А. С. ЕСИПОВ

bhv

ИНФОРМАТИКА и ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ

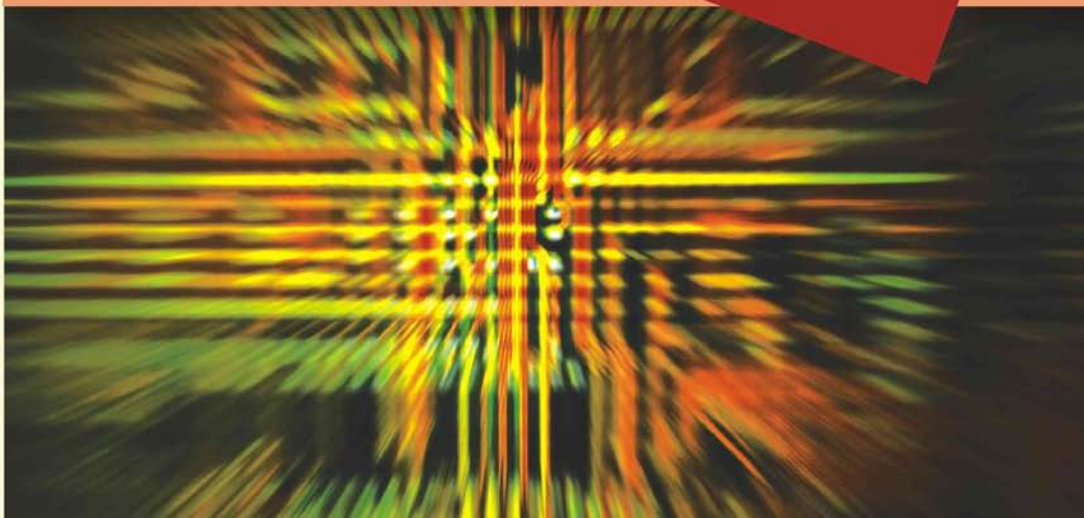
ДЛЯ УЧАЩИХСЯ ШКОЛ И КОЛЛЕДЖЕЙ

БАЗОВЫЙ И УГЛУБЛЕННЫЙ УРОВЕНЬ

ЯЗЫК ПРОГРАММИРОВАНИЯ VISUAL BASIC

ЗАДАЧИ ДЛЯ САМОСТОЯТЕЛЬНОГО РЕШЕНИЯ

ИНФОРМАТИКА И ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ



Александр Есипов

Информатика и информационные технологии для учащихся школ и колледжей

Санкт-Петербург
«БХВ-Петербург»
2004

УДК 681.3.06(075.3)
ББК 32.973я721
Е83

Есипов А. С.

Е83 Информатика и информационные технологии для учащихся школ и колледжей. — СПб.: БХВ-Петербург, 2004. — 480 с.: ил.

ISBN 5-94157-537-8

В учебном пособии рассматриваются следующие разделы школьного курса информатики и информационных технологий: устройство и основные принципы работы компьютера, информация и информационные процессы, обработка числовой и текстовой информации, работа с базами данных, алгоритмизация и программирование на языке Visual Basic. В первой части материал излагается на более простом, базовом уровне, во второй части — на углубленном. Все разделы учебника сопровождаются многочисленными примерами с комментариями и задачами для самостоятельного решения.

Пособие предназначено для учащихся 7—9 классов общеобразовательной школы, материал второй части может быть полезен учащимся 10—11 классов и студентам средних профессиональных ОУ

УДК 681.3.06(075.3)
ББК 32.973я721

Группа подготовки издания:

Главный редактор	*Екатерина Кондукова*
Зам. главного редактора	*Людмила Еремеевская*
Зав. редакцией	*Григорий Добин*
Редактор	*Галина Смирнова*
Компьютерная верстка	*Натальи Смирновой*
Корректор	*Наталия Першакова*
Дизайн серии	*Инны Тачиной*
Оформление обложки	*Игоря Цырульникова*
Зав. производством	*Николай Тверских*

Лицензия ИД № 02429 от 24.07.00. Подписано в печать 30.07.04.
Формат 70×100 1/16. Печать офсетная. Усл. печ. л. 38,7.
Тираж 3000 экз. Заказ №
"БХВ-Петербург", 190005, Санкт-Петербург, Измайловский пр., 29.

Гигиеническое заключение на продукцию, товар № 77.99.02.953.Д.001537.03.02 от 13.03.2002 г. выдано Департаментом ГСЭН Минздрава России.

Отпечатано с готовых диапозитивов
в ГУП "Типография "Наука"
199034, Санкт-Петербург, 9 линия, 12

ISBN 5-94157-537-8

Содержание

Предисловие

Материал учебника соответствует требованиям Министерства образования Российской Федерации к результатам обучения информатике и информационным технологиям, охватывает все важные разделы предмета и по отдельным вопросам несколько выходит за его пределы.

Углубленное изложение материала в некоторых разделах учебника согласуется с вопросами сборников тестовых заданий для выпускников средней школы и абитуриентов вузов, с программами профильных общеобразовательных курсов [22]. Этот материал также можно успешно использовать в работе факультативов. Во всех главах учебника представлены многочисленные примеры с комментариями и задачи для самостоятельного решения.

Учебник разделен на две части. Главы первой части предназначены для начинающих и знакомят с персональным компьютером, работой на нем, основными понятиями предмета информатики. На примерах приложений Калькулятор, Блокнот и графический редактор Paint рассматривается работа с приложениями, с операционной системой и файлами. Обучаемые знакомятся со свойствами информации и способами ее кодирования. На элементарном уровне дается материал по используемым в компьютерах системам счисления, логическим элементам и функциям.

Дальнейшие главы первой части учебника знакомят с основным принципом работы компьютера и его устройствами. Кратко рассматривается история формирования структуры персонального компьютера и совершенствования пользовательского интерфейса.

Большое внимание в этой части уделено компьютерным технологиям. После краткого обзора назначения и состава прикладного программного обеспечения на простых примерах рассматривается работа с текстовым редактором Word, таблицами Excel и базами данных Access.

Завершается первая часть главами, посвященными алгоритмизации и программированию. Практика показывает, что эти разделы школьной информатики, как бы их не называли, были и пока остаются наиболее важными, позволяющими направленно формировать у обучаемых логическое мышление, системный, операционный подход к решению задач различного характера и степени сложности. Этим разделам в учебнике уделено большое внимание.

Выбор автором в качестве базового языка программирования самого распространенного в мире языка Visual Basic 6.0 (VB-6) объясняется многими его преимуществами. К ним следует отнести простоту языка, его доступность,

лаконичность и наглядность записи программ, легкость чтения и понимания программ на начальных этапах изучения языка и многое другое. Эти качества позволяют успешно применять Visual Basic 6.0 при начальном знакомстве с программированием в общеобразовательных заведениях. Наиболее успешно это происходит там, где обучаемые имели предварительное знакомство с языком QBasic, изучение которого начинается в 5—6 классах [22].

Главы второй части учебника на примерах знакомят с управляющими элементами VB-6 и методами построения приложений. В них также рассматриваются дополнительные вопросы, связанные с определением количества информации. На более высоком уровне обучаемые знакомятся с арифметическими и логическими основами построения и работы компьютеров, с алгеброй логики и решением логических задач.

Приложение 1 содержит материал по работе с файлами, папками, ярлыками. В приложении 2 приводятся сведения о синтаксисе функции форматирования `Format`.

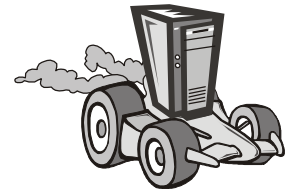
Автор выражает искреннюю признательность сотруднику Областного института развития образования Савицкой Валентине Григорьевне за оказанные ею помощь и поддержку.

Замечания по учебнику и предложения автор просит присылать в Областной институт развития образования при Комитете общего и профессионального образования Ленинградской области по адресу: 196137, Санкт-Петербург, Чкаловский пр., д. 25-а, кабинет информатики, **loiro@soros.spb/ru**.

ЧАСТЬ I

Базовый уровень

Глава 1



Первое знакомство с компьютером

1.1. Персональный компьютер

Компьютером называют электронное устройство, способное выполнять заданную последовательность действий по приему, хранению, преобразованию и выдаче информации.

Состав компьютера

В состав компьютера обычно входят: системный блок, монитор (дисплей), клавиатура, манипулятор "мышь", гарнитура (наушники и микрофон). Также к системному блоку могут быть подключены: звуковые колонки, принтер, сканер, другие устройства.

Включение и выключение питания

На передней панели системного блока находится кнопка включения питания (Power) и кнопка перезагрузки компьютера (Reset). Заметим, что выключать компьютер кнопкой включения питания крайне нежелательно. Выключение выполняется через главное меню командой **Завершение работы**.

При включении компьютера из памяти загружается *операционная система* — набор программ, управляющих работой компьютера и обеспечивающих связь пользователя с компьютером. Заметим, что пользователями в информатике обычно называют тех, кто использует компьютер в своих целях: решает простые и сложные вычислительные задачи, сочиняет музыку, путешествует по сети Интернет, набирает и оформляет различные тексты и т. п.

На передней панели дисплея также имеется кнопка включения питания и элементы управления настройкой изображения. Обычно питание дисплея отключается автоматически после выключения компьютера.

Дисководы внешней памяти

На переднюю панель системного блока выведены органы управления дисководами внешней памяти компьютера: карманы для вставки трехдюймовых гибких дискет и карманы для компакт-дисков.

Клавиатура

На клавиатуре расположено более ста клавиш. Клавиши объединены в группы по назначению: информационные, управляющие, командные и т. п. Как правило, используется двойное и тройное обозначение информационных клавиш. Клавиатуры адаптируются под национальные алфавиты. Подробнее о клавиатуре будет рассказано в *гл. 2*.

Манипулятор "мышь"

Это второй по важности после клавиатуры инструмент работы пользователя. Перемещение мыши по коврику сопровождается согласованным перемещением указателя мыши на экране дисплея. Кнопки мыши служат для подачи команд щелчками на выбранных объектах. Кроме того, мышью можно перемещать на экране объекты и их границы.

1.2. С чего начать?

Рабочий стол

После включения компьютера на экран дисплея выводится картинка *рабочего стола* (рис. 1.1). Вид этой картинки зависит от установленной в компьютере операционной системы и от желаний пользователя. Обычно на рабочем столе находится несколько *значков* и *ярлыков*. Ярлыки отмечены маленькими стрелками. Значки и ярлыки служат для открытия или быстрого запуска соответствующих им папок, программ, документов.

Среди них можно, например, увидеть ярлыки таких папок:

- **Мои документы** — папка используется по умолчанию для сохранения документов, рисунков и других работ пользователя. Содержит в себе папку Мои рисунки;
- **Мой компьютер** — в папке отображается содержимое дисков памяти (гибкого диска — дискеты, жесткого диска — винчестера и компакт-диска — CD-ROM-диска);
- **Корзина** — в корзине хранятся документы и программы, которые были удалены пользователем. Там они сохраняются до очистки корзины или возврата их на прежнее место.

В зависимости от предварительной настройки вид рабочего стола может несколько различаться. Но имеются и общие для различных вариантов оформления стола элементы.

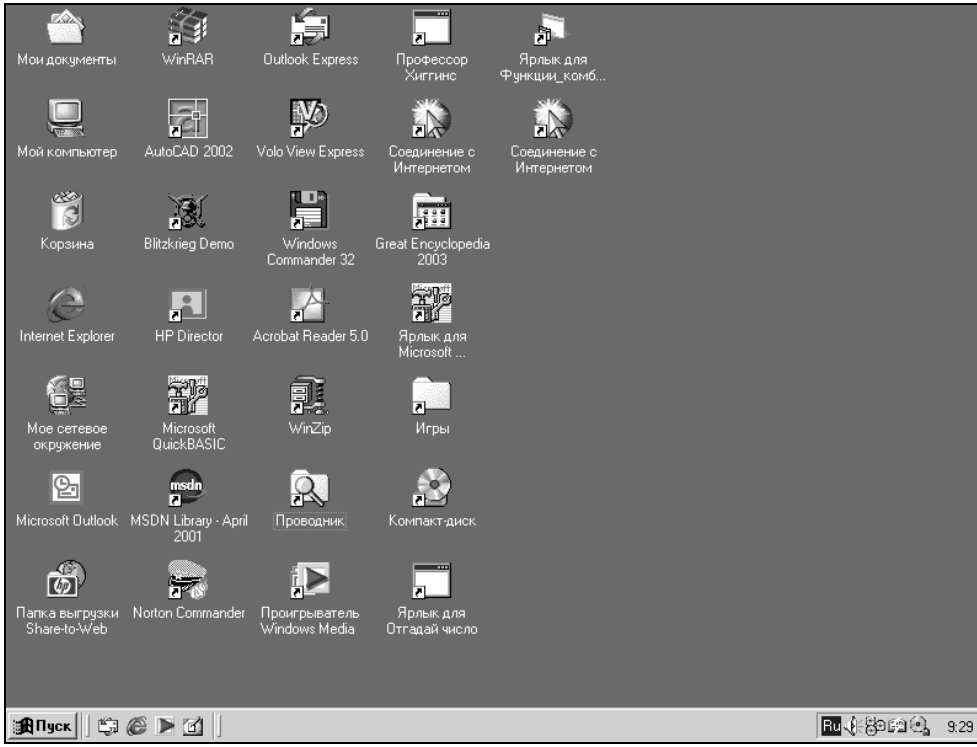


Рис. 1.1. Рабочий стол системы Windows

Панель задач

Внизу рабочего стола расположена *панель задач*. Слева на ней находится кнопка **Пуск**, которая открывает *главное меню*. Через главное меню выполняются быстрый запуск программ, поиск файлов, получение справок, настройка устройств, а также завершение работы компьютера.

Одновременно может быть запущено несколько программ, открыто несколько документов. После запуска программ на панель задач выводятся соответствующие этим программам кнопки. Эти кнопки удобно использовать для быстрого перехода от одной программы к другой.

С помощью расположенной на панели задач кнопки **Свернуть все окна** можно быстро свернуть все открытые окна, очистив при этом рабочий стол.

В правой части панели задач расположен *системный лоток*, в котором могут быть размещены: циферблат системных часов, кнопка переключателя языка **Русский | Английский** и другие кнопки.

Если подводить указатель мыши к расположенным на панели задач кнопкам, то рядом с ними в желтом прямоугольнике будет появляться *всплывающая подсказка*. Такие подсказки или сообщают о назначении этой кнопки, или уточняют значение какого-либо параметра. Так, если расположить указатель мыши на циферблате системных часов, то всплывет информация о дате — например, *26 марта 2004*.

Если выполнить быстрый двойной щелчок мышью[1] (ее *левой* кнопкой) на циферблате системных часов, то откроется окно **Свойства: Дата и время**, в котором можно внести изменения в дату и время. Щелчок *правой* кнопкой мыши на свободном пространстве панели задач открывает окно меню с командами настройки рабочего стола.

Для переключения клавиатуры с русского языка на английский нужно щелкнуть мышью на значке индикации языка и в открывшемся списке выбрать нужную строку.

Если подвести мышь к кнопке **Пуск**, то всплывет подсказка *Начните работу с нажатия этой кнопки*. Вот и ответ на вопрос, поставленный в названии этого раздела: "С чего начать?"

1.3. Приложение Калькулятор

Знакомство с приложениями удобно начать с Калькулятора. При всей своей простоте он содержит множество элементов, характерных для более сложных приложений.

Приложение Калькулятор предназначено для выполнения простых и сложных вычислений. В соответствии с этим он имеет два вида: обычный и инженерный.

Запуск Калькулятора и элементы его окна

Нажатие кнопки **Пуск** открывает главное меню. В последовательно раскрывающихся меню выберем пункты: **Программы | Стандартные | Калькулятор** (рис. 1.2).

Щелчок мышью на имени программы выводит на экран окно Калькулятора (рис. 1.3).

---

[1] В дальнейшем под щелчком мыши без указания кнопки будет пониматься щелчок левой кнопкой мыши. Щелчок правой кнопкой мыши будет оговариваться.

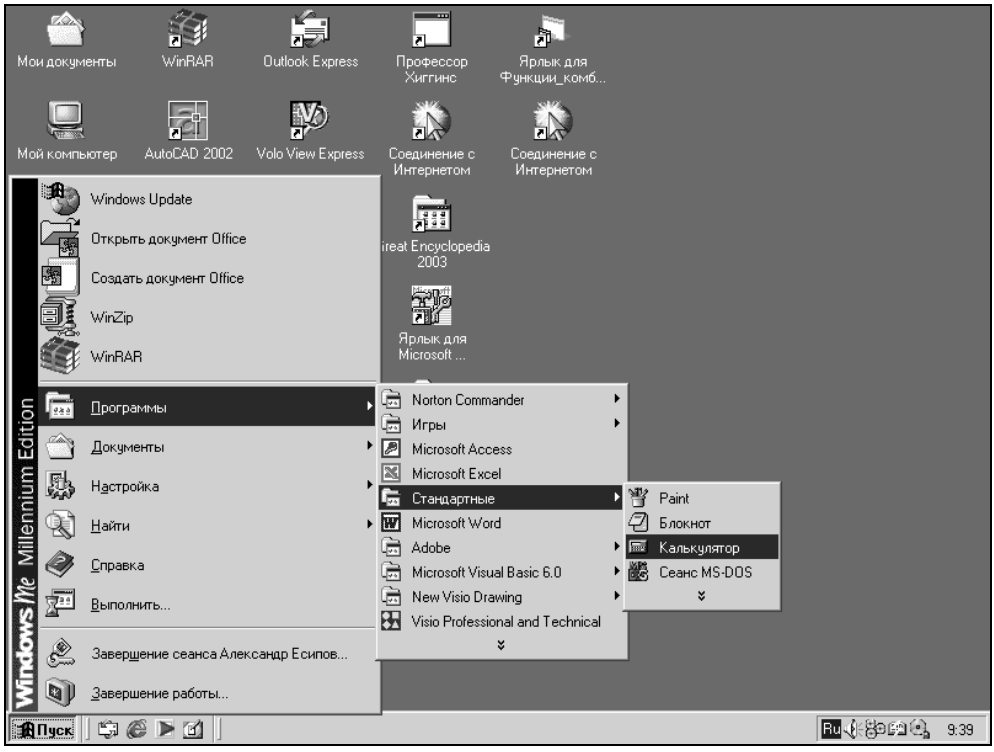


Рис. 1.2. Запуск приложения Калькулятор

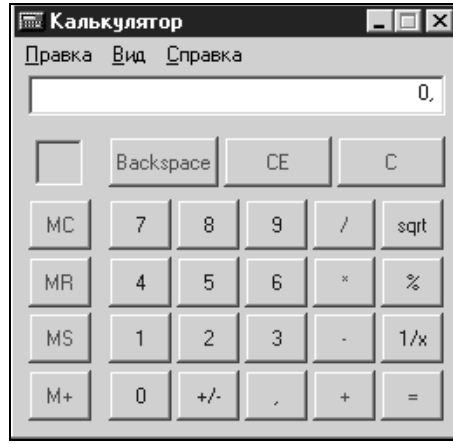


Рис. 1.3. Окно приложения Калькулятор

Заголовок окна

Верхняя строка окна калькулятора — это его *заголовок*. Если установить на синее поле заголовка указатель мыши, то, перемещая мышь при нажатой левой кнопке, окно можно переместить по экрану в удобное для работы место. В левой части заголовка находится кнопка *системного меню*. Команды системного меню в основном дублируют команды, подаваемые с помощью кнопок правой части заголовка[2]: **Свернуть**, **Развернуть** и **Закрыть**.

Команда **Свернуть** отправляет калькулятор в виде кнопки на панель задач. Щелчок мышью на этой кнопке на панели задач возвращает окно калькулятора на экран. Кнопка **Развернуть** у калькулятора не работает, поэтому размеры окна фиксированы и не могут быть изменены. Кнопка **Закрыть** удаляет калькулятор с экрана и панели задач.

Ниже заголовка в окне калькулятора помещена строка *меню*, под которой находится текстовое окно записи чисел и вывода результатов вычислений.

Меню Калькулятора

Меню **Правка** содержит две команды: **Копировать** (<Ctrl>+<C>) и **Вставить** (<Ctrl>+<V>). В скобках записаны комбинации клавиш клавиатуры, одновременные нажатия которых дублируют эти две команды. Команда **Копировать** отправляет копию записанного в текстовом окне числа в буфер памяти для хранения. Команда **Вставить** считывает число из буфера памяти и вставляет его в текстовое окно.

Меню **Вид** содержит команды **Обычный** и **Инженерный**, переключающие вид калькулятора.

Справочная система Калькулятора

Команда меню **Справка | Вызов справки** вызывает справочную систему калькулятора (рис. 1.4).

Изменение размеров окна

Окно справки несколько отличается от окна калькулятора. Вы можете не только перемещать окно справки по рабочему столу, "ухватив" мышью за заголовок, но и изменять его размеры. В правом верхнем углу окна работает средняя кнопка — **Развернуть/Восстановить**, позволяющая развернуть окно справки во весь рабочий стол и вернуть ему нормальные размеры. Если при нормальных размерах окна подвести указатель мыши (стрелку) к границе окна до превращения его в двойную стрелку, то при нажатой левой кнопке мыши эту границу можно перемещать.

---

[2] Названия команд всплывают при подведении курсора мыши к соответствующей кнопке.

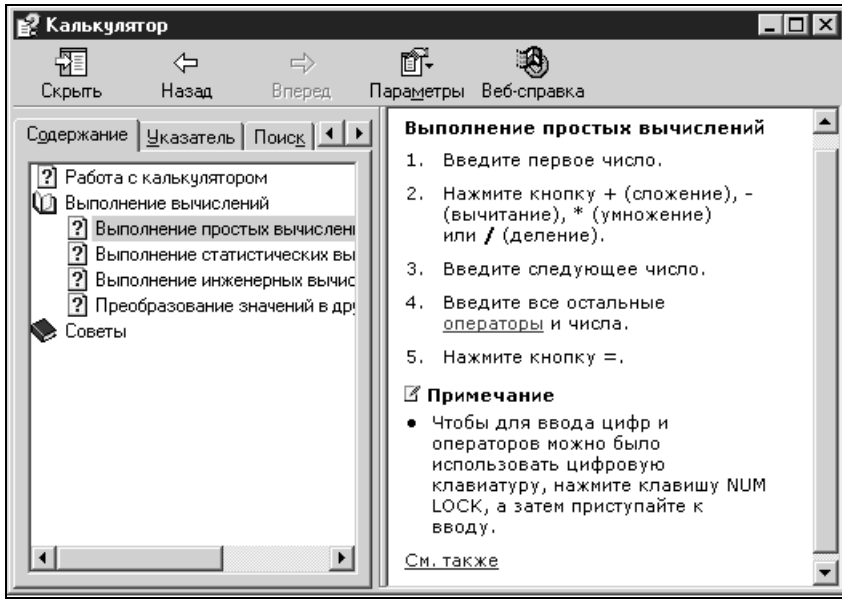


Рис. 1.4. Окно справочной системы калькулятора

Назначение вкладок окна вопросов

Окно справки разделено на две части. Левая часть служит для задания справочной системе вопросов, правая часть — для вывода ответов. Границу между ними также можно перемещать. Левая часть имеет несколько *вкладок* — вариантов задания вопросов, — становящихся активными при щелчке на их названиях.

Вкладка **Содержание** выводит на экран информацию, напоминающую оглавление справочника. При выборе щелчком мыши вопроса текст ответа выводится в текстовое окно правой части.

Вкладка **Указатель** выводит все вопросы, связанные с использованием калькулятора. На этой вкладке предлагается ввести в текстовое окно ключевое слово для организации поиска ответа. Можно поставить вопрос, выделив нужную тему в списке разделов и нажав кнопку **Показать**. Для ускорения выбора темы можно использовать расположенную у правой границы окна вопросов полосу прокрутки.

Вкладка **Найти** также требует ввести для поиска нужной темы ключевое слово.

В некоторых вариантах окна справки может быть и вкладка **Избранное**. Она открывает список разделов, которыми пользователь интересовался в последнее время. Этот список можно выборочно или полностью очищать.

Работа с калькулятором

Рассмотрим работу калькулятора в обычном режиме. Инженерный режим будет рассмотрен в *гл. 3*, *4* и *20*. Для включения обычного режима нужно открыть меню **Вид** и выбрать пункт **Обычный**.

Команда **Справка | Вызов справки | Содержание | Выполнение вычислений | Выполнение простых вычислений** открывает статью справочной системы, в которой по пунктам расписано выполнение простых вычислений. Заметим, что структура и содержание справочной системы калькулятора могут различаться в зависимости от применяемой в компьютере операционной системы.

Вводить числа и выполнять операции над ними можно двумя способами. Во-первых, можно нажимать на кнопки калькулятора с помощью мыши. Во-вторых, это можно делать, нажимая эквивалентные клавиши клавиатуры. Соответствие кнопок клавишам записано в таблице, раскрываемой командой **Справка | Вызов справки | Содержание | Полезные советы | Клавиши, эквивалентные кнопкам калькулятора**.

В этой таблице можно увидеть, что:

- кнопке **C** калькулятора соответствует клавиша <Esc> (Отменить), расположенная в левом верхнем углу клавиатуры);
- кнопке **CE** — клавиша <Delete> (Очистить);
- кнопке **=** (Равно) — клавиша <Enter> (Ввод, подтверждение) и т. д.

Аналогично соответствуют друг другу кнопки и клавиши цифр и арифметических действий.

Из справки можно узнать, что кнопки **MS**, **MR**, **MC** и **M+** служат для работы с памятью калькулятора:

- **MS** — занесение числа в память;
- **MR** — вызов из памяти;
- **MC** — очистка памяти;
- **M+** — прибавление числа к содержимому памяти.

Правая кнопка мыши

Много полезной информации можно получить с помощью правой кнопки мыши. Подведите указатель мыши к какой-нибудь кнопке калькулятора и щелкните на ней правой кнопкой. Рядом с указателем на экран выводится прямоугольник с вопросом: "Что это такое?" Переместите указатель на поле вопроса до окрашивания его в синий цвет и сделайте на нем щелчок правой или левой кнопкой. Появляется желтый прямоугольник всплывающей подсказки, уже знакомой нам по рабочему столу. В нем дана подробная информация о назначении кнопки и особенностях ее работы.

Примеры вычислений

Введем следующие обозначения. Пусть запись <24>, <+>, <32>, <=> указывает, что в такой последовательности вводились числа и нажимались кнопки действий над ними и кнопка вывода результата.

Пример 1

Выполнить действия: 7 * 143 * 1665 / 3. Звездочка (*) — знак умножения, наклонная черта (/) — знак деления.

При выполнении такой последовательности действий необязательно выводить кнопкой <=> на экран промежуточные результаты. Поэтому вторая строка короче:

<7>, <*>, <143>, <=>, <*>, <1665>, <=> </>, <3>, <=>

<7>, <*>, <143>, <*>, <1665>, </>, <3>, <=>

В обоих случаях будет выведен правильный ответ — шесть отличных отметок.

Пример 2

Выполнить действия: (72 − 36) / (6 * 3).

При работе с памятью нужно помнить, что в память записывается число из строки ввода чисел и вывода результатов. В эту же строку число выводится из памяти. Проверьте на калькуляторе результаты приведенных ниже последовательностей действий:

- <6>, <*>, <3>, <MS> вводит в память число 3;
- <6>, <*>, <3>, <=>, <MS> вводит в память число 18;
- <6>, <*>, <3>, <=>, <MS>, <72>, <−>, <36>, </>, <MR> выводит из памяти в строку результатов число 18;
- <6>, <*>, <3>, <=>, <MS>, <72>, <−>, <36>, </>, <MR>, <=> выводит в строку результатов правильный результат — число 2.

Упражнения

Выполните действия:

1. 5 * 7 * 2 − 12 (ответ: 58);
2. (15 * 3 + 4 * 25) / 5 (ответ: 29);
3. (200 − 13 * 14) + (25 * 7 − 19) (ответ: 174).

Глава 2

Работа на компьютере

2.1. Файлы и каталоги

Файлы

Файлом называют объединение под одним именем информации, как правило, связанной единым смыслом и назначением. В качестве файла могут выступать программа, текст, иллюстрация, другая информация. Файл может не содержать информацию, быть пустым.

Характеристиками файла служат:

- *имя* файла — например, TETRIS, Расписание уроков, Рисунок, Дом. Имя файла ограничено длиной в 256 символов. При записи имен файлов запрещено использовать символы: /, \, |, :, <, >, " и ?;
- *тип* файла — определяется его назначением, например: Текстовый документ, Видеозапись, Точечный рисунок. Тип файла характеризуется его *расширением*, которое обычно записывается через точку после имени файла и содержит небольшое число символов. Например: *.txt, *.com, *.bmp и т. п. Здесь символ * указывает любое имя файла. Каждый тип файла имеет свой значок;
- *размещение* файла — это адрес места в памяти компьютера, где файл хранится. Например, запись C:\Мои документы\Игры\TETRIS означает, что файл игры TETRIS находится в папке Игры, которая расположена в папке Мои документы на диске памяти C:;
- *приложение* — имя программы, в которой файл был создан и которая может его открыть;
- *размер* файла (в байтах) — например, текстовый файл в простейшем случае требует для своего хранения в памяти компьютера столько байт, сколько в нем символов, включая пробелы;
- *дата* создания и дата последнего изменения файла.

Кроме перечисленных общих характеристик некоторые файлы могут иметь свойства, связанные со статистикой. Например, среди свойств документа

текстового редактора Word на экран выводится количество страниц в документе, число абзацев, строк, слов, знаков (символов), пробелов и т. п. Файлы типа Звукозапись сопровождаются характеристиками записи: Длительность (в секундах), Формат звука (например, 16 бит, Стерео).

Папки и каталоги

В памяти современных компьютеров десятки и более тысяч файлов. Пусть файлы — это книги. Представьте себе библиотеку, в которой книги стоят на полках неупорядоченные по авторам и назначению. Для поиска нужной книги в такой библиотеке приходится затрачивать много времени. В худшем случае потребуется перебрать все книги библиотеки.

Для сокращения времени поиска файлов их помещают в папки. Папкам присваивают имена. Каждая папка может содержать файлы и другие папки с файлами. Такие вложения папок в папки могут быть многократными. Поэтому структура связей между папками и файлами напоминает дерево, где папки — это ветви и сучки, а файлы — листья.

Файловой системой называют часть операционной системы, которая обеспечивает выполнение операций над файлами. В набор этих операций входят создание и удаление файла, его чтение и запись, наименование и переименование, копирование и редактирование, другие операции. В *приложении 1* рассмотрены все основные действия с файлами, папками и ярлыками.

Файловой структурой называют совокупность файлов и взаимосвязей между ними. Это способ организации файлов. Обычно под файловой структурой понимается каталог всех файлов, находящихся в памяти компьютера.

Каталог жесткого диска, как правило, имеет древовидную многоуровневую структуру. Иначе мы получаем необозримое и трудно управляемое множество файлов. Нижний уровень дерева каталога называют корневым каталогом. Обычно на уровне корневого каталога компьютера находятся папки Мои документы, Мой компьютер, Корзина и другие разделы файловой структуры.

Щелчок мышью на знаке плюс в строке **Мой компьютер** раскрывает следующий уровень этого раздела каталога. Это уровень всех дисков компьютера и панели управления. При этом знак плюс заменяется знаком минус. Щелчок на знаке минус закрывает раскрытый каталог. Щелчок на знаке плюс в строке диска [C:] раскрывает каталог этого диска, позволяя таким образом переходить к следующему уровню, и т. д.

Каждый диск имеет свой корневой каталог, который можно сравнить с ветвями, отходящими от ствола дерева. От этих ветвей расходятся ветви каталога второго уровня, дочернего по отношению к родительскому, корневому каталогу. И так далее, до листьев-файлов (рис. 2.1).

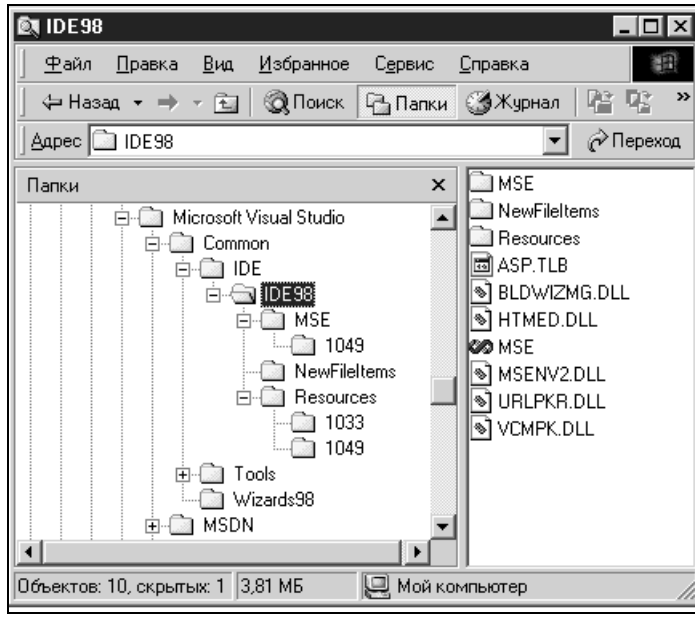


Рис. 2.1. Пример дерева каталога

На рис. 2.1 показана раскрытая часть дерева каталога, вложенного в папку Microsoft Visual Studio. Вертикальные пунктирные линии обозначают уровни вложения папок. Третья слева линия — это уровень корневого каталога — жесткого диска C: (винчестера). Содержимое выделенной и открытой папки IDE98 показано в правом окне. В папку вложены три папки и семь файлов.

Древовидная организация файловой структуры улучшает наглядность в размещении множества файлов и каталогов, позволяет работать сразу с целой группой файлов, входящих в один каталог. Она также ускоряет доступ к требуемому файлу.

При работе с файлами для обеспечения доступа к файлу необходимо передавать компьютеру маршрут его поиска по каталогам. *Полным маршрутом (путем)* к файлу называют последовательность каталогов, ведущую от диска к этому файлу. Каталоги в записи пути разделяются символом \, например:

C:\Мои документы\Издательство BHV\Информатика 2004\Глава 2.doc

2.2. Текстовый редактор Блокнот

Текстовыми редакторами называют приложения (программы), с помощью которых можно создавать текстовые документы и работать с ними — просматривать, исправлять, редактировать, отправлять на печать и т. п. Прило-

жение Блокнот — наиболее простой из текстовых редакторов. Он позволяет работать с текстовыми документами небольшого объема.

При работе с этим простым текстовым редактором мы познакомимся с клавиатурой и основными действиями с папками и файлами.

Окно текстового редактора Блокнот

Приложение Блокнот запускается командой **Пуск | Программы | Стандартные | Блокнот**. С запуском Блокнота на экран выводится его окно (рис. 2.2).

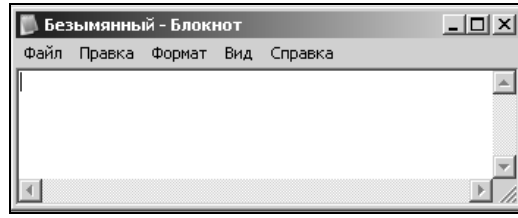


Рис. 2.2. Окно текстового редактора Блокнот

Одновременно на панели задач появляется кнопка со значком приложения Блокнот и именем документа. В начале работы всем документам присваивается имя Безымянный.

Окно приложения Блокнот имеет знакомые по работе с Калькулятором и справочной системой Калькулятора элементы. Это заголовок с кнопкой системного меню, названием документа (**Безымянный**), названием приложения (**Блокнот**) и тремя кнопками: **Свернуть**, **Развернуть/Восстановить** и **Закрыть**. При подведении к этим кнопкам указателя мыши появляется всплывающая подсказка.

Напомним, что окно приложения Блокнот можно перемещать ("ухватив" за заголовок), и также можно изменять размеры окна, перемещая мышью его границы. Кроме того, окно можно развернуть во весь экран, восстановить, свернуть, отправив в виде кнопки на панель задач, и окно можно закрыть. Полезно проверить все эти возможности.

Ниже заголовка расположено меню приложения Блокнот. Каждый пункт меню может открывать свое подменю с помощью "быстрой клавиши" клавиатуры (по одной букве в названиях пунктов подчеркнуто). Например, чтобы открыть меню **Файл**, нужно нажать вместе две клавиши <Alt> и <Ф> (сначала <Alt>, затем, не отпуская ее, нажать <Ф>). Эту команду записывают так: <Alt>+<Ф>. Клавиатура при этом должна быть переключена на русский язык (значок **Ru** на лотке в правой части панели задач).

Последовательно открывая пункты меню, познакомьтесь с их содержимым. Некоторые пункты меню **Правка** записаны блеклым шрифтом. Эти команды

будут выделены и станут доступны, когда начнется работа с документом и будет что копировать, удалять и т. п.

Обязательно познакомьтесь с содержанием справочной системы. Обращайтесь к справке при затруднениях в работе с Блокнотом. Этим можно существенно сэкономить время и избежать ошибок.

Начнем с того, что зададим документу имя и выберем место для его сохранения.

Первое сохранение документа

В меню **Файл** выберем пункт **Сохранить как**. По этой команде открывается диалоговое окно **Сохранить как** (рис. 2.3).

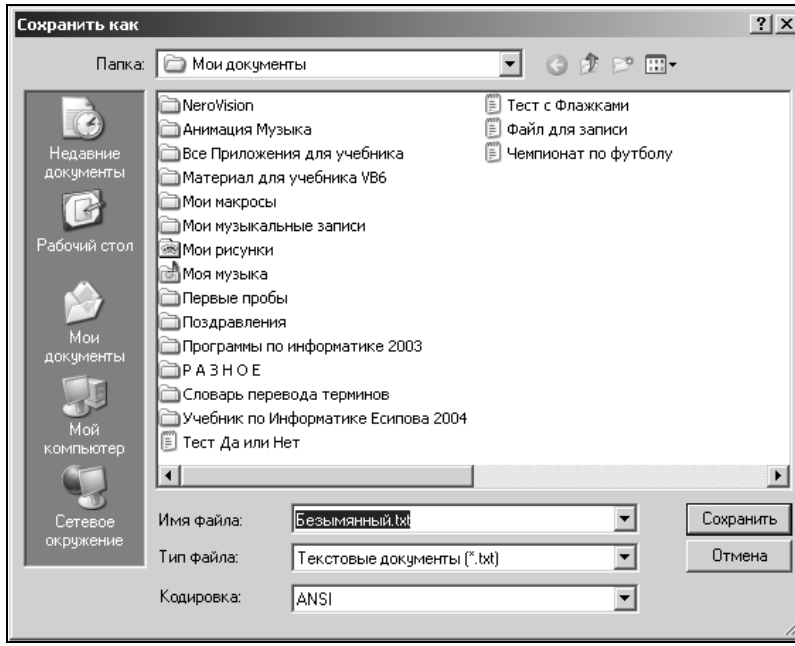


Рис. 2.3. Диалоговое окно **Сохранить как**

В правой части заголовка окна **Сохранить как** расположены кнопка **Закрыть** (с крестом) и кнопка **Справка** (со знаком вопроса). Щелчок на кнопке **Справка** добавляет к указателю (стрелке) мыши знак вопроса. Если подводить такой указатель к элементам окна и щелкать на них мышью, то на экран будет выводиться справка о назначении этого элемента. О назначении элементов окна также можно узнать из всплывающей подсказки. На рис. 2.3 показана всплывающая подсказка кнопки **Создание новой папки**.

Под заголовком окна в раскрывающемся списке **Папка** отображается название выбранной папки. По умолчанию операционная система предлагает сохранить документ в папке Мои документы. Это первый уровень каталога компьютера — корневой каталог.

В левой части окна находятся пять кнопок быстрого доступа к элементам файловой структуры.

Почти все поле окна занимает список папок и файлов второго уровня каталога. Это содержимое папки Мои документы. В списке только один текстовый файл — Чемпионат по футболу. Если раскрыть расположенный внизу окна список **Тип файла** и выбрать в нем строку **Все файлы (*.*)**, то в списке могут появиться файлы других типов.

Создание новой папки

Чтобы не засорять каталог россыпью файлов различного типа и назначения, можно создать новую папку, назвать ее, например, Первые опыты, и записывать в нее создаваемые документы.

В окне **Сохранить как** щелчок на кнопке **Создание новой папки** добавляет в список новый значок папки с именем Новая папка. Имя записано инверсным шрифтом — белыми буквами на синем фоне. Такое изображение слова называют *выделенным*. Начинайте записывать новое имя папки — "Первые опыты". С началом записи старое имя стирается. Выделенное старое имя можно также удалить клавишей <Delete>.

При записи имени все буквы можно найти в трех рядах левой части клавиатуры. Запись прописных (заглавных) букв выполняется при нажатой клавише <Shift>. Пробел между словами вставляется длинной клавишей в нижнем ряду клавиатуры.

Запись файла в папку

Откроем щелчком мыши созданную нами папку Первые опыты, подготовив ее для записи пока Безымянного документа. При этом имя папки показывается в поле списка **Папка**. Эта папка пока пустая. Вложенных в нее файлов и папок нет.

Поместим указатель мыши на текстовое поле **Имя файла**. Вид указателя изменился. Проведя по имени Безымянный мышью, выделим его. Имя при выделении примет вид — белые буквы на черном фоне.

Запишем новое имя документа, например, Блокнот Шаг 1. С началом записи по выделенному старому имени оно стирается. Нажмем на кнопку **Сохранить**. Окно сохранения документа закрывается. В памяти компьютера записан пустой текстовый файл. Путь к нему (его адрес) такой: C:\Мои документы\Первые опыты\Блокнот Шаг 1.txt.

Переименование файла, папки

В процессе работы может возникнуть желание переименовать папку или файл. Это можно сделать разными способами. Будем использовать знакомое диалоговое окно **Сохранить как**, открываемое одноименной командой меню **Файл**.

В окне **Сохранить как** в папке Мои документы находим нашу папку Первые опыты и делаем на имени папки щелчок правой кнопкой мыши. Открывается так называемое *контекстное меню*, вид и содержание которого меняются в зависимости от ситуации во время его вызова.

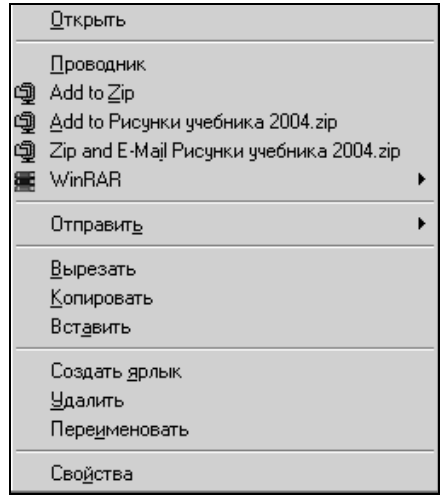


Рис. 2.4. Контекстное меню правой кнопки

Среди пунктов меню имеется много таких, назначение которых понятно из названия. Это **Открыть**, **Отправить**, **Копировать**, **Удалить**, **Переименовать** и т. д. Последний пункт **Свойства** открывает диалоговое окно **Свойства**, в котором можно познакомиться со всеми свойствами папки или файла.

Выбор пункта **Переименовать** удаляет окно контекстного меню, выделяет имя папки Первые опыты и окружает его прямоугольной рамкой. Запишем в эту рамку новое имя папки, например, Первые пробы. После записи нового имени утвердим переименование щелчком мыши вне имени папки.

Двойным щелчком мыши откроем папку Первые пробы и переименуем файл Блокнот Шаг 1.txt, присвоив ему, например, имя Блокнот Проба 1 (расширение останется прежним). В итоге получили новый путь к создаваемому текстовому документу: C:\Мои документы\Первые пробы\Блокнот Проба 1.txt.

Работа с клавиатурой

Познакомимся с клавиатурой и наберем простой текст документа Блокнот Проба 1.

Как правило, начинающий пользователь набирает текст, работая одним пальцем и подолгу выискивая нужные клавиши. Этого не избежать. Все так начинали. Квалифицированные пользователи используют слепой метод набора, когда взгляд следит не за клавиатурой, а за вводимым текстом. Приблизиться к этому уровню можно с помощью клавиатурных тренажеров.

Клавиатурных тренажеров множество. Имеются детские тренажеры с увлекательными картинками. Имеются и профессиональные. Все они требуют правильной постановки рук, некоторой привязки пальцев к определенным клавишам. Напомним, что переучиваться труднее, чем научиться.

Клавиатура

Кнопки клавиатуры объединены в блоки. Верхний ряд состоит из пяти блоков. Первый блок — одна клавиша <Esc> (Отмена). Три следующие блока — 12 функциональных клавиш. Они используются в разных программах для подачи различных команд. Первая клавиша последнего (пятого) блока <Print Screen SysRq> служит для снятия в буферную память копии картинки экрана для дальнейшего использования. Остальные две клавиши рассмотрим позже.

Под клавишами верхнего ряда размещены четыре блока клавиш. Левый, самый большой блок служит для набора текста. Последняя клавиша его верхнего ряда (<Backspace> или <←>) стирает символ слева от текстового курсора. Остальные клавиши многофункциональны (до четырех символов). Вид символа зависит от языка (**Русский/Английский**) и от клавиши <Shift> (нажата или нет).

В левой части блока находятся клавиши: табуляции — <Tab>, включения прописных букв — <Caps Lock>, переключения функций клавиш — <Shift>, две управляющие клавиши: <Ctrl> (контроль) и <Alt> (альтернатива). Между ними — клавиша вывода на экран главного меню Windows (дублирует кнопку **Пуск** на панели задач).

В правой части блока размещены клавиша ввода и подтверждения <Enter>, клавиши <Shift>, <Alt> и <Ctrl> — дублирующие, но не во всем равноценные клавишам в левой части блока. Там же находится клавиша вывода главного меню и клавиша контекстного меню. В нижней части блока — клавиша пробела.

Следующий блок из шести клавиш содержит клавиши: <Insert> — вставить (позволяет вставить справа от курсора новый символ вместо старого), <Delete> — удалить, <Home> — в начало, <End> — в конец, <Page Up> — пере-

ход на страницу текста назад (вверх), <Page Down> — переход на страницу вперед (вниз).

Блок из четырех клавиш со стрелками (←, →, ↑ и ↓) состоит из клавиш перемещения по экрану текстового курсора — вертикальной мерцающей линии.

Последний, правый блок клавиш удобен при вводе большого количества числовых данных. Клавиша <Num Lock> переключает клавиши с режима ввода цифр на режим перемещения курсора. Режим ввода цифр индицируется светящимся светодиодом, расположенным в правой верхней части клавиатуры. Там же размещена индикация клавиш <Caps Lock> и <Scroll Lock>. Блок имеет клавиши четырех арифметических действий и клавишу <Enter>.

Набор текста

Начав набирать текст, можно увидеть, что вводимые с клавиатуры русские буквы имеют непонятный вид. Исправить это можно, выбрав нужный *шрифт* — набор символов одного рисунка (стиля).

Среди множества шрифтов различают группы: **антиква** (обыкновенные, с засечками) и **гротеск** (рубленые, без насечек). Засечки помогают связывать буквы в слова и быстрее читать тексты. Рубленые шрифты используются, как правило, для заголовков.

В окне приложения Блокнот командой меню **Правка | Шрифт** откройте диалоговое окно **Шрифт** (рис. 2.5).

Диалоговое окно **Шрифт** — это стандартное диалоговое окно, используемое во многих приложениях. Оно имеет несколько списков и текстовых окон для выбора шрифта и его параметров. Используя кнопку **Справка**, добавляющую знак вопроса к указателю мыши, познакомьтесь с элементами окна. Списки **Шрифт** и **Размер** имеют полосы прокрутки, ускоряющие поиск нужного параметра. При выборе параметра он записывается в текстовом поле над своим списком.

В окне отображены результаты выбора шрифта Times New Roman типа антиква (с засечками), обычного начертания и размером 12 пунктов (один пункт равен 0,376 мм). При выборе шрифта необходимо проверять, имеется ли в составе шрифта набор символов кириллицы.

Из шрифтов типа гротеск (рубленые) можно просмотреть следующие: Arial, Courier, Arial Unicode MS.

Набор текста старайтесь выполнять обеими руками, поделив блок клавиш набора текста примерно пополам по границе: для левой руки — клавиши <6>, <T>, <G>, <B> и для правой руки — <7>, <Y>, <H>, <N>. Клавиша <Пробел> нажимается большими пальцами обеих рук. Поиск нужной клавиши выполняйте рассеянным взглядом, одновременно просматривая 2—4 клавиши.

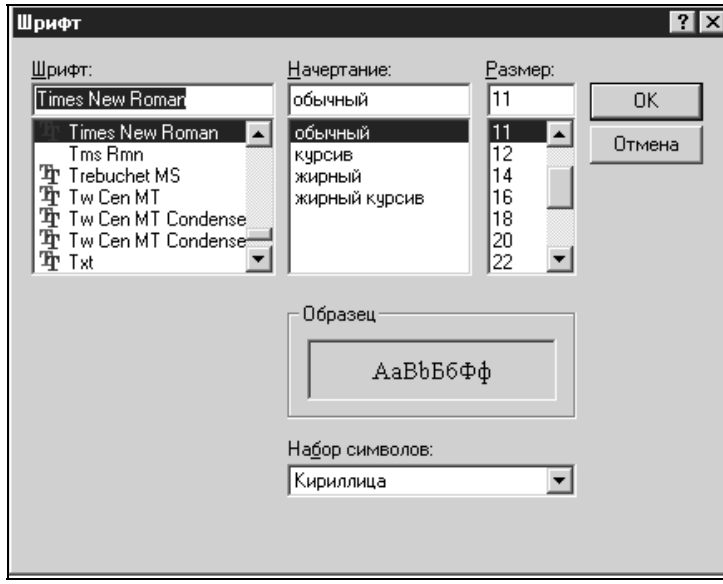


Рис. 2.5. Стандартное диалоговое окно **Шрифт**

Выполним набор следующего текста:

КОМПЬЮТЕР (англ. *computer*, от лат. *computo* — считаю) машина для приема, переработки, хранения и выдачи информации в электронном виде, которая может воспринимать и выполнять сложные последовательности вычислительных операций по заданной инструкции — программе. (Большая энциклопедия Кирилла и Мефодия, 2003).

Набранный в приложении Блокнот текст показан на рис. 2.6.

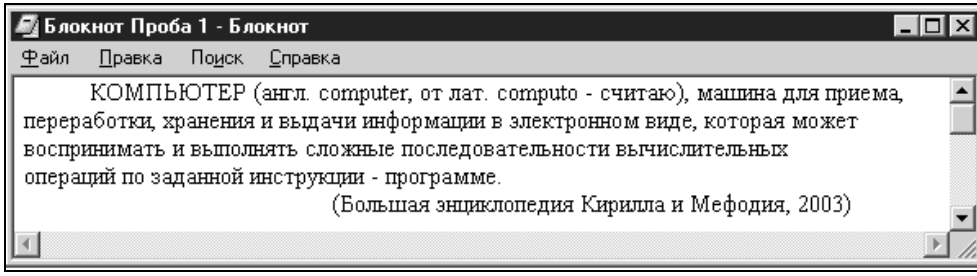


Рис. 2.6. Текстовый документ Блокнот Проба 1 в окне приложения Блокнот

Символы текста при наборе вставляются справа от текстового курсора — мерцающей вертикальной черты.

При наборе текста мало знать и быстро находить нужные символы. Имеется множество полезных команд, подаваемых компьютеру нажатием на определенные клавиши или сочетания клавиш. Рассмотрим некоторые из них.

- Для перемещения курсора по тексту можно использовать клавиши со стрелками или мышь. Клавиша <Home> перемещает курсор в начало строки, <End> — в конец строки, <Page Up> — на страницу вверх, <Page Down> — на страницу вниз.
- Переход от строчных букв к прописным и обратно: постоянно — клавиша <Caps Lock>, временно — ввод буквы при нажатой клавише <Shift>.
- Команды перехода от латинского алфавита к русскому зависят от настройки клавиатуры — <Ctrl>+<Shift> или <Alt>+<Shift>. Для проверки или установки настройки нужно подать команду **Пуск | Настройка | Панель управления | Клавиатура**. В раскрывшемся окне **Свойства: Клавиатура** выбрать вкладку **Язык**, на которой в блоке переключателей **Переключение раскладок** выбрать желаемый вариант перехода.
- Команда стирания неправильно введенного символа: слева от курсора — клавиша со стрелкой (<BackSpase>) под клавишей <F12>, справа от курсора — клавиша <Delete>.
- Многие действия, производимые с текстом, начинаются с выделения его фрагмента. Выделить символ или несколько символов можно, проведя по ним мышью с нажатой левой кнопкой, или клавишами перемещения курсора при нажатой клавише <Shift>. Слово выделяет двойной щелчок на нем мышью. Выделенный фрагмент можно вырезать, копировать, вставить, удалить, т. е. сделать все, что предлагает контекстное меню, раскрываемое щелчком правой кнопки мыши или клавишей контекстного меню клавиатуры.

При вводе текста в редакторе Блокнот отсутствует автоматический переход на следующую строку. Этот переход выполняется клавишей <Enter>.

После ввода текста следует подать команду **Файл | Сохранить**. После этого можно закрыть Блокнот кнопкой с крестом в правой части заголовка.

Если попытаться закрыть приложение без предварительного сохранения, то на экран выводится специальное диалоговое окно (рис. 2.7). Восклицательный знак в желтом треугольнике предупреждает о возможных серьезных последствиях при невнимательном отношении к сообщению и вопросу в окне.

Щелчком на кнопке **Да** сохраним введенный в Блокнот текст и закроем приложение.

Желательно проверить, все ли было сделано правильно. Вновь запустим приложение. Командой **Файл | Открыть** выведем на экран диалоговое окно **Открыть**. В папке Мои документы найдем папку Первые пробы. Двойным

щелчком на имени папки откроем ее. Двойным щелчком на имени файла Блокнот Проба 1 откроем созданный текстовый документ.

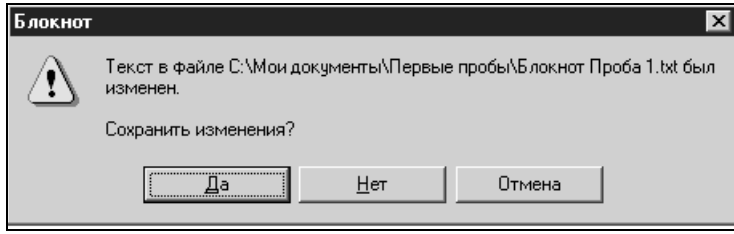


Рис. 2.7. Диалоговое окно, появляющееся перед закрытием приложения

2.3. Проводник

Проводником называют программу операционной системы Windows, позволяющую выполнять с папками и файлами все возможные операции по открытию файлов, их копированию, переименованию, перемещению, удалению и многое другое.

Вызвать Проводник можно многими способами. Простейший из них — это щелкнуть на кнопке **Пуск** правой кнопкой мыши. Проводник можно найти среди стандартных программ, список которых открывается командой **Пуск | Программы | Стандартные**. Часто Проводник можно запустить щелчком мыши на выведенном на рабочий стол значке Проводника — лупа на фоне папки.

На рис. 2.1 было показано окно Проводника с деревом папок каталога. На рис. 2.8 показано окно с каталогом папки Мои документы, раскрытой папкой Первые пробы и файлом Блокнот Проба 1.

Каталог в окне Проводника на рис. 2.8 раскрыт только до второго уровня. Знак плюс перед значком папки указывает на то, что она содержит вложенные папки. Щелчок на знаке плюс или двойной щелчок на папке раскрывает ее структуру. Щелчок на минусе — прячет вложенные папки.

Одна папка раскрыта. Это Первые пробы. Перед ней нет знаков плюс и минус, что указывает на отсутствие вложенных в нее папок. Содержимое открытых папок показывается в правом окне Проводника. Имя последней в дереве каталога открытой папки выносится в строку заголовка окна.

Работая с компьютером, познакомьтесь с элементами окна Проводника, с его меню. Много информации можно получить из справочной системы.

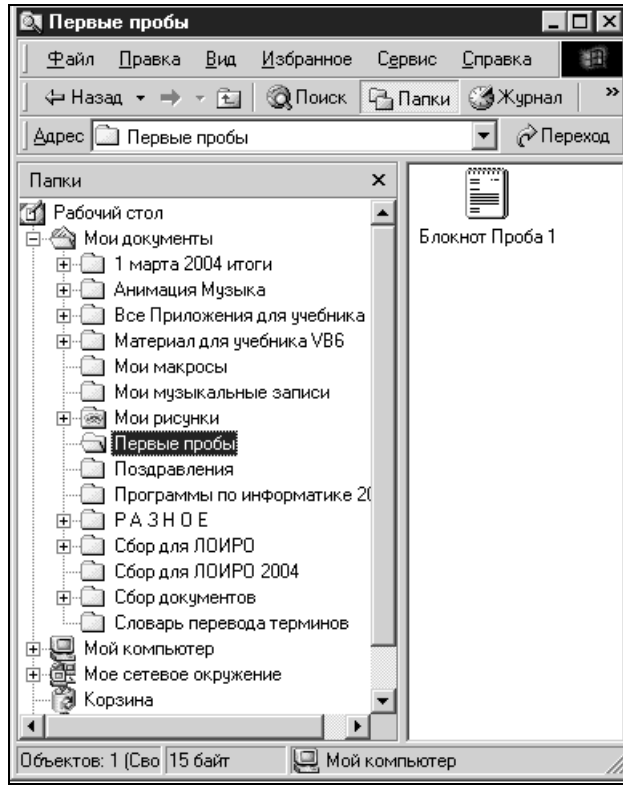


Рис. 2.8. Окно Проводника
с раскрытой папкой Первые пробы

2.4. Графический редактор Paint

Графический редактор Paint используется для работы с точечными (растровыми) рисунками формата JPG, GIF или BMP. Это один из наиболее простых графических редакторов, что позволяет быстро освоить работу с ним. С его помощью можно создавать и редактировать достаточно сложные черно-белые и цветные рисунки.

Окно графического редактора Paint и справочная система

Запуск редактора Paint можно выполнить командой **Пуск | Программы | Стандартные | Paint**. Окно редактора показано на рис. 2.9.

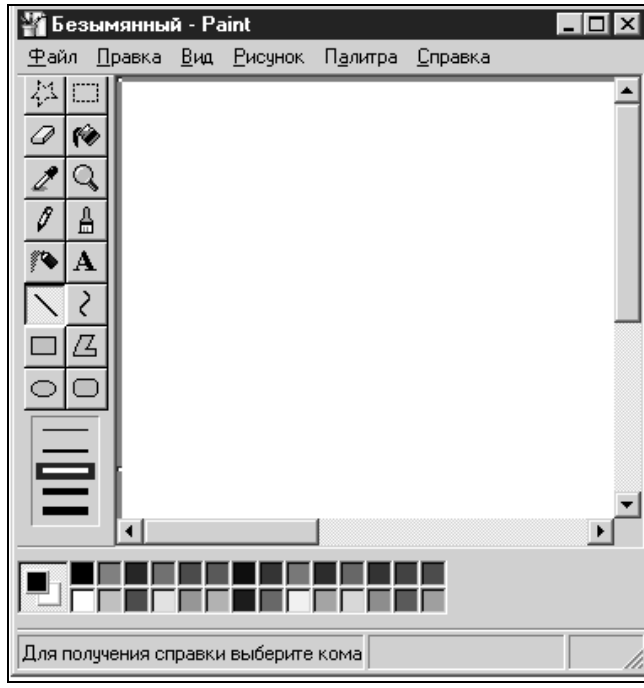


Рис. 2.9. Окно графического редактора Paint

Окно редактора Paint имеет стандартные элементы окон Windows-приложений. Это заголовок с кнопками **Свернуть**, **Развернуть/Восстановить** и **Закрыть**. Ниже заголовка — главное меню редактора. Почти всю левую часть окна занимает панель инструментов и расположенное под ней диалоговое окно настройки инструментов. В нижней части окна редактора размещена палитра выбора цвета рисования.

Меню редактора Paint

Последовательно раскрывая подменю главного меню, познакомимся с их содержанием.

- Меню **Файл**, кроме известных по работе с файлами команд (**Создать**, **Сохранить**, **Сохранить как** и т. п.), содержит и команды подготовки результатов работы к печати и их печати. В нижней части меню помещен список имен последних рисунков, с которыми работал редактор. Щелчок мышью на имени рисунка выводит его на рабочее поле редактора.
- Меню **Правка** начинается очень полезной для новичков командой **Отменить**, которая очищает рабочее поле от результатов последних действий

пользователя (до трех действий). Следующая за ней команда **Повторить** может вернуть результаты лишних отмен (также до трех отмен).

- Ниже расположена группа команд для работы с рисунком и его фрагментами. Выделенный рисунок или его фрагмент (о выделении будет рассказано ниже) можно **Вырезать** или **Копировать** в буфер обмена, **Вставить** из буфера на рабочее поле, **Очистить выделение**. Можно также **Выделить все**.
- Меню **Вид** содержит команды, определяющие вывод в окна редактора **Панели инструментов**, **Палитры**, **Строки состояния** и окна с параметрами шрифта (**Панель атрибутов текста**). Здесь же помещены команды выбора масштаба изображения рисунка и его просмотра.
- Меню **Рисунок** начинается с группы команд, изменяющих положение, размеры и внешний вид рисунка: **Отразить/повернуть**, **Растянуть/наклонить** и **Обратить цвета**. Здесь же имеется полезная команда **Очистить**, которую удобно применять, например, после вырезки нужного фрагмента рисунка перед его вставкой в рабочее поле.
- Меню **Палитра** не содержит вложенных подменю. Команда **Палитра** выводит на экран диалоговое окно **Изменение палитры**.
- Меню **Справка**. Справочная система является одним из главных элементов редактора Paint. Она дает возможность не носить с собой учебник и получать быстрые подсказки по всем вопросам работы с редактором.

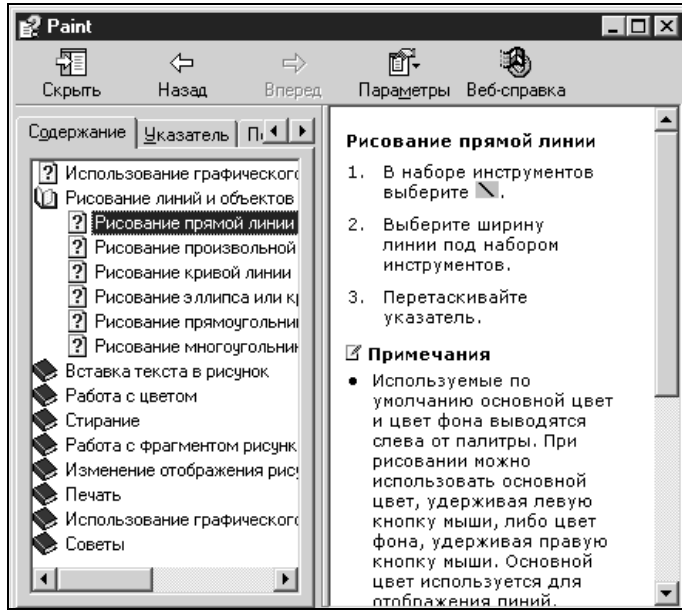


Рис. 2.10. Окно справочной системы редактора Paint

Подайте команду **Справка | Вызов справки**. На экран выводится окно справочной системы (рис. 2.10). Если окно справки развернуто, то кнопкой **Восстановить** и перемещением границ окна справки задайте размер окна примерно такой же, как на рисунке. Это позволит видеть одновременно и окно справки, и элементы окна редактора. Заметим, что структура материала в справочной системе может несколько различаться в зависимости от версии операционной системы.

Уже первое знакомство с содержанием справки убеждает в том, что в ней есть краткие и понятные ответы на все вопросы по работе с рисунками. Поэтому знание справочной системы и умение уверенно с ней работать позволяют быстро освоить все тонкости работы с графическим редактором.

Знакомство с работой редактора Paint

Рисование линий и объектов

Этот раздел справки (на рис. 2.10 он показан в виде раскрытой книги) включает в себя несколько статей, содержание которых выводится в правое окно справки.

Начните со статьи **Рисование прямой линии**. Прочитайте содержание статьи и сверните окно справки кнопкой **Свернуть** в правой части заголовка окна справки Paint. Этим вы отправите окно справки на панель задач. Так будет проще вернуть справку в том же виде на экран.

Очистив рабочее поле, поупражняйтесь в рисовании линий и объектов. Если будут встречаться затруднения, обращайтесь за помощью к справочной системе. Для этого будет достаточно щелкнуть мышью на кнопке справки **Paint** на панели задач (если вы туда ее отправили).

Вставка текста в рисунок

Это следующий раздел справки. После выделения места под текст в меню **Вид** становится доступной команда **Панель атрибутов текста**. С ее помощью можно выбрать размер и внешний вид текста.

Работа с цветом

Это один из интересных разделов справки. Цвет оживляет рисунок, делает его более красивым. Но излишнее увлечение цветом может и испортить рисунок. При работе с цветом следует соблюдать меру.

Раздел **Работа с цветом** содержит девять статей, охватывающих все действия, так или иначе связанные с цветом. Из них четыре первых статьи нужно прочитать обязательно, закрепляя прочитанное действиями с палитрой, инструментами и рисунками. С остальными статьями этого раздела можно будет познакомиться позже.

Стирание

Стирание — это действие, без которого трудно обойтись при создании рисунков с использованием инструментов редактора. Раздел содержит статьи, в которых рассматриваются все возможные варианты стирания — от стирания инструментом **Ластик** до стирания выделенных областей и всего рисунка.

Работа с фрагментом рисунка

Этот раздел содержит статьи по выделению, копированию и вставке фрагмента рисунка, его сохранению в отдельном файле.

Изменение отображения рисунка на экране

Статьи этого раздела знакомят с командами меню **Вид** и **Рисунок**, которые позволяют изменять размеры рисунка, растягивать рисунок по горизонтали и/или вертикали, поворачивать его на заданный угол, изменять масштаб изображения и т. д. К этим командам следует вернуться после освоения элементарных действий с инструментами и командами редактора.

Печать

Этот раздел — один из самых коротких — знакомит с командой предварительного просмотра рисунка перед печатью и командой его печати.

Использование графического редактора Paint с другими программами

В этом разделе справочная система знакомит с возможностями вставки в рисунок изображения из файла с помощью команды **Правка | Вставить из файла**.

Советы

Этот последний раздел справочной системы не требует пояснений.

Примеры работы с редактором

Инструменты для черчения

Это **Линия**, **Кривая**, **Прямоугольник**, **Эллипс**, **Скругленный прямоугольник** и **Многоугольник** (рис. 2.11).

Для изображения фигуры нужно нажать левую кнопку мыши, курсор которой установлен на соответствующем значке на панели инструментов, и, удерживая ее нажатой, протащить указатель мыши по рабочему полю.

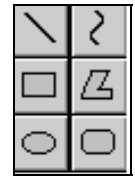


Рис. 2.11. Значки инструментов **Линия**, **Кривая**, **Прямоугольник**, **Эллипс**, **Скругленный прямоугольник** и **Многоугольник**

При нажатой клавише <Shift>:

- линия рисуется строго горизонтально, вертикально или под углом 45 градусов;
- прямоугольники рисуются в виде квадратов;
- эллипс рисуется в виде круга.

Кривая

Для уверенной работы инструментом **Кривая** нужна тренировка. Рисование каждой отдельной кривой выполняется за три такта, в три приема. Рассмотрим примеры.

Нажмите кнопку **Кривая**. Проведите горизонтальную прямую линию (первый такт). Захватите мышью за середину линии и при нажатой левой кнопке мыши перетащите ее вверх (второй такт). Не смещая мышь, сделайте еще один щелчок (третий такт). Получилась кривая 1, показанная на рис. 2.12.

Перетаскивать линию мышью не обязательно. Достаточно щелкнуть мышью в точке, в которую желательно переместить линию. Проведите горизонтальную прямую линию (первый такт). Щелкните мышью над левым концом отрезка (второй такт) и под его правым концом (третий такт). Получилась кривая 2, показанная на рис. 2.12.

Для получения кривой линии не обязательно начинать с рисования отрезка прямой. Представьте себе равнобедренный треугольник. Щелкните в месте его предполагаемой вершины (прямая, вырожденная в точку, — первый такт). Второй щелчок сделайте в месте предполагаемого левого края основания (второй такт), третий щелчок — в месте правого края основания (третий такт). Получили крупную каплю дождя (рис. 2.12, вариант 3). Повторите рисунки, показанные на рис. 2.12.

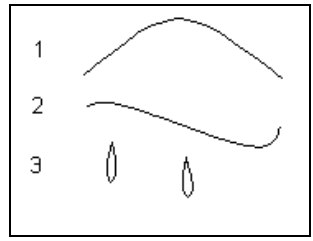


Рис. 2.12. Примеры кривых линий

Многоугольник

Многоугольник рисуется в следующей последовательности. Сначала рисуется первая сторона многоугольника, как это можно сделать инструментом **Линия**. Затем делаете щелчки мышью в тех точках, где предполагаются вершины вашего многоугольника. При этом точки автоматически соединяются линиями. В последней точке делаете двойной щелчок.

Палитра цветов

Палитра служит для выбора цвета рисования (рис. 2.13). В левой части палитры один над другим помещены два квадрата с краской. Цвет левого квадрата (первый цвет) выбирается в палитре левой кнопкой мыши, цвет правого квадрата (второй цвет) — правой кнопкой. С нажатой левой кнопкой мышь рисует первым цветом, с нажатой правой кнопкой — вторым цветом.

Рис. 2.13. Палитра

При желании изменить палитру нужно соответствующей командой меню **Палитра** вызвать диалоговое окно **Изменение палитры**, в котором нажать кнопку **Определить цвет**. О действиях по изменению палитры можно прочитать в разделе справочной системы **Работа с цветом**, в статье **Создание нестандартных цветов**.

Инструменты для рисования и записи текстов

К ним относятся инструменты: **Ластик**, **Заливка**, **Выбор цветов** (**Пипетка**), **Масштаб** (**Лупа**), **Карандаш**, **Кисть**, **Распылитель** и **Надпись** (рис. 2.14).

Рис. 2.14. Значки инструментов
Ластик, **Заливка**, **Выбор цветов** (**Пипетка**), **Масштаб** (**Лупа**),
Карандаш, **Кисть**, **Распылитель** и **Надпись**

Карандаш

Инструмент **Карандаш** рисует тонкие линии (толщиной в один пиксел). Для получения строго горизонтальных и вертикальных линий можно чертить при нажатой клавише <Shift>. При рисовании карандашом курсор мыши имеет форму карандаша.

Распылитель

Инструмент **Распылитель** (**Аэрограф**) закрашивает рисунок, набрасывая на него мелкие капли (брызги) краски. Плотность закраски зависит от выбранного размера закрашиваемого пятна и от времени распыления. Размер пятна выбирается в окне под панелью инструментов. При распылении указатель мыши имеет форму баллончика. На рис. 2.15 показан рисунок, созданный с помощью инструментов **Карандаш** и **Распылитель**.

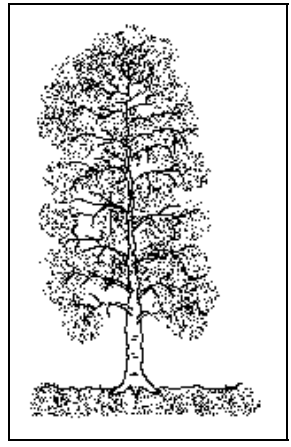


Рис. 2.15. Результат работы инструментов
Карандаш и **Распылитель**

Кисть

Инструмент **Кисть**, в отличие от инструмента **Карандаш**, позволяет рисовать широкими линиями (мазками) разного размера. Вид и размер кисти выбирается в окне свойств под панелью инструментов. Напомним, что левой кнопкой мыши кисть рисует первым цветом, а правой кнопкой — вторым цветом.

Заливка

Инструмент **Заливка** используется для закрашивания фона, на котором будет создаваться рисунок, и закрашивания замкнутых областей (деталей) рисунка. Замкнутость границы области закрашивания обязательна. Иначе краска будет "разливаться" по всему экрану. Если такое произойдет — подавайте команду **Правка | Отменить** и выясняйте причину.

Ластик

Инструмент **Ластик** служит для стирания рисунка. При этом учитывается, что рисунок может быть нарисован на каком-нибудь фоне. Поэтому перед стиранием нужно проверить, совпадает ли цвет фона со вторым цветом. Если совпадает, то левой кнопкой мыши будет выполняться стирание. Если не совпадает, то стирание левой кнопкой превращается в рисование вторым цветом.

При желании не стереть, а только заменить цвет рисунка можно воспользоваться правой кнопкой мыши. С нажатой правой кнопкой вторым цветом закрашиваются только детали рисунка, нарисованные первым цветом. При этом все детали, нарисованные другими цветами, и фон рисунка сохраняют свой цвет.

Размер ластика можно выбрать в окне свойств под панелью инструментов. При выборе ластика указатель мыши заменяется на залитый вторым цветом квадратик выбранного размера. Все действия по стиранию можно отменить командой **Правка | Отменить**.

Пипетка

Этот инструмент необходим, если имеются сомнения при выборе первого или второго цвета. Возможен случай, когда такого цвета, которым нарисован вызванный из памяти рисунок, нет в палитре. С помощью пипетки этот цвет можно присвоить первому цвету (левой кнопкой) или второму цвету (правой кнопкой).

Надпись

Инструмент **Надпись** служит для помещения на рисунок текста. Для этого нужно растянуть рамку с мигающим текстовым курсором и записать в нее текст. На экран выводится панель с параметрами текста (рис. 2.16).

Рис. 2.16. Панель атрибутов текста

Перед записью или после записи текста для него можно задать цвет, размер, начертание шрифта. Окраску шрифта определяет первый цвет. Записанный текст можно редактировать: удалять и вставлять символы, выделять, копировать отдельные его части или весь текст. Размеры рамки можно изменять. Рамку с текстом можно переместить мышью в нужное место рисунка. Все действия с текстом можно выполнять до его утверждения щелчком мыши вне рамки.

При выборе инструмента **Надпись** ниже панели инструментов появляются две кнопки с рисунками. При нажатии верхней кнопки текст вводится на фоне второго цвета. При выборе нижней кнопки текст вводится на фоне белого цвета.

Выделение

Выделение прямоугольных областей и областей произвольной конфигурации служит одним из часто применяемых приемов работы с рисунком. С выделения начинаются операции вырезки, копирования, удаления фрагментов рисунка, их перемещения, изменения цвета. Два инструмента выделения показаны на рис. 2.17.

Рис. 2.17. Значки инструментов выделения

Щелчок правой кнопкой на выделенном пунктирной рамкой фрагменте рисунка выводит на экран контекстное меню, в котором имеется большой набор операций, применимых к выделению. При отсутствии выделения контекстное меню не выводится.

Выделение прямоугольной области выполняется растягиванием пунктирной рамки. Выделение области произвольной конфигурации выполняется путем обвода мышью желаемого контура. По завершении обводки редактор ограничивает этот контур прямоугольной рамкой. Но выполненное выделение остается и его, например, можно переместить в нужное место или удалить.

Изменение размеров и формы рисунка

Команда **Отразить/Повернуть** меню **Рисунок** выводит на экран диалоговое окно **Отражение и поворот** (рис. 2.18).

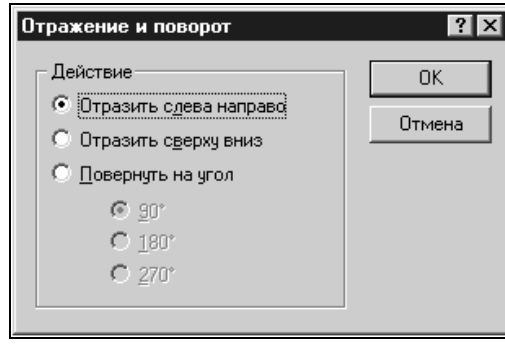


Рис. 2.18. Окно Отражение и поворот

Если выделить рисунок и выбрать в диалоговом окне **Отражение и поворот** строку **Отразить слева направо**, то в месте выделения окажется его зеркальное отображение: левые и правые детали рисунка поменяются местами. При выборе строки **Отразить сверху вниз** поменяются местами верхние и нижние детали рисунка. Команда **Повернуть на угол** поворачивает выделенный рисунок на заданный угол.

Команда **Отразить/Повернуть** полезна при рисовании объектов, имеющих симметричное расположение деталей. Например, при рисовании бабочек с их удивительно красочным рисунком крыльев без команды **Отразить** обойтись трудно.

Вторая команда меню **Рисунок — Растянуть/Наклонить —** выводит на экран диалоговое окно **Растяжение и наклон** (рис. 2.19).

Изменять размеры выделенных фрагментов рисунка можно, перемещая границу прямоугольника выделения. Более точно это делается командами окна **Растяжение и наклон**. Заметим, что изменение размеров сказывается на качестве изображения и не всегда позволяет вернуть исходное качество рисунка.

Некоторые стандартные фигуры (эллипсы, прямоугольники) и другие элементы рисунков трудно четко и правильно изобразить инструментами **Карандаш** и **Кисть** с наклоном в пространстве на заданные углы по горизонтали и вертикали. Команда **Наклонить** позволяет выполнить такие рисунки с наклоном на заданное число градусов.

Желание увидеть свой рисунок в необычном виде дает возможность команда **Обратить цвета** меню **Рисунок**. Команда меняет все цвета на обратные: красный на голубой, зеленый на сиреневый, черный на белый и т. д.

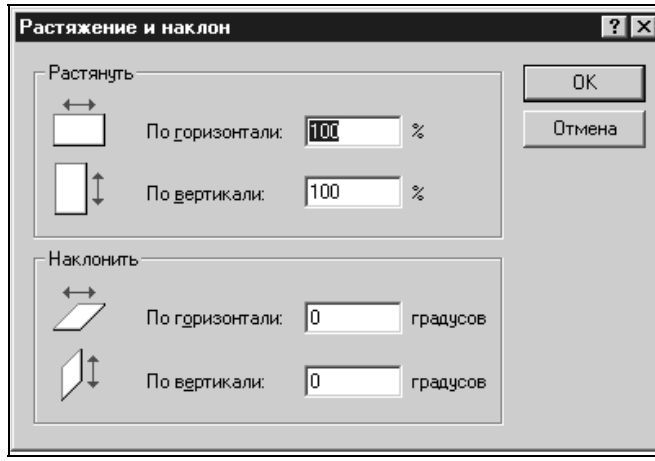


Рис. 2.19. Диалоговое окно **Растяжение и наклон**

Если возникает желание точнее воспроизводить мелкие детали рисунка, удобно пользоваться масштабом. В этом случае выбирают масштаб командой **Вид | Масштаб | Крупный** (400) или командой **Вид | Масштаб | Другой** открывают диалоговое окно **Масштаб** (рис. 2.20), в котором выбирают масштаб 600 или даже 800. Команда **Вид | Масштаб | Показать сетку** выводит на рабочее поле сетку, каждая ячейка которой — пиксел. При этом рисование мелких деталей фрагмента рисунка сводится к заполнению ячеек сетки инструментами **Карандаш** или **Кисть**.

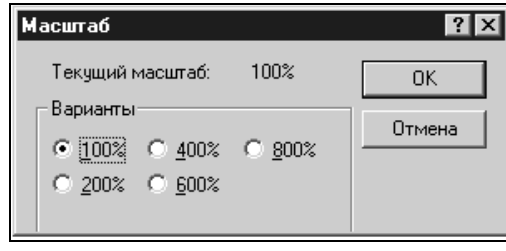


Рис. 2.20. Диалоговое окно **Масштаб**

Сохранение и печать рисунков

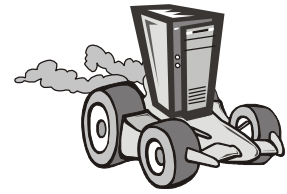
Обычно работа над рисунком завершается его сохранением. Перед сохранением рисунок можно просмотреть, развернув его во весь экран командой

Вид | Просмотреть рисунок. Возврат к стандартному виду окна редактора выполняется щелчком в любом месте рисунка.

Для первого сохранения рисунка подается команда **Файл | Сохранить как**. Редактор, как правило, предлагает сохранить рисунок в папке Мои документы/Мои рисунки. В этой папке желательно создать папку, например, Мои первые рисунки, и записывать рисунки, присваивая им имена, связанные с их содержанием. При последующих сохранениях рисунка, например, после его просмотра или доработки достаточно подать команду **Файл | Сохранить**.

Перед печатью рисунок просматривают. Для этого подают команду **Файл | Предварительный просмотр**. Рисунок отображается на листе формата А4 так, как он будет отпечатан на принтере.

Глава 3



Информация. Информатика

3.1. Общие сведения об информации и информатике

Информация

Информацией называют сведения о предметах, процессах и явлениях окружающего нас мира.

Информацию человек воспринимает с помощью органов чувств. Зрение, слух, обоняние, вкус и осязание позволяют нам, например, получить информацию о размере, форме, цвете, твердости, аромате и вкусе яблока.

Человек не только воспринимает информацию, но и сам является ее источником, призводя ее с помощью речи, жестов, письма, графики. Сам образ человека несет информацию, по которой его узнают, отличают от других людей.

Источниками информации для человека служат окружающие его предметы и явления. Много информации он получает по телевидению, радио, из печати, читая книги, просматривая фильмы, в разговорах с другими людьми.

Со второй половины XX века наблюдается "информационный взрыв". Развитие науки и техники, литературы и искусства привели к значительному увеличению количества информации. Одновременно существенно возросли темп жизни и цена времени. Человек не в силах одновременно слушать и смотреть сотни и тысячи теле- и радиопередач, читать сотни и тысячи издаваемых ежедневно книг, газет и журналов. Остро встала проблема сортировки и выбора информации, разработки средств ее передачи и переработки. На помощь пришли новейшие средства телекоммуникаций, важными элементами которых являются компьютеры.

В соответствии с увеличившимся значением информации и средств телекоммуникаций, а также углубленным пониманием роли информации и информационных процессов не только в жизни человечества, но и во вселенском масштабе, меняется взгляд на научную картину мира.

Известны два поля дальнего взаимодействия — электромагнитное, осуществляющее взаимодействие между электрически заряженными частицами, и гравитационное (поле тяготения), осуществляющее взаимодействие между массами. В настоящее время ученые к этим полям добавляют *информационное поле*. Современная научная картина мира строится на трех основополагающих понятиях: *вещества*, *энергии* и *информации*.

Информацию выдают, воспринимают, используют, перерабатывают все представители животного и растительного мира, все автоматически работающие механизмы и устройства.

Важно отметить, что информационные процессы в живой природе, обществе, в различных механизмах и устройствах имеют много общего и подчиняются одним и тем же законам. Это позволяет рассматривать и изучать их с единых позиций, строить единую для всех них теорию информации.

Информатика

Информатикой называют отрасль науки, изучающую структуру и общие свойства информации, а также вопросы, связанные с ее сбором, хранением, поиском, преобразованием, распространением и использованием в различных сферах деятельности человека.

Информационными технологиями называют раздел информатики, занимающийся разработкой, внедрением и использованием технических и программных средств, с помощью которых наиболее эффективно выполняются разнообразные операции по обработке информации во всех сферах человеческой деятельности.

Среди областей применения информационных технологий следует выделить экономику и финансы, науку и производство, просвещение и культуру, социальную сферу и оборону, развлечения и средства связи.

Информационная система

Информация не может существовать без выдающих ее объектов, передаваться без соответствующих объектов, способных ее переносить, и восприниматься без объектов, способных ее принимать, понимать и использовать. Простейшая *информационная система* состоит из источника информации, канала связи и приемника информации (рис. 3.1).

Примерами информационных систем могут служить: книга и читатель, светофор и водитель (каналы связи зрительные), передающая радиостанция и радиоприемник (канал связи — радиоволны), радиоприемник и слушатель (канал связи звуковой) и т. п.

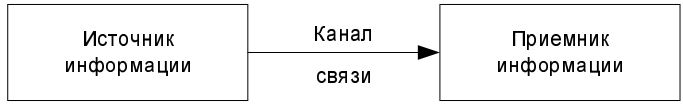


Рис. 3.1. Простейшая информационная система

3.2. Свойства информации

К свойствам информации можно отнести ее важность (ценность, полезность), достоверность (истинность, правильность), полноту, оперативность (своевременность), понятность (доступность) и некоторые другие ее качества.

Важность

Полученная в разговоре по телефону информация может быть новой и важной для собеседников или сообщающей известные им сведения. Информация о возможных заморозках имеет различную важность для огородника, шофера и для водителя трамвая. В этом проявляется субъективность в оценке информации.

Можно сказать, что цена информации определяется связанными с ней материальными или духовными приобретениями или потерями. Более ценной информации соответствует и больший выигрыш или меньший проигрыш.

Достоверность

Это свойство связано с истинностью и ложностью информации. Известно, что истина объективна по содержанию (не зависит от человека), но субъективна по форме, так как является результатом деятельности человека.

Истина относительна. Она отражает предмет не полностью, ввиду ограниченности человеческих знаний. В процессе познания человеком она стремится к истине абсолютной, полностью исчерпывающей предмет познания.

Например, великий Пифагор считал, что молнии мечет на Землю разгневанный Зевс. Теперь же каждый школьник знаком с электричеством. Долгое время за истину принималась геоцентрическая система Птолемея. В настоящее время истинность, достоверность информации о гелеоцентричности Солнечной системы бесспорна.

Полнота

Полнота информации о предмете, процессе, явлении зависит от ее количества, подробности, всесторонности. Понятие полноты информации о предмете так же субъективно и относительно, как и понятие истины. Информа-

цию даже о простейшем предмете невозможно исчерпать полностью. Всегда можно что-то добавить, уточнить.

Оперативность

Оперативность — это своевременность информации. С этой характеристикой информации мы встречаемся практически ежедневно. Все средства массовой информации стремятся оперативно получать информацию и передавать ее зрителям, слушателям, читателям. Устаревшая информация нас, как правило, мало интересует.

Доступность

Это свойство информации связано с возможностью ее воспринимать, понимать и использовать. Сообщение, передаваемое вам, например, азбукой Морзе, если вы ее не знаете, не несет никакой информации.

Можно заметить, что все перечисленные характеристики информации взаимосвязаны, взаимозависимы. Например, ценность информации определяется ее достоверностью, полнотой и оперативностью.

3.3. Кодирование и единицы информации

Языки и алфавиты

Различают естественные и искусственные (формальные) языки. *Естественные языки* развивались веками и служат для общения людей между собой. *Формальные языки* разрабатываются для специальных применений. Примером формальных языков могут служить языки программирования, языки кодирования информации для ее передачи, хранения и т. п.

Каждый язык имеет свой *алфавит*. Под алфавитом языка понимают набор используемых символов. Под *мощностью алфавита* понимают количество составляющих алфавит символов.

Кодом называют совокупность знаков (символов), предназначенных для представления информации в соответствии с определенными правилами. Такое представление называют *кодированием*. Кодируют информацию с целью ее передачи, хранения, преобразования.

Одно и то же понятие на различных языках может кодироваться различными способами. Например, слово "стол" — это код в русском алфавите всем известного предмета мебели. В других языках, в других алфавитах этот предмет кодируется иначе.

Звук *а*, издаваемый человеком, кодируется в некоторых языках буквой А. Буква А в азбуке Морзе кодируется так: "• —" (точка, тире). В компьютере

буква A латинского алфавита в привычной для нас десятичной системе кодируется числом 65. В свою очередь, число 65 в "привычной" для компьютера двоичной системе (цифры только 0 и 1) кодируется так: 01000001.

В естественных языках традиционно сложились некоторые неоднозначности. Например, под словом "коса" понимается и девичья коса, и речная отмель, и инструмент для скашивания травы. Подобные неоднозначности в формальных языках, как правило, недопустимы.

Различны и алфавиты языков кодирования. Это буквы A, B, C, D,... и буквы А, Б, В, Г, Д,... в латинском и русском алфавитах; точка и тире в азбуке Морзе, арабские цифры 0, 1, 2, 3, ... , 9, с помощью которых записываются числа, кодирующие различные количества; красный, желтый и зеленый цвет в светофоре; цифры 0 и 1 в компьютере и т. п.

Количество и графическое изображение символов в алфавитах естественных языков определяются характерными особенностями языка, историей его развития, традициями. Например, русский алфавит имеет 33 буквы, латинский — 26, итальянский — 21, армянский — 39, арабский — 28, китайский — несколько тысяч иероглифов. Количество и изображение знаков в формальных языках могут также существенно различаться.

Наименьший по числу знаков алфавит имеет только один знак. Пусть этот знак **1** (единица). Тогда три цвета светофора можно закодировать, например, так: красный — 1, желтый — 11, зеленый — 111. Такой алфавит самый неэкономичный по записи кодов. В этом легко убедиться, если попытаться записать в этом алфавите, например, число десять: 1111111111.

Двоичный алфавит

В информатике и вычислительной технике широко используется алфавит, имеющий два знака, например, 1 и 0. Этим знакам в логике и технике приводят в соответствие понятия — да и нет, истина и ложь, включено и выключено. Такой алфавит называют *двоичным* или *бинарным*. В соответствии с этим введена и наименьшая единица информации — *бит* (англ. *bit*, от *binary* — двоичный и *digit* — знак).

Одного бита информации достаточно, чтобы передать слово "да" или "нет", закодировать, например, состояние электролампочки. Кстати, на некоторых выключателях пишут "1 — включено" и "0 — выключено". Взгляд на выключатель снимает для нас неопределенность в его состоянии. При этом мы получаем количество информации, равное одному биту.

Пример 1

Пусть требуется отгадать задуманное число из набора натуральных чисел от 1 до 16, задавая вопрос: "В первой половине находится задуманное число?" и получая в ответ "Да" или "Нет".

Задав первый вопрос и услышав первый ответ, получаем один бит информации. При этом неопределенность уменьшается в два раза (из 16 чисел остается только 8). После второго вопроса получаем еще один бит информации, уменьшающий неопределенность еще в два раза (из 8 чисел осталось только 4). После получения еще одного бита информации неопределенность снова уменьшилась вдвое (из 4 чисел осталось 2 числа). Последний, третий вопрос позволяет отгадать число.

Для отгадывания потребовалось получить четыре ответа, четыре бита информации. Столько же бит требуется для оптимального кодирования каждого из шестнадцати чисел.

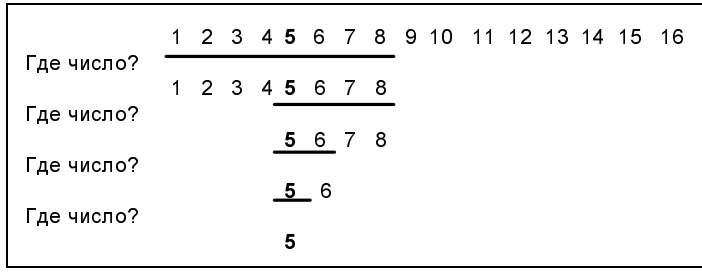


Рис. 3.2. Пример процедуры отгадывания числа

На рис. 3.2 показана последовательность шагов отгадывания числа 5. На вопросы "Где число?" дается ответ в виде жирной линии. При решении задачи использован широко применяемый, быстрый прием поиска путем деления всего множества чисел пополам.

Двоичное слово. Байт

Если нужно закодировать в двоичном алфавите красный, желтый и зеленый цвет светофора, то требуется уже два бита. Закодировать три цвета можно, например, так: 00, 01 и 10. Сообщение о том, что включен, например, красный цвет светофора, содержит информации больше одного бита.

Для кодирования четырех сторон света (север, восток, юг и запад) требуется также два бита: 00, 01, 10, 11. Поэтому сообщение о том, какая выбрана сторона света, содержит ровно два бита информации.

При кодировании восьми углов куба потребуется три бита: 000, 001, 010, 011, 100, 101, 110, 111. При кодировании от 9 до 16 объектов потребуется уже четыре бита, от 17 до 31 — 5 бит, от 32 до 63 — 6 бит, от 64 до 127 — 7 бит.

Последовательность символов называют *словом*. Можно сделать вывод: чем больше требуется закодировать объектов, тем длиннее требуемое двоичное слово. Восьмибитовое двоичное слово называется *байтом*. С помощью байта можно закодировать 256 различных объектов.

До недавнего времени байта было достаточно, чтобы закодировать все символы текста в русском и латинском алфавите: буквы, цифры, знаки препинания, управляющие сигналы — все то, что передавалось компьютеру с клавиатуры.

С развитием информатики байт начал сдерживать возможность увеличения количества используемых символов. В настоящее время завершается переход на двухбайтовое кодирование символов. 16-битовое двоичное слово позволяет закодировать 65 536 символов и команд.

Единицы информации

На практике используются более объемные, производные единицы количества (объема) информации:

1 байт = 8 бит;

1 килобайт (Кбайт) = 2^{10} байтов = 1024 байтов;

1 мегабайт (Мбайт) = 2^{10} килобайтов = 1024 килобайтов;

1 гигабайт (Гбайт)= 2^{10} мегабайтов = 1024 мегабайтов;

1 терабайт (Тбайт)= 2^{10} гигабайтов = 1024 гигабайтов,

и т. д.

Пример 2

Подсчитаем объем памяти, требуемый для записи и хранения в памяти компьютера книги "Программируем на языке Quick BASIC 4.5" Г. А. Зельднера. В книге 420 страниц. На каждой странице в среднем по 40 строк. В каждой строке по 60 символов. Итого в книге $60 \times 40 \times 420 \approx 1\ 000\ 000$ знаков. Каждый знак требует для записи 1 байт памяти. Поэтому для записи всей книги нужно около 1 Мбайт. А это всего одна гибкая дискета с объемом памяти 1,44 Мбайт.

Часто ошибочно отождествляют понятия количества и объема информации. Поясним их различие на примере. Пусть имеется большой энциклопедический словарь в 1400 страниц и такого же объема книга, на каждой странице которой все строки заполнены вопросительными знаками. Объем словаря и книги одинаков — 1400 страниц текста. Поэтому для хранения словаря и книги требуется одинаковый объем памяти. Количество же информации в словаре и книге существенно различается. Всю информацию книги можно записать одной фразой: "1400 страниц с вопросительными знаками". Но даже эта фраза имеет сомнительную ценность и для многих людей не несет никакой полезной информации.

В целях экономии памяти при записи информации на хранение (при архивации) широко используют процедуру ее сжатия. Так, информацию словаря

можно сжать примерно в десять раз. При этом следует понимать, что путем сжатия уменьшается не количество информации, а только ее объем.

3.4. Кодирование графики и звука

С помощью компьютера рисуют, сочиняют музыку; компьютер понимает человеческий голос и сам может отвечать пользователю. Так как вся информация в компьютере обрабатывается и сохраняется в цифровом виде, то и графическая, и звуковая информация должны быть представлены в цифровом виде. С этой целью ее кодируют.

Кодирование растровой графики

Весь экран дисплея делится на точки — пикселы, подсвечивание которых создает видимое отображение текста и рисунков. Различают растровую и векторную графику.

При растровом способе создания, хранения и отображения графических объектов приходится иметь дело со всеми точками, входящими в изображение. Например, если цветная фотография, полученная с помощью цифровой фотокамеры, имеет размеры 1600×1200 точек, то в памяти компьютера необходимо хранить информацию (цвет, яркость) о каждой из точек фотографии.

Информационный объем растрового изображения (Q) определяется как произведение числа входящих в изображение точек (N) на информационный объем одной точки (q), который зависит от количества возможных цветов, т. е. $Q = N \cdot q$.

При черно-белом изображении q = 1 бит (например, 1 — точка подсвечивается и 0 — точка не подсвечивается). Поэтому для хранения черно-белого (без оттенков) изображения размером 100×100 точек требуется 10 000 бит. Если между черным и белым цветом имеется еще шесть оттенков серого (всего 8), то информационный объем точки равен 3 бита ($\log_2 8 = 3$). Информационный объем такого изображения увеличивается в три раза: Q = 30 000 бит.

Цветное изображение получается за счет различной яркости трех основных цветов — красного, синего и зеленого. Цветные изображения могут отображаться в различных режимах, соответственно изменяется и информационный объем точки:

Режим	Информационный объем точки
16 цветов	$q = \log 216 = 4$ бита
256 цветов	$q = \log 2256 = 8$ бит = 1 байт
65 536 цветов	$q = \log 265536 = 16$ бит = 2 байта
16 777 216 цветов	$q = \log 216\,777\,216 = 24$ бит = 3 байта

Умножение информационного объема точки на количество точек дает значительную величину. Поэтому актуальной становится задача расчета объема памяти (видеопамяти), необходимой для хранения изображения (битовой карты) со всего экрана монитора. Рассмотрим два примера.

Пример 3

При размерах экрана (разрешении) 640×480 точек и количестве цветов 256 ($q = 1$ байт) объем видеопамяти должен быть не менее $Q = 640 \cdot 480 \cdot 1$, т. е. около 300 Кбайт.

Пример 4

Для экрана размером 1280×1024 точки и количестве цветов 16 777 216 ($q = 3$) имеем $Q = 1280 \cdot 1024 \cdot 3$, что примерно равно 3,75 Мбайта.

Современные компьютеры имеют объем видеопамяти 8, 16 и более Мбайт, что позволяет хранить несколько графических изображений размером во весь экран.

Кодирование векторной графики

Векторное изображение состоит из набора элементарных деталей — графических примитивов. Это линии, прямоугольники, окружности, дуги и т. п. Положение этих элементов на экране определяется координатами точек. Например, отрезок прямой задается координатами концов, окружность — координатами центра и радиусом и т. д. Кроме координат задаются цвет отображаемых деталей, толщина линий и другие характеристики.

Информация о векторном рисунке кодируется обычным способом, как хранятся тексты, формулы, числа, т. е. хранится не графическое изображение, а только координаты и характеристики изображения его деталей. Поэтому для хранения векторных изображений требуется существенно меньше памяти, чем для изображений растровых.

При запуске программы с векторным рисунком он создается каждый раз вновь, после чего в растровом виде может сохраняться в видеопамяти. При просмотре графических изображений они считываются компьютером из видеопамяти на экран дисплея с частотой 50—60 и более раз в секунду.

Кодирование звука

Человек слышит звук с частотами от 16 герц (колебаний в секунду) до 20 килогерц. До недавнего времени звуковые сигналы передавались, записывались на граммофонных пластинках и магнитных лентах и воспроизводились в виде амплитудных колебаний. При передаче радио- и телевизионных программ использовалась амплитудная или частотная модуляция так называе-

мой несущей частоты радиосигнала низкочастотными звуковыми сигналами. На приемных устройствах звуковые сигналы усиливались и подавались на репродукторы и динамики. В процессе передачи по каналам связи на звуковые сигналы могут накладываться различной природы внешние электромагнитные воздействия, которые искажают сигналы.

Значительное увеличение быстродействия компьютеров и других цифровых устройств, увеличение памяти компьютеров позволяют переходить на цифровую передачу и воспроизведение аудио и телевизионных передач, сохранять их в памяти компьютера, хранить на лазерных CD- и DVD-дисках. При этом существенно повышается надежность и качество воспроизведения.

Задания для самостоятельной работы

1. Придумать примеры: а) информации, имеющей различную ценность для различных людей, что подтверждает субъективность в оценках информации; б) информации, ценность которой существенно зависит от ее оперативности; в) информации, ценность которой зависит от ее полноты.
2. Определить число бит, потребное для кодирования в двоичном алфавите: а) всех пальцев на руках и ногах; б) всех номеров квартир стапятидесятиквартирного дома.
3. Сколько бит информации нужно получить, чтобы отгадать загаданное число из: а) 16 последовательно расположенных чисел натурального ряда; б) 32 чисел.
4. Выбрать любую книгу и подсчитать потребный для ее записи объем памяти.

Глава 4

Системы счисления

4.1. Общие сведения

Вся информация в компьютере кодируется числами. Кодируются данные вычислительных задач и буквы алфавитов, рисунки и музыка, кодируются управляющие сигналы и вся другая информация, с которой работает компьютер.

Кодируется информация двоичными кодами, использующими символы 1 и 0. Такой способ представления чисел называют двоичной системой счисления. *Системой счисления* называют совокупность приемов построения, записи и наименования чисел.

История развития способов счета насчитывает тысячелетия. Менялись и средства счета: пальцы, камешки, узелки, счеты, арифмометры, компьютеры. Естественно было желание ученых и инженеров проектировать вычислительные устройства, работающие в привычной для нас десятичной системе. Так и происходило, пока эти устройства были механическими.

Первые электронные вычислительные машины на реле уже строились на основе двоично-десятичной системы, в которой каждая десятичная цифра кодировалась в двоичной системе. В настоящее время компьютеры работают с информацией, представленной, как правило, в двоичной системе, имеющей перед другими системами большие преимущества.

Кроме двоичной системы при работе компьютера используются десятичная система, восьмеричная и шестнадцатеричная системы.

Десятичная система

Наиболее известна широко применяемая на практике десятичная система. Это *позиционная* система счисления. Количество, определяемое цифрой числа, зависит от позиции этой цифры в записи числа. Например, в записи числа $A_{(10)} = 333$ одна и та же цифра 3 определяет различные количества: триста, тридцать и три.

К *непозиционным* системам относят римскую систему счисления. Например, в числе XXII количество, определяемое цифрами X и I, не зависит от их положения в записи числа.

Десятичная система имеет 10 цифр (0, 1, 2, ..., 9), что и определило название системы и ее важнейшую характеристику — *основание системы*. Обозначим ее буквой ***p***. Для десятичной системы $p = 10$.

Формула разложения числа по степеням основания

Пусть в десятичной системе задано некоторое число $A_{(10)} = 3745$. Каждая позиция, занимаемая цифрами, называется *разрядом* числа. Разряды имеют названия и номера: разряд единиц, разряд десятков, разряд сотен и т. д. Названия разрядов определяют их *вес*: единицы, десятки, сотни, тысячи. Заметим, что вес разряда равен степени основания в этом разряде.

Заданное число, не изменяя его количества, можно записать следующими способами:

$A_{(10)} = 3745;$

$A_{(10)} = 3000 + 700 + 40 + 5;$

$A_{(10)} = 3 \cdot 1000 + 7 \cdot 100 + 4 \cdot 10 + 5 \cdot 1.$

Последняя запись представляет собой сумму произведений цифр числа на вес разрядов. Эту запись называют *формулой разложения числа*. При знакомстве со степенями чисел формулу разложения можно записать короче:

$A_{(10)} = 3 \cdot 10^3 + 7 \cdot 10^2 + 4 \cdot 10^1 + 5 \cdot 10^0 = 3 \cdot 10^3 + 7 \cdot 10^2 + 410 + 5.$

Запишем целое четырехразрядное десятичное число и полученную формулу его разложения в общем виде:

$A_{(10)} = a_3a_2a_1a_0 = a_3 \cdot 1000 + a_2 \cdot 100 + a_1 \cdot 10 + a_0 \cdot 1 = a_3 \cdot 10^3 + a_2 \cdot 10^2 + a_1 \cdot 10^1 + a_0 \cdot 10^0.$

В этой записи переменные a_3, a_2, a_1 и a_0 — цифры разрядов числа, а числа 3, 2, 1 и 0 — номера разрядов.

4.2. Системы счисления в компьютерах

Ввод информации в компьютер с клавиатуры и вывод из него результатов вычислений на экран дисплея и принтер производится, как правило, в привычной для нас десятичной системе. Хранение и преобразование информации современные компьютеры выполняют в двоичной системе ($p = 2$). В качестве вспомогательных используются восьмеричная ($p = 8$) и шестнадцатеричная ($p = 16$) системы.

Двоичная система счисления

Основание двоичной системы $p = 2$ определяет и число цифр: 0 и 1. Формула разложения целого четырехразрядного числа для двоичной системы следующая:

$A_{(2)} = a_3a_2a_1a_0 = a_3 \cdot 2^3 + a_2 \cdot 2^2 + a_1 \cdot 2^1 + a_0 \cdot 2^0 = a_3 \cdot 8 + a_2 \cdot 4 + a_1 \cdot 2 + a_0 \cdot 1.$

При первом знакомстве с двоичной, восьмеричной и шестнадцатеричной системами существенно помогает таблица соответствия записей числа в различных системах (табл. 4.1).

Таблица 4.1. *Соответствие записей числа в различных системах*

$p = 10$	$p = 2$	$p = 8$	$p = 16$	$p = 10$	$p = 2$	$p = 8$	$p = 16$
0	0	0	0	9	1001	11	9
1	1	1	1	10	1010	12	A
2	10	2	2	11	1011	13	B
3	11	3	3	12	1100	14	C
4	100	4	4	13	1101	15	D
5	101	5	5	14	1110	16	E
6	110	6	6	15	1111	17	F
7	111	7	7	16	10000	20	10
8	1000	10	8				

Восьмеричная и шестнадцатеричная системы счисления

Основание восьмеричной системы $p = 8$. Цифры: 0, 1, 2, 3, 4, 5, 6 и 7. Основание шестнадцатеричной системы $p = 16$. Цифры: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F.

Формулы разложения целых четырехразрядных чисел в этих системах следующие:

$A_{(8)} = a_3 \cdot 8^3 + a_2 \cdot 8^2 + a_1 \cdot 8^1 + a_0 \cdot 8^0 = a_3 \cdot 512 + a_2 \cdot 64 + a_1 \cdot 8 + a_0 \cdot 1,$

$A_{(16)} = a_3 \cdot 16^3 + a_2 \cdot 16^2 + a_1 \cdot 16^1 + a_0 \cdot a^0 = a_3 \cdot 4096 + a_2 \cdot 256 + a_1 \cdot 16 + a_0 \cdot 1.$

4.3. Перевод чисел из одной системы в другую

Перевод с использованием формулы разложения

В основе способа лежит использование значений веса разрядов чисел. Веса пяти разрядов чисел с основаниями 10, 2, 8 и 16 приведены в табл. 4.2.

Действия при переводе выполняются в новой системе, поэтому способ удобно использовать для перевода чисел в десятичную систему. В дальнейшем будем применять его при проверках правильности перевода чисел другими способами.

Для перевода двоичного числа в десятичную систему достаточно просуммировать веса разрядов, в которых стоят единицы. Например, если $A_{(2)} = 101$, то для перевода нужно получить сумму $A_{(10)} = 4 + 1 = 5$.

Таблица 4.2. *Веса разрядов чисел*

n	4	3	2	1	0
n = 10	10 000	1000	100	10	1
n = 2	16	8	4	2	1
n = 8	4096	512	64	8	1
n = 16	65 536	4096	256	16	1

Пример 1

Дано: $A_{(2)} = 1101$. Найти: $A_{(10)}$.

Решение.

Записываем формулу разложения двоичного числа:

$A_{(2)} = a_3a_2a_1a_0\ a_3 = \ a_3 \cdot 8 + a_2 \cdot 4 + a_1 \cdot 2 + a_0 \cdot 1.$

Подставляем в формулу значения разрядов заданного двоичного числа и выполняем действия:

$A_{(10)} = 1 \cdot 8 + 1 \cdot 4 + 0 \cdot 2 + 1 \cdot 1 = 13.$

Заглянув в табл. 4.1, убедимся, что получен правильный результат. Двоичному числу $1101_{(2)}$ соответствует десятичное число $13_{(10)}$.

Ответ: $A_{(10)} = 13$.

Заметим, что решение Примера 1 можно было записать проще: $A_{(10)} = 8 + 4 + 1 = 13$. Действительно, при переводе достаточно суммировать только

вес тех разрядов числа, где стоят единицы. Поэтому сколько единиц в двоичной записи числа — столько слагаемых. Будем использовать это правило при решении других примеров.

Пример 2

Дано: $A_{(2)} = 100111$. Найти: $A_{(10)}$.

Решение.

Запишем формулу разложения двоичного числа:

$A_{(2)} = a_5a_4a_3a_2a_1a_0 = a_5 \cdot 32 + a_4 \cdot 16 + a_3 \cdot 8 + a_2 \cdot 4 + a_1 \cdot 2 + a_0 \cdot 1$.

Запишем сумму степеней основания разрядов, в которых стоят единицы, и выполним действия:

$A_{(10)} = 32 + 4 + 2 + 1 = 39$.

Ответ: $A_{(10)} = 39$.

При переводе чисел в десятичную систему из восьмеричной и шестнадцатеричной систем приходится суммировать не веса разрядов, а произведения веса разрядов на цифры числа в этих разрядах.

Пример 3

Дано: $B_{(8)} = 135$. Найти: $B_{(10)}$.

Решение.

Формула разложения числа в восьмеричной системе следующая:

$B_{(8)} = b_2b_1b_0 = b_2 \cdot 64 + b_1 \cdot 8 + b_0$.

Подставляем в формулу значения разрядов заданного восьмеричного числа:

$B_{(10)} = 1 \cdot 64 + 3 \cdot 8 + 5 = 93$.

Ответ: $B_{(10)} = 93$.

Пример 4

Дано: $C_{(16)} = 2A$. Найти: $C_{(10)}$.

Решение.

Формула разложения числа в шестнадцатеричной системе:

$C_{(16)} = c_1c_0 = c_1 \cdot 16 + c_0$.

Подставляем значения разрядов заданного числа:

$C_{(10)} = 2 \cdot 16 + 10 = 42$.

Ответ: $C_{(10)} = 42$.

Перевод целых чисел делением на основание

Правило перевода чисел заключается в следующем. Сначала исходное число, а затем получающиеся частные делим на основание новой системы. Действия выполняем в старой системе. Записываем последнее частное и остатки в порядке, обратном порядку их получения. Полученное число является записью заданного числа в новой системе.

Пример 5

Дано: $A_{(10)} = 35$. Найти: $A_{(2)}$.

Решение:

```
_ 35  | 2
  34  _ 17  | 2
    1   16  _ 8  | 2
          1   8  _ 4  | 2
              0    4  _ 2  | 2
                   0    2    1
                        0
```

Ответ: $A_{(2)} = 100011$.

Пример 5 наглядно показывает последовательность действий при переводе целых чисел. Проверку правильности полученного результата удобно выполнить, используя формулу разложения числа:

$A_{(10)} = 1 \cdot 32 + 0 \cdot 16 + 0 \cdot 8 + 0 \cdot 4 + 1 \cdot 2 + 1 = 32 + 2 + 1 = 35$.

Так как действия выполняются в старой системе, рассмотренное правило удобно использовать при переводе чисел из десятичной системы. В примерах 6 и 7 выполняется перевод чисел из десятичной системы соответственно в восьмеричную и шестнадцатеричную системы. Проверку правильности полученных результатов легко выполнить обратным переводом с использованием формулы разложения.

Пример 6

Дано: $A_{(10)} = 95$. Найти: $A_{(8)}$.

Решение:

```
_ 95  | 8
  88  _11  | 8
    7   8    1
        3
```

Ответ: $A_{(8)} = 137$.

Пример 7

Дано: $A_{(10)} = 45$. Найти $A_{(16)}$.

Решение:

$$\begin{array}{r|l} 45 & 16 \\ \underline{32} & 2 \\ 13 & \end{array}$$

Ответ: $A_{(16)} = 2D$.

Поразрядные способы перевода

Перевод чисел существенно упрощается, если основания старой (p) и новой (q) систем связаны соотношением $p = q^k$ или $p^k = q$, где k — целое число. Это, например, системы восьмеричная и двоичная ($k = 3$), шестнадцатеричная и двоичная ($k = 4$), девятеричная и троичная ($k = 3$) и т. д.

Рассмотрим на примерах перевод чисел из восьмеричной и шестнадцатеричной систем в двоичную и обратно.

Пример 8

Дано: $A_{(8)} = 132$. Найти: $A_{(2)}$.

Решение.

Для получения результата нужно каждую восьмеричную цифру заданного числа записать тремя двоичными разрядами — триадой ($k = 3$):

$A_{(8)} = \quad 1 \quad 3 \quad 2;$

$A_{(2)} = 001 \quad 011 \quad 010;$

Ответ: $A_{(2)} = 1011010$.

Пример 9

Дано: $B_{(16)} = 20E$. Найти: $B_{(2)}$.

Решение.

В этом примере каждая шестнадцатеричная цифра записывается четырьмя двоичными разрядами — тетрадой ($k = 4$):

$B_{(16)} = \quad 2 \quad 0 \quad E;$

$B_{(2)} = 0010 \quad 0000 \quad 1110;$

Ответ: $B_{(2)} = 1000001110$.

Пример 10

Дано: $C_{(2)} = 11001111$. Найти: $C_{(8)}$ и $C_{(16)}$.

Решение.

Для перевода в восьмеричную (шестнадцатеричную) систему нужно разбить заданное двоичное число, начиная с младших разрядов, на триады (тетрады), при необходимости дополняя их нулями. Затем каждую триаду (тетраду) записать цифрами восьмеричной (шестнадцатеричной) системы:

$C_{(2)} = 11001111$; $C_{(2)} = 1011111010$;

$C_{(2)} = 011\ 001\ 111$; $C_{(2)} = 0010\ 1111\ 1010$;

$C_{(8)} = 3\ \ 1\ \ 7$; $C_{(16)} = 2\ \ F\ \ A$;

Ответ: $C_{(8)} = 317$. Ответ: $C_{(16)} = 2FA$.

Поразрядные способы перевода чисел можно использовать для сокращения действий при переводе числа, например, из десятичной системы в двоичную. Для этого число сначала переводят в восьмеричную систему, а затем из восьмеричной системы поразрядно в двоичную систему.

Если в качестве промежуточной системы использовать двоичную, то существенно упрощается перевод из восьмеричной системы в шестнадцатеричную и обратно. Это показано в следующем примере.

Пример 11

Дано: $A_{(8)} = 36\ 057$. Найти: $A_{(16)}$.

Решение:

$A_{(8)} = 3\ \ 6\ \ 0\ \ 5\ \ 7$

$A_{(2)} = 11\ 110\ 000\ 101\ 111$

$A_{(2)} = 0011\ 1100\ 0010\ 1111$

$A_{(16)} = 3\ \ C\ \ 2\ \ F$

Ответ: $A_{(16)} = 3C2F$.

Быстрый способ перевода, использующий устный счет

Записав единицу, приписываем к ней справа нули (1, 10, 100, 1000, 10000, ...) и переводим в десятичную систему. Получаем числа 1, 2, 4, 8, 16, ... С приписыванием справа нуля двоичное число увеличивается вдвое. Если же приписать единицу, то число увеличится вдвое плюс единица.

Пример 12

Дано: $A_{(2)} = 1010011$. Найти: $A_{(10)}$.

Решение.

Начиная со старших разрядов, последовательно открывая разряды числа, получаем:

1, 2, 4 + 1 = 5, 10, 20, 40 + 1 = 41, 82 + 1 = 83.

Ответ: $A_{(10)} = 83$.

4.4. Арифметические действия в двоичной системе

Все правила выполнения арифметических действий в любой позиционной системе совпадают с правилами для десятичной системы. Рассмотрим сложение и умножение чисел в двоичной системе. Таблицы сложения (табл. 4.3) и умножения (табл. 4.4) для двоичной системы предельно просты.

Таблица 4.3. *Сложение чисел*

a	*b*	*a* + *b*
0	0	0
0	1	1
1	0	0
1	1	10

Таблица 4.4. *Умножение чисел*

a	*b*	*a* · *b*
0	0	0
0	1	0
1	0	0
1	1	1

Рассмотрим простейшие примеры сложения и умножения двоичных чисел. Параллельно будем выполнять те же действия в десятичной системе. Это поможет контролировать правильность результата.

Пример 13

Дано: $A = 1010_{(2)} = 10_{(10)}$; $B = 111_{(2)} = 7_{(10)}$.

Найти: $C_{(2)} = A_{(2)} + B_{(2)}$.

Решение:

$$\begin{array}{r} A_{(2)} = \quad 1010 \\ +\ B_{(2)} = \quad 111 \\ \hline C_{(2)} = 10001 \end{array} \qquad \begin{array}{r} A_{(10)} = 10 \\ +\ B_{(10)} = \ 7 \\ \hline C_{(10)} = 17 \end{array}$$

Ответ: $C_{(2)} = 10001$

Пример 14

Дано: $A = 1101_{(2)} = 13_{(10)}$. B $= 1011_{(2)} = 11_{(10)}$.

Найти: $C_{(2)} = A_{(2)} \cdot B_{(2)}$.

Решение:

$\times A_{(2)} = 1101$	$\times A_{(10)} = 13$
$B_{(2)} = 1011$	$B_{(10)} = 11$
1101	+ 13
+ 1101	13
0000	$C_{(10)} = 143$
1101	
$C_{(2)} = 10001111$	

При сложении чисел необходимо правильно формировать переносы в старшие разряды с учетом переносов из младших разрядов чисел. При умножении суммируются частичные произведения (произведения множимого на цифры разрядов множителя). Заметим, что в двоичной системе частичные произведения формируются очень просто: они равны или сдвинутому влево множимому, если цифра множителя равна единице, или нулю, если цифра множимого равна нулю.

4.5. Системы счисления и калькулятор

На рис. 4.1 показан инженерный вариант окна калькулятора. Он имеет кнопки цифр и переключатели, позволяющие выполнять перевод чисел из одной системы в другую.

Ниже окна отображения вводимых чисел и результатов вычислений в рамке размещены четыре переключателя систем счисления: **Hex** — шестнадцатеричная, **Dec** — десятичная, **Oct** — восьмеричная и **Bin** — двоичная. Одновременно может быть выбран только один переключатель.

При выборе системы счисления изменяется набор цифровых клавиш. В двоичной системе работают только клавиши 1 и 0. В восьмеричной — 8, в десятичной — 10. В шестнадцатеричной — все 16 цифровых клавиш.

Перевод чисел выполняется в три приема.

1. Переключателем выбирается исходная (старая) система.
2. *В*водится число.
3. Переключателем выбирается новая система. При этом число переводится в эту систему.

Число — это абстрактное выражение количества. После ввода числа переключатели позволяют познакомиться с его записью еще в трех системах счисления. Все это хороший пример кодирования одного понятия (конкретного числа) в четырех различных алфавитах.

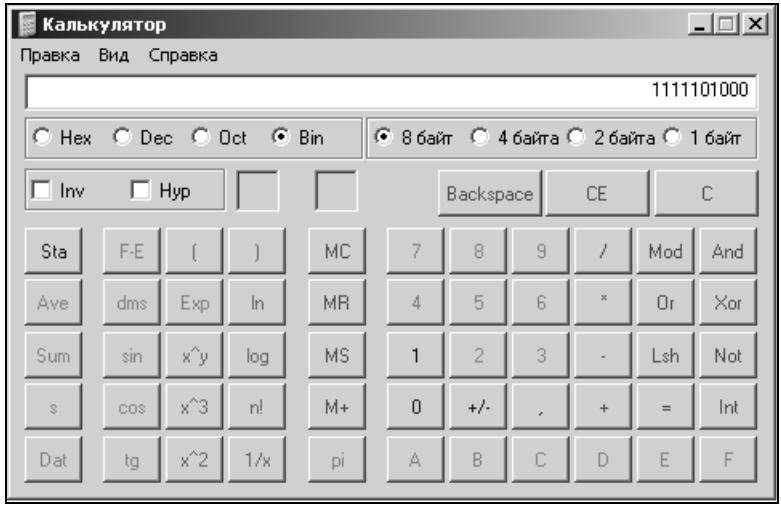


Рис. 4.1. Калькулятор в инженерном режиме

Калькулятор позволяет выполнять над числами в различных системах арифметические действия. Проверьте эту возможность.

4.6. Сравнение систем счисления

На первый взгляд вне конкуренции находится очень хорошо знакомая нам десятичная система. Однако она используется только при вводе информации в компьютер и выводе из него. Это обусловлено следующими причинами.

При хранении и передаче информации каждую цифру необходимо представлять некоторой физической величиной — например, амплитудой напряжения, тока, направлением намагниченности магнитного материала и т. п. В условиях помех, чем больше число градаций этих физических величин (для десятичной системы оно равно десяти), тем больше вероятность переходов от одной градации к другой и появления ошибок. Это приводит к уменьшению надежности хранения и передачи информации. Возможность появления таких ошибок минимальна при использовании двоичной системы.

При кодировании информации в двоичной системе наиболее просто технологически реализуются электронные схемы, выполняющие операции над

числами (транзистор открыт или закрыт, импульс тока есть или нет, участок поверхности магнитного диска намагничен или размагничен). К тому же и действия над двоичными числами, как было показано, выполняются также весьма просто.

К недостаткам двоичной системы следует отнести необходимость и трудоемкость перевода чисел из десятичной системы при вводе информации в компьютер и в десятичную систему при выводе результатов. Отметим также, что двоичная система неэкономна по записи чисел. Она требует больше разрядов, чем запись того же числа в других системах.

Задания для самостоятельной работы

Перевести числа:

1. Дано: $A_{(2)} = 101011$. Найти: $A_{(8)}$, $A_{(10)}$, $A_{(16)}$.
2. Дано: $B_{(8)} = 150$. Найти: $B_{(2)}$, $B_{(10)}$, $B_{(16)}$.
3. Дано: $C_{(10)} = 87$. Найти: $C_{(2)}$, $C_{(8)}$, $C_{(16)}$.
4. Дано: $D_{(16)} = A0$. Найти: $D_{(2)}$, $D_{(8)}$, $D_{(10)}$
5. Дано: $F_{(5)} = 304{,}12$. Найти: $F_{(10)}$.
6. Дано: $G_{(10)} = 47{,}8$. Найти: $G_{(3)}$.
7. Дано: $H_{(\text{р.с.с.})} = \text{MDCCCXII}$. Найти: $H_{(10)}$.

Выполнить действия:

1. Дано: $A_{(2)} = 10101$, $B_{(2)} = 111$. Найти: $C_{(2)} = A_{(2)} + B_{(2)}$.
2. Дано: $D_{(2)} = 1101$, $E_{(2)} = 1011$. Найти: $F_{(2)} = D_{(2)} \cdot E_{(2)}$.

Глава 5

Логические функции и элементы

5.1. Логические операции и функции

Высказыванием называют повествовательное предложение, в отношении содержания которого можно сказать, что оно *истинно* или *ложно*. Высказывания также называют *суждениями*.

Примеры истинных высказываний: "Вода — жидкость", "роман Война и мир написал Лев Толстой", "Число 100 в 10 раз больше числа 10" и т. п.

Примеры ложных высказываний: "Кошка — насекомое", "Число 2 больше числа 5", "Река Нева впадает в озеро Байкал" и т. п.

Об истинности или ложности высказываний "$Z > 10$" и "Дождь идет" сказать ничего нельзя. Нужна дополнительная информация о значении числа Z, о месте и времени события "Дождь идет". Эти высказывания могут рассматриваться как двоичные переменные, о значениях которых (истина или ложь) ничего не известно.

Обычно высказывания обозначают прописными буквами латинского алфавита: A, B, C, D. В этом заслуга древнегреческого философа Аристотеля, который одним из первых стал использовать для обозначения высказываний подобную символику. Это был первый шаг к формализации логических рассуждений.

Будем записывать высказывание, например, так: A = "$Z > 10$". После присвоения переменной Z конкретного значения можно будет сказать о высказывании, что оно истинно или ложно. Например, если присвоить Z значение 5, то высказывание A = "$Z > 10$" = Ложь. Если присвоить Z значение 25, то высказывание A = Истина. Значения Ложь и Истина для краткости заменяют соответственно буквами Л и И, а также цифрами 0 и 1. Последний способ будем использовать чаще всего.

Уже древними философами была замечена важная роль в разговорном языке таких логических связок, как "И", "ИЛИ", отрицания "НЕ" и т. п. Эти связки выступают в качестве *логических операций* над высказываниями. Одни операции объединяют простые высказывания в сложные. Другие — изменяют ис-

тинность высказываний. При этом истинность сложных высказываний зависит от истинности или ложности входящих в него простых высказываний.

Из множества логических операций рассмотрим только три основные: отрицание, логическую сумму и логическое произведение. В арифметике операциям сложения и умножения соответствуют функции сумма и произведение. В логике операциям: отрицание, логическая сумма и логическое произведение соответствуют функции: *инверсия*, *дизъюнкция и конъюнкция*.

Отрицание (инверсия)

Рассмотрим операцию *логическое отрицание* (НЕ). Пусть дано высказывание *А* = "Тигр — хищник". Это высказывание истинное, поэтому можно записать: *А* = Истина. Применим к этому высказыванию операцию отрицания. Запись НЕ А следует понимать, как "Неверно, что тигр — хищник" или как "Тигр — не хищник". Получили, что НЕ *А* = Ложь. Короче все это можно было записать так: *А* = Истина, НЕ *А* = Ложь, или *А* = 1, НЕ *А* = 0.

Вывод: отрицание истинного высказывания приводит к ложному высказыванию. Верно и обратное — отрицание ложного высказывания приводит к истинному высказыванию. Запишем это простое правило в виде таблицы (табл. 5.1). Такие таблицы называют *таблицами истинности*. Данная таблица задает соответствующую операции отрицания функцию инверсия.

Таблица 5.1. *Таблица истинности инверсии*

А	НЕ *А*
0	1
1	0

Инверсия — это функция одного аргумента. В качестве аргумента в ней может выступать простое или сложное высказывание. При записи логических формул обозначающее операцию отрицание слово НЕ заменяется чертой над отрицаемым аргументом, т. е. вместо НЕ *А* пишут: $\overline{A}$. Эта черта — знак операции отрицания. Часто функцию инверсия отождествляют с операцией и говорят: функция отрицание.

Логическая сумма (дизъюнкция)

Результат логической суммы (дизъюнкции) отличается от результата сложения двух одноразрядных двоичных чисел. Логическая сумма двух высказываний *А* и *В* может быть записана так: *А* ИЛИ *В*. Это выражение является сложным (составным) высказыванием. Его истинность зависит от истинно-

сти простых высказываний A и B. Обозначим это новое высказывание буквой C.

Высказывание $C = A$ ИЛИ B — Истина, если хотя бы одно из входящих в него высказываний — Истина, и высказывание $C = A$ ИЛИ B — Ложь, если все входящие в него высказывания — Ложь. Это правило записано в таблице истинности (табл. 5.2), которая задает соответствующую логической сумме функцию дизъюнкция.

Таблица 5.2. *Таблица истинности дизъюнкции*

A B	A ИЛИ B
0 0	0
0 1	1
1 0	1
1 1	1

Дизъюнкция — это функция двух и более аргументов. В качестве аргументов также могут выступать простые или сложные высказывания. При записи логических формул вместо слова ИЛИ ставят знак $\vee$. Это знак операции логического сложения. Например, вместо $C = A$ ИЛИ B записывают $C = A \vee B$.

Пусть $A = X > 10$ и $B = X < Y$. При этом переменным X и Y присвоили следующие значения: $X = 5$, $Y = 7$. В этом случае имеем: $A = X > 10 = 5 > 10 = 0$, $B = X < Y = 5 < 7 = 1$. Логическая сумма этих высказываний по таблице истинности равна 1, т. е. $C = A \vee B = 0 \vee 1 = 1$.

Логическое произведение (конъюнкция)

Результат логического произведения (конъюнкции) совпадает с результатом произведения арифметического. Логическому произведению соответствует функция конъюнкция. Таблица истинности конъюнкции очень проста (табл. 5.3).

Таблица 5.3. *Таблица истинности конъюнкции*

A B	A И B
0 0	0
0 1	0
1 0	0
1 1	1

Конъюнкция — это функция двух и более аргументов. Она равна 1 только в случае, если все высказывания равны 1. При записи логических формул в качестве знака операции логического умножения используют знак $\wedge$ или, как и в обычной алгебре, точку между сомножителями.

Для переменных $X = 5$, $Y = 7$ высказывание $A = X < Y = 1$, а высказывание $B = X > Y = 0$. Поэтому логическое произведение этих простых высказываний в соответствии с таблицей истинности (см. табл. 5.3) равно 0: $C = A \wedge B = 1 \wedge 0 = 0$ или $C = A \cdot B = 1 \cdot 0 = 0$.

Рассмотренные функции служат основой для записи более сложных логических функций большого числа аргументов. Доказано, что функций конъюнкция, дизъюнкция и инверсия достаточно, чтобы записать любую сколь угодно сложную логическую функцию.

Логические функции широко применяются в языках программирования. При составлении программ они используются для записи логических условий.

5.2. Логические элементы и схемы

Все перерабатывающие информацию устройства компьютера представляют собой логические схемы, построенные из логических элементов. В том числе используются и элементы, выполняющие логические операции отрицание, логическое сложение и логическое умножение. Их называют соответственно *инвертор*, *дизъюнктор* и *конъюнктор*. Из этих элементов можно построить все логические схемы компьютера.

Графическое обозначение элементов показано на рис. 5.1 *а*, *б*, *в*.

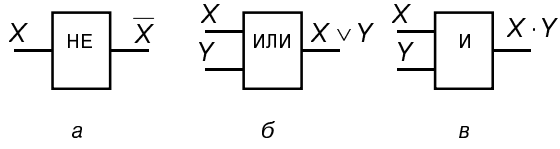


Рис. 5.1. Логические элементы:
а — инвертор; *б* — дизъюнктор; *в* — конъюнктор

В схемах компьютера на входах и выходах логических элементов действуют электрические сигналы. Этими сигналами кодируются логические значения 0 и 1. Например, за 0 принимают сигнал, напряжением 0 вольт, за 1 принимают сигнал, напряжением +3 вольта. Элемент НЕ (инвертор) при таком кодировании работает так: если на входе 0 вольт, то на выходе +3 вольта, если на входе +3 вольта, то на выходе 0 вольт.

Замечательно то, что за логические значения 0 и 1 можно принимать значения 0 и 1 чисел в двоичной системе счисления и строить логические схемы, выполняющие различные операции над двоичными числами.

Рассмотрим простой пример построения схемы одноразрядного сумматора. На входы сумматора поступают значения двух одноразрядных двоичных чисел X и Y. На выходах сумматора вырабатываются значения суммы этих чисел S и переноса P. Таблица истинности функций S и P (табл. 5.4) может рассматриваться как таблица работы двоичного сумматора. Модель сложения и условное обозначение такого сумматора показаны на рис. 5.2.

Таблица 5.4. *Таблица работы сумматора*

X Y	S	P
0 0	0	0
0 1	1	0
1 0	1	0
1 1	0	1

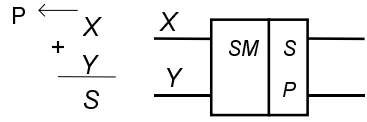


Рис. 5.2. Модель сложения чисел и условное обозначение сумматора

Для построения логической схемы сумматора выпишем из таблицы формулы функций S и P. Будем рассматривать только те строки, где функции принимают значение 1:

$$S = \overline{X} \cdot Y \vee X \cdot \overline{Y}; \qquad P = X \cdot Y.$$

Проверим соответствие формулы суммы таблице работы. Подставляя в формулу суммы наборы значений слагаемых X и Y, можно убедиться в ее правильности:

$$S = \overline{X} \cdot Y \vee X \cdot \overline{Y} = \overline{0} \cdot 0 \vee 0 \cdot \overline{0} = 1 \cdot 0 \vee 0 \cdot 1 = 0 \vee 0 = 0;$$

$$S = \overline{X} \cdot Y \vee X \cdot \overline{Y} = \overline{0} \cdot 1 \vee 0 \cdot \overline{1} = 1 \cdot 1 \vee 0 \cdot 0 = 1 \vee 0 = 1;$$

$$S = \overline{X} \cdot Y \vee X \cdot \overline{Y} = \overline{1} \cdot 0 \vee 1 \cdot \overline{0} = 0 \cdot 0 \vee 1 \cdot 1 = 0 \vee 1 = 1;$$

$$S = \overline{X} \cdot Y \vee X \cdot \overline{Y} = \overline{1} \cdot 1 \vee 1 \cdot \overline{1} = 1 \cdot 0 \vee 0 \cdot 1 = 0 \vee 0 = 0.$$

Формула переноса P — это формула конъюнкции. Поэтому для схемной реализации переноса потребуется всего один логический элемент — конъюнктор (элемент И). Для схемной реализации суммы потребуется два инвертора (элемента НЕ), два конъюнктора и один дизъюнктор (элемент ИЛИ). Логическая схема сумматора показана на рис. 5.3.

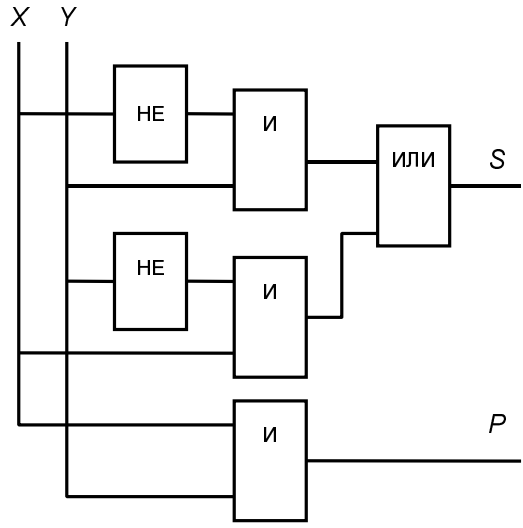


Рис. 5.3. Логическая схема сумматора

Глава 6

Принцип работы компьютера

6.1. Общие сведения о компьютере

Компьютером называют электронное устройство, способное автоматически выполнять заданную последовательность действий по приему, хранению, преобразованию и выдаче информации.

Появление такого рода устройств явилось существенной необходимостью. Наблюдаемый в последние десятилетия "информационный взрыв" потребовал создания соответствующих средств для переработки колоссальных объемов информации, связанных с экономикой, наукой и техникой, другими отраслями деятельности человека.

Компьютеры пришли на смену таким средствам вычислений, как счеты, арифмометры, логарифмические линейки, другие счетные приборы и устройства. Среди предшественников компьютеров называют механические вычислительные устройства французского математика Паскаля, английского инженера Бэббиджа.

Современные компьютеры обладают многими замечательными свойствами. Важнейшие из них: высокое быстродействие, большой объем памяти, высокая точность вычислений, достоверность результатов. К ним следует добавить высокую надежность, малые размеры, вес и потребление энергии.

Вся информация в компьютере делится на *данные* и *программу* их обработки. Под данными понимаются исходные данные, промежуточные результаты, результаты решения задач. Программа состоит из последовательности *команд* компьютеру на выполнение действий с данными.

Простейшая схема компьютера показана на рис. 6.1. Эта схема позволит пояснить основной принцип работы компьютеров — принцип программного управления.

Из схемы видно, что компьютер состоит из следующих устройств:

- *Запоминающее устройство* (ЗУ) — это память компьютера. Оно предназначено для хранения информации: данных и программ. ЗУ состоит из *ячеек памяти*, способных хранить число или команду программы. Каждая

ячейка памяти имеет свой *адрес*. Чтение информации из ячеек может выполняться многократно. При записи в ячейку предыдущая информация стирается (как в бытовом магнитофоне).

- *Арифметическо-логическое устройство* (АЛУ) служит для выполнения действий над числами.
- *Устройство управления* компьютером (УУ) выбирает из ЗУ команду за командой и вырабатывает сигналы для их выполнения. АЛУ и УУ образуют *процессор* компьютера.
- *Устройства ввода и вывода* информации — это устройства соответственно ввода исходных данных, программ их обработки и вывода результатов работы компьютера. *Пульт управления* служит для подачи на компьютер различных управляющих сигналов.

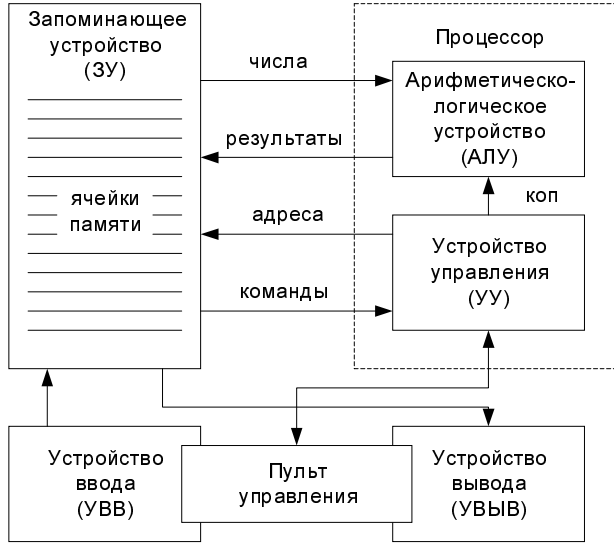


Рис. 6.1. Упрощенная структурная схема компьютера

6.2. Программирование на машинном языке

Одной из характеристик компьютера является набор используемых команд, называемый *системой команд*. Важнейший параметр команд — их *адресность*. Используются трехадресные, двухадресные, одноадресные команды, команды с переменной адресностью, безадресные команды. На рис. 6.2 показана структура наиболее естественной и понятной трехадресной команды.

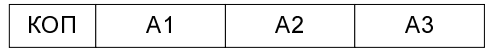


Рис. 6.2. Структура трехадресной команды.

- КОП — код операции, закодированный числом символ операции (действия) над числами;
- A1 — адрес ячейки ЗУ, где хранится первый операнд (первое число);
- A2 — адрес ячейки ЗУ, где хранится второй операнд (второе число);
- A3 — адрес ячейки ЗУ, куда должен быть записан результат операции.

К операциям в компьютере относятся не только арифметические действия над числами, но и логические операции сравнения чисел, передачи чисел, печати результатов и многие другие. Выберем для примеров упрощенную систему команд и поставим в соответствие выбранным операциям их коды (табл. 6.1).

Таблица 6.1. *Коды операций*

Операция	КОП
сложение	01
вычитание	02
умножение	03
деление	04
печать	05

Допустим, что память нашего гипотетического компьютера имеет 1000 ячеек с адресами от 000 до 999. Тогда команда, например, сложения двух чисел может выглядеть так:

01 205 206 320,

где 205 и 206 — адреса ячеек памяти, в которых хранятся слагаемые; 320 — адрес ячейки памяти, куда будет записана сумма.

Поставим себе задачу составить на машинном языке в выбранной системе команд программу вычисления функции:

$$F = (a^2 - b)/c + 5d.$$

При программировании на машинном языке приходится самостоятельно решать задачу размещения в памяти компьютера данных и программы. Пусть данные занимают последовательно ячейки, начиная с адреса 201, а программа — ячейки с адреса 301. Остальные незанятые ячейки могут служить для записи промежуточных результатов (рабочие).

В табл. 6.2 показано размещение в памяти компьютера данных, а в табл. 6.3 — размещение программы. Программа составлена в соответствии с последовательностью выполнения действий при вычислении заданной функции *F*.

Таблица 6.2. *Данные*

Адреса	Содержимое
201	*a*
202	*b*
203	*c*
204	*d*
205	*5*
206	*F*
207	рабочая
208	рабочая

Таблица 6.3. *Программа*

Адрес	Команды				Примечания
	КОП	А1	А2	А3	
301	03	201	201	207	a^2
302	02	207	202	207	$a^2 - b$
303	04	207	203	207	$(a^2 - b)/c$
304	03	205	204	208	$5d$
305	01	207	208	206	*F*
306	05	206	000	000	печать *F*

Так составлялись программы, и так выглядела их запись для компьютеров первого поколения (на электронных лампах), когда языки программирования высокого уровня (Фортран, Алгол) еще только разрабатывались и начинали применяться.

Составление программ было делом утомительным, малоэффективным, требовало большого внимания, сопровождалось большим количеством ошибок, которые приходилось "вылавливать" самим программистам.

6.3. Принцип программного управления

Принцип работы компьютера проследим на примере выполнения составленной программы по упрощенной схеме на рис. 6.1. Каждая команда программы в общем случае выполняется за 6 шагов.

1. Из памяти считывается в начале первая, затем очередная команда программы и передается в УУ для расшифровки.
2. Адреса А1, А2 и А3 адресной части команды из УУ передаются в ЗУ.
3. В ЗУ из ячеек с адресами А1 и А2 считываются числа и передаются в АЛУ.
4. Код операции (КОП) передается в АЛУ, где над числами выполняется операция.
5. Результат операции передается из АЛУ в ЗУ, где записывается в ячейку с адресом А3.
6. Если все команды выполнены, то компьютер прекращает работу, если нет, то выполняется переход к шагу 1.

Рассмотренный пример кодирования информации, распределения памяти, составления и записи программы и работы компьютера при ее выполнении наглядно демонстрирует основной принцип работы компьютера — *принцип программного управления*. Этот принцип был сформулирован американским ученым Дж. фон Нейманом и служит основой для построения компьютеров.

Принцип программного управления включает:

- деление информации на данные и программу их обработки;
- кодирование данных и программы числами;
- запись данных и программы в память компьютера;
- последующую автоматическую работу компьютера по выполнению программы.

6.4. Свойства компьютера

Компьютеры занимают центральное место среди устройств, имеющих дело с информацией. Они способны воспринимать информацию, хранить ее, преобразовывать в соответствии с заданной программой в требуемый вид, выдавать результаты преобразования.

Среди замечательных свойств компьютера в начале данной главы были выделены высокое быстродействие, большой объем памяти и надежность, обеспечивающая достоверность результатов работы. Рассмотрим их подробнее.

- Быстродействие. Было время, когда, удивляясь быстродействию компьютера "Урал" (100 операций типа сложения многоразрядных чисел в секун-

ду), подсчитывали, сколько людей-вычислителей он заменяет. Современные компьютеры выполняют сотни миллионов и более таких операций в секунду.

Учеными в свое время было поставлено много задач, на решение которых "вручную" требовались столетия. Сейчас, благодаря компьютеру, многие из этих задач решены. Некоторые задачи науки и техники требуется решать в *реальном времени*. Например, на коррекцию траектории ракеты, выводящей на орбиту спутник, отводятся секунды. Только компьютер способен решать подобные задачи.

- Объем памяти. Если первые компьютеры имели память несколько тысяч слов, то в память современного компьютера можно записать, например, содержание всех книг большой библиотеки. Это миллиарды слов. Из такой библиотеки можно практически мгновенно вызывать на экран монитора любую книгу и нажатием клавиши перелистывать ее страницы.
- Достоверность результатов. Если громоздкую задачу по арифметике задать ученикам одного класса и предложить решить ее за ограниченное время, то результаты решения могут значительно различаться. При этом ошибок и различий будет больше, если задачу задать на последнем уроке. Скажется усталость. При решении такой задачи компьютером ответ будет получен намного быстрее. И сколько бы раз не запускали компьютер, результат будет один и тот же. Компьютер не устает. Он свободен от эмоций. При правильно составленной программе решения задачи компьютер не допускает ошибок и выдает достоверные, правильные результаты решения.

К положительным свойствам компьютера также следует добавить небольшие размеры, вес и малое потребление энергии.

В 1950-х годах ученые обсуждали философский вопрос: Может ли машина мыслить? Насмотревшись фантастических боевиков о роботах, не спешите ответить на этот вопрос утвердительно. Чтобы ответить на него, нужно точно определить все составляющие его понятия, т. е. договориться, что понимать под словами **может**, **машина** и **мыслить**. Например, компьютеры играют в шахматы на уровне чемпиона мира, доказывают сложнейшие теоремы. Можно ли считать, что они мыслят? Пока это сказать нельзя. Программу игры и принципы доказательств составляют математики и программисты.

6.5. Поколения компьютеров

Вычислительные устройства, упрощающие и автоматизирующие процесс вычислений, разрабатывались, создавались и внедрялись как насущная необходимость, как веление времени. Так Паскаль создал свое вычислительное устройство, желая облегчить работу отцу — сборщику налогов, проводивше-

му ночи за подсчетом собранных денег. Это же можно сказать и о Бэббидже, решавшем важную задачу автоматизации расчетов склонений на мореходных картах, монопольным изготовителем которых в те времена являлась Англия. А один из первых электромеханических табуляторов решал сложную задачу обработки результатов переписи населения США. Но все эти устройства обычно относят к предыстории создания устройств, ставших незаменимыми помощниками человека, устройств, совсем недавно чаще называемых электронными вычислительными машинами (ЭВМ). С их появлением и начинается отсчет поколений компьютеров.

- Первое поколение. К нему относят компьютеры *на электронных лампах*. Компьютеры были громоздкими, занимали громадные помещения, потребляли десятки киловатт электроэнергии и обладали низкой надежностью. Ученые того времени ввели понятие "тирании больших количеств". Компьютеры имели сотни ламп, тысячи резисторов, конденсаторов, паяных соединений и контактов разъемов. При ограниченной надежности этих деталей возникала ситуация, когда каждый момент времени в компьютере был неисправен хотя бы один из его элементов. Поэтому при очень большом числе элементов с малой надежностью компьютер мог все время находиться в неисправном состоянии.
- Второе поколение. Некоторые проблемы компьютеров первого поколения были решены с появлением *транзисторов*. Уменьшились размеры. Существенно уменьшилась потребляемая мощность. За счет применения печатного монтажа несколько увеличилась надежность. Однако продолжалось использование отдельных, соединяемых в схемы пайкой резисторов, конденсаторов, трансформаторов.
- Третье поколение. Существенно повысилась надежность компьютеров с внедрением *интегральных схем* сначала малой, затем средней степени интеграции. В едином технологическом процессе на кристалле полупроводника изготавливаются транзисторы, резисторы и соединения их в логические схемы. Поэтому надежность сложнейшей логической схемы из сотен и тысяч элементов в интегральном исполнении практически равна надежности одного элемента. Значительно уменьшились размеры и потребляемая мощность.
- Четвертое поколение. Кардинально проблемы надежности, потребляемой мощности, веса, размеров, стоимости компьютеров были решены при внедрении *больших интегральных схем* (БИС). В настоящее время существенно увеличивается сложность БИС, их быстродействие. На одном кристалле кремния изготавливается вся схема процессора современных компьютеров. Применение больших интегральных схем способствовало появлению персональных компьютеров.

Задания для самостоятельной работы

Запишите программу вычисления одной из следующих функций:

1. $f = (2a^2 - b) / (b - c)$.
2. $f = ax^2 + bx + c$; *c*) $f = 7x^4 - a / (b + c)$.
3. $f = 3a / 2b + 2abc$.

Распределите память компьютера, имеющую 1000 ячеек, между данными и программой. Для записи программы используйте рассмотренные выше трехадресную систему команд и список операций.

Глава 7

Общие сведения о компьютерах

7.1. Структурная схема персонального компьютера

Упрощенная структурная схема персонального компьютера (ПК) показана на рис. 7.1.

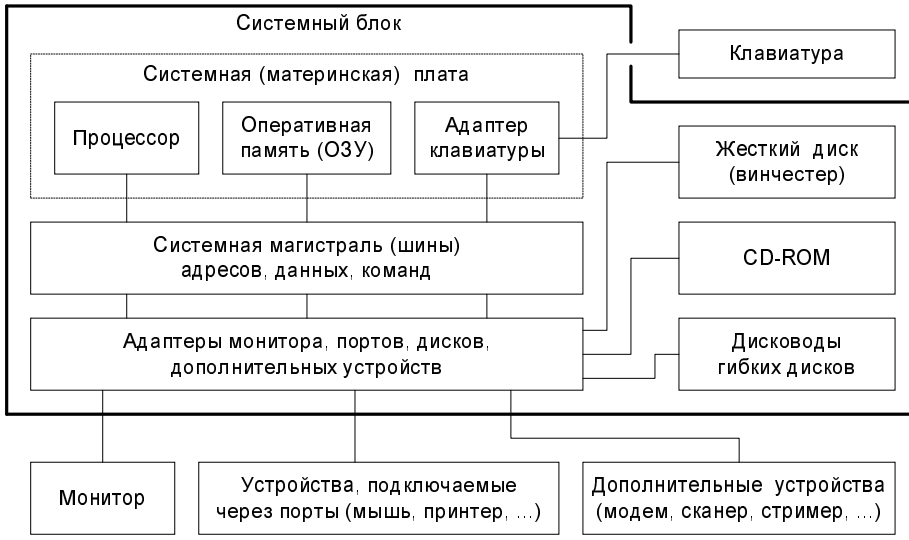


Рис. 7.1. Структурная схема ПК

Современные персональные компьютеры строятся по *магистрально-модульному* принципу, принципу *открытой архитектуры*. Вся схема компьютера состоит из устройств-модулей, соединяемых между собой с помощью *стандартного интерфейса* — стандартных разъемов и кабелей (магистралей). Это позволяет пользователю увеличивать, изменять и модернизировать ком-

плектность компьютера добавлением или заменой блоков-модулей на более производительные или обладающие большей памятью.

В системном блоке ПК находится *системная (материнская) плата*, на которой расположены процессор, оперативная память, *чипсет* и множество разъемов для связи с другими устройствами. Чипсет — это микросхемы, управляющие основными устройствами ПК, обменом информации между ними. Чипсет во многом определяет производительность ПК.

Основная характеристика материнской платы — это ее типоразмер, называемый *форм-фактором*. Он определяет геометрические размеры платы, расположение разъемов и т. п. Современным популярным форм-фактором материнских плат является ATX (Advanced Technology extended). На такой плате располагаются контроллеры и разъемы большинства работающих с компьютером устройств. Плата обеспечивает включение и выключение питания компьютера программным способом, опрос датчиков температуры процессора и управление работой его вентилятора, а также имеет другие полезные свойства.

Процессор выполняет в ПК основную нагрузку по преобразованию информации. Быстродействие процессора определяет быстродействие компьютера и может составлять десятки и сотни миллионов элементарных операций с секунду. *Разрядность процессора* (16, 32, 64 двоичных разрядов) определяет разрядность обрабатываемых двоичных чисел.

Производительность процессоров определяется как количество операций над числами за секунду. Каждая операция выполняется за несколько тактов. Время выполнения одного такта определяется тактовой частотой процессора, которая измеряется в сотнях и тысячах мегагерц. Кроме того, производительность зависит от структуры процессора, позволяющей распараллеливать вычисления и совмещать их во времени.

В качестве единиц измерения производительности используют:

- МИПС (MIPS — Mega Instruction Per Second) — миллион операций (инструкций) над числами с фиксированной точкой в секунду;
- МФЛОПС (MFLOPS — Mega Floating Operation Per Second) — миллион операций над числами с плавающей точкой в секунду;
- ГФЛОПС (GFLOPS — Giga Floating Operation Per Second) — миллиард операций над числами с плавающей точкой в секунду.

Информация между устройствами компьютера передается по проводам — *магистралям* (шинам). Различают адресные шины, командные шины и шины данных. Число разрядов адресной шины определяет количество адресов запоминающего устройства — *адресное пространство*. Число разрядов шины данных определяется, как правило, разрядностью процессора.

Принцип открытой архитектуры обеспечивается платами и слотами расширения возможностей ПК. *Платы расширения* еще называют картами или адаптерами. *Слоты* — это разъемы, в которые вставляются платы расширения, например, оперативная память.

Адаптеры (контроллеры) устройств, подключаемых к материнской плате и внешним портам ПК, обеспечивают управление работой этих устройств. Все адаптеры связаны с процессором и памятью ПК посредством системной магистрали и территориально расположены в системном блоке.

Сетевой адаптер служит для включения компьютеров в локальную сеть, позволяющую пользователю получать доступ к информации в других компьютерах.

Видеоплата (видеокарта) обеспечивает связь с монитором и управление его работой. На плате помещена видеопамять монитора.

Аудиоплата (Sound Blaster, звуковая карта) позволяет с помощью компьютера воспроизводить и редактировать музыку и речь. Аудиоплата связана со звуковыми колонками, наушниками, микрофоном.

Модем — это устройство для обмена информацией с другими компьютерами через телефонную сеть. Модемы (МОдуляторы-ДЕМодуляторы) различаются скоростью передачи информации, которая может составлять 2400, 9600 и более бод (1 бод равен 1 бит в секунду). Модемы по исполнению бывают встроенными в системный блок компьютера или внешними, подключаемыми к компьютеру через коммуникационный порт.

Внешние устройства ПК подключаются к *портам*, представляющим собой разъемы на задней стенке системного блока. Различают параллельные и последовательные порты.

Параллельные порты обеспечивают передачу информации параллельным кодом, например, побайтно, по восьми проводам. Это наиболее быстрый способ передачи. К параллельному порту подключается, например, принтер. Эти порты обозначаются как LPT1, LPT2 и LPT3 (LinePrinTer — линии принтера).

Последовательные порты служат для подключения линий, передающих информацию на большие расстояния последовательным кодом, по одному проводу. Это коммуникационные порты COM1, COM2, ... (COMmunication port).

В настоящее время многие последовательные и параллельные порты устарели и используются только для совместимости с разъемами ранее разработанных устройств. Используется параллельный порт с расширенными функциями *ECP* (Extended Capabbility Port), для подключения принтеров, сканеров, плоттеров используется улучшенный параллельный порт *EPP* (Enhanced Parallel Port).

Из последних разработок следует отметить последовательную универсальную шину *USB* (Universal Serial Bus), позволяющую подключать множество устройств даже без выключения питания. Находит также применение инфракрасная линия связи IrDA (Infrared Data Association), обеспечивающая беспроводную связь с ПК клавиатуры, мыши, принтера и других устройств.

В системном блоке имеется *блок питания*, обеспечивающий все схемы компьютера необходимыми номиналами напряжений. Блок питания охлаждается воздушным вентилятором. *Системные часы* с аккумулятором обеспечивают подсчет текущего времени.

На передней панели системного блока находятся кнопки включения/выключения ПК и кнопки переключения режимов работы. Здесь же расположены карманы для гибких и лазерных дисков.

Остальные устройства будут рассмотрены ниже.

7.2. Память компьютера

Триггеры

Самый низкий уровень памяти занимают элементы, способные находиться только в двух устойчивых состояниях и хранить значение одной двоичной единицы информации — один бит. Это триггеры (рис. 7.2).

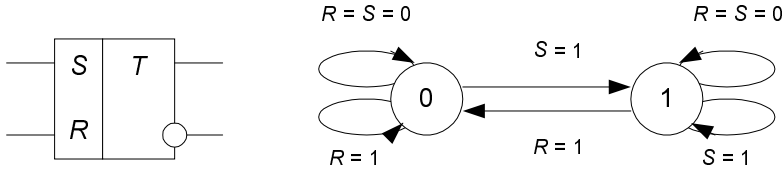


Рис. 7.2. Условное обозначение и графическая схема переходов триггера

Триггер имеет два входа: *R*-вход — вход установки схемы в нулевое состояние, *S*-вход — запись единицы, установка в единичное состояние. При нулевом состоянии триггера на выходе Q нулевой сигнал ($Q = 0$), на выходе $\overline{Q}$ — единичный ($\overline{Q} = 1$). Это прямой и инверсный выходы триггера.

Показанная на рис. 7.2 схема триггера является основой для построения более сложных схем элементов памяти и схем с памятью. Триггеры служат основными элементами памяти быстродействующих запоминающих устройств, реализованных в интегральном исполнении. С выключением питания содержимое такой памяти теряется.

Регистры и счетчики

Следующий уровень занимают элементы, способные хранить многоразрядные двоичные числа. Это регистры и счетчики. Для их построения используются триггеры. Так, для построения регистра, способного хранить один байт информации, требуется 8 триггеров. В регистрах хранятся данные, поступающие в процессор для выполнения с ними различных операций. Регистры и счетчики способны не только хранить, но могут также преобразовывать двоичную информацию, например, сдвигать числа, инвертировать их разряды, подсчитывать количество поступивших на вход сигналов и т. д.

Кэш-память

Кэш-память — это сверхоперативная память. Она занимает промежуточное положение между регистрами процессора и ОЗУ, которое в связи со сравнительно невысоким быстродействием может несколько сдерживать работу процессора. Поэтому уже в первых ПК ставилось несколько "быстрых" регистров для хранения промежуточных результатов и другой информации, которая могла потребоваться в ближайшее время. Со временем "быстрые" регистры развились в быстродействующую кэш-память объемом в 256, 512 и более Кбайт.

Оперативная память

Следующий уровень памяти — это ОЗУ (RAM, Random Access Memory — память с произвольным доступом). Это основная, наиболее загруженная работой память компьютера. Часть ОЗУ при работе ПК отводится для хранения операционной системы — тех ее разделов, которые обеспечивают загрузку и тестирование ПК, а также выполнение других системных задач. Объем ОЗУ современных ПК равен 32, 64, 128 и более Мбайт.

Постоянная память

В постоянное запоминающее устройство — ПЗУ (ROM, Read Only Memory — память только для чтения) информация записывается при изготовлении, однократно. В ПЗУ записываются программы BIOS (Basic Input/Output System — базовая система ввода/вывода). Это программы работы компьютера после его включения, среди которых, в частности, находятся программы проверки аппаратуры тестированием и программы загрузки операционной системы. Расположены ОЗУ и ПЗУ на системной плате ПК.

Видеопамять

Монитор имеет собственную память, используемую для того, чтобы хранить созданные изображения. Это видеопамять, размещаемая в адаптере монито-

ра. Процессор записывает в нее изображение, которое затем и выводится на экран монитора. Объем видеопамяти — десятки Мбайт.

Гибкие дискеты

Внешней по отношению к материнской плате памятью является память на дисках. Это следующий уровень памяти, как по объему, так и по использованию. Дисковая память — память с произвольным доступом. Диски делятся на съемные (гибкие диски, или дискеты, и компакт-диски) и несъемные (жесткие диски, так называемые "винчестеры"). Гибкие съемные дискеты размером 3,5 дюйма (89 мм) удобны для документирования результатов работы, переноса документов и программ с компьютера на компьютер, выполнения архивных копий. Они представляют собой гибкие пластиковые покрытые тонким магнитным слоем диски, расположенные внутри защитных конвертов. Запись информации ведется с помощью головок на магнитную поверхность с обеих сторон диска. Информация записывается по концентрическим окружностям — дорожкам. Дорожки разделены на сектора. Чем выше скорость вращения диска, тем больше скорость обмена информацией. Объем памяти наиболее распространенных трехдюймовых дискет составляет 1,44 Мбайт. На этих дискетах имеется переключатель, разрешающий или запрещающий запись информации.

Успешно проводятся разработки по увеличению объема памяти гибких дисков. Уже поступили в продажу разработанные фирмой Sony Electronics новые диски емкостью в 200 Мбайт, что в 140 раз превышает емкость широко в настоящее время используемых дискет на 1,44 Мбайт. Более распространены разработанные другими фирмами дисководы на 120 и 100 Мбайт. Важно отметить, что новые дисководы позволяют использовать также и старые дискеты.

Жесткие диски (HDD — Hard Disk Drive)

Жесткие диски — это главная рабочая память большого объема и длительного хранения. Накопители на жестких дисках (винчестеры) служат для хранения информации, необходимой для обеспечения работы компьютера (операционной системы, программ обслуживания) и работы пользователя (пакетов прикладных программ, различных редакторов, трансляторов языков программирования, создаваемых пользователем документов). Винчестеры встроены в системный блок ПК. Объем памяти жестких дисков может составлять от нескольких сотен Мбайт до нескольких десятков Гбайт. Скорость обмена информацией жестких дисков на порядок выше, чем гибких.

Компакт-диски

Сочетанием достоинств гибких дисков (съемность) и жестких дисков (большой объем памяти) обладают компакт-диски. Наиболее распространены диски с алюминиевой поверхностью. Это CD-ROM (Read Only Memory — только читаемая память). Один такой диск обладает емкостью 450-ти трехдюймовых дискет. Этого достаточно, чтобы сохранять на нем до четверти миллиона страниц текста. Лазерные диски CD-R (Recordable — диск с записью) позволяют однократно записать информацию с последующим многократным ее считыванием. Перезаписываемые CD-RW-диски (CD Rewritable) допускают тысячекратную перезапись. Компакт-диски находят большое распространение для тиражирования программных продуктов, учебных пособий, видеофильмов, игр.

Стримеры

Важность и ценность информации, хранящейся на винчестере компьютера, бывает чрезвычайно высока. Потеря ее из-за воздействия, например, вирусов может быть невосполнимой. Для резервного копирования информации находят применение стримеры. Это накопители на магнитных лентах. Они более дешевы, чем устройства для записи и чтения компакт-дисков, и обладают достаточно большой емкостью (десятки Гбайт). Однако стримеры уступают компакт-дискам по надежности и длительности сроков хранения. Гарантия сохранности информации на компакт-дисках дается не менее чем на 30 лет. Магнитные ленты подвержены высыханию, короблению, воздействию магнитных полей.

7.3. Логические схемы компьютера

Логические элементы

На самом низком уровне находятся логические элементы, реализующие элементарные логические (булевы) функции. Это конъюнкторы (элемент И), дизъюнкторы (ИЛИ), инверторы (НЕ), элементы Шеффера (И-НЕ), Пирса (ИЛИ-НЕ) и др.

Логические схемы

Из логических элементов низкого уровня строятся логические схемы, реализующие более сложные логические функции, такие, например, как схема многоразрядного сумматора двоичных чисел. Из них же строятся все элементы памяти.

Элементы с памятью

Это триггеры, регистры и счетчики. На входах и выходах элементов с памятью, как правило, стоят логические схемы записи, чтения и преобразования информации.

Функциональные блоки

Это объединения логических схем и схем с памятью для выполнения сложных функций. Примером функционального блока может служить так называемый "блок арифметики", в простейшем варианте состоящий из трех регистров (регистра первого числа, регистра второго числа и регистра результата) и сумматора.

Устройства

Устройства — это сложные, состоящие из множества логических схем и схем с памятью, функционально завершенные устройства, решающие задачи по преобразованию информации и согласованию работы других устройств, работой которых они управляют. Обычно выделяют арифметическо-логическое устройство (АЛУ), центральным элементом которого является упомянутый выше блок арифметики, устройство управления (УУ), устройства ввода и вывода информации УВв и УВыв, устройства памяти. Совокупность АЛУ и УУ образует процессор компьютера.

Все перечисленные выше элементы, кроме УВв и УВыв, образуют системный блок компьютера. Рассмотрим элементы, находящиеся вне системного блока.

7.4. Внешние устройства компьютера

Монитор

Монитор (дисплей) служит для вывода текстовой и графической информации. Мониторы бывают цветные и черно-белые (монохромные). Для вывода текста экран монитора разбивается на знакоместа, в каждое из которых может быть выведен любой из знаков (буквы, цифры, символы). Количество строк и знакомест в строке зависит от режима работы экрана. На цветном мониторе возможно окрашивание текста, графики и фона, на котором они выводятся.

К основным характеристикам мониторов относят: размеры экрана, разрешающую способность экрана, размеры точек — пикселов экрана (PICture'S ELement — элемент картинки), объем видеопамяти. Размеры экранов мони-

торов задаются длиной диагонали экрана в дюймах. Наиболее распространены экраны 15" и 17". В системах проектирования для получения качественного графического изображения используют мониторы, имеющие размер до 21". Разрешающая способность экрана определяется количеством точек на экране по его горизонтали и вертикали. Чем больше точек, тем выше качество изображения текстов и графики. В современных компьютерах разрешающая способность экранов мониторов доведена до 800×600, 1024×768, 1600×1200 и выше. От размера точки на экране зависит четкость изображения. На качественных мониторах величина зерна может составлять от трети (0,31, 0,28,) до четверти (0,25) миллиметра. Качество изображения существенно влияет на комфортность работы пользователя, на утомляемость его зрения.

Широкое распространение, особенно для портативных ПК, начинают получать плоские жидкокристаллические экраны. Они свободны от вредных излучений, имеют меньший вес и объем.

Клавиатура

Клавиатура предназначена для ввода информации. Расположение и число клавиш на клавиатурах разных типов могут несколько различаться. Как правило, это 101 клавиша с расположением, как у пишущей машинки. Клавиатуры адаптируются под национальные алфавиты.

Принтеры

Принтеры служат для вывода информации на бумагу или пленку. Принтеры способны выводить тексты, рисунки, графики, другие изображения. Различают черно-белые и цветные принтеры. По принципам работы различают матричные, струйные и лазерные принтеры. Рассмотрим их вкратце.

- *Матричные принтеры* имеют головку с матрицей иголок. Головка перемещается вдоль распечатываемой строки, нужные для данного символа иголки в нужные моменты ударяют по красящей ленте, оставляя на бумаге изображение этого символа. Важной характеристикой принтеров является скорость печати, которая у матричных принтеров невысока и составляет от 10 до 60 секунд на страницу.
- *Струйные принтеры* формируют изображение с помощью разбрызгивания выдуваемых из микросопел специальных чернил. Они обеспечивают более высокое качество печати по сравнению с матричными принтерами. Важным достоинством струйных принтеров является простота организации цветной печати — достаточно сменить картридж с чернилами. Скорость печати струйных принтеров сравнима со скоростью матричных и составляет от 15 до 100 секунд на страницу.

- *Лазерные принтеры* обеспечивают наиболее качественную печать. В них используется принцип ксерографии. Изображение переносится на бумагу со специального барабана. На барабане изображение формируется путем электризации поверхности лучом лазера, перемещением которого управляет компьютер. К электризованным участкам притягиваются частички краски, которая затем переносится на бумагу. Скорость печати составляет от 5 до 15 секунд на страницу. Быстродействующие лазерные принтеры печатают от 15 до 40 страниц в минуту.

Мышь

Мышь — это манипулятор для ввода информации в компьютер. Для перемещения стрелки-указателя мыши по экрану мышь перемещают по ровной поверхности, как правило, по специальному коврику. Мышь вслед за клавиатурой становится главным инструментов работы с компьютером. Имеются манипуляторы другого типа — например, в виде вращаемого рукой шара (трекбол). Имеются мыши с дистанционной, беспроводной передачей сигналов компьютеру.

Сканер

Сканер предназначен для считывания текстовой и графической информации в компьютер. Просматривая изображение, сканер создает его электронную копию. Различают настольные (страничные и планшетные) сканеры и ручные. Настольные сканеры обрабатывают весь лист целиком. Для считывания информации ручным сканером нужно вручную проводить считывающим устройством по листу с этой информацией. Сканеры бывают черно-белые и цветные. Сканеры различаются по разрешающей способности, количеству различаемых ими цветов или оттенков серого цвета, а также по скорости сканирования и предельным размерам сканируемого документа.

Графический планшет

Графический планшет служит для ввода чертежей в компьютер. Используется в системах автоматизации проектирования (в САПРах).

Плоттер

Плоттер (*to plot* — вычерчивать чертеж), или графопостроитель, используется для вывода из компьютера чертежей на бумагу и пленку. Применяется для получения высококачественных архитектурных чертежей, географических карт, художественной графики. Различают планшетные плоттеры и плоттеры барабанного типа. По принципу действия различают плоттеры перьевые, карандашные, струйные и электростатические. Используются в САПРах.

Устройства распознавания речи

С помощью устройств распознавания речи получаемый на выходе микрофона аналоговый сигнал кодируется в цифровой форме и записывается в память ПК. В памяти находятся образцы слов, с которыми сравнивается вновь полученное слово. Это позволяет идентифицировать речь конкретного человека (например, в криминалистике).

Устройства распознавания речи находят применение в обучении иностранным языкам. В одной из таких систем на экран выводятся графические образы слов с правильным произношением и слов, произносимых обучаемым, что позволяет выполнять коррекцию.

Источники бесперебойного питания

Источники бесперебойного питания обеспечивают кратковременное продолжение работы при сбое или полном отключении питания в сети.

7.5. Классификация компьютеров

По назначению и уровню производительности компьютеров их можно условно подразделить на компьютеры, встроенные в различные изделия, персональные компьютеры и суперкомпьютеры.

Микрокомпьютеры, *микропроцессоры* встраиваются в системы управления технологическими процессами, в станки с микропрограммным управлением, в системы вооружения, транспортные системы, в бытовую технику и т. п.

Персональные компьютеры наиболее широко известны. Широк и спектр их модификаций: карманные (электронные записные книжки), блокнотные (notebook), настольные варианты персональных компьютеров. Все это компьютеры, различающиеся быстродействием, видом и объемом памяти, средствами ввода и вывода информации, уровнем оснащенности мультимедийными средствами.

Суперкомпьютеры предназначены для решения задач, требующих сверхвысокой производительности вычислительных средств. К ним относятся, например, *суперкомпьютеры "Эльбрус"*.

Отметим, что большой вклад в разработку архитектуры современных и перспективных компьютеров внесли российские ученые и инженеры. Так, идеи, заложенные в известные всему миру многопроцессорные вычислительные системы большой производительности "Эльбрус", по многим параметрам на десятилетие опережают аналогичные разработки западных фирм. Это отмечают ведущие западные специалисты. В частности, Pentium-III по своей архитектуре и параметрам — почти точная копия компьютера фирмы "Эльбрус" Эль-90. Компьютеры "Эльбрус" начали выпускаться задолго до

того, как реализованные в них современные архитектурные принципы начали только обсуждаться на Западе.

В суперкомпьютерах "Эльбрус-1" (1978 год), "Эльбрус-2" (1984 год), "Эльбрус-3" (1991 год) и E2K ("Эльбрус-2000") высокая производительность достигается глубоким распараллеливанием как на программном, так и на структурном уровнях.

Программное распараллеливание ведется на уровне задач (многозадачность) и на уровне операций и микроопераций (распараллеливание и совмещение выполнения во времени). Структурное распараллеливание состоит в многопроцессорности (до 10 процессоров) и наличии специализированных блоков выполнения арифметических операций и других устройств, поддерживающих программное распараллеливание.

Одна из последних разработок компании "Эльбрус" — суперкомпьютер E2K — по материалам зарубежных изданий по своим параметрам существенно превосходит разрекламированный на западе суперкомпьютер Merced фирмы Intel.

Компьютеры ряда "Эльбрус" являются *суперскалярами*. Они ориентированы на решение самого широкого круга задач. Их архитектура намного сложнее *векторных* суперкомпьютеров, для которых характерной задачей является, например, задача прогнозирования погоды.

При решении этой задачи на всю поверхность земного шара как бы набрасывается сетка, каждой ячейке которой соответствует вектор с параметрами погоды: температура, давление, характер облачности, направление и скорость ветра, другие параметры. Имея такие данные, можно рассчитать погоду в этих областях через некоторый промежуток времени. Чем чаще сетка и чем выше быстродействие компьютера, тем точнее результаты. Компьютеру требуется решать однотипную задачу для множества точек. Поэтому архитектура векторных суперкомпьютеров представляет собой матрицу из нескольких сотен процессоров, управляемых одним устройством управления и синхронно работающих по одной программе.

7.6. Перспективы развития компьютеров

Современные персональные компьютеры обладают *быстродействием* в сотни миллионов операций в секунду, оперативной памятью 64, 128, 256 Мбайт, жестким диском с *объемом памяти* 40, 60, 120 и более Гбайт; они имеют эффективные *внешние устройства*: высококачественные дисплеи, струйные и лазерные цветные принтеры, сканеры, модемы для обеспечения работы в сетях и другое оборудование.

Со временем, в перспективе *быстродействие* компьютеров будет все повышаться, а размеры их — уменьшаться. Возможно, однако, что повышение

быстродействия компьютеров начнет сдерживаться теоретическими и техническими ограничениями, а уменьшение размеров — удобством использования этих компьютеров.

Компьютеры-книги, *компьютеры-бумажники* будут подключены к глобальной сети и позволят практически мгновенно получать любую нужную информацию, от точного места нахождения владельца компьютера на земном шаре до температуры его тела и величины артериального давления.

Глобальную сеть можно будет сравнивать с *информационной магистралью*, позволяющей решать множество задач, не выходя из дома. Много научных, финансовых, учебных учреждений не будут иметь своих зданий с помещениями для размещения сотрудников и студентов, которые будут работать и обучаться дома. Связь с ними будет осуществляться через глобальную сеть.

Широкое внедрение микрокомпьютеров охватит все стороны жизни человека. В жизнь войдут управляемые голосом бытовые приборы и устройства, идентификация владельца по тембру голоса и отпечаткам пальцев при открывании дверей и проведении финансовых операций, синхронные переводы с иностранных языков и многое другое.

Системное программное обеспечение компьютеров будет ориентировано на дальнейшее приближение их к запросам пользователя, на простоту и удобство работы с ними. В системах Windows уже сейчас реализуются и будут развиваться: переход при управлении действиями от текста к графике, многооконность, стандартизация однотипных операций при работе, как с системой, так и с прикладными программами, внедрение принципа гипертекста в справочные системы и в работу с Web-страницами, поддержка работы с мультимедийными продуктами. Все это улучшает пользовательский интерфейс при одновременном снижении требований к квалификации пользователя.

Дальнейшее развитие системного обеспечения будет направлено на все более полное удовлетворение индивидуальных запросов пользователей. Пользовательский интерфейс будет подстраиваться, гибко реагировать на пожелания владельца компьютера. Все более активно будет использоваться ввод и вывод информации голосом. Пользователь будет задавать компьютеру вопросы, спорить с ним, давать задания. Компьютер будет следить за временем, напоминать пользователю о встречах, записывать содержание бесед и т. п.

Прикладное программное обеспечение уже в настоящее время удовлетворяет многим запросам пользователей. Например, уже последние версии текстового редактора Word фирмы Microsoft не только воспринимают многие подаваемые голосом команды форматирования, но и позволяют вводить голосом тексты. Спектр предлагаемых программ широк как по назначению, так и по уровню применения, охватывающему области от бытовых прикладных программ до профессиональных.

Все более активно внедряемое в жизнь *мультимедийное* оборудование и программное обеспечение будут расширять возможности и области применения компьютеров. Под технологиями мультимедиа понимается эффективное совместное использование высококачественных видеоизображений (в том числе телевизионных), графики, текста, анимации, различных видео- и звуковых эффектов, высококачественного звукового сопровождения.

Технологии мультимедиа будут находить все более широкое применение в различных обучающих, игровых, развлекательных и рекламных программах, в *системах виртуальной реальности*. Со временем даже стандартные компьютеры будут способны синтезировать достаточно реалистичные изображения и управлять ими. Используя технологии виртуальной реальности, пользователь сможет создать своего "виртуального двойника", придать ему желаемый внешний вид, речь, манеру поведения, подключить его к мощному банку данных и вести с ним беседы на интересующие его темы.

Глава 8

Программное обеспечение компьютеров

8.1. Общие сведения о программном обеспечении

Программным обеспечением компьютера называют совокупность имеющих отношение к компьютеру программных продуктов. Программное обеспечение делят на три категории: системное, прикладное и инструментальное.

Системное программное обеспечение — это комплекс системных программ, обеспечивающих эффективное функционирование компьютера и удобство пользователя по управлению его работой. Основа системного программного обеспечения — операционные системы.

К *прикладному программному обеспечению* относят программные продукты, специально разработанные для автоматизации и повышения эффективности работы в различных областях деятельности человека: экономике, науке, искусстве, бизнесе, обучении, развлечениях и других областях. Такие программные продукты называют приложениями.

Инструментальное программное обеспечение включает программные продукты, предназначенные для создания программ, относящихся к системному и прикладному программному обеспечению. В качестве таких инструментальных систем выступают системы программирования, языки программирования.

8.2. Системное программное обеспечение

К системному программному обеспечению относят:

- операционные системы: RT-11, ФОДОС, MS-DOS фирмы Microsoft, IBM OS/2, UNIX, Mac OS фирмы Apple, Windows NT, Windows 95, Windows 98, Windows 2000, Windows XP и др.;
- системные оболочки: Volkov Commander, Norton Commander, Windows 3.* и др.;

- вспомогательные системы: упаковщики, антивирусные программы, программы оптимизации хранения информации на дисках, другие программы.

Дисковая операционная система (DOS)

Операционной системой (ОС) называют набор программ, загружаемых с включением компьютера и управляющих его работой. ОС обеспечивает диалог пользователя с компьютером и эффективное управление его работой. Появление операционных систем можно считать первым шагом в развитии программного пользовательского интерфейса — системы программ, обеспечивающих удобное и эффективное общение пользователя с компьютером.

Операционная система RT-11 была разработана в 1972 году фирмой DEC для компьютеров PDP-11. В нашей стране на базе этой системы были разработаны ее версии (РАФОС, ОС ДВК, ФОДОС). Они использовались в персональных компьютерах ДВК (Диалоговых Вычислительных Комплексах) и школьных компьютерах УКНЦ и БК-0011.

Первая дисковая операционная система (DOS — Disk Operating System) была выпущена в 1981 году одновременно с появлением персонального компьютера IBM PC. До недавнего времени наиболее распространена была операционная система MS-DOS фирмы Microsoft. MS-DOS совершенствовалась, появлялись новые ее версии, обозначаемые номерами, например, MS-DOS 5.0, MS-DOS 6.22. В настоящее время наиболее широко используются операционные системы Windows.

Работа пользователей в DOS уходит в прошлое, как ушло программирование на машинных языках. Совершенствование пользовательского интерфейса сначала с появлением системных оболочек (Norton Commander, Windows 3.* и др.), затем — систем Windows существенно упростило общение пользователей с компьютером. Сейчас трудно предположить, что начинающий пользователь будет предпочитать работу в DOS, для которой необходимо помнить множество команд и правила их записи, работе в оболочках и, тем более, в Windows. Но это история информатики и вычислительной техники, а историю надо знать, чтобы иметь возможность прогнозировать развитие, заглядывать в будущее. Это понимают и составители многочисленных контролирующих тестов, включающих в них вопросы с элементарными сведениями о DOS

Загрузка и командный язык DOS

Операционные системы Windows содержат в своем составе и систему DOS версии 7.0 и выше. Эта система эмулируется (создается) операционной системой Windows и служит для поддержки ранних программных разработок.

Команда **Пуск | Программы | Стандартные | Сеанс MS-DOS** выводит на экран показанное на рис. 8.1 окно **Сеанс MS-DOS**, в котором имеется приглашение подавать команды DOS: C:\WINDOWS>. В некоторых из послед-

них версий Windows общение с DOS начинается с команды **Пуск | Программы | Стандартные | Командная строка**, которая выводит на экран окно **Командная строка**.

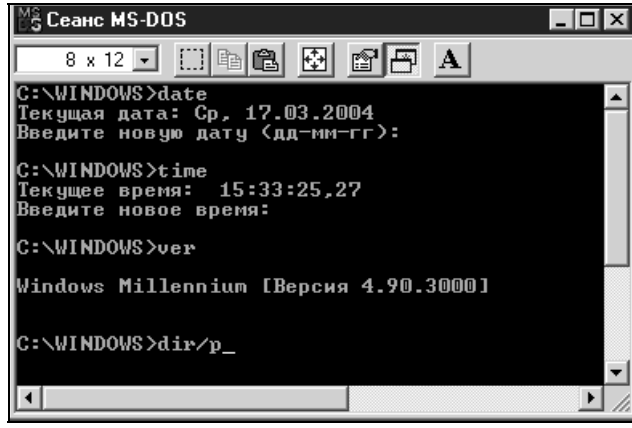


Рис. 8.1. Окно **Сеанс MS-DOS**

Рассмотрим некоторые из команд DOS. Всего их около сорока.

- По команде **date** компьютер сообщает дату, насчитанную к моменту запроса внутренними часами компьютера. Запись команды следующая:

  C: WINDOWS \ > date <Enter>

  На эту команду компьютер, например, отвечает:

  Текущая дата: Ср, 17.03.2004

  Введите новую дату (дд-мм-гг)

  При желании изменить значение даты следует ввести ее по предложенному формату.

- Проверить внутреннее время компьютера можно командой **time**:

  C: WINDOWS \ > time <Enter>

  Текущее время: 11:31:27,16

  Компьютер предлагает ввести новое время.

- Желая проверить версию операционной системы, нужно подать команду **ver**:

  C: \ WINDOWS > ver <Enter>

  Windows 98 [Версия 4.10.1998]

- Многие команды имеют многочисленные так называемые ключи. Каждый ключ определяет модификацию команды, т. е. вариант ее исполне-

ния. Например, у команды выдачи оглавления диска **dir** имеются ключи: **/w** — выдача оглавления диска в 5 колонок, **/p** — вывод оглавления поэкранно с переходом на продолжение вывода нажатием любой клавиши. Запись такой команды, например, с ключами p и w следующая:

C: \ WINDOWS > dir / p / w <Enter>

- Команда **edit** выводит на экран удобный текстовый редактор. Откройте редактор и убедитесь в его предельной простоте.
- Выход из DOS осуществляется командой **exit** (C:\WINDOWS>exit).

С другими командами DOS можно познакомиться в специальной литературе. Среди них имеются так называемые внешние команды DOS. Они обращаются к программам-утилитам, которые работают и в DOS, и в Windows.

Программы-утилиты

Утилиты — это вспомогательные программы, обеспечивающие выполнение множества важных, часто необходимых действий с устройствами и программами компьютера. К ним относятся, например, программы-упаковщики (сжатия информации), антивирусные программы, программы диагностики компьютера, программы оптимизации размещения информации на дисках и многие другие программы. Каждая из программ запускается соответствующими, как правило, одноименными командами. Многие утилиты имеют свои "оболочки" в виде диалоговых окон, обеспечивающих пользователю возможность выбирать режимы работы программы и демонстрацию этой работы. Утилиты обеспечивают выполнение внешних команд DOS. В системе Windows наиболее важные программы-утилиты можно вызвать из списка, открываемого командой **Пуск | Программы | Стандартные | Служебные**.

Если в окне **Сеанс MS-DOS** подать команду **fdisk**, то запустится одноименная программа-утилита разбиения винчестера на логические диски, форматирования этих дисков, подготовки их к использованию. Команда **scandisk** запускает программу-утилиту, которая проверяет диск и сообщает о его состоянии. Команда **defrag** запускает программу-утилиту, которая устраняет фрагментацию информации на диске.

Рассмотрим, например, функции программы **defrag**. Эта программа входит в состав служебных программ всех версий Windows. Файлы имеют разные размеры. В процессе работы некоторые из них удаляются, перемещаются. На их место записываются новые файлы — возможно, большего размера. В результате на диске могут образовываться пустые места, а некоторые файлы будут размещаться в нескольких участках диска, т. е. будут фрагментированы. Это приводит к замедлению работы компьютера при чтении и записи файлов ввиду излишних механических перемещений головок ввода/вывода жесткого диска. Программа **defrag** устраняет фрагментацию, перемещая файлы в компактном виде в начало диска. При оптимизации размещения

информации на диске на экран дисплея может выводиться карта диска, наглядно демонстрирующая процесс работы программы.

Следует отметить, что не все программы-утилиты способны работать в среде Windows. Запуская программу на исполнение, нужно твердо знать, что она не приведет к искажению или потере данных в компьютере.

Командный язык DOS позволяет работать с файлами, каталогами, вспомогательными программами. Он дает возможность вызывать на экран монитора нужную пользователю информацию о компьютере, его устройствах; работать с файлами, распечатывать их содержимое и т. п. Однако все эти действия связаны с необходимостью запоминать и безошибочно записывать команды значительной длины. Это затрудняло работу, занимало много времени. В помощь пользователю начали разрабатываться различные программы-оболочки, позволяющие упростить работу с файловой системой, сделать ее более наглядной.

Оболочки операционных систем

Для облегчения работы пользователей разрабатывались различного вида оболочки DOS. Одной из наиболее популярных программ-оболочек была разработанная фирмой Peter Norton Computing программа Norton Commander.

Norton Commander от DOS отличает большая наглядность отображения каталогов, простота освоения и обеспечиваемое ею удобство работы пользователя при выполнении различных действий. Это следующий шаг по совершенствованию программного пользовательского интерфейса и приближению компьютера к менее квалифицированному пользователю.

Если подать команду **Пуск | Программы | Norton Commander**, то можно познакомиться с окном программы Norton Commander для Windows, которое несколько отличается от окна аналогичной программы для DOS.

Окно состоит из двух панелей. Наличие двух панелей обеспечивает удобную работу с файловыми структурами одного диска или двух дисков. При подаче многих команд (например, на форматирование дискет или копирование файлов) открываются диалоговые окна, позволяющие задавать требуемые режимы выполнения этих операций. Очень просто выполняются архивация и разархивация файлов, удаление файлов, множество других операций.

С более полными возможностями программы по управлению информационными панелями и работе и файлами и каталогами можно познакомиться в справочной литературе и в справочной системе программы.

Следует все же заметить, что сохранение оболочки Norton Commander в системе Windows — это дань фирмы Microsoft пользователям со стажем, привыкшим с этой оболочкой работать. Все перечисленные операции и многие другие более эффективно выполняются в операционной системе Windows.

Графические оболочки

Важным шагом в совершенствовании пользовательского интерфейса явилось появление первых версий графических оболочек Windows — прообразов операционных систем Windows. Появление графического интерфейса GUI (Graphical User Interface) покорило неквалифицированных пользователей визуализацией объектов и операций, удобством работы с файловой системой.

Появление графических оболочек Windows определило разработку фирмой Microsoft системы программирования Бейсик третьего поколения — Visual Basic, ориентированной на создание Windows-приложений. За короткое время Visual Basic стал одним из наиболее востребованных языков программирования. Как и многие другие разработки фирмы Microsoft, его отличают мощность и простота освоения.

Операционные системы Windows

Следующим важным шагом в обеспечении максимального удобства работы пользователя с компьютером явилась разработка операционных систем Windows.

Новая идеология создания пользовательского интерфейса основана на многооконности, высокой визуализации и стандартизации элементов управления окнами, меню, справочниками. Широко используются такие принципы работы с окнами, кнопками, значками, меню, как "Drang-and-Drop" — "переместить и оставить", "Point-and-Click" — "указать и щелкнуть", "pull-down" — ниспадающие меню, "Plug and Play" — "вставь и играй" (точнее: "подключай и используй").

Первый вариант Windows был выпущен фирмой Microsoft в 1985 году как программа, управляющая отображением. Этим началась серия версий графических оболочек Windows 3.*x*, основанных на 16-разрядной DOS. Затем появились 32-разрядные операционные системы Windows. Разрабатывались новые программы, работающие в среде Windows. Среди них можно назвать текстовый редактор Word для Windows, табличный редактор Excel и базу данных Access. Последние версии Windows являются операционными системами, ориентированными на наиболее полное удовлетворение запросов пользователей Интернета.

Общие принципы построения и работы операционных систем Windows позволяют без затруднений переходить от работы с одними версиями к освоению других. Отличие состоит в некоторых нововведениях. Каждая последующая версия Windows даже при значительном расширении функций все ближе к неквалифицированному пользователю. Наблюдается существенное упрощение работы, сведение некоторых операций к привычным в быту действиям с реальными объектами. Например, папки можно открывать, закры-

вать, перемещать, убирать в портфель. Документы можно открывать, просматривать, отправлять в "мусорную корзину". Если это было сделано ошибочно, то можно вытащить документ из корзины и положить на прежнее место.

О работе с операционной системой Windows написаны книги, учебники. Но наиболее простой и эффективный метод освоения системы — это начать с ней работать.

Следует помнить, что пользователь подавляющую часть времени работает не ***с*** системой Windows, а ***в*** системе Windows. Поэтому изучение системы не самоцель, а необходимость быстро находить прикладные программы, выполнять их запуск, сохранять созданные документы и т. п. Начав работать с прикладными программами, пользователь постепенно в силу необходимости все глубже осваивает и работу с операционной системой. В этом помогают подробные встроенные справочники, которые позволяют без особых затруднений освоить работу с этими перспективными системами.

В *главах 1* и *2* можно было познакомиться с основными объектами системы Windows и правилами работы с ними. Остановимся подробнее на справочной системе.

Справочная система Windows

Успешная работа с множеством программ и приложений во многом зависит от знания справочной системы и умения ею пользоваться. Справочную систему можно вызвать командой **Пуск | Справка**. При вызове справки открывается окно **Справка Windows**. Вид окна в зависимости от версии операционной системы Windows может несколько варьироваться, но принципы работы со справкой в разных версиях если и различаются, то незначительно.

Даже первое знакомство со справочной системой раскрывает много полезных ее возможностей. Например, стало традицией включать в справочную систему краткие учебники и руководства по системе Windows. В них можно найти информацию по всем вопросам работы с Windows. Можно даже найти краткое, снабженное анимационными картинками руководство для начинающего пользователя по использованию мыши.

Справочная система, как правило, имеет три вкладки (раздела): **Содержание**, **Указатель** и **Поиск**. Для начинающих полезно на вкладке **Содержание** хотя бы поверхностно познакомиться с имеющимся справочным материалом, прочитав разделы **Использование справки**, **Поиск раздела**.

При активной работе с компьютером пользователю часто приходится выполнять множество различных операций с папками и документами. Этих операций — десятки. К ним относятся создание файлов и каталогов, их копирование, удаление, перемещение и т. п. Каждая операция требует определенной последовательности действий. Для начинающего желательно иметь

подсказки по выполнению этих действий. Можно, конечно, выделить в учебнике десяток страниц и расписать все операции по пунктам (один из вариантов выполнения некоторых операций дан в *гл. 2*). Но стоит ли копировать справочную систему Windows? К тому же при работе с компьютером справочная система всегда рядом, чего нельзя сказать об учебнике.

Откройте Справку Windows по команде **Пуск | Справка** и перейдите на вкладку **Указатель**. В текстовое поле ввода под названием **Введите ключевое слово для поиска** введите слово "файл". В нижнем окне появляется перечень наиболее важных действий с файлами. Выберите, например, **копирование**, щелкнув мышью на этой строке и на кнопке **Показать** (или сделав двойной щелчок на строке). В правое окно выводится подробная справка по копированию файла.

Просмотрите справки по копированию, поиску, сжатию, переименованию, сохранению, удалению, восстановлению после удаления, архивации, распаковке файлов и другим действиям с ними. Убедитесь, что имеются различные варианты подачи команд на выполнение этих действий. Один из удобных вариантов — использование правой кнопки мыши. Щелчок правой кнопки на имени файла или каталога раскрывает обширное меню действий с ними или получения полезной информации о них.

8.3. Общие сведения о прикладных программах

К прикладному программному обеспечению относят программы или так называемые приложения, предназначенные для автоматизации работы в различных областях человеческой деятельности. Перечислим некоторые из них и дадим им краткую характеристику.

Редакторы текстов

Редакторы текстов (текстовые редакторы) служат для создания различного вида текстовых документов: писем, статей, рефератов, отчетов, других работ. Использование для этой цели компьютеров дает практически неограниченные возможности по набору, редактированию, форматированию текстов, проверке орфографии, поиску синонимов, вставке графики и выполнению множества других операций.

Редакторы текстов различаются своими размерами, возможностями. Наибольшей популярностью пользуется мощный и удобный редактор Word фирмы Microsoft. Подробнее о нем будет рассказано ниже. Широко используются редакторы Лексикон, Multi Edit, Writer. Заметим, что простые текстовые редакторы содержатся во многих операционных системах и оболоч-

ках. В список стандартных программ системы Windows включаются текстовые редакторы WordPad и Блокнот.

Следует отметить, что начальный выбор текстового редактора — дело серьезное. К нему легко привыкнуть, а при переходе к другому, более мощному и удобному редактору невольно придется согласиться с тем, что переучиваться труднее, чем учиться. *Глава 9* знакомит с современным редактором Microsoft Word.

Электронные таблицы

Электронные таблицы используются для создания больших таблиц данных и работы с ними. Числовая и текстовая информация записывается в клетках таблицы, после чего с числовой информацией можно выполнять многочисленные операции поиска, сортировки, определения максимальных и средних значений, построения различного вида наглядных диаграмм. Если текстовые редакторы иногда называют текстовыми процессорами, то за электронными таблицами закрепилось название табличных процессоров.

К табличным процессорам относятся такие популярные программы, как Lotus 1-2-3 для DOS и для Windows, Quattro Pro для DOS и для Windows, Excel для Windows. О программе Excel для Windows будет рассказано ниже.

Базы данных

Системы управления базами данных (СУБД) служат для автоматизации создания в систематизированном виде больших информационных массивов и работы с ними по вводу, сортировке, компоновке, поиску, выводу данных, выполнению других операций.

Одной из первых популярных программ такого назначения считают dBase. С нею совместимы такие системы, как Fox Pro фирмы Microsoft для DOS и для Windows, язык программирования СУБД Clipper. Следует отметить СУБД Paradox также для DOS и Windows. В *гл. 11* рассматривается перспективная база данных Microsoft Access.

Интернет-программы

Множество программ обслуживают Интернет. В их числе — программы PowerPoint и Outlook. Первая предназначена для создания из различного рода данных экранных презентаций с помощью слайдов. Вторая программа — это программа управления рабочим временем и электронной почтой. К ним относится и программа создания Web-страниц FrontPage, сетевые приложения Internet Explorer, Netscape Communicator.

Графические редакторы

Графические редакторы применяются для создания, преобразования и компоновки рисунков на экране дисплея. В графических редакторах все графические средства языка Бейсик и многие другие возможности организованы в удобную для использования систему. Создаваемые и имеющиеся в библиотеке рисунки можно включать в другие документы — подготовленные, например, текстовыми редакторами или издательскими системами. Среди графических редакторов можно отметить Paint, Paintbrush, профессиональные редакторы CorelDRAW, Adobe Photoshop, Animator Pro, Photo Editor.

Кратко перечислим другие прикладные программы.

Интегрированные системы. Объединяют в себе возможности электронных таблиц, текстовых и графических редакторов, систем управления базами данных. К ним относятся Symphony, Frame Work, Open Access, Microsoft Works для DOS и для Windows, Мастер.

Системы автоматизации проектирования (САПР). Используются при проектировании машин, механизмов, строительных конструкций. Позволяют выполнять высококачественные чертежи, т. е. автоматизируют наиболее трудоемкую часть работ по проектированию. К САПР относится популярная программа AutoCad, программы ArchiCAD и CADdy.

Системы искусственного интеллекта. Это переводы с одного языка на другой, проверка орфографии, распознавания образов, экспертные системы, системы робототехники. К ним относятся программы ПРОМТ, Socrat, FineReader, МультиЛекс и др.

Бухгалтерские системы. Автоматизируют бухгалтерский труд. К бухгалтерским системам относятся Турбо-Бухгалтер, Бухгалтерия малого предприятия, многочисленные другие системы.

Антивирусные программы (Dr.Web), *Архиваторы* (Arj), *Музыкальные редакторы*, *Издательские системы*, *Обучающие системы* и многие другие прикладные программы.

Глава 9

Текстовый редактор Word

9.1. Начальное знакомство с приложениями

Овладение программами типа Word, Excel, Access, PowerPoint, Outlook и многими другими в настоящее время не только престижно. Это, почти без преувеличения, следующий уровень грамотности после чтения и письма. Учитывая большое разнообразие услуг, предоставляемых этими приложениями, трудно ожидать быстрого их освоения в полном объеме возможностей. Но следует учесть также, что по статистике 80% пользователей используют Word не более чем на 30%. И этого бывает вполне достаточно для выполнения большого ассортимента высококачественных работ с различного вида документами, насыщенными формулами, таблицами, рисунками, диаграммами и другими элементами.

Основные принципы построения и работы, основные объекты приложений от версии к версии почти не изменяются. Появляются некоторые дополнительные возможности, незначительно меняются и увеличиваются в числе меню, окна, панели инструментов, названия и изображения кнопок, и только. Поэтому знакомиться с основными принципами построения и работы прикладной программы можно на любой ее версии. Если же приложение изучается для дальнейшей работы с ним, то желательно выбрать последнюю версию. Следует отметить, что дополнительные возможности, которыми обрастают прикладные программы от версии к версии, усложняют их, делают все более профессиональными, что несколько затрудняет их освоение.

В помощь пользователю придаются мощные и удобные справочные системы, сопровождающие каждый программный продукт для системы Windows. Умелое пользование помощью — это залог успешного освоения элементарных начал работы с приложениями. Приложения Word, Excel и Access версий 7.0, 97 и 2000, работающие с операционной системой Windows 95 и выше, имеют унифицированные справочные системы, что позволяет быстрее осваивать работу с ними.

Обилие информации в справочной системе не должно озадачить начинающего пользователя. Способ освоения работы с приложением путем чтения и запоминания правил неприемлем, так как он занимает много времени и в итоге не всегда приводит к желаемому результату. Оптимальный путь к освоению приложения — начать с ним работать, при необходимости обращаясь за помощью к его справочной системе. При этом после освоения одного приложения знакомство с другими происходит намного быстрее. Опыт самостоятельного изучения прикладных программ трудно переоценить.

Если ранние версии приложений различались номерами (номер версии, номер модификации), то последние пакеты Microsoft Office и входящие в них приложения обозначают годом их появления.

9.2. Текстовый редактор Word

Запуск приложений и выход из них

Приложения Word, Excel и Access различных версий во многом унифицированы. Одинаковы многие элементы окон, панелей инструментов, названия кнопок и пунктов меню. Одинаков запуск программ приложений и завершение работы с ними. Одинаков принцип организации и работы справочной системы приложений. Все это позволяет несколько унифицировать и знакомство с ними.

Запустить приложения в Windows можно различными способами:

- последовательно выбрать мышью кнопки и пункты меню **Пуск**, **Программы**, **Microsoft Word** (**Microsoft Excel**, **Microsoft Access**);
- дважды щелкнуть мышью на ярлыке приложения на рабочем столе Windows;
- командами **Пуск | Документы** раскрыть список имен последних использованных документов и щелчком мыши на выбранном документе запустить приложение, в котором он был создан;
- кнопкой на панели инструментов Office.

Как и множество других операций, выход из приложения выполняется также многими способами:

- нажать комбинацию клавиш <Alt>+<F4>, что обеспечивает выход из любых Windows-программ;
- открыть в главном меню пункт **Файл** и подать команду **Выход** (нижний пункт меню **Файл**);
- дважды щелкнуть мышью на значке системного меню в левом верхнем углу окна или на кнопке **Закрыть** в правом верхнем углу окна.

Элементы окна Word

После запуска текстового редактора Word на поле рабочего стола выводится его окно (рис. 9.1). Многие из стандартных элементов окна знакомы по их описанию в *гл. 1, 2*. Это, например, заголовок в верхней строке. В начале заголовка расположен значок системного меню, в конце — кнопки **Свернуть**, **Восстановить/Развернуть** и **Закрыть**. Ниже заголовка — главное меню. Ниже главного меню расположены панели инструментов.

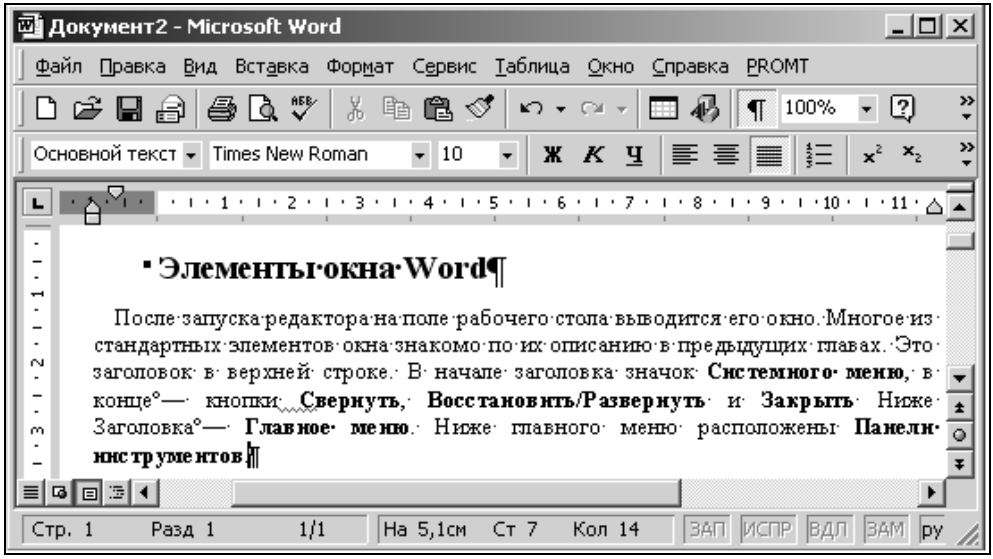


Рис. 9.1. Окно текстового редактора Word

В нижней части окна редактора расположена *строка состояния*. Над строкой состояния и в правой части окна — *полосы прокрутки*. В начале строки с нижней полосой прокрутки — кнопки *рабочих режимов просмотра документа*. Слева и сверху окна редактирования текста расположены *линейки* установки позиций табуляции.

Главное меню

Главное меню предназначено для подачи различных команд по работе с документами и элементами приложения. Пункты меню открываются щелчком мыши. Открывающееся краткое подменю можно расширить, щелкнув мышью на двойной стрелке внизу окна или немного задержавшись в нижней части поля окна. Некоторые пункты подменю имеют справа стрелки (шев-

роны). Щелкнув мышью на них, открываем подменю второго уровня. Закрыть подменю можно, щелкнув мышью вне его окна.

Панели инструментов

Панели инструментов состоят из кнопок, нажатие которых приводит к выполнению соответствующих команд. Панели можно настраивать, добавляя и убирая кнопки. Панели инструментов способны автоматически подстраиваться, убирая редко используемые кнопки.

Наводя указатель мыши на рисунки кнопок, ознакомьтесь с их назначением. Всплывающая подсказка появляется в прямоугольном окне на желтом фоне. Более подробную справку о назначении кнопки можно получить так. Откройте пункт главного меню **Справка**. Щелкните мышью на пункте меню **Что это такое?**. К стрелке мыши при этом добавится знак вопроса. При щелчке мышью по выбранной кнопке появится окно с расширенной справкой о ее назначении.

Большинство панелей можно вызывать по требованию пользователя. Их можно перемещать по окну приложения в места, где они не будут заслонять выполняемую работу, и закрывать нажатием кнопки **Закрыть**. Щелчок правой кнопкой мыши на панели инструментов вызовет окно со списком всех панелей приложения. Выберите, к примеру, панель **Рисование**. Подведите мышь к границе появившейся панели до появления двойной стрелки. Удерживая левую кнопку мыши, проведите мышь вправо и влево. При этом размеры окна будут изменяться. Поставьте стрелку мыши на заголовок окна и с нажатой левой кнопкой мыши переместите окно вправо вверх. Нажимая на шевроны, убедитесь в больших возможностях панели **Рисование**. Кнопкой **Закрыть** закройте панель.

Вызов и настройка панелей инструментов могут быть выполнены различными способами. Подайте команду **Вид | Панели инструментов | Настройка**. В окне **Настройка** три вкладки: **Панели инструментов**, **Команды** и **Параметры**. Выберите первую вкладку. Она позволяет отобразить или скрыть выбранную панель. Вызовите некоторые из панелей на экран. Затем закройте их.

Вкладка **Параметры** дает возможность, например, записать панель **Стандартная** и панель **Форматирование** в разных строках. Это разделение более привычно для пользователей, привыкших к ранним версиям редактора. К тому же оно позволяет вывести большее число кнопок. Для начинающих освоение приложения можно порекомендовать разделить эти две панели.

Если щелкнуть мышью на шевроне **Другие кнопки** в конце панели инструментов, появится окно с дополнительными кнопками или окно **Добавить**

или удалить кнопки, от которых по шеврону можно перейти к меню всех возможных кнопок панели. Добавьте, а затем удалите одну-две кнопки.

Строка состояния

Строка состояния расположена внизу окна редактора. Записи в ней имеют следующее значение (слева направо):

- положение текстового курсора в документе (например, запись "Стр 3 Разд 1" говорит о том, что курсор находится на странице 3 раздела 1, а запись "3/9" указывает, что курсор находится на странице 3, а всего в документе 9 страниц);
- положение текстового курсора на текущей странице документа (например, запись "на 14,1 см Ст 25 Кол 32" означает, что курсор находится в 14,1 сантиметрах от верхнего края листа, на строке 25 и в колонке 32 текста документа на этой странице).

Режимы просмотра документа

Режимы просмотра документа задаются кнопками, расположенными слева от полосы горизонтальной прокрутки. К ним относятся режимы (слева направо): обычный, разметки страницы, структуры, Web-документа. По умолчанию предлагается режим разметки страницы, в котором документ отображается на экране так, как он будет напечатан. Если кнопками режимов просмотра долго не пользоваться, то они могут быть спрятаны автоматически. В этом случае их можно найти среди пунктов меню **Вид**.

Справочная система

Всплывающие подсказки уже использовались нами для выяснения назначения кнопок панелей инструментов. Их появление можно задать или отменить командой **Настройка | Параметры | Отображать подсказки для кнопок**. Команда **Справка | Что это такое?** также использовалась для получения расширенной информации о кнопках.

Помощник

Помощник представляет собой анимационную картинку, обращение к которой дает возможность получать советы и обращаться с запросами к справочной системе. Если Помощник на экране отсутствует, вызвать его можно командой **Справка | Показать помощника**. При желании скрыть Помощника следует подать команду **Справка | Скрыть помощника**.

Щелчок мышью на Помощнике вызывает окно с предложением ввести в текстовое поле вопрос. После ввода термина, определяющего суть вопроса, следует нажать кнопку **Найти**. Помощник в своем окне предложит несколько разделов по заданному вопросу. Выбор темы открывает окно с текстом справки.

Вид Помощника можно выбрать в диалоговом окне **Помощник**. Для этого нужно щелкнуть на рисунке Помощника правой кнопкой мыши. В появившемся меню следует выбрать пункт **Параметры**, что приведет к раскрытию окна **Помощник**. Окно имеет две вкладки. Во вкладке **Коллекция** можно просмотреть и выбрать внешний вид Помощника. Вкладка **Параметры** служит для задания особенностей работы Помощника. В этой вкладке имеется флажок **Использовать помощника**. При желании отключить Помощника нужно снять этот флажок, убрав знак "птичка" щелчком мыши на нем.

Справка по Microsoft Word вызывается при отключенном Помощнике командой **Справка по Microsoft Word** меню **Справка**. В диалоговом окне Справки **Справка Microsoft Word** три вкладки: **Содержание**, **Мастер ответов** и **Указатель**. Окно разделено на две части. Левая часть служит для поиска справки, правая — для ее отображения. Для удобства работы с содержимым окон можно перераспределить их ширину. Это можно сделать, смещая границу между ними мышью с нажатой левой кнопкой.

Вкладка **Содержание** представляет собой "библиотеку книг" с информацией по всем вопросам работы с документами и приложением (рис. 9.2). Щелчок на знаке плюс перед книгой раскрывает ее разделы, главы, содержание.

Справочные тексты отмечены рисунком листа со знаком вопроса. Щелчок на знаке вопроса выводит текст справки в правую часть диалогового окна. Если слово или часть текста справки выделены синим цветом, то подведенный к ним указатель мыши принимает вид раскрытой ладони. Щелчок мышью выводит на экран дополнительную информацию по рассматриваемому вопросу.

Вкладка **Мастер ответов** предлагает ввести в текстовое поле **Выберите действие** искомый вопрос. Программа на этот вопрос отвечает предложением **Выберите раздел**. Текст справки по выбранному разделу отображается в правой части окна.

Вкладка **Указатель** раскрывает в левой части окна три текстовых поля. В первое поле предлагается ввести искомый термин. Во втором поле с каждой вводимой буквой информация меняется до совпадения введенных символов термина с ключевым словом указателя. После совпадения следует на-

жать кнопку **Найти**. В третьем текстовом поле предлагается на выбор несколько тем, содержащих информацию по заданному вопросу. Выбор темы выполняется щелчком мыши на ее названии. При этом в правую часть окна выводится текст справки.

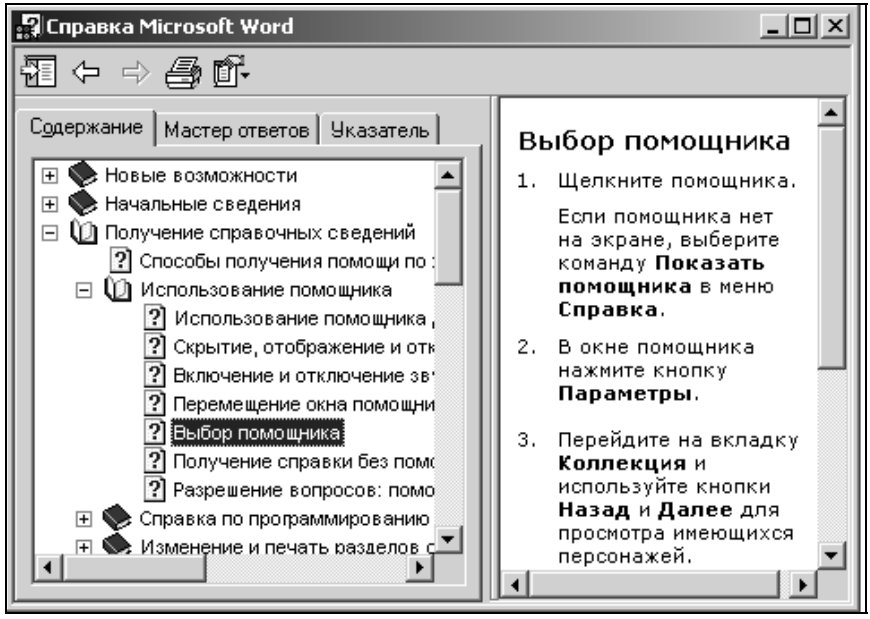


Рис. 9.2. Справочная система. Вкладка **Содержание** диалогового окна **Справка Microsoft Word**

Элементарные начала работы с текстом

Найдите на стандартной панели инструментов кнопку **Создать** (крайняя слева) и щелкните на ней мышью. В верхней строке окна появится заголовок — начальное имя текста, промежуточное до присвоения ему имени пользователем. Как правило, это имя Документ 1.

Рассмотрим некоторые приемы работы с документом при вводе его текста. Начнем с уяснения того, что переход на следующую строку при вводе текста выполняется автоматически, и только для завершения абзаца нужно нажать клавишу <Enter>.

Перемещения

Перемещения по документу выполняются различными способами.

Клавиши перемещения курсора дают возможность перемещаться построчно и посимвольно:

- <Home> — в начало строки текста;
- <End> — в конец строки;
- <Ctrl>+<Home> — в начало документа;
- <Ctrl>+<End> — в конец документа;
- <PgUp> — на одну экранную страницу вверх;
- <PgDn> — на одну экранную страницу вниз.

Удобно перемещаться по документу с помощью мыши и полос прокрутки. Вся длина полосы прокрутки примерно соответствует размеру документа. Поэтому движок полосы прокрутки указывает на положение расположенного на экране фрагмента в документе. В нижней части вертикальной полосы прокрутки находятся кнопки постраничного перемещения по документу — **Предыдущая страница** и **Следующая страница**. Там же находится кнопка **Выбор объекта перехода**.

Отмена действий

Начиная работу с редактором, невозможно избегнуть ошибок в действиях, командах. Среди кнопок стандартной панели инструментов имеется кнопка **Отменить ввод**[1], которая позволяет с каждым нажатием возвращаться на один шаг работы с документом назад, тем самым исправляя допущенные ошибки.

Удаление

Удаление символов, слов, фрагментов текста выполняется кнопками клавиатуры <Back Space> (другое ее обозначение стрелка <←>) — удаляется символ слева от курсора) и <Delete> — удаляется символ справа от курсора и выделенный фрагмент.

Замена символа

В ситуациях, когда возникает необходимость заменить несколько одиночных символов в разных местах документа, можно воспользоваться клавишей клавиатуры <Insert>. После ее нажатия вводимый с клавиатуры символ заменяет символ, стоящий справа от курсора. Режим замены индицируется на

[1] Названия всех кнопок видны во всплывающем желтом прямоугольнике при наведении курсора мыши на кнопку.

строке состояния в нижней части окна. Отказ от режима замены — повторное нажатие клавиши <Insert>.

Буфер обмена

Если имеются сомнения в целесообразности удаления, то фрагмент можно отправить в буфер обмена для временного хранения. Для этого после выделения фрагмента (о выделении будет рассказано ниже) требуется щелкнуть по кнопке **Вырезать** (с изображением ножниц) стандартной панели инструментов или нажать комбинацию клавиш <Shift>+<Delete>. Из буфера фрагмент можно многократно скопировать в место положения курсора щелчком по кнопке **Вставить** или нажатием на комбинацию <Shift>+<Insert>.

Отправить фрагмент текста в буфер обмена можно и без его удаления. Для этого нужно выделить фрагмент и щелкнуть по кнопке **Копировать** или нажать комбинацию клавиш <Ctrl>+<Insert>. После этого опять появляется возможность многократного копирования фрагмента из буфера в места положения курсора.

Смена языка

Переход от латинского шрифта к русскому выполняется при одновременном нажатии клавиш <Shift>+<Ctrl> или нажатием кнопки в правом нижнем углу рабочего стола (на лотке в правой части панели задач, которая остается доступной во время работы с любым приложением Windows). Возможны другие варианты смены языка — они зависят от применяемой программы-русификатора.

Непечатаемые знаки

Ввод непечатаемых знаков (символов, не видных на печати) можно выполнять при нажатой кнопке **Непечатаемые знаки** стандартной панели инструментов (кнопка со знаком ¶). При этом на экране точками заполняются пробелы между словами, а символ ¶ будет завершать абзацы, организуемые нажатием клавиши <Enter>. Заметим, что при наборе текста нежелательно ставить более одного пробела подряд.

Ввод текста

Перед вводом текста выберите в главном меню команду **Вид | Разметка страницы**. На панели инструментов **Форматирование** с помощью кнопок **Стиль**, **Шрифт** и **Размер**, раскрывающих списки вариантов выбора, выберите стиль **Обычный** и шрифт **Times New Roman** размером в **12** пунктов. Затем введите приведенный ниже текст "Золотое сечение".

В результате получим на экране следующий текст:

Золотое сечение.

Золотым сечением называют такое деление отрезка на две неравные части, при котором отношение всего отрезка к большей его части равно отношению большей части к меньшей. Это отношение равно 1,618. Иногда рассматривают обратное отношение — отношение меньшей части к большей. Оно равно 1/1,618 = 0,618.

Это знаменитое отношение известно человеку многие тысячелетия и является фундаментальной мировой константой. Удивительные математические свойства "золотой пропорции" создали вокруг нее ореол таинственности и мистического поклонения.

Золотые пропорции находят в египетских пирамидах, эллинских храмах. Стремясь к гармонии, удобству, рациональности, поиску новых чудесных свойств, к золотому сечению обращаются специалисты практически всех наук и искусств.

Удивительные тайны открывает природа перед любознательными и настойчивыми.

Приведем введенный текст в относительный порядок.

Выделение текста

Множество операций над текстом и его частями начинаются с выделения. После выделения текст можно, например, дать курсивом, подчеркнуть, набрать другим шрифтом, сохранить в буфер, скопировать, удалить и т. п. Снять выделение можно, щелкнув мышью в любом месте окна редактирования. Способов выделения много:

- выделить часть текста можно, проведя по нему мышью с нажатой левой кнопкой;
- слово выделяется двойным щелчком по нему мышью;
- предложение выделяется щелчком на нем мыши при нажатой клавише <Ctrl>;
- строка текста выделяется щелчком мыши напротив выбранной строки на полосе выделения (свободном поле слева от строки); если при этом нажать клавишу <Shift>, то выделение выполнится до положения курсора;
- абзац выделяется тройным щелчком мыши в любом месте абзаца или двойным щелчком на полосе выделения;

- ❒ весь текст выделяется тройным щелчком мыши на полосе выделения или одинарным щелчком при нажатой клавише <Ctrl>;
- ❒ выделить элементы текста можно также при нажатой клавише <Shift> клавишами перемещения курсора.

Выделение можно выполнять с помощью клавиши <F8>. Нажатие этой клавиши активизирует режим расширения выделения (слово, предложение, абзац, документ). При этом в строке состояния (внизу окна приложения) активизируется кнопка **ВДЛ**. Снять режим расширения можно двойным щелчком мыши на кнопке **ВДЛ**.

Перетаскивание мышью

Выделенные фрагменты документов можно перемещать методом Drang & Drop (перетащить и положить). Для этого нужно поставить указатель мыши на выделенный фрагмент и при нажатой левой кнопке перетащить его в выбранное место. При этом указатель мыши дополняется прямоугольником. Фрагмент вставляется в место, отмечаемое пунктирной вертикальной линией, возникающей при подведении указателя мыши к этому месту. Форматирование текста

Выделите введенный текст. Откройте подменю **Формат** главного меню. Выберите в нем пункт **Абзац**. В открывшемся диалоговом окне **Абзац** перейдите на вкладку **Отступы и интервалы**. Выберите в списке **Выравнивание** пункт **по ширине**, а в списке **первая строка — Отступ** и с помощью поля со списком значений **на** установите значение отступа 0,5 см. После установки нажмите кнопку **ОК**.

Не снимая выделения, выпролните команду **Сервис | Язык | Расстановка переносов**. Установите в открывшемся диалоговом окне флажок **Автоматическая расстановка переносов** и нажмите **ОК**. После этого снимите выделение. Текст принимает следующий вид:

Золотое сечение

Золотым сечением называют такое деление отрезка на две неравные части, при котором отношение всего отрезка к большей его части равно отношению большей части к меньшей. Это отношение равно 1,618. Иногда рассматривают обратное отношение — отношение меньшей части к большей. Оно равно 1/1,618=0,618.

Это знаменитое отношение известно человеку многие тысячелетия и является фундаментальной мировой константой. Удивительные математические свойства "золотой пропорции"

создали вокруг нее ореол таинственности и мистического поклонения.

Золотые пропорции находят в египетских пирамидах, эллинских храмах. Стремясь к гармонии, удобству, рациональности, поиску новых чудесных свойств, к золотому сечению обращаются специалисты практически всех наук и искусств.

Удивительные тайны открывает природа перед любознательными и настойчивыми.

Проверка правописания

При вводе текста обращали на себя внимание подчеркнутые красными и зелеными волнистыми линиями слова и предложения. Таким образом редактор контролирует правописание каждого слова, расстановку переносов и т. д. Если слово подчеркнуто красным, но при этом правильно написано, то, возможно, его нет в словаре редактора. Снять выделение можно, щелкнув на подчеркнутом слове правой кнопкой и выбрав в открывшемся окне команду **Пропустить все**.

В окне, вызываемом правой кнопкой, предлагаются для замены ошибочного слова правильно написанные варианты. При желании заменить слово правильно написанным словом из предложенного списка нужно щелкнуть на нем мышью.

Зеленые волнистые подчеркивания указывают на то, что редактор сомневается в соблюдении грамматических правил и предлагает проверить, например, расстановку знаков препинания.

Продолжая работать с текстом и желая оценить работу редактора при проверке орфографии, введите в текст одно-два ошибочных слова, например "отнашение" и "искуств". Откройте командой **Сервис | Правописание** диалоговое окно **Правописание**. Проверка орфографии начнется с того места в тексте, где находится курсор, или с выделенного фрагмента. Редактор найдет ошибочные слова и предложит заменить их на правильно написанные; проверит расстановку знаков препинания и согласование слов; даст советы по улучшению текста. Команда **Сервис | Язык | Тезаурус** поможет подобрать синонимы.

Можно поработать и с заголовком текста. Выделите его, запишите полужирным шрифтом размером в 14 пунктов и нажмите на кнопку выравнивание **По центру** панели **Форматирование**. В результате получите следующий заголовок:

Золотое сечение

Правая кнопка мыши

Команды, в соответствии с которыми были выполнены некоторые из преобразований текста, можно подавать, используя контекстное меню, предлагаемое при нажатии правой кнопки мыши. Поставив курсор мыши на одно из слов текста, нажмите ее правую кнопку. Появляется меню со списком возможностей. Выберите, например, пункт **Синонимы**. Редактор предложит несколько синонимов слова, на котором установлен курсор.

Вставка символов и формул

При работе с текстом часто возникает необходимость вставить нестандартные, отсутствующие на клавиатуре символы. Командой **Вставка | Символ** откройте диалоговое окно **Символ** с двумя вкладками, содержащими каждая большой набор символов. Чтобы вставить выбранный символ в текст, нужно выделить его щелчком мыши и нажать кнопку **Вставить**. При этом выбранный символ вставляется в место положения курсора.

Простые формулы можно набирать с помощью клавиатуры, не прибегая к специальным средствам. Например, в формуле $f(x,y) = 2\sin^2 5x^3$ или в формуле $a_{(2)} = 1011$ требуется только знать, как поднять показатель степени (<Shift>+<Ctrl>+<=>) и опустить индекс (<Ctrl>+<=>). Возврат в исходное положение осуществляется повторным нажатием на соответствующую комбинацию клавиш.

При активной работе со степенями и индексами можно на панель форматирования вывести дополнительные кнопки **Верхний индекс** (x^2) и **Нижний индекс** (x_2). Для этого нужно щелкнуть по шеврону на панели форматирования и активизировать соответствующие пункты меню.

Для набора сложных, многоуровневых формул, содержащих специальные символы, требуется использовать пункт **Объект** меню **Вставка**. Предварительно нужно курсор поставить в то место текста, куда должна быть вписана формула. В раскрывшемся на вкладке **Создание** диалоговом окне **Вставка объекта**, в списке **Тип объекта** выберите пункт **Microsoft Equation 3.0**. Ознакомьтесь с большими возможностями вызванной программы. С ее помощью не составляет труда записать, например, такие формулы:

$$f(x,y) = 0{,}5ax^3 + \frac{\sqrt{1 + \frac{|\cos^3 by|}{ab}}}{3ab}; \qquad F(A,B,C) = \overline{A} \vee B\overline{C} \vee A\overline{B}C.$$

Для тренировки наберите эти формулы. Полезно знать, что для внесения исправлений в формулу, созданную с помощью программы Equation 3.0, достаточно щелкнуть на формуле мышью.

Работа с таблицами

В набираемый текст можно вставить таблицу. Простую таблицу можно создать, щелкнув на кнопке **Добавить таблицу** стандартной панели инструментов. Предварительно нужно поместить курсор в место будущего расположения таблицы. В появившейся сетке следует выделить мышью нужное количество строк и столбцов и щелкнуть на сетке мышью.

Таблица может быть также создана командой главного меню **Таблица | Нарисовать таблицу** или командой **Таблица | Добавить | Таблица**. В первом случае на экран выводится окно с инструментами для создания таблицы и работы с ней. Во втором случае появляется окно **Вставка таблицы**, где запрашиваются характеристики таблицы.

После ввода требуемых значений на экране получаем готовый для заполнения макет таблицы (для простоты выбрана таблица (табл. 9.1), размером 3×3).

Таблица 9.1. *Макет таблицы*

Удалить таблицу можно командой **Таблица | Удалить | Таблица**. Перед этим таблицу нужно выделить. Выделить таблицу можно двумя щелчками мыши. Первый — по таблице, второй — по появившемуся рядом с таблицей значку (крестик в квадратике)[2].

Перемещение по клеткам таблицы выполняется клавишами передвижения курсора и клавишей <Tab>. Поместив курсор в любую клетку таблицы, снова откройте подменю **Таблица**. Меню изменилось. Появились возможности выделения, добавления, удаления элементов таблицы, сортировки данных по возрастанию, убыванию, возможности выполнять другие действия.

После заполнения таблицы командой **Таблица | Автоформат** откройте диалоговое окно **Автоформат**. В нем предлагается (с демонстрацией) множество вариантов обрамления таблицы. Выбрав вариант **Сетка 5**, получим, например, таблицу, показанную ниже (табл. 9.2).

---

[2] Этот способ выделения работает только в том случае, если для документа установлен вид **Разметка страницы**.

Таблица 9.2. *Вариант оформления таблицы*

Время года	$\mathbf{T_{max}}$	$\mathbf{T_{min}}$
Зима	2	−37
Лето	31	12

Столбцы таблицы можно сужать и расширять, подводя к ним указатель мыши и перемещая при нажатой левой кнопке границу столбца в новое место. Командой **Таблица | Автоподбор** открывается диалоговое окно с командами форматирования размеров столбцов и строк.

В таблицу легко добавлять строки и столбцы. Команда **Таблица | Добавить** раскрывает меню с командами добавления. Предварительно требуется выделить столбец или строку таблицы, рядом с которыми желательно выполнить вставку.

Выделение элементов таблицы выполняется обычным образом. Столбцы можно выделять, подводя сверху курсор мыши к столбцу до появления черной стрелки и нажимая на кнопку мыши. Для выделения ячеек таблицы мышь нужно подводить к ним слева. Выделив всю таблицу, можно ее удалить, отправить в буфер, скопировать в новое место.

С содержимым таблицы можно выполнять все операции форматирования текстов. Можно выбирать шрифт, выбирать расположение текста в столбце (слева, справа, по центру) и т. п. Кроме того, в меню **Таблица** имеется пункт **Сортировка**, позволяющий расположить информацию в таблице в алфавитном или в порядке возрастания или убывания значений.

Сортировать можно не только содержимое таблиц, но и списки. Если записать в столбец несколько слов или чисел, выделить их и подать команду **Таблица | Сортировка**, то откроется окно, в котором достаточно будет указать, что и как нужно сортировать, и нажать кнопку **OK**.

Для перемещения таблицы по документу нужно щелкнуть на ней мышью. При этом около левого верхнего угла таблицы появляется квадратик с крестиком внутри.

Установив указатель мыши на этот значок, при нажатой левой кнопке таблицу можно переместить в выбранное место.

При желании оформить таблицу так, чтобы текст обтекал ее, нужно выделить таблицу и выбрать в меню **Таблица** пункт **Свойства таблицы**. Затем в появившемся одноименном окне перейти на вкладку **Таблица** и в ней выбрать вариант обтекания таблицы текстом.

Вставка рисунков

Редактор Word позволяет вставлять в текст готовые рисунки из библиотеки рисунков, вставлять рисунки, созданные с помощью других программ, и рисовать простые рисунки с помощью собственных средств редактора.

Чтобы вставить рисунок, выполните команду **Вставка | Рисунок | Картинки** или на панели инструментов **Рисование** щелкните на кнопке **Добавить картинку**. В открывшемся диалоговом окне **Вставка картинки** выберите рисунок и вставьте его в документ. За справками обратитесь к справочной системе.

В каталогах Clip Callery, Clipart и других обычно находятся десятки различных рисунков, которые можно просматривать на демонстрационном экране и вставлять в место расположения курсора простым нажатием кнопки **OK** или через буфер обмена. Так было сделано с рисунком, показанным на рис. 9.3.

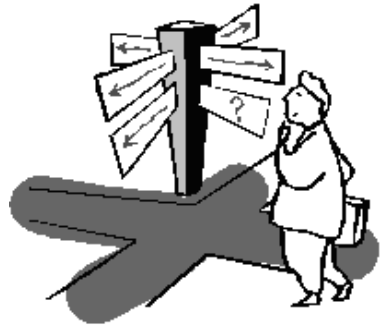


Рис. 9.3. Рисунок каталога Clipart

Для обтекания рисунка текстом следует выделить рисунок щелчком мыши и выполнить команду **Формат | Рисунок**. В раскрывшемся диалоговом окне **Формат рисунка** можно перейти на вкладку **Положение**, в которой выбрать вид обтекания рисунка текстом. На вкладке **Положение** имеется кнопка **Дополнительно**, нажатие которой открывает окно **Дополнительная разметка**. Если в этом окне перейти на вкладку **Обтекание текстом**, то можно установить не только вид обтекания, но и расстояния в сантиметрах от рисунка до текста.

При выделении рисунка на экран выводится панель инструментов **Настройка изображения** с кнопками, позволяющими выполнять с рисунком различные действия. Панель можно вывести щелчком правой кнопки мыши на стандартной панели инструментов с последующим выбором строки **Настройка изображения**. Среди кнопок панели имеется кнопка **Формат рисун-**

ка, которую также можно использовать для выполнения рассмотренной выше операции.

Рисование

Панель инструментов для рисования открывается нажатием кнопки **Рисование** на панели **Стандартная**. Панель обычно размещается внизу, под окном редактирования. Все рисованные элементы прикрепляются к сетке, у которой шаги по горизонтали и вертикали задаются в окне **Привязка к сетке**, открываемом командой **Действия | Сетка**.

Для изображения линии нужно щелкнуть на кнопке **Линия**, подвести мышью крестик к месту начала линии, нажать на кнопку мыши и провести ее к концу линии. При необходимости нарисовать несколько линий или других элементов подряд следует дважды щелкнуть на соответствующей кнопке. После проведения линии по ее концам остаются отметки выделения (квадратики). При их наличии можно линию удалить клавишей <Delete>, перемещать линию мышью при нажатой левой кнопке, перемещать клавишами перемещения курсора с шагом, соответствующим интервалам сетки, а также изменять ее размеры. Аналогично можно поступать с другими рисованными объектами.

Если на панели **Рисование** нажать кнопку с белой стрелкой **Выбор объектов**, то линию можно будет поворачивать и трансформировать ее размеры. Это же относится к другим рисованным элементам. Выделить несколько элементов рисунка можно щелчками мыши на этих элементах при нажатой клавише <Shift>. Выделение удобно выполнять при нажатой кнопке **Выбор объектов**.

Нажатая клавиша <Shift> позволяет выполнять более строгое рисование элементов. При этом линии рисуются под углами с некоторой дискретностью, прямоугольники — как квадраты, овалы — как окружности.

Выделив линию, нажмите кнопку **Свободное вращение**. Квадратики выделения при этом превратились в закрашенные зеленым цветом круги. Если подвести курсор мыши к одному из концов линии и с нажатой левой кнопкой перемещать его, то линия будет вращаться.

Элементы рисунков можно изображать линиями различной толщины, линиями со стрелками, пунктирами. При использовании цвета можно выбирать цвет рисованных деталей и цвет заливки (закрашивания) рисованных фигур. Имеются большие наборы автофигур. Кнопки панели дают возможность создавать тени (кнопка **Тень**), объемов (кнопка **Объем**), делать надписи (кнопка **Надпись**) и т. д. Познакомьтесь с этими возможностями самостоятельно.

Дополнительные возможности редактора

Команда **Правка | Найти** раскрывает диалоговое окно, позволяющее записать слово, которое нужно в тексте найти или найти и заменить на другое записываемое в окне слово. Вызвать это окно можно кнопкой **Найти** (в виде рисунка бинокля) на стандартной панели инструментов.

Имеется возможность снабжать многостраничные документы закладками. С помощью команды **Найти** закладки позволяют быстро находить в документе нужные места.

Команда главного меню **Окно | Разделить** делит рабочее поле редактора на две части, в которых можно раздельно просматривать, сравнивать и редактировать различные части одного документа. Команда **Окно | Упорядочить все** позволяет вывести на экран два и более различных документов и поочередно работать с ними.

Редактор предоставляет пользователю большие возможности по оформлению текста различными видами отступов, интервалов, рамок, буквиц и т. п. Имеется большое количество различных стилей оформления документов.

Программа редактора позволяет организовывать в пределах документа гиперссылки. Редактор имеет средства для создания макросов. Этот вопрос будет рассмотрен в *гл. 16*.

Важным достоинством редактора Word 2000 можно считать обеспечение пользователя эффективными средствами создания и публикации Web-документов. В помощь начинающим редактор предоставляет Мастера Web-страниц. Для его вызова достаточно подать команду **Файл | Создать** и в открывшемся окне **Создание документа** перейти на вкладку **Web-страницы** и щелкнуть мышью на значке **Web-мастер**.

Работа с файлами

Сохранение документов

Сохранение вновь созданного документа в памяти выполняется командой **Сохранить как** меню **Файл**. При этом открывается диалоговое окно, в котором следует выбрать место, куда нужно поместить документ, и записать имя файла, в котором документ будет сохранен (имя документа). При сохранении документа, уже имеющего имя, достаточно в меню **Файл** выбрать пункт **Сохранить** или нажать кнопку **Сохранить**.

Автосохранение

При отсутствии надежного, бесперебойного питания компьютера, с целью исключить возможность утраты наработанного материала следует включать автосохранение документа через равные, задаваемые пользователем промежутки времени. Для этого нужно подать команду **Файл | Сохранить как |**

Сервис | Параметры, в окне **Сохранение** установить флажок **Автосохранение каждые** и задать количество минут. Это же можно сделать командами **Сервис | Параметры | Сохранение**.

Вызов документа из памяти

Открыть документ для просмотра, редактирования, печати или других действий можно двумя способами: нажать на кнопку **Открыть** стандартной панели инструментов или активизировать пункт **Файл | Открыть** главного меню. В диалоговом окне следует отыскать или записать имя открываемого файла и вызвать его в окно редактирования нажатием кнопки **OK** или двойным щелчком мыши на его имени.

В меню **Файл** имеется список имен четырех последних документов, которые вызывались редактором. Щелчок мышью на имени вызывает документ в окно редактирования.

Печать документа

Перед печатью документ желательно просмотреть, нажав на кнопку **Предварительный просмотр** стандартной панели инструментов. При этом документ отображается на экране в том виде, в каком он будет отпечатан. Окно просмотра имеет свою панель инструментов, позволяющих обеспечить эффективность просмотра и подготовки документа к печати.

Команду печати документа можно подать разными способами. Во-первых, это делается нажатием кнопки **Печать** стандартной панели инструментов. Во-вторых, по команде **Файл | Печать**. В-третьих — комбинацией клавиш <Ctrl>+<P>. Последние две команды открывают диалоговое окно с параметрами печати, которые требуется выбрать.

Перед подачей команды печати следует подготовить принтер — включить его, вставить бумагу.

Глава 10

Электронные таблицы Excel

10.1. Общие сведения

Программа Excel предназначена для работы с числовыми данными, автоматизации расчетов, организации учета и различного вида отчетности.

Excel дает возможность системно и упорядоченно сосредоточить в едином комплексе колоссальные объемы числовой и текстовой информации; позволяет сортировать и обрабатывать эту информацию, организовывать поиск нужной информации, обобщать и наглядно отображать зависимости элементов этой информации.

Широкий набор средств форматирования, богатый выбор шрифтов, возможность вставки текстов и рисунков, выполненных другими программами, и многое другое отвечают самым высоким запросам пользователей в области профессионального оформления таблиц.

Приложение Excel можно запустить, подавая последовательно команды **Пуск | Программы | Microsoft Excel**. Другие варианты запуска приложений Office рассмотрены в *гл. 9*.

Элементы окна Excel

Увидев окно Excel (рис. 10.1), обнаруживаем, что большинство его элементов знакомо по описаниям окна Windows и редактора Word. Сказывается стандартизация — основа идеологии Microsoft. Имеются и новые понятия: рабочие книги и рабочие листы. *Рабочие книги* — это файлы Excel. Они могут состоять из нескольких рабочих листов. *Рабочие листы* — это сами таблицы, диаграммы и модули Visual Basic.

Документ Excel (рабочая книга) состотт из одного или нескольких рабочих листов. Листы представляют собой *таблицы*, состоящие из *строк* и *столбцов*, пересечения которых образуют *ячейки*. В ячейки записываются данные в виде текста, чисел и формул. Каждая ячейка имеет свой *адрес*, состоящий из имени рабочего листа, номера строки и номера столбца.

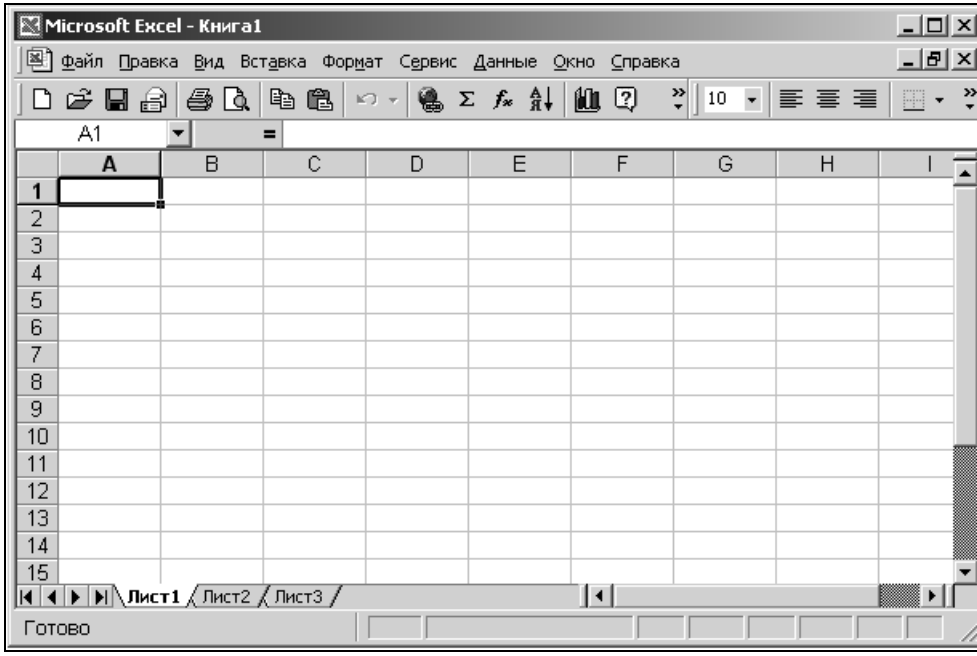


Рис. 10.1. Окно программы Excel

Верхняя строка окна — заголовок, например, **Microsoft Excel — Книга 1**. Имена `Книга 1`, `Книга 2`, ... программа Excel присваивает рабочим книгам автоматически. При записи таблицы в память компьютера пользователь присваивает им свои имена.

В начале строки заголовка находится значок-кнопка системного меню. Щелкнув на кнопке мышью, раскрываем меню и знакомимся с его командами. Среди них команда **Закрыть**, по которой выполняется выход из Excel. Выйти из программы можно также двойным щелчком мыши на кнопке системного меню или на кнопке **Закрыть** в правой части заголовка, или командой **Файл | Выход**.

Ниже заголовка расположено главное меню, рассмотренное при знакомстве с окном редактора Word. Ниже главного меню расположены панели инструментов.

Ниже строки панелей инструментов находится строка, в которой слева расположено *поле имени*, где отображаются координаты выбранных ячеек или блоков ячеек таблицы. Если в поле имени записать координаты ячейки, то эта ячейка станет выбранной и окажется в рабочем окне. Правая часть этой

строки — это *строка формул*. В этой строке дублируется все, что записывается в выбранную ячейку. Она же является строкой редактирования.

Вся рабочая область окна занята чистыми ячейками рабочего листа — таблицей. Столбцы озаглавлены буквами, строки — числами. Всего на рабочей странице 256 столбцов и 16 384 строки.

Таблица сверху и слева обрамлена соответственно *заголовками столбцов* (A, B, C, D, ...) и *заголовками строк* (1, 2, 3, 4, ...).

Справа и внизу таблицы расположены полосы прокрутки. Щелчками мыши на стрелках по концам полос можно построчно и по колонкам перемещаться по рабочему листу. Захватив бегунок мышью, можно перемещаться быстрее. Бегунок полос прокрутки к тому же выполняет функции индикатора расположения видимого рабочего поля в таблице.

Окно с заголовком, например, Книга 1 может состоять из нескольких рабочих листов (таблиц, диаграмм). При открытии книги загружаются все ее рабочие листы. Их *ярлычки* расположены ниже таблицы: **Лист 1**, **Лист 2**, и т. д. Рабочим листам также как и книгам можно дать более выразительные имена. Для этого нужно щелкнуть правой кнопкой мыши на ярлычке листа и выбрать команду **Переименовать**.

Ярлычки рабочих листов имеют свои кнопки прокрутки. Они расположены слева от ярлычков. При отсутствии ярлычков в окне их можно ввести командой **Сервис | Параметры | Вид | Ярлычки листов**.

Строка состояния в нижней части программы отображает сведения о выполняемой операции. В левой части строки сообщение **Готово** говорит, что программа готова к работе и ожидает ввода новых команд. В правой части строки состояния отображается состояние клавиш <Num Lock>, <Scroll Lock> и <Caps Lock>.

Панели инструментов

Щелчок правой кнопкой мыши на стандартной панели инструментов открывает окно со списком панелей. Выберите нижнюю строку. При этом открывается диалоговое окно с тремя вкладками: **Панели инструментов**, **Команды** и **Параметры**. Откройте вкладку **Панели инструментов** и оставьте выбранными только две панели — **Стандартная** и **Форматирование**. С остальными панелями будем знакомиться при необходимости их использования. Последовательно подводя стрелку мыши к кнопкам панелей инструментов, можно ознакомиться с их назначением. Функции кнопок появляются в квадратике рядом с кнопкой. Для более подробной информации о назначении кнопок можно использовать команду **Справка | Что это такое?**.

Справочная система

О справочной системе приложений и работе с ними было кратко рассказано в *гл. 9*. Ознакомьтесь со справочной системой Microsoft Excel. Просмотрите все три вкладки: **Содержание**, **Мастер ответов** и **Указатель**. Остановитесь на вкладке **Содержание**. Если в этой вкладке последовательно открыть несколько книг, то можно убедиться в обширности и качестве справочного материала. Напомним, что умелое пользование справкой — это залог успешного освоения работы с программой. При желании работать с Помощником вызовите его командой **Справка | Показать помощника**.

10.2. Простейшие действия с таблицей

Перемещения и выделения

Перемещение по рабочей книге состоит в выборе рабочих листов. Рабочим листам соответствуют ярлычки листов, которые отображаются в нижней части окна. Щелчок мышью на ярлычке вызывает лист на рабочее поле книги. В примере, который ниже будет предложено выполнить, рабочая книга состоит из одного рабочего листа.

Для выбора ячейки рабочего листа достаточно щелкнуть на ней мышью. При этом ячейка становится активной и с ней можно выполнять различные действия: запись, удаление, редактирование, копирование, перемещение информации и т. п. Имя активной ячейки запишется в поле имени.

Строка или столбец выделяются щелчком мыши на их именах. При этом все выделенные ячейки, кроме активной, закрашиваются синим цветом. В поле имени записывается имя активной ячейки.

Блок ячеек выбирается щелчком мыши на крайней ячейке области и протаскиванием мыши с нажатой кнопкой до противоположной крайней ячейки блока.

Вся таблица выделяется щелчком мыши на кнопке в левом верхнем углу рабочего листа (на пересечении номеров столбцов и номеров строк).

Для выделения нескольких не связанных между собой фрагментов таблицы можно пользоваться клавишами <Shift> и <Ctrl>.

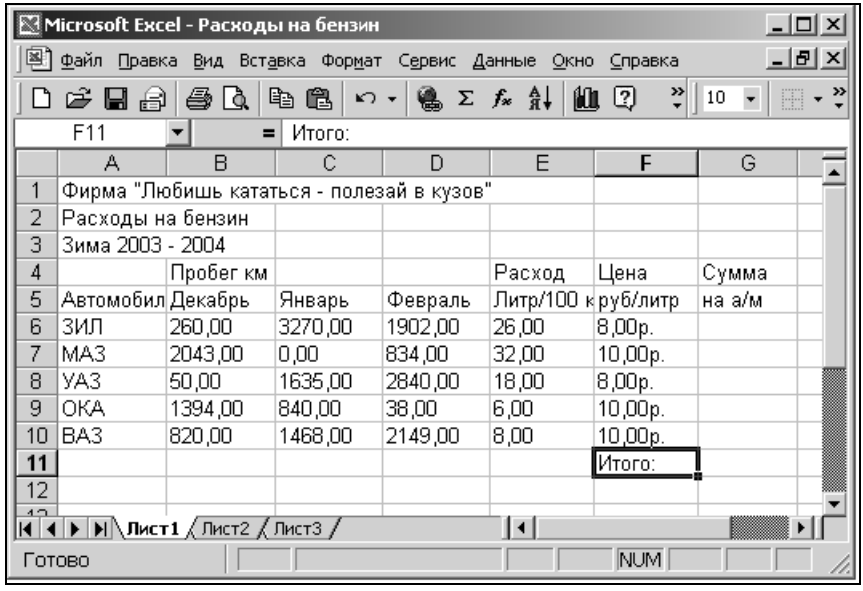
Снятие выделения выполняется щелчком мыши вне выделенного объекта.

Перемещение по рабочему листу осуществляется с помощью полос прокрутки, а по ячейкам рабочего листа — с помощью клавиш управления курсором, клавиши <Tab> и мыши.

Заполнение таблицы

Подадим команду **Файл | Сохранить как**, выберем место для сохранения таблицы и сохраним ее под именем `Расходы на бензин`. Имя таблицы при этом записалось в заголовке окна.

Выберем предлагаемый по умолчанию шрифт Areal Cyr размером 10 пунктов и введем данные в таблицу учета расходов на бензин в фирме "Любишь кататься — полезай в кузов" (рис. 10.2).



	A	B	C	D	E	F	G
1	Фирма "Любишь кататься - полезай в кузов"						
2	Расходы на бензин						
3	Зима 2003 - 2004						
4		Пробег км			Расход	Цена	Сумма
5	Автомобил	Декабрь	Январь	Февраль	Литр/100 к	руб/литр	на а/м
6	ЗИЛ	260,00	3270,00	1902,00	26,00	8,00р.	
7	МАЗ	2043,00	0,00	834,00	32,00	10,00р.	
8	УАЗ	50,00	1635,00	2840,00	18,00	8,00р.	
9	ОКА	1394,00	840,00	38,00	6,00	10,00р.	
10	ВАЗ	820,00	1468,00	2149,00	8,00	10,00р.	
11						Итого:	
12							

Рис. 10.2. Таблица `Расходы на бензин`

Ввод текста

Ввод информации можно выполнять непосредственно в ячейку или в строку формул. При этом если формула записывается в строке формул, то она дублируется в ячейке, а если запись выполняется в ячейке, то она дублируется в строке формул. Для помещения курсора в ячейку достаточно дважды щелкнуть на ней мышью или нажать на клавишу <F2>. Для перехода в строку формул нужно щелкнуть мышью на строке формул.

Начнем ввод с текстовой информации. Это заголовок таблицы, заголовки строк и столбцов. По умолчанию они идут в формате **Общие**. Убедиться в этом можно, подав команду **Формат | Ячейки | Число**. При вводе текст выравнивается по левому краю ячейки.

Выделим щелчком мыши ячейку `A1`. При этом она окажется в более жирной рамке. Введем в ячейку текст: Фирма "Любишь кататься — полезай в кузов". Вводимый текст дублируется в строке формул. Пользуясь обычными приемами работы с текстом, при вводе его можно изменять, исправлять, удалять.

С началом ввода информации перед строкой формул появляются три кнопки. Кнопка **Ввод** (с зеленой птичкой) соответствуют клавише <Enter> на клавиатуре компьютера, кнопка **Отмена** (с красным крестиком) — клавише <Esc>. Кнопка **Изменить формулу** служит для открывания палитры формул, которая помогает при записи формул.

После ввода текст в ячейке нужно закрепить клавишей <Enter> на клавиатуре или кнопкой **Ввод** перед строкой формул. Информация в ячейке закрепляется также щелчком мыши на любой другой ячейке или переходом к следующей ячейке клавишами управления курсором или клавишей <Tab> (при активизации ячейки справа).

Вводимая информация может не помещаться в ячейке полностью. При этом если соседняя ячейка справа заполнена, то часть информации обрезается, но не пропадает. Чтобы вся информация была видна, нужно увеличить ширину столбца. Это будет предложено сделать ниже, при форматировании таблицы.

Ввод чисел

Выделим блок ячеек `B6:E10` (это крайние ячейки по противоположным углам блока), подадим команду **Формат | Ячейки | Число | Числовой | OK**. Введем данные пробегов автомашин и средний расход бензина на 100 км.

Выделив блок ячеек `F6:G10`, подадим команду **Формат | Ячейки | Число | Денежный | OK**. Введем цену за литр горючего. При вводе числа выравниваются по правому краю ячеек.

Исправление, удаление, перемещение и копирование

Исправления в ячейке начинаются с ее выделения. При этом содержимое ячейки дублируется в строке формул. Щелчок мышью на строке формул или двойной щелчок на ячейке вносит в них текстовый курсор, после чего содержимое ячейки редактируется обычными способами.

Для удаления информации из ячеек достаточно их выделить и нажать клавишу <Delete>.

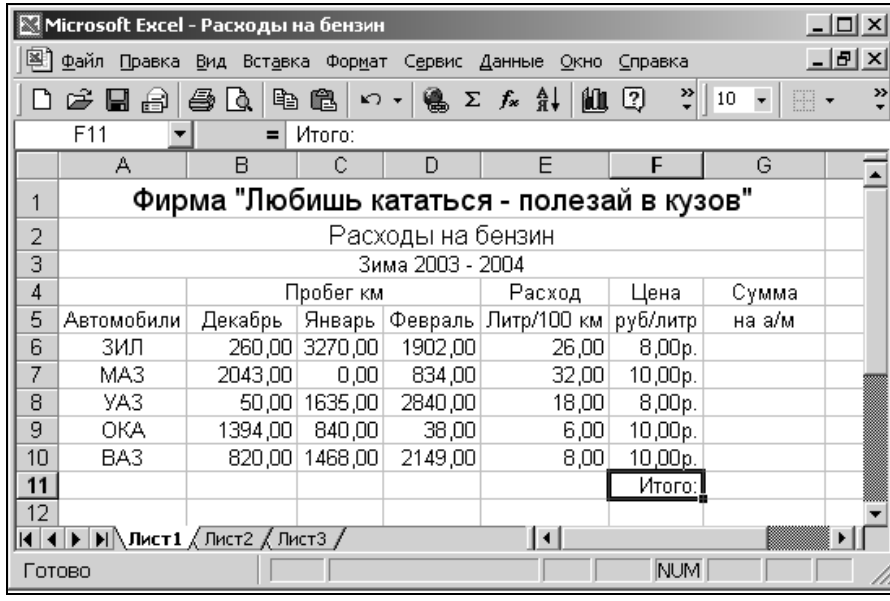
Перемещение и копирование информации выполняется с помощью кнопок **Вырезать**, **Копировать** и **Вставить** стандартной панели инструментов. Перед этим ячейку или блок ячеек нужно выделить.

Форматирование содержимого таблицы

Приведем таблицу к более наглядному виду. Начнем с заголовка. Выделим блок ячеек `A1:G3` и подадим команду **Формат | Ячейки | Выравнивание | по горизонтали | по центру выделения | ОК**. Заголовок при этом разместится по центу таблицы. Выделив заголовок в первой строке, выберем размер шрифта 14 пунктов. Во второй строке выберем размер шрифта текста 12 пунктов. У заголовка можно изменить цвет шрифта.

Выравнивание ширины столбцов и высоты строк

Если подвести указатель мыши к вертикальной линии, разделяющей заголовки столбцов, то указатель примет вид двойной стрелки. С нажатой левой кнопкой мыши эти границы нужно переместить так, чтобы все тексты полностью размещались в своих ячейках. Программа переместит границу в нужное место автоматически, если по границе дважды щелкнуть мышью.



	A	B	C	D	E	F	G
1	Фирма "Любишь кататься - полезай в кузов"						
2	Расходы на бензин						
3	Зима 2003 - 2004						
4		Пробег км			Расход	Цена	Сумма
5	Автомобили	Декабрь	Январь	Февраль	Литр/100 км	руб/литр	на а/м
6	ЗИЛ	260,00	3270,00	1902,00	26,00	8,00р.	
7	МАЗ	2043,00	0,00	834,00	32,00	10,00р.	
8	УАЗ	50,00	1635,00	2840,00	18,00	8,00р.	
9	ОКА	1394,00	840,00	38,00	6,00	10,00р.	
10	ВАЗ	820,00	1468,00	2149,00	8,00	10,00р.	
11						Итого:	
12							

Рис. 10.3. Таблица после форматирования

Высота строк изменяется по тем же правилам. При этом указатель мыши следует подводить к границам между именами строк. Заметим, что размер строки в таблице является функцией размера шрифта. Чем больше размер шрифта текста в строке, тем больше высота строки.

Выделим блок ячеек B4:D4 и подадим команду **Формат | Ячейки | Выравнивание | по горизонтали | по центру выделения | OK**. Заголовок Пробег км разместится по центру трех столбцов. При нажатой клавише <Ctrl>, выделив ячейки A5:G5, A6:A10 и E4:G4, командой **Формат | Ячейки | Выравнивание | по горизонтали | по центру** можно разместить тексты по центрам своих столбцов. Слово "Итого" разместите **по правому краю ячейки**.

В результате получим таблицу, показанную на рис. 10.3.

Вычисления в таблице

Вычисления в таблице — это одна из основных функций программы Excel. Так как вычисления выполняются не с конкретными числами, а с содержимым ячеек таблицы, то результаты вычислений изменяются автоматически вместе с изменениями содержимого ячеек. Вычисления производятся по формулам, вводимым пользователем. Имеется множество функций, упрощающих получение результатов при расчетах.

Упражнение

Выделите блок ячеек B6:B11 и щелкните на кнопке **Автосумма** на стандартной панели инструментов. В ячейке B11 окажется суммарный пробег автомобилей за декабрь. Если выделить блок ячеек B6:D11, то получим сумму в каждом месяце. Удалите полученные суммы.

Суммы можно вычислить иначе. Выделите ячейку B11 под столбцом пробегов за декабрь и нажмите кнопку **Автосумма**. Блок ячеек B6:B10 окружается штриховой мерцающей рамкой, а в ячейке B11 записывается название функции и адрес блока: =SUMM(B6:B10). Нажатие клавиши <Enter> вписывает сумму в ячейку.

Выделите ячейку C11 и щелкните на кнопке **Вставка функций** на стандартной панели инструментов. Открывается окно **Мастер функций — шаг 1 из 2**. Познакомьтесь с категориями предлагаемых функций. Найдите функции СРЗНАЧ (среднее арифметическое), МАКС (максимальное значение), МИН (минимальное значение). Выберите одну из них и нажмите кнопку **OK**. В появившемся окне опять нажмите кнопку **OK**. В ячейке C11 записывается искомый результат. Удалите результаты вычислений.

Ввод формул

Выделим ячейку G6. В нее требуется записать стоимость горючего, израсходованного автомобилем ЗИЛ за три месяца зимы. На 100 км этот автомобиль расходует в среднем 26 литров по 8 рублей за литр.

Запись формулы начнем со знака равенства: `=(B6+C6+D6)/100*E6*F6`. Запись формул удобно выполнять в строке формул, так как слева расположены кнопки **Отмена** и **Ввод**. После записи формулы по команде **Ввод** результат записывается в ячейку.

В остальные ячейки столбца записывать формулы нет необходимости. Формулу можно в них копировать. Выделим ячейку `G6` и кнопкой **Копировать** на стандартной панели занесем содержимое (формулу) в буфер. Выделим остальные ячейки столбца (`G7:G10`) и кнопкой **Вставить** поместим формулы в эти ячейки. Вставляемая формула не меняется. Изменяются только адреса входящих в формулу ячеек. В выделенных ячейках окажутся записанными требуемые результаты.

Осталось только вычислить итоговую сумму, что легко можно сделать с помощью кнопки **Автосумма** на стандартной панели инструментов.

Выделяя блоки, используем возможности расположенной на панели форматирования кнопки **Границы**. В результате получаем таблицу, показанную на рис. 10.4.

Microsoft Excel

Файл Правка Вид Вставка Формат Сервис Данные Окно Справка

C8 = 1635

Расходы на бензин

	A	B	C	D	E	F	G
1	Фирма "Любишь кататься - полезай в кузов"						
2	Расходы на бензин						
3	Зима 2003 - 2004						
4		Пробег км			Расход	Цена	Сумма
5	Автомобили	Декабрь	Январь	Февраль	Литр/100 км	руб/литр	на а/м
6	ЗИЛ	260,00	3270,00	1902,00	26,00	8,00р.	11 298,56р.
7	МАЗ	2043,00	0,00	834,00	32,00	10,00р.	9 206,40р.
8	УАЗ	50,00	1635,00	2840,00	18,00	8,00р.	6 516,00р.
9	ОКА	1394,00	840,00	38,00	6,00	10,00р.	1 363,20р.
10	ВАЗ	820,00	1468,00	2149,00	8,00	10,00р.	3 549,60р.
11						Итого:	31 933,76р.
12							

Готово NUM

Рис. 10.4. Результаты вычислений в таблице

Абсолютная и относительная адресация

При копировании формулы из ячейки `G6` в остальные ячейки столбца `G` адреса ячеек в формуле изменялись в соответствии с номерами следующих строк. Программа полагает, что следующие входящие в формулу данные расположены аналогично шестой строке — в тех же столбцах. В этом случае говорят об относительной адресации.

Нередки случаи, когда некоторое количество адресов в формуле должно оставаться постоянным. Тогда применяют абсолютную адресацию. Чтобы программа различала абсолютные и относительные адреса, к абсолютным адресам добавляется знак доллара, например так: `$D$5`.

Пример

Если формулу подсчета расходов на горючее на автомашину записать так: `=(B6+C6+D6)/100*$E$6*F6`, то расход горючего на 100 км у всех автомашин будет принят равным 26 литрам.

Вставка и удаление ячеек, строк и столбцов

В построенную таблицу можно при необходимости вставить в нужные места нужное количество ячеек, строк и столбцов. Это делается командами главного меню **Вставка | Ячейки**, **Вставка | Строки**, **Вставка | Столбцы** соответственно.

При вставке столбца следует выделить столбец, перед которым должен быть вставлен пустой, и подать команду. При выделении не всего столбца, а нескольких его ячеек появляется диалоговое окно **Добавление ячеек** с вопросами к пользователю о том, что сдвигать и куда сдвигать ячейки (вправо или вниз) при освобождении места для вставки.

Удалить ненужные ячейки, строки и столбцы можно командой **Правка | Удалить**. При этом если удаляется не весь столбец или не вся строка, то появляется диалоговое окно **Удаление ячеек** с вопросами о заполнении освободившегося места.

Работа с буфером обмена

После выделения ячейки, группы ячеек или всей таблицы их можно вырезать (отправить в буфер обмена), скопировать, переместить. Для этого удобно использовать кнопки **Вырезать**, **Копировать** и **Вставить** стандартной панели инструментов. Эти же команды имеются в подменю главного меню **Правка**. Кроме того, можно пользоваться известными комбинациями клавиш: <Shift>+<Delete> — вырезать, <Ctrl>+<Insert> — копировать, <Shift>+ +<Insert> — вставить.

Правая кнопка мыши

Если после выделения ячейки или блока ячеек щелкнуть правой кнопкой мыши, то откроется *контекстно-зависимое меню* ("контекстное меню"). В нем имеется полный набор команд для действий с выделенным фрагментом таблицы.

Перемещение ячеек и блоков ячеек

Переместить выделенный блок ячеек также можно несколькими способами. Способ "перетащить и оставить" (Drang&Drop) заключается в следующем. Выделяем блок ячеек. Подводим указатель мыши к рамке блока и при нажатой левой кнопки перетаскиваем рамку в выбранное место. Если при этом удерживать нажатой клавишу <Ctrl>, то рядом с указателем мыши появится знак + и выделенный фрагмент будет скопирован.

Построение диаграмм

Программа Excel предоставляет пользователю удобные средства создания диаграмм, наглядно представляющих содержимое построенных таблиц. Рассмотрим процесс построения диаграмм для таблицы расходов на горючее с помощью Мастера диаграмм.

Мастер диаграмм построит диаграмму за четыре шага. Перед вызовом Мастера нужно выбрать в таблице данные, которые желательно в диаграмме отобразить. Выделим в таблице блок ячеек `A5:D10` и на стандартной панели инструментов нажмем кнопку **Мастер диаграмм**. На экране появляется первое диалоговое окно Мастера диаграмм с заголовком:

Мастер диаграмм (шаг 1 из 4): тип диаграммы

Окно имеет две вкладки. Откроем вкладку **Стандартные**. Перебирая разновидности **типов** диаграмм, просматриваем их **вид**. Выберем **Тип:** | **Гистограмма** и **Вид:** — второй рисунок в первом ряду. Нажимаем на кнопку **Далее**. Появляется следующее окно мастера:

Мастер диаграмм (шаг 2 из 4): источник данных диаграммы

Так как данные были выбраны перед вызовом мастера, то кнопкой **Далее** переходим к следующему диалоговому окну мастера:

Мастер диаграмм (шаг 3 из 4): параметры диаграммы

Это самое насыщенное вкладками окно мастера. Выберем только две вкладки. Во вкладке **Заголовки** впишем в текстовое поле ввода **Название диаграммы** заголовок "Пробег а/м. Зима 2003-2004". Во вкладке **Линии сетки** установим флажки в строчках **основные линии**. Переходим к следующему окну:

Мастер диаграмм (шаг 4 из 4): размещение диаграммы

В этом окне предлагается альтернатива выбора того, на каком листе поместить диаграмму: **Отдельном** или на **Имеющемся**. Выбираем второй вариант и нажимаем кнопку **Готово**. На поле таблицы появляется диаграмма.

Изменение размеров и места размещения диаграммы

Выделяется диаграмма щелчком мыши в области диаграммы на свободном от деталей месте. Снимается выделение щелчком вне области диаграммы. Выделенная диаграмма окружена восемью размерными маркерами.

Для изменения размеров диаграммы нужно захватить мышью выбранный размерный маркер и при нажатой левой кнопке мыши перетащить его в нужное место.

Для перемещения диаграммы следует ее выделить, поместить мышь в любое свободное место области диаграммы и при нажатой кнопке перетащить ее в выбранное место.

Если при щелчке мышью на отдельных элементах диаграммы они окружаются размерными маркерами, то с ними также можно выполнять действия по изменению размеров и перемещению.

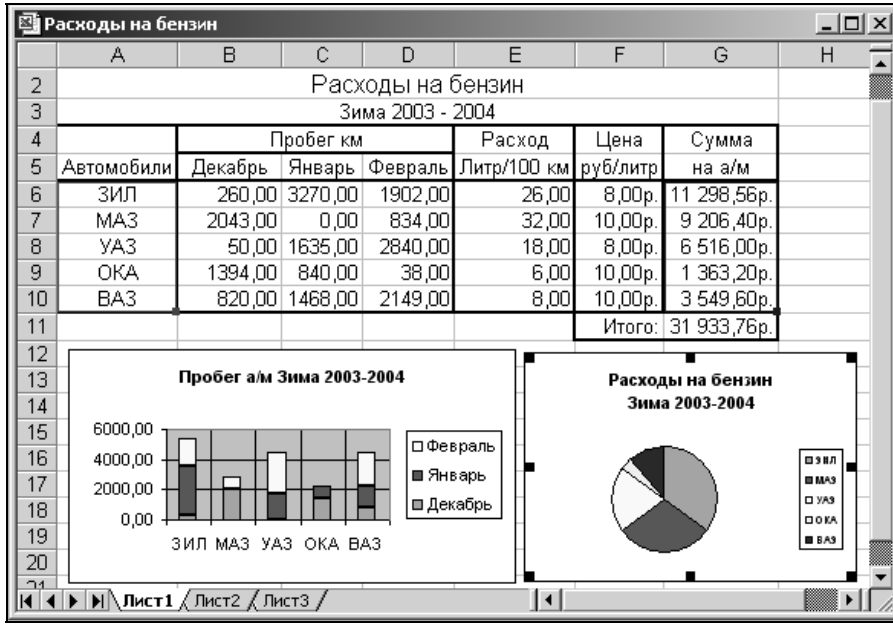


Рис. 10.5. Диаграммы типа "гистограмма" и "круговая"

Упражнение

Уменьшите размеры созданной диаграммы и переместите ее в нижний левый угол экрана. При нажатой клавише <Ctrl> выделите два блока ячеек: `A6:A10` и `G6:G10`. С помощью Мастера диаграмм создайте круговую диаграмму расходов на горючее. Расположите ее в правом нижнем углу экрана (рис. 10.5).

Работа с файлами

Сохранение рабочей книги

Для сохранения результатов работы с таблицей подайте команду **Файл | Сохранить как**. На экране появится диалоговое окно с предложением присвоить файлу имя, под которым Рабочая книга должна быть записана в память компьютера. После выбора места сохранения и записи имени следует нажать <Enter> или кнопку **OK**.

Если рабочая книга уже имеет имя, то для ее сохранения достаточно щелкнуть мышью на кнопке **Сохранить**.

Вызов рабочей книги из памяти

Для открытия требуемого файла можно подать команду **Файл | Открыть** или щелкнуть мышью по кнопке **Открыть**. В появившемся диалоговом окне нужно выбрать имя файла и дважды щелкнуть на нем мышью или щелкнуть мышью на кнопке **OK**.

Задания для самостоятельной работы

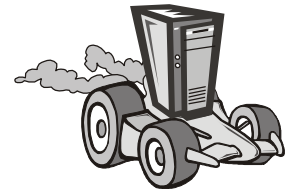
Фирма Microsoft, как правило, сопровождает созданные ею приложения не только достаточно полными и удобными в работе справочными системами, но и отдельными файлами-примерами. При этом следует учитывать, что в разных моделях компьютеров и версиях приложения наличие и состав этих файлов могут различаться.

Во-первых, эти примеры можно найти в списке файлов папки C:\Program Files\Microsoft Office\Office\Samples. В этом списке могут быть записаны файлы: Samples (примеры) и Solvsamp. Они отмечены характерным для приложения Excel значком. Первый файл на листе Excel содержит несколько примеров для Microsoft Excel. Их полезно просмотреть. Переход от примера к примеру выполняется выбором ярлычков, расположенных ниже таблицы. Второй файл содержит пример "Поиск решения", с которым желательно познакомиться подробнее и выполнить предлагаемые в нем действия.

Во-вторых, с примерами можно познакомиться, подав команду **Пуск | Найти | Файлы и папки**. В диалоговом окне **Результаты поиска** в текстовом поле ввода **Искать имена Файлов или Папок** нужно записать *.xls (найти все Excel-файлы) и нажать на кнопку **Найти**. Здесь список файлов может быть обширнее. Среди них можно увидеть и файлы Samples и Solvsamp. Кроме них интересно просмотреть содержимое следующих файлов:

- MAPSTATS — содержит представляемые корпорацией MapInfo данные демографического характера по всем странам мира. Эта информация будет интересна учащимся, увлекающимся географией, историей, политикой.
- COMMON — демонстрирует Excel-документ с двумя не без юмора заполненными рабочими листами: "Сведения о сотрудниках" и "Товары и услуги".
- XL8GALRY — содержит 20 образцов диаграмм, с которыми полезно ознакомиться для того, чтобы использовать их в своей практике.
- FUNCS — представляет собой полный словарь функций, операторов, команд и служебных слов приложения Excel. Словарь выполнен в виде двух столбцов. Слева термины на русском языке, справа — на английском. Информацию в левом столбце можно упорядочить по возрастанию или по убыванию. При желании с помощью приема, описанного выше в данной главе, столбцы можно поменять местами. При этом, возможно, потребуется снять защиту листа.

Глава 11



База данных Access

11.1. Общие сведения о базах данных

Данными называют информацию, представленную в определенной форме, позволяющей выполнять ее прием, передачу, хранение, преобразование и использование.

Базой данных называют систематизированное, упорядоченное объединение больших объемов данных, собранных с целью их хранения, накопления, организации различных видов поиска и выдачи в определенной форме.

Примерами баз данных могут служить: адреса клиентов в адресной книге, сведения о наличии товаров на складе или на оптовой базе, формы со сведениями о сотрудниках в отделе кадров учреждения и т. п. Простейшую базу данных мы видим в конце школьного классного журнала, где записаны сведения об учениках и их родителях. К базам данных можно отнести и записную книжку с фамилиями, адресами и номерами телефонов.

Одной из современных прикладных программ создания баз данных и работы с ними является приложение Microsoft Access. Это система управления базами данных (СУБД). Она создает *реляционные* базы данных, сохраняющие связанные данные в одном месте, что упрощает их поиск, анализ, защиту и выдачу по запросам.

Перечислим основные объекты базы данных Access.

Таблицы — это наборы данных, объединенных общим смыслом, характером, назначением. (Примером данных таблицы могут служить данные о сотрудниках.) В таблицах эти данные сохраняются. Данные в таблице располагаются в *полях* (столбцах) и *записях* (строках). Каждое отдельное поле содержит одно какое-либо сведение обо всех сотрудниках, например, дату рождения. Отдельная запись содержит все сведения об одном сотруднике. Обычно база данных состоит из нескольких таблиц.

Запросы — собирают затребованные данные из одной или нескольких таблиц по заданным критериям. Учитывая значительные по величине размеры таблиц реальных баз данных, осуществлять поиск нужной информации вручную, без использования запросов практически невозможно.

Формы — выводят данные из таблиц и запросов в удобном для чтения и анализа формате. Как правило, форма выводит информацию, соответствующую одной записи в таблице. Формы, кроме того, облегчают работу по вводу информации в базу данных.

Отчеты — отображают на пронумерованных страницах с заголовками данные таблиц или запросов в удобном для чтения формате. Отчеты используются для просмотра и редактирования данных и вывода их на печать.

Страницы доступа — облегчают доступ к часто используемым данным.

Макросы — автоматизируют выполнение конкретных операций с данными.

Модули — содержат программы на языке программирования Visual Basic, применяемые для решения специальных задач баз данных.

11.2. Создание базы данных

Программа Access по действиям, выполняемым при построении базы данных, отличается от программ Word и Excel некоторой жесткостью. Не всегда можно отменить или изменить выбранный путь построения базы, выполнить возврат назад. Нередко начинающим пользователям приходится повторять действия по построению базы данных с самого начала. Поэтому перед созданием большой базы данных, содержащей несколько таблиц, должен быть выполнен этап предварительного планирования. При создании рассматриваемой ниже простой учебной базы данных желательно следовать предлагаемой последовательности действий.

Интересно создавать документы, имеющие практическое применение. Можно, например, в кабинете информатики учебного заведения создать базу данных своего класса или учебной группы. Такая база может содержать следующие виды данных (или *поля*): фамилия, имя, адрес, дата рождения, телефон, увлечения. В дальнейшем, на традиционных встречах выпускников эту базу можно исправлять, добавлять в нее новые поля, такие как: место учебы, работы, семейное положение, поля с другой информацией. При этом всегда имеется возможность распечатать обновленную базу и перенести ее на свой компьютер.

В качестве примера создадим простую базу данных `Адресная книга` с одной таблицей `Адреса 11а класса` всего для пяти учащихся.

Запуск программы

Запуск программы Access выполняется рассмотренными при работе с другими приложениями способами *(см. гл. 9, 10)*. После запуска приложения на экране появляется диалоговое окно программы Microsoft Access с отключен-

ными главным меню и стандартной панелью инструментов. В центре этого окна расположено диалоговое окно, показанное на рис. 11.1.

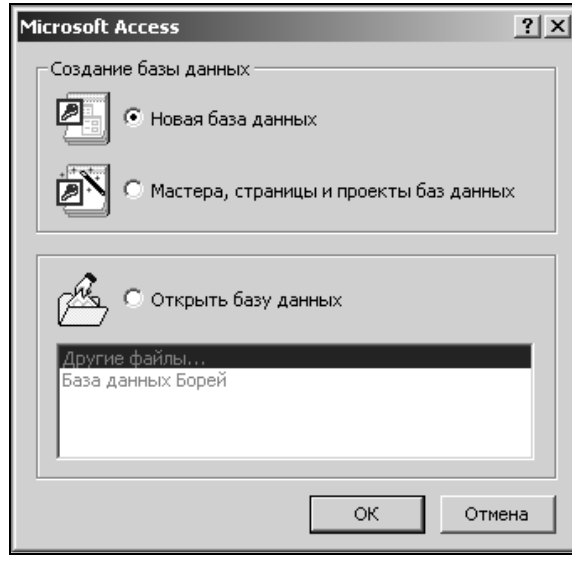


Рис. 11.1. Диалоговое окно **Microsoft Access** с приглашением к работе

Диалоговое окно **Microsoft Access** имеет две панели: **Создание базы данных** и **Открыть базу данных**. Для получения справок можно щелкнуть мышью на кнопке со знаком вопроса, а затем сначала на одной панели, затем на другой.

Включим переключатель **Новая база данных** и щелкнем по кнопке **OK**. Появляется диалоговое окно **Файл новой базы данных** (рис. 11.3).

Если на экране отсутствует окно, показанное на рис. 11.1, то начать работу по созданию базы данных можно иначе. При щелчке на кнопке **Создать** на стандартной панели инструментов появляется диалоговое окно **Создание** (рис. 11.2).

В диалоговом окне **Создание** две вкладки: **Общие** и **Базы данных**. На вкладке **Базы данных** предлагается несколько вариантов готовых шаблонов баз данных. При выборе значка с именем шаблона в окне **Просмотр** демонстрируется картинка со структурой базы. Откажемся от шаблонов и выберем **Общие | База данных | OK**.

Следующее диалоговое окно **Файл новой базы данных** (рис. 11.3) имеет одинаковый вид для обоих вариантов начала работы.

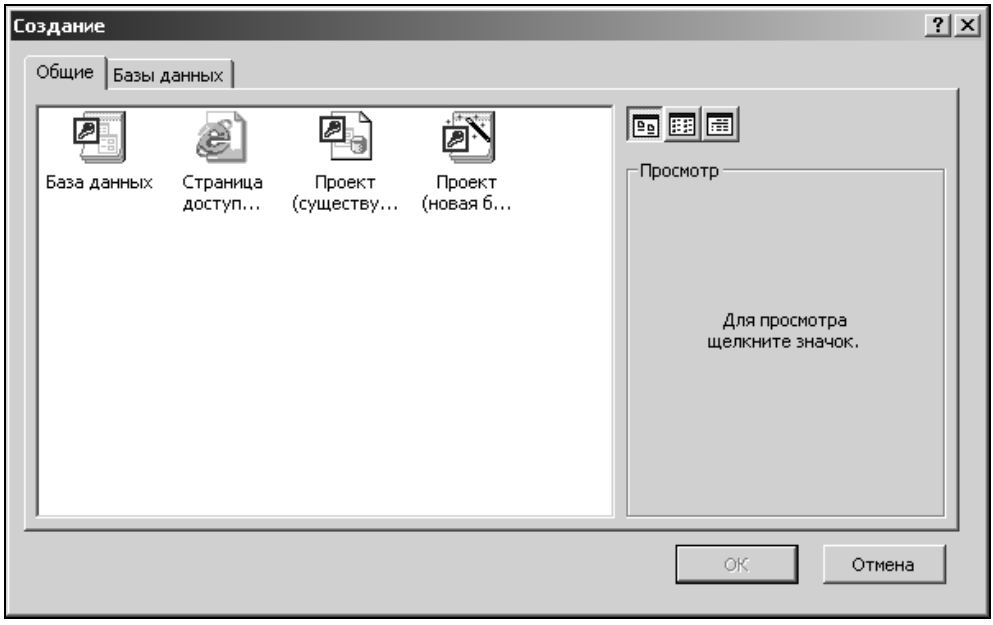


Рис. 11.2. Диалоговое окно **Создание**

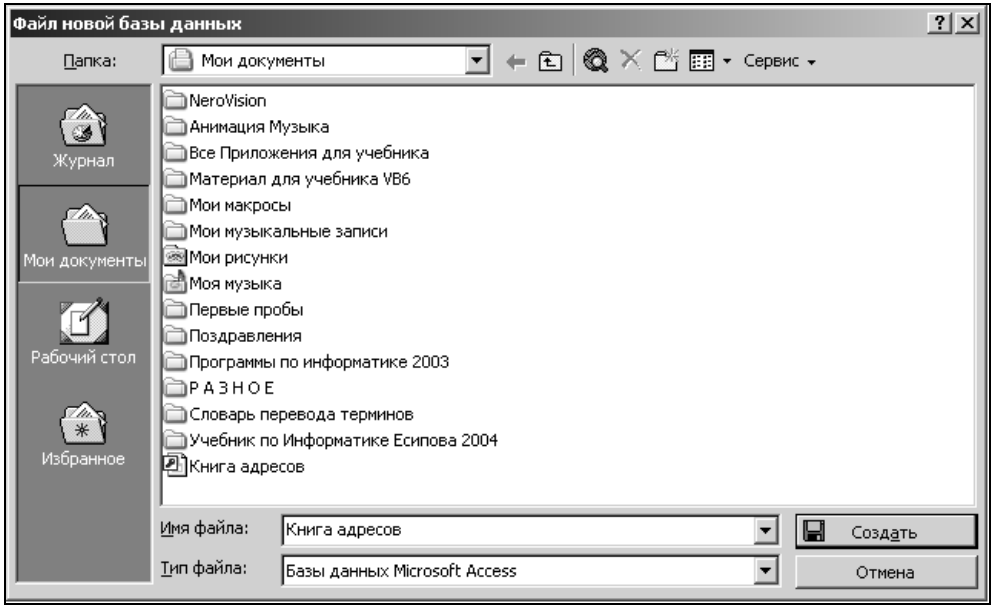


Рис. 11.3. Диалоговое окно **Файл новой базы данных**

В этом окне предлагается указать, в какой папке и под каким именем будет сохраняться создаваемая база данных. В поле ввода **Имя файла** запишем `Книга адресов` и нажмем кнопку **Создать**.

Обратим внимание на то, что файлы базы данных Access имеют расширение mdb.

С нажатием кнопки **Создать** открывается новое диалоговое окно — **Книга адресов : база данных** (рис. 11.4). Это *главное окно* приложения Access, к которому будем неоднократно возвращаться для создания очередного объекта базы данных.

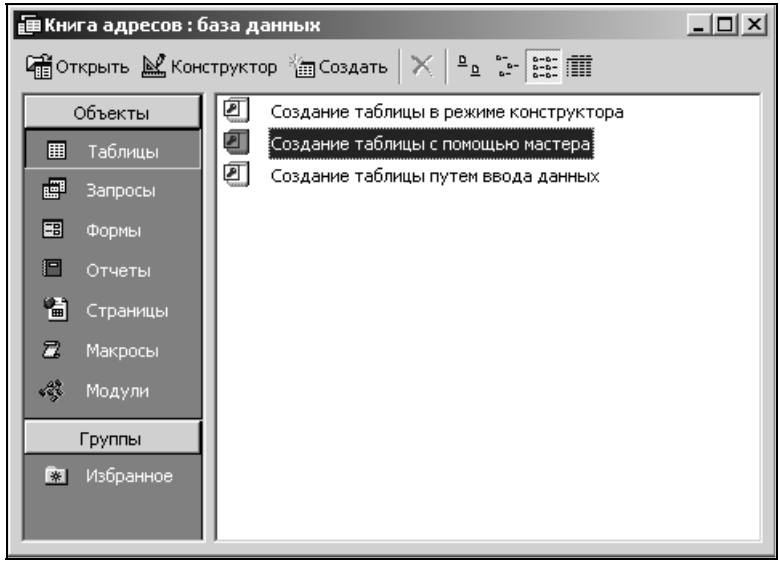


Рис. 11.4. Главное окно **Книга адресов : база данных**

В левой части окна расположена группа кнопок вкладки **Объекты**. Справа — предложения вариантов создания объектов.

Создание таблицы с помощью мастера

Выберем в главном окне **Книга адресов : база данных** объект **Таблицы** и дважды щелкнем на кнопке **Создание таблицы с помощью мастера**.

На экране появляется первое диалоговое окно **Создание таблиц** (рис. 11.5). Прочитайте рекомендации в верхней части окна. Ниже (в рамке) расположены два переключателя: **Деловые** и **Личные**. Просматривая образцы баз данных, можно познакомиться с предлагаемыми для этих баз образцами полей.

После просмотра образцов выберем категорию **Личные**. В списке **Образцы таблиц** выберем категорию **Адреса**.

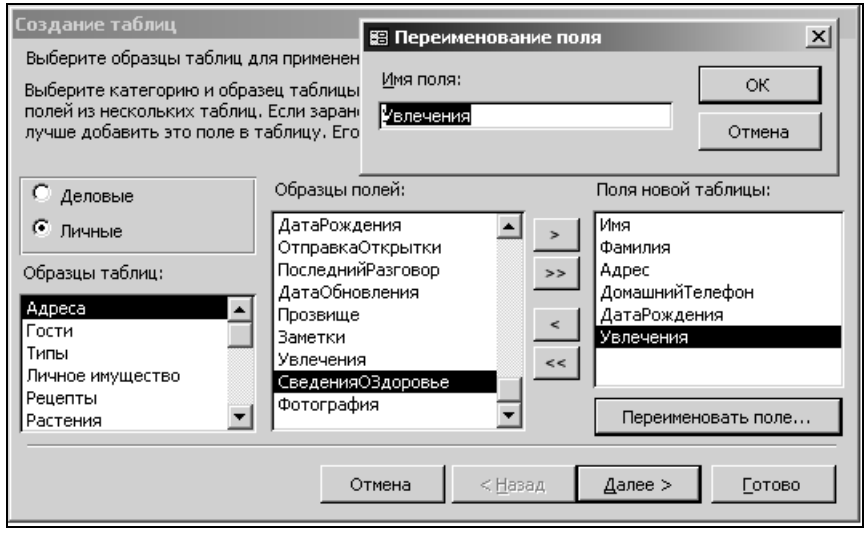


Рис. 11.5. Первое диалоговое окно **Создание таблиц** с вызванным окном **Переименованиеь поля**

Напомним, что столбцы таблицы в базах данных — это *поля*, а строки с записанной в них информацией — *записи*.

Формируем список названий полей создаваемой таблицы. Для этого в списке **Образцы полей** (он расположен правее списка **Образцы таблиц**) выделяем желаемые названия полей и с помощью кнопок со стрелками перемещаем их в список **Поля новой таблицы**. Выберем поля: **Имя**, **Фамилия**, **Адрес**, **ДомашнийТелефон**, **ДатаРождения**, **Увлечения**.

Кнопкой **Переименовать поле** вызывается окно, позволяющее дать выделенному полю другое название. Переименуем поле `Увлечения` в `Хобби`. По завершении создания списка полей нажмем кнопку **Далее**.

На экране появляется второе диалоговое окно **Создание таблиц**. В нем предлагается задать имя новой таблицы. Впишем в текстовое поле ввода **Задайте имя для новой таблицы** имя `Адреса 11а`. Под этим именем таблица будет сохранена в базе данных `Книга адресов`.

База данных может содержать несколько таблиц. Каждая из них должна иметь свое имя. Следующие таблицы создаваемой базы могут иметь, например, такие имена: `Адреса родственников и знакомых`, `Адреса магазинов`, `Адреса учреждений` и т. п.

После задания имени таблицы включим переключатель **Microsoft Access автоматически определяет ключ** и нажмем кнопку **Далее**.

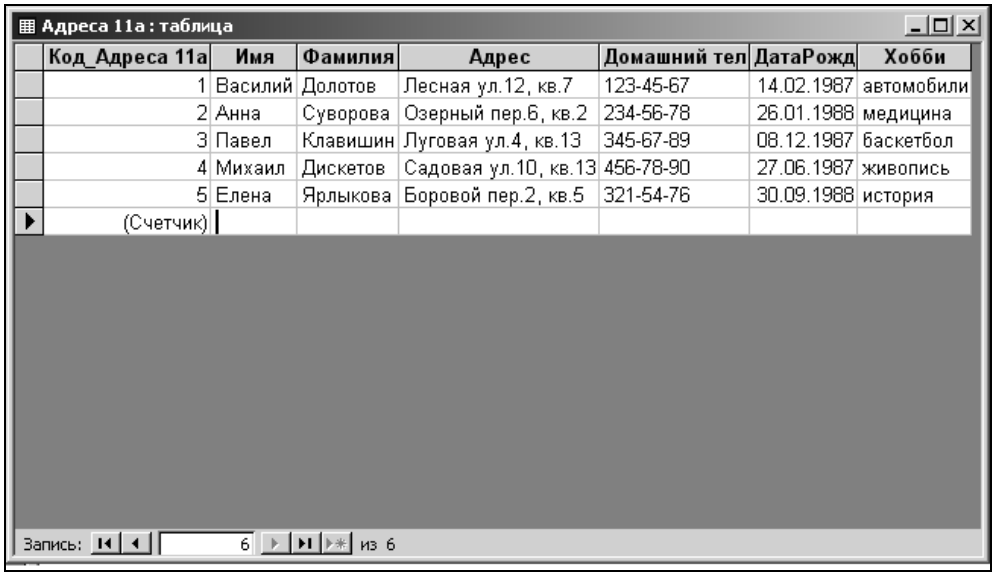
Появляется третье диалоговое окно **Создание таблиц**. В нем сообщается, что указаны все сведения, необходимые для создания таблицы с помощью мастера, и спрашивается, в частности: как вводить сведения в таблицу — непосредственно или с помощью формы, создаваемой мастером. Второй способ более удобен, более профессионален (с ним мы познакомимся ниже). Выберем первый вариант, позволяющий непосредственно вводить данные в таблицу и видеть все вводимые записи.

Имеющиеся в диалоговых окнах кнопки **Назад** дают возможность возвращаться к предыдущим окнам для внесения изменений. Просмотрим еще раз принятые решения, вернемся к последнему окну **Создание таблиц** и нажмем кнопку **Готово**.

На экране появляется окно **Адреса 11а : таблица** (рис. 11.6) с макетом таблицы. Таблица имеет выбранные на предыдущих шагах названия столбцов (полей) и готова для заполнения данными.

Рис. 11.6. Макет готовой для заполнения таблицы `Адреса 11а`

Вид таблицы `Адреса 11а` после заполнения и форматирования показан на рис. 11.7.



Адреса 11а : таблица

Код_Адреса 11а	Имя	Фамилия	Адрес	Домашний тел	ДатаРожд	Хобби
1	Василий	Долотов	Лесная ул.12, кв.7	123-45-67	14.02.1987	автомобили
2	Анна	Суворова	Озерный пер.6, кв.2	234-56-78	26.01.1988	медицина
3	Павел	Клавишин	Луговая ул.4, кв.13	345-67-89	08.12.1987	баскетбол
4	Михаил	Дискетов	Садовая ул.10, кв.13	456-78-90	27.06.1987	живопись
5	Елена	Ярлыкова	Боровой пер.2, кв.5	321-54-76	30.09.1988	история
(Счетчик)						

Запись: 6 из 6

Рис. 11.7. Заполненная и отформатированная таблица `Адреса 11а`

Перемещение по строкам (записям) и столбцам (полям) таблицы осуществляется с помощью мыши, клавиш управления курсором клавиатуры, клавиш <Tab> и <Enter>.

Код адреса проставляется в виде порядковых номеров записей автоматически, с началом ввода информации в строку. Для исправлений в заполненных ячейках курсор в них удобно вводить мышью. Действуют известные правила выделения, копирования и удаления.

Главное меню приложения Access и стандартная панель инструментов с переходом к диалоговому окну **Адреса 11а** активизировались. Появилась возможность использовать их при проверке орфографии записей, форматирования таблицы, сортировки записей и т. п.

Некоторые тексты в полях могут быть частично скрыты, не умещаться в рамках поля. Раскрыть полностью имена полей и тексты в них можно командой **Формат | Ширина столбца | По ширине данных**. Для отдельных полей это можно сделать с помощью мыши. Если подвести мышь к линии границы между именами полей, то появляется двойная стрелка. Двойной щелчок мышью расширяет или сужает столбец до нужных размеров.

Сортировка записей выполняется очень просто. Для этого нужно выделить поле, данные которого требуется упорядочить, и щелкнуть на кнопке панели инструментов **Сортировка по возрастанию** или на кнопке **Сортировка по**

убыванию. Сортировке поддаются как числовые поля (**ДатаРождения**), так и текстовые (**Имя**, **Фамилия**).

Кнопкой **Закрыть** закрываем окно с таблицей и возвращаемся к главному окну **Книга адресов : база данных**, в правой части которого последней строкой уже записано имя созданной таблицы `Адреса 11а`. Двойной щелчок мышью на имени раскрывает только что созданную таблицу. Возврат к главному окну — по нажатию кнопки **Закрыть**.

Таблицы служат главным хранилищем информации в базах данных. Учитывая очень большие размеры таблиц в реальных базах данных, в явном виде (целиком) они, как правило, не используются. Для доступа к определенным выборкам данных из таблиц служат запросы. Для ввода и редактирования данных в записях таблицы служат формы. Вывод данных на печать организуется с помощью отчетов.

Создание запросов с помощью мастера

Запросы выбирают из таблиц сведения по определенным критериям. Значение запросов трудно оценить при работе с очень простой базой данных, состоящей из одной таблицы `Адреса 11а`. Обычно база данных состоит из нескольких таблиц значительного размера. Поэтому сделать из них нужную выборку бывает очень трудно. В этих случаях на помощь приходят запросы.

В главном окне программы Access (см. рис. 11.4) выберем следующий объект — **Запросы** и строку **Создание запроса с помощью мастера**. На экране появляется диалоговое окно **Создание простых запросов** (рис. 11.8).

В диалоговом окне **Создание простых запросов** из списка **Доступные поля** предлагается перенести в окно **Выбранные поля** имена полей, которые будут определять простую выборку. Перенесем имена: `Имя`, `Фамилия`, `ДомашнийТелефон`, `ДатаРождения`. Для перехода к следующему диалоговому окну нажмем кнопку **Далее**.

Второе диалоговое окно **Создание простых запросов** требует задать имя запроса. В текстовое поле ввода **Задайте имя запроса** впишем имя `Поздравления` и нажмем кнопку **Готово**.

В итоговом диалоговом окне (рис. 11.9) количество полей в выведеной по запросу таблице по сравнению с исходной таблицей уменьшилось и соответствует количеству полей, выбранных при построении запроса. Напомним, что в таблицах реальных баз данных количество полей может быть труднообозримым. Поэтому такое уменьшение полей в запросе бывает полезным.

Уменьшение количества отображаемых полей — это наиболее простой вариант построения запроса. Более сложные запросы позволяют использовать и более сложные критерии. Например, можно создать запрос о днях рождения

в заданном месяце, отобрать все записи, содержащие название определенной улицы, и т. п. Такие запросы удобно создавать в режиме конструктора.

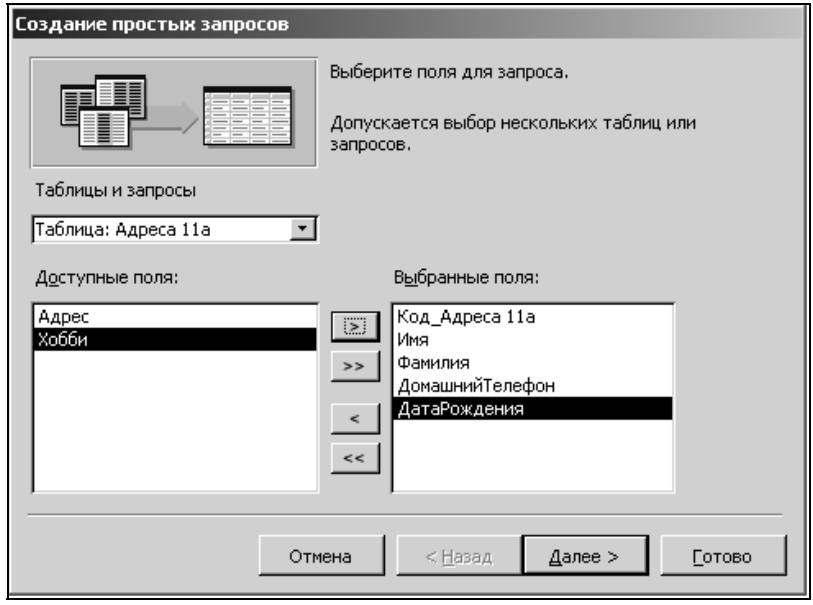


Рис. 11.8. Первое диалоговое окно **Создание простых запросов**. Выбор полей

Рис. 11.9. Запрос `Поздравления`

Используя различные критерии, можно создать несколько запросов. Имена запросов после создания вписываются в меню запросов главного окна базы

данных. Созданные запросы вызываются на вкладке **Запросы** главного окна **Книга адресов : база данных** щелчком мыши на имени запроса.

Создание форм с помощью мастера

По внешнему виду формы напоминают бланки, предназначенные для заполнения требуемой информацией. Поэтому формы удобно использовать для ввода информации в базу данных, ее редактирования и исправления. Кроме этого, формы можно применять для последовательного или выборочного просмотра отдельных записей таблиц.

Вернувшись к диалоговому окну **Книга адресов : база данных** (см. рис. 11.4), выбираем вкладку **Формы** и дважды щелкаем на значке в строке **Создание форм с помощью мастера**. Этой командой вызывается первое диалоговое окно **Создание форм** (рис. 11.10), предлагающее выбрать поля из списка доступных.

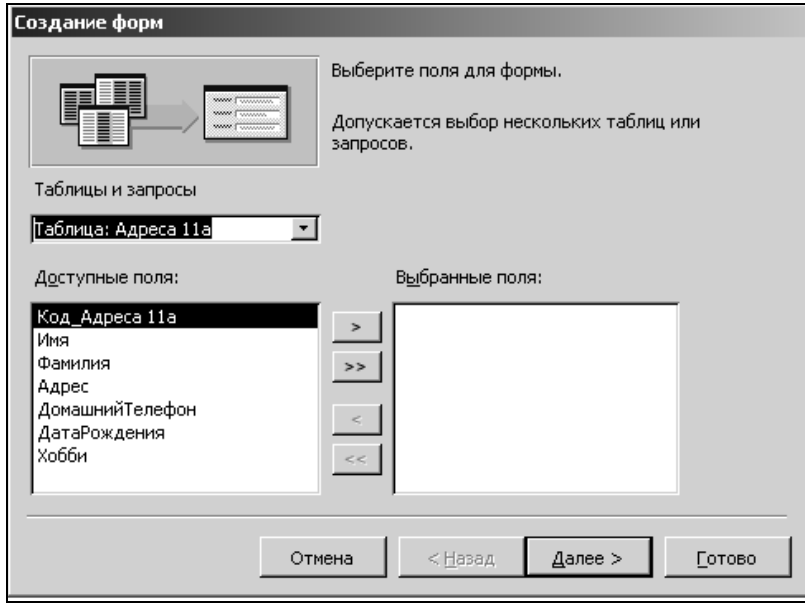


Рис. 11.10. Первое диалоговое окно **Создание форм**. Выбор полей формы

Двойной стрелкой перенесем все доступные поля в окно выбранных полей и нажмем кнопку **Далее**. На экран выводится второе диалоговое окно Мастера создания форм с предложением выбрать внешний вид формы. С помощью переключателя **в один столбец** выберем соответствующий вид и нажмем кнопку **Далее**.

На экране — третье диалоговое окно Мастера создания форм, в котором предлагается выбрать стиль отображения формы. В окне, расположенном в левой части диалогового окна, можно просмотреть предлагаемые стили. Выберем стиль **Камень** и нажмем на кнопку **Далее**.

В четвертом, последнем диалоговом окне мастер предлагает назвать создаваемую форму. Введем в текстовое поле ввода **Задайте имя формы** имя `Адреса 11a` и нажмем кнопку **Готово**.

Созданная мастером с учетом внесенных требований форма на рис. 11.11 показана на фоне главного окна **Книга адресов : база данных** и окна программы **Microsoft Access**.

Пользуясь кнопками со стрелками в нижней части окна, можно переходить в начало и в конец таблицы базы данных, а также вызывать записи таблицы последовательно, одну за другой.

Рис. 11.11. Форма `Адреса 11a` на фоне главного окна

Формы позволяют организовать поиск нужной информации. Щелкнув на имени выбранного поля (например, `Фамилия`), щелчком на кнопке **Найти** на

стандартной панели инструментов (в виде бинокля) вызываем диалоговое окно **Поиск и замена**. В нем переходим на вкладку **Поиск**, где в текстовое поле ввода **Образец** вписываем искомую фамилию. Поиск начинается с нажатием кнопки **Найти далее**. Закрыв окно, обнаруживаем форму для заданной фамилии.

Можно представить себе сложность задачи поиска по книгам учета данных на сотрудника учреждения, имеющего несколько десятков тысяч сотрудников, или данных на товар в оптовой базе, имеющей десятки тысяч наименований. С помощью компьютера эти задачи решаются практически мгновенно.

Создание отчетов с помощью мастера

Отчеты служат для вывода информации из базы данных на печать. Для создания отчета будем привлекать Мастера отчетов. Рассмотрим второй способ перехода к процедуре создания объектов базы данных. Выбрав в окне **Книга адресов : база данных** вкладку **Отчет**, нажмем кнопку панели инструментов **Создать**. В открывшемся диалоговом окне **Новый отчет** выберем **Мастер отчетов**. При этом открывается первое диалоговое окно **Создание отчетов** (рис. 11.12), где предлагается выбрать для отчета поля. Учитывая, что выбор у нас небольшой, с помощью двойной стрелки выберем все поля таблицы `Адреса 11а`. Нажмем кнопку **Далее**.

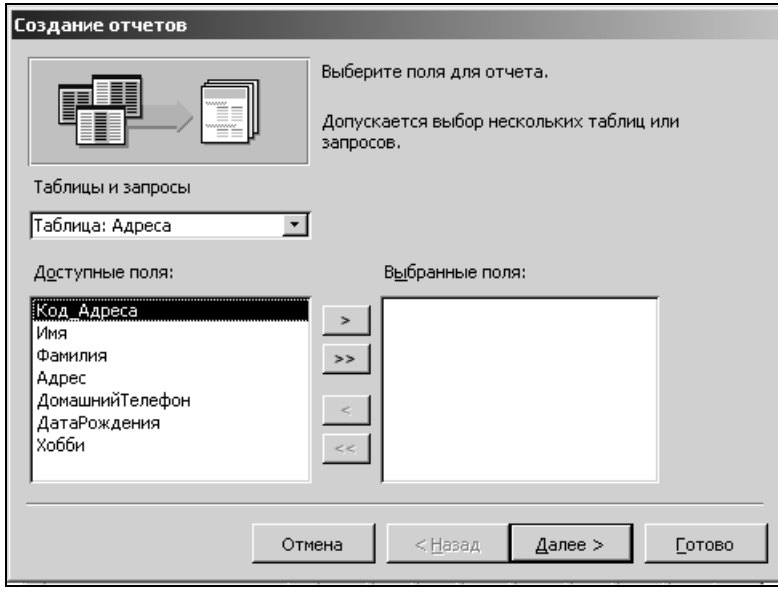


Рис. 11.12. Первое диалоговое окно **Создание отчетов**. Выбор полей

В новом окне — втором диалоговом окне **Создание отчетов** предоставляемой мастером процедуры создания отчета предлагается добавить уровни группировки с помошью соответствующего списка. Откажемся от других вариантов группировки данных при выводе на печать и просто нажмем кнопку **Далее**.

Следующее, третье диалоговое окно **Создание отчетов** предлагает задать сортировку данных. Откажемся от сортировки и нажмем кнопку **Далее**.

В четвертом диалоговом окне **Создание отчетов** Мастер отчетов предлагает выбрать макет для создаваемого отчета. Частично изменим предлагаемый в окне вариант и выберем с пмощью переключателей группы **Макет** — макет **в столбец**, а ориентация пусть останется — **книжная**.

Следующие два диалоговых окна (пятое и шестое), раскрывающиеся по нажатию кнопки **Далее**, служат для выбора стиля отображения отчета и задания его имени.

Выберем стиль обычный и имя `Адреса 11а`. В пятом окне после выбора стиля нажмем кнопку **Далее**, в шестом окне — **Готово**.

На рис. 11.13. показана часть созданного отчета. В таком виде он выводится на печать.

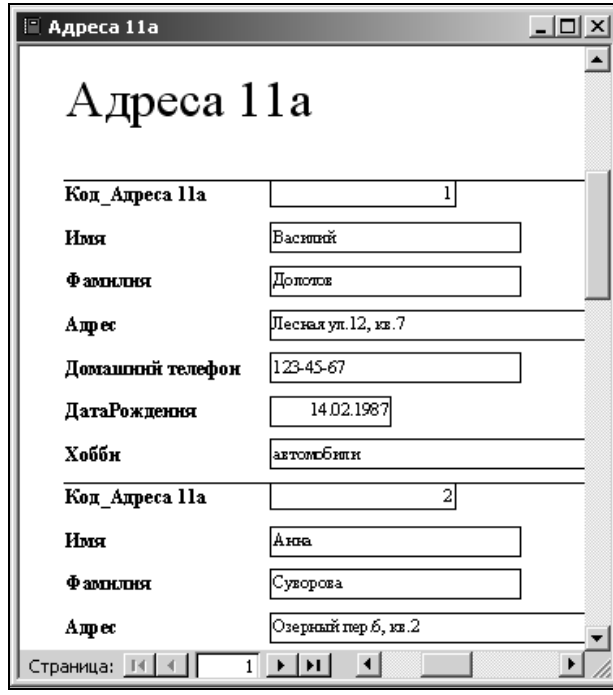


Рис. 11.13. Отчет `Адреса 11а`

Рассмотренный пример создания базы данных `Книга адресов` не охватил некоторые добавочные возможности и особенности построения объектов базы данных. К ним можно отнести вопросы связывания таблиц, вычислений в запросах, создания диаграмм и некоторые другие. Мало использовались панели инструментов. Как правило, рассматривался только один вариант подачи команд. Для более основательного знакомства с базами данных следует обращаться к специальной литературе.

Задания для самостоятельной работы

1. Создать базу данных планет Солнечной системы. Именами полей могут быть: название планеты, ее размеры, вес, удаленность от Солнца, число спутников, другие характеристики.
2. Создать базу данных европейских стран с полями: площадь, население, столица, денежная единица, религия и т. д.
3. Придумать и создать свою учебную базу данных, содержащую интересные справочные данные.

Глава 12

Алгоритмы и алгоритмизация задач

12.1. Понятие алгоритма. Свойства алгоритма

Алгоритмом называют строгую, полностью определенную последовательность действий с изменяемыми исходными данными, выполнив которую, получим искомый результат, решим поставленную задачу.

Пример 1

Записать алгоритм вычисления функции $F = \dfrac{ax + b}{cy}$.

В дальнейшем дроби часто будем записывать более экономно, как в языках программирования — в одну строку: $F = (ax + b)/(cy)$. При записи алгоритма будем использовать правила приоритета выполнения арифметических действий, *операцию присваивания* и вспомогательные переменные. Знак операции присваивания следующий: " := ". Запись a := 5 читают так: "Пусть *a* равно 5" или "Переменной *a* присвоим значение 5".

Запись алгоритма может быть такой:

1. z1 := ax;
2. z2 := z1 + b;
3. z3 := cy;
4. F := z2 / z3.

Использование вспомогательных переменных (z1, z2 и z3) упрощает записи алгоритмов вычисления сложных функций, делает их более обозримыми. В рассмотренной простой задаче они введены для примера.

Свойства алгоритмов

К свойствам алгоритмов относят: определенность, дискретность, результативность и массовость. Все эти свойства вытекают из определения алгоритма. Рассмотрим их подробнее.

Определенность требует от алгоритма быть строгим, четким, понятным. Все действия и символы операций должны быть или общепринятыми, или заранее четко и однозначно определены. Не допускаются двусмысленности, неоднозначности.

Например, последовательность действий:

1. y := a # b;
2. z := y @

не алгоритм, т. к. операции со знаками # и @ не определены.

Дискретность требует от алгоритма пошаговой записи и выполнения.

Результативность алгоритма предполагает обязательное получение результата. При этом, как говорят, "отрицательный результат — тоже результат". Например, если компьютер выдает на экран дисплея предусмотренное алгоритмом решения сообщение "Решение невозможно ввиду отрицательного подкоренного выражения", то такой алгоритм обладает свойством результативности. Если же запуск на выполнение программы, записанной в соответствии с некоторым алгоритмом, приводит к бесконечным вычислениям, то с результативностью такого алгоритма не все в порядке.

Массовость требует от алгоритма возможности применения его при различных значениях исходных данных, т. е. предполагается, что алгоритм должен содержать переменные величины.

Например, запись:

1. y := 3 + 5;
2. z := 2 × y

нельзя считать алгоритмом, т. к. она не удовлетворяет свойству массовости.

Одна из задач предмета информатики заключается в том, чтобы проследить преобразование информации по цепочке:

задача — алгоритм — программа — компьютер — результат решения.

- Задача формулируется на обычном разговорном языке, в виде формул, соотношений, зависимостей. Это этапы постановки, выбора модели и формализации задачи.
- Алгоритм разрабатывается и записывается одним из способов записи, на одном из формальных языков. Этап называют этапом разработки и записи алгоритма.

- Программа — это тот же алгоритм, но записанный на понятном компьютеру языке — языке программирования. Программа записывается на этапе программирования задачи.
- Компьютер переводит введенную программу с языка программирования на внутренний язык компьютера — язык машинных команд и решает задачу. Это этап решения задачи.
- Результат выдается компьютером в предусмотренном программой виде и анализируется пользователем.

12.2. Способы записи алгоритмов

Выбор способа записи зависит от характера задачи. Алгоритм вычислительного характера можно записать формулой, последовательностью формул. Алгоритм заваривания чая удобно записать словами в пронумерованных пунктах. Алгоритм решения квадратного уравнения будет наиболее понятен при записи словами и формулами.

Из формальных способов записи алгоритмов чаще других будем использовать язык блок-схем и алгоритмический язык. Заметим, что программа также является записью алгоритма на языке программирования.

Запись алгоритмов словами

Словесная запись алгоритма наиболее проста, не требует строгих форматов, правил. Обычно используется запись пронумерованными пунктами. Запишем словами, например, алгоритм решения известной детской логической задачи "Волк, коза, капуста и перевозчик".

Пример 2

Постановка задачи следующая: "На левом берегу реки находятся волк, коза, капуста и перевозчик с лодкой. Перевозчик должен переправить всех на правый берег так, чтобы не оставлять наедине волка с козой и козу с капустой. Как это сделать?"

1. Начало алгоритма.
2. Переправить на правый берег козу, оставив на левом волка и капусту.
3. Вернуться на левый берег, оставив козу на правом берегу.
4. Переправить на правый берег капусту.
5. Вернуться на левый берег с козой.
6. Переправить на правый берег волка.
7. Вернуться на левый берег, оставив на правом волка и капусту.

8. Переправить на правый берег козу.
9. Все в сборе на правом берегу.
10. Конец алгоритма.

В качестве упражнений запишите словами алгоритм перехода улицы, алгоритм заварки чая, алгоритмы решения других задач.

Блок-схемы алгоритмов

Записи алгоритмов на языке блок-схем обладают большой наглядностью. Хорошо просматривается структура алгоритма. Блок-схема представляет собой соединенные линиями блоки различной конфигурации (пример блоков показан на рис. 12.1). Вид блоков и последовательность их соединения соответствуют типу и последовательности действий алгоритма.

Ограничимся сокращенным набором блоков.

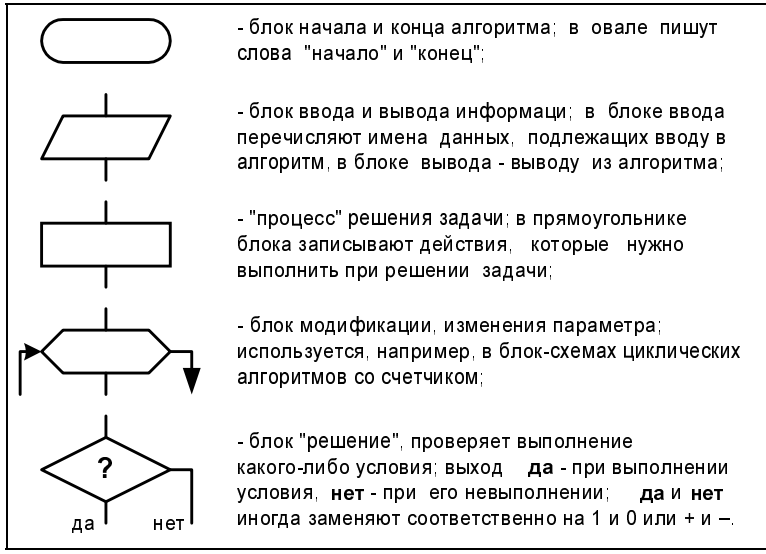


Рис. 12.1. Блоки блок-схем алгоритмов

Алгоритмический язык

Алгоритмический язык — это язык, предназначенный для записи алгоритмов. Как и любой другой язык, он включает: набор символов (алфавит), правила записи алгоритмов (синтаксис) и правила истолкования записей (семантику).

Учебный алгоритмический язык использует русский алфавит, синтаксис и семантику наиболее применяемых языков программирования. Поэтому он является как бы ступенью для понимания и освоения многих языков программирования.

Алгоритмический язык ввиду своей распространенности, простоты и универсальности широко используется для записи алгоритмов решения учебных задач, а также различного рода учебных проверочных и контролирующих тестов. Заметим, что правила записи алгоритмов на школьном алгоритмическом языке могут различаться, но это не затрудняет их чтение и понимание.

Запись алгоритмов на алгоритмическом языке требует определенной строгости и четкости. Все служебные слова выделяются (например, подчеркиваются). Необходимо выдерживать правила смещения записей друг относительно друга. Это смещение подчеркивает структуру алгоритма, делает его легко читаемым, более понятным.

В обобщенной схеме одного из вариантов записи алгоритма на алгоритмическом языке названия содержимого записей взяты в треугольные скобки:

алг <название алгоритма> (<список переменных и их типы>)

арг <список аргументов>

рез <список результатов>

нач <ввод вспомогательных переменных>

<присваивание начальных значений>

<последовательность действий в соответствии с алгоритмом>

выв <список данных, выводимых на печать >

кон

В информатике все величины по своему типу делятся на: числовые, строковые и логические. Строковые величины еще иногда называют литерными, символьными, текстовыми

Среди числовых величин различают:

- вещественные числа, обозначение — вещ X;
- целые числа, обозначение — цел X;
- натуральные числа, обозначение — нат X.

Величины строкового типа обозначаются стр X или лит X и представляют собой символы или наборы символов (тексты). Как правило, они записываются в кавычках:

стр X := "вариант 5", лит S := "счастливый билет".

Логические величины могут принимать только два значения: 1 — True (истина), и 0 — False (ложь). Будем их обозначать так: лог X.

12.3. Линейные алгоритмы

Различают три типа алгоритмов: *линейные*, *ветвящиеся* и *циклические*. Их названия определяются входящими в них типовыми алгоритмическими конструкциями, которые также называют *базовыми структурами*. К основным базовым структурам относят: *следование*, *ветвление* и *цикл*. Доказано, что этих трех основных базовых структур достаточно, чтобы построить алгоритм любой сложности.

Самые простые по структуре — линейные алгоритмы. Они не имеют ветвлений и циклов. В блок-схемах отсутствуют блоки "решение" и обратные связи, позволяющие многократно выполнять некоторые действия.

Пример 3

Для двух целых чисел A и B записать на алгоритмическом языке и языке блок-схем алгоритм определения их суммы S, разности R, произведения P и среднего арифметического SR.

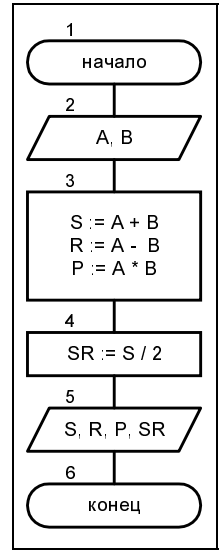


Рис. 12.2. Блок-схема линейного алгоритма

Решение.

На алгоритмическом языке алгоритм решения задачи записан ниже. Блок-схема алгоритма показана на рис. 12.2.

```
алг Пример 1 (цел A, B, S, R, P, вещ SR)
        арг A, B
        рез S, R, SR
нач
        S := A + B
        R := A − B
        P := A * B
        SR := S / 2
        выв S, R, P, SR
кон
```

Задачи для самостоятельного решения

1. Переменной A присвоить ее значение, увеличенное на величину C.
2. Переменной A присвоить ее значение, увеличенное в C раз.
3. Поменяйте друг с другом значения двух переменных A и B, воспользовавшись для временного хранения третьей переменной C.
4. Поменяйте друг с другом значения двух переменных, не используя третью переменную.
5. Имеются две переменные A и B. Переменной A присвойте их сумму, а переменной B — их разность. Третью переменную не использовать.
6. Чему станет равна переменная A в результате выполнения следующего алгоритма: 1) A := 2; 2) A := 3A − A; 3) A := 4A − A.
7. Определите, чему будут равны переменные A и B в результате выполнения алгоритма: 1) A := 3; 2) B := 5; 3) A := 2A − B; 4) B := 5A − B; 5) A := A − A.

Для следующих задач запишите алгоритмы их решения на алгоритмическом языке и в виде блок-схем:

1. Заданы стороны треугольника. Определите его периметр и площадь.
2. Заданы стороны прямоугольника. Определите его площадь, периметр и длину диагонали.
3. Заданы радиус основания и высота цилиндра. Определите площадь его поверхности и объем.

4. Заданы длина, ширина и высота параллелепипеда. Определите его объем и площадь поверхности.
5. Два отрезка прямой на плоскости заданы координатами своих концов. Запишите алгоритм определения суммы длин этих отрезков.

12.4. Ветвящиеся алгоритмы

Ветвящиеся алгоритмы содержат базовую структуру *ветвление*. Они содержат блок "решение", который может иметь два и более альтернативных выходов. При работе алгоритма в зависимости от выполнения условий выбирается один из этих выходов, и выполняются соответствующие ему действия. Ветвящиеся алгоритмы, как правило, включают в себя более простую базовую структуру — *следование*.

Пример 4

Вычислить значение величины c, определяемое по формулам: $c = a + b$, если $a \le b$, и $c = a - b$, если $a > b$.

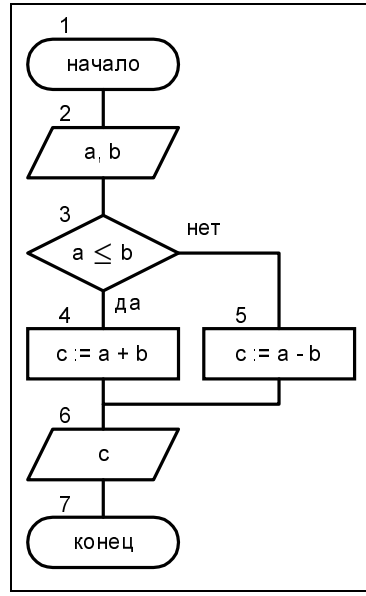


Рис. 12.3. Блок-схема ветвящегося алгоритма

Блок-схема алгоритма показана на рис. 12.3. Запись алгоритма на алгоритмическом языке следующая:

```
алг Задача. (вещ a,b,c)
    арг a, b
    рез c
нач
    если a ≤ b
        то c := a + b
        иначе c := a − b
    все
    выв c
кон
```

Рассмотрим работу алгоритма решения задачи по его блок-схеме. Пусть блок 2 вводит значения данных $a = 5$, $b = 2$. Блок 3 проверяет условие $a \le b$. Условие не выполняется. Поэтому следующим выполняется действие в блоке 5: $c = a - b = 5 - 2 = 3$. Блок 6 выводит результат $c = 3$. Конец алгоритма.

Проследите работу алгоритма при $a = b = -4$, $a = 3$ и $b = -3$, $a = b = 0$, других значениях исходных данных.

Пример 5

Выполним алгоритмизацию шуточной задачи "Счастливый билет".

Постановка задачи

Автобусный билет считают "счастливым", если сумма трех первых цифр номера билета равна сумме трех последних цифр. Требуется записать алгоритм определения билета, имеющего счастливый номер.

Формализация задачи

Введем обозначения и запишем основные соотношения.

Пусть цифры билета: a, b, c, d, e, f.

Суммы цифр: $C1 := a + b + c$; $C2 := d + e + f$.

Кроме того, обозначим: S := "билет счастливый", N := "билет несчастливый", R – результат.

В итоге имеем:

$$R = \begin{cases} S, & \text{если } C1 = C2; \\ N, & \text{если } C1 \neq C2. \end{cases}$$

Запись алгоритма

Запись алгоритма на алгоритмическом языке следующая:

```
алг Счастливый билет (цел a, b, c, d, e, f, стр R)
    арг a, b, c, d, e, f
    рез R
нач цел C1, C2, стр S, N
    стр S := "билет счастливый"; стр N := "билет несчастливый"
    C1 := a + b + c; C2 := d + e + f
    если C1 = C2
        то R := S
        иначе R := N
    все
    выв R
кон
```

При записи блок-схемы (рис. 12.4) показан способ отображения связи блоков при переносах части блоков на другой лист или другую страницу.

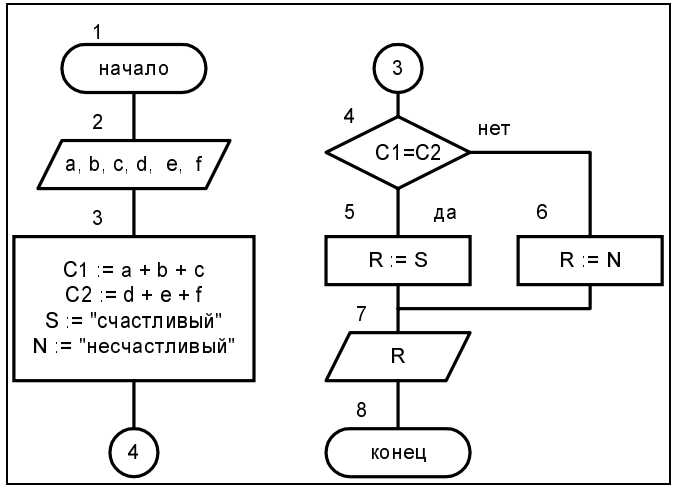


Рис. 12.4. Блок-схема алгоритма "Счастливый билет"

Полная и сокращенная формы ветвления

В обычном разговорном языке в условном предложении "если не будет дождя, то мы пойдем гулять, иначе будем читать книги" содержится два действия, из

которых выполняется только одно. Условное предложение "если не будет дождя, то мы пойдем гулять" содержит одно действие. Первое предложение соответствует полной форме ветвления, второе — сокращенной форме.

Блок-схемой и на алгоритмическом языке полная и сокращенная формы записываются следующим образом (рис. 12.5).

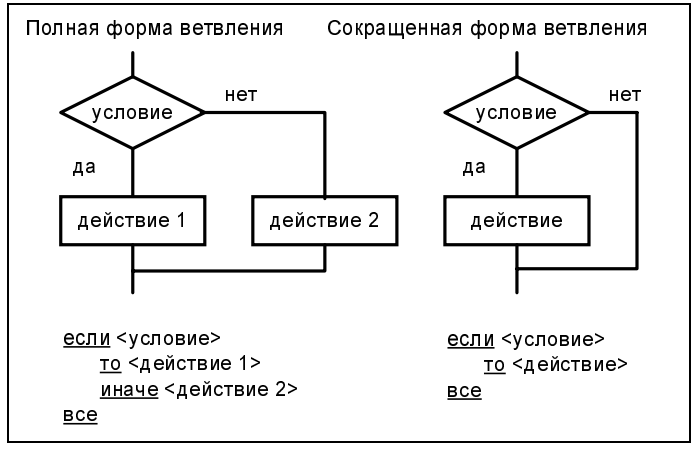


Рис. 12.5. Полная и сокращенная формы ветвления

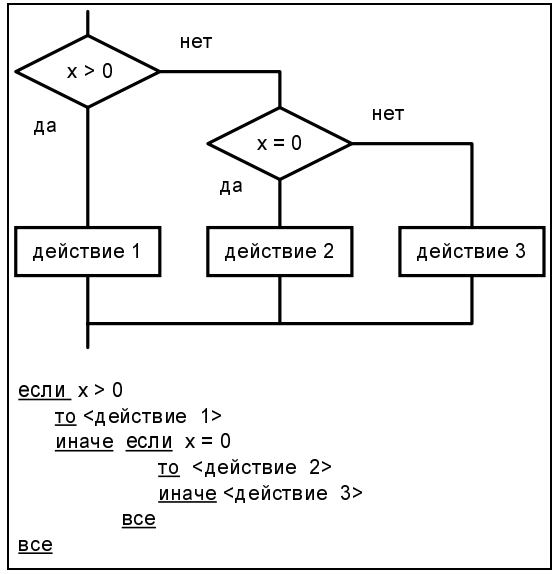


Рис. 12.6. Вариант построения блок-схемы ветвления на три выхода

Нередко в задачах проверяются условия, соответствующие трем и более выходам. Например, если выполнение условий $x > 0$, $x = 0$, $x < 0$ требуют трех различных действий, то структура ветвления может быть такой, как показано на рис. 12.6.

Если число альтернатив больше трех, то в таких случаях, как правило, используют конструкцию *"выбор"* или *"выбор — иначе"*.

Ветвление типа "выбор"

выбор
  при условие 1: действия 1
  при условие 2: действия 2
    . . .
  при условие N: действия N
все

Ветвление типа "выбор — иначе"

выбор
  при условие 1: действия 1
  при условие 2: действия 2
    . . .
  при условие N: действия N
  иначе действия N + 1
все

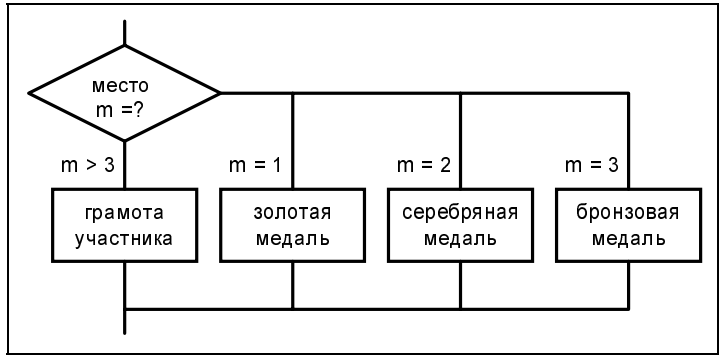


Рис. 12.7. Блок-схема алгоритма *"выбор — иначе"*

В соответствии с рассмотренными на алгоритмическом языке вариантами ветвлений изображаются и блок-схемы. Ниже приведен пример записи и блок-схема (см. рис. 12.7) алгоритма решения задачи для варианта "выбор — иначе".

выбор
  при 1-е место: золотая медаль

при 2-е место: серебряная медаль

при 3-е место: бронзовая медаль

иначе: грамота участника

все

Применение сложных условий

Условия, имеющие знаки отношения (>, <, =, ≤, ≥), рассматриваются как логические переменные, которые могут принимать значение True (истина), если условие выполняется, или False (ложь) в противном случае. В блок-схемах выходы блока "решение" соответствующие выполнению или невыполнению условия соответственно помечаются словами и знаками *да* (1, +) — истина и *нет* (0, −) — ложь.

Алгоритмы многих задач записываются намного проще, если в них используются сложные, составные условия. Для связи простых условий будем применять логические связки И, ИЛИ, НЕ. Например, если требуется задать проверку условия нахождения величины z в границах отрезка $[a, b]$, то в алгебре это записывают так: $a \le z \le b$, а на алгоритмическом языке это можно записать так:

если z ≥ a И z ≤ b

то z находится в границах отрезка [a, b]

все

В случае если, например, нужно проверить, находится ли z в хотя бы одном из отрезков $[a, b]$ и $[c, d]$, запись будет следующей:

если z ≥ a И z ≤ b ИЛИ z ≥ c И z ≤ d

то z находится в одном из отрезков [a, b] или [c, d]

все

Рассмотрим пример решения задачи с использованием сложного условия.

Пример 6

Заданы длины a, b и c трех отрезков прямой (a, b и c — целые, положительные числа). Можно ли использовать эти отрезки в качестве сторон треугольника?

Решение.

Условие, при выполнении которого треугольник построить нельзя, следующее:

a > b + c ИЛИ b > a + c ИЛИ c > a + b

Условие, при котором треугольник построить можно, записывается двумя вариантами:

НЕ (a > b + c ИЛИ b > a + c ИЛИ c > a + b) или

a < b + c И b < a + c И c < a + b.

Для первого варианта алгоритм решения задачи записывается так:

```
алг Построение треугольника (цел a, b, c, стр z)
    арг a, b, c
    рез z
нач
    если НЕ (a > b + c ИЛИ b > a + c ИЛИ c > a + b)
      то z := "Можно"
      иначе z := "Нельзя"
    все
    выв z
кон
```

Задания для самостоятельного решения

Замечание

При решении задач желательно придерживаться рассмотренной последовательности действий. После *постановки задачи* следует этап *разработки алгоритма* её решения, который совмещается с *формализацией задачи*. При формализации вводятся обозначения, присваиваются имена переменным и константам, записываются расчетные формулы, соотношения. При хорошей формализации задачи увеличивается вероятность ее правильного решения. К тому же без соответствующих определений значения введенных обозначений со временем забываются и на восстановление их смысла требуется время. Четкая и подробная формализация задачи упрощает запись алгоритма. На этапе *записи алгоритма* также следует строго придерживаться рассмотренных выше правил.

Записать алгоритмы решения следующих задач.

1. Заданы два разных по величине числа. Определить, какое из этих чисел больше.
2. При вводе двух чисел выводится одно из трех сообщений: первое число больше, второе число больше, числа равны.

3. Заданы три числа. Определить, есть ли среди них хотя бы одна пара равных по величине (на выходе "Да") или нет такой пары (на выходе "Нет").
4. Заданы три натуральных числа. Вывести "Нет", если среди чисел нет равных, "Да", если есть одна пара равных, и "Все равны", если все числа равны.
5. Заданы длины трех отрезков прямой. Определить, могут ли они являться сторонами треугольника.
6. Известны длины сторон двух треугольников *a*, *b*, *c* и *d*, *e*, *f*. Определить, какой треугольник (первый или второй) имеет бóльшую площадь.
7. Заданы длины сторон треугольника. Определить, является ли этот треугольник равнобедренным.
8. Заданы длины сторон треугольника. Определить, является ли этот треугольник прямоугольным.
9. Заданы три различных по величине числа. Определить, какое из них наибольшее.
10. Заданы три положительных числа *a*, *b* и *c*. Определить, являются ли они последовательно стоящими элементами арифметической (геометрической) прогрессии. Если да, то напечатать значение разности (знаменателя) прогрессии.
11. Заданы три стороны треугольника x, y и *z*. Определить, является ли треугольник прямоугольным. Если да, то напечатать сообщение о том, какая сторона служит гипотенузой.
12. Заданы длины *a*, *b*, *c* и *d* четырех отрезков прямой. Проверить, могут ли эти отрезки быть сторонами квадрата, прямоугольника.

12.5. Структуры данных. Массивы

Структуры данных определяют классификацию данных и отношения между ними. Структуры могут быть простыми (элементарными) и сложными (составными). Различают следующие структуры: константу, переменную, массив, запись и таблицу. Константу и переменную можно считать элементарными данными, единичными, неделимыми элементами более сложных систем организации данных.

Константа — это число, текст или логическое значение, которые не изменяются в процессе реализации алгоритма, решения задачи на компьютере.

Переменная — это единица организации данных, которой в процессе обработки информации могут присваиваться различные значения. Переменная имеет имя — *идентификатор* и тип: числовой, строковый, логический. Переменные могут сопровождаться словами, указывающими на их тип.

Массив — это объединенное одним именем (идентификатором массива) множество однотипных элементов. К основным параметрам массивов относят его тип (числовой, строковый, логический), размерность (одномерный, двухмерный и т. д.) и размер (количество элементов массива в каждом измерении).

Виды записи массива на различных языках могут различаться. Например, массив, задающий значения роста учащихся некоторого класса, имеющего N учеников, на алгоритмическом языке может быть задан одномерным массивом так: нат R[1:N]. Величина N — максимальный номер элементов массива. Этот же массив на языке программирования может быть записан так: `r(1 TO n)`.

Элементами массива являются переменные с номерами (индексами). Имена переменных совпадают с именем массива. Пусть задан массив роста каждого ученика класса. Тогда массив R[1:N] можно раскрыть следующим образом:

R[1:N] = [156, 162,... ,R[i},..., 164];

R[1:N] = [R[1], R[2],...,R[i],...,R[N]],

где индексы (в квадратных скобках) определяют номера элементов массива. R[i] — любой (i-й) элемент массива.

В примере рассмотрен одномерный (линейный) массив. Примером двухмерного массива может служить таблица умножения нат T[1:9,1:9]. В нем каждый элемент T[i, j] равен произведению индексов. Индексы в двухмерном массиве определяют положение элемента в таблице: i — номер строки, j — номер столбца. Заметим, что не только двухмерный, но и одномерный (линейный) массив иногда называют таблицей и на алгоритмическом языке пишут, например, таб цел G[1:N].

Запись — это такая структура организации данных, которая позволяет объединять данные разных типов. Элементами записи являются поля. Запись и поля записи имеют имена. Каждое поле может быть переменной, массивом, записью более низкого уровня. Примером записи может служить одна строка таблицы.

Таблица может быть определена как объединение записей (строк) или как объединение массивов (столбцов, полей). При этом массивы могут быть разного типа. В приведенной ниже таблице (табл. 12.1) собраны сведения об учениках класса: номер по порядку, фамилия, год рождения, рост, вес и участие в спортивной жизни школы (1 — да, 0 — нет).

Таблица 12.1. *Пример таблицы*

N	Фамилия *F*[1:*N*]	Год рожд. *G*[1:*N*]	Рост *R*[1:*N*]	Вес *W*[1:*N*]	Спорт *S*[1:*N*]
1	Иванов	1984	156	52.2	0
2	Петров	1983	162	61.5	1

Таблица 12.1 *(окончание)*

N	Фамилия *F*[1:*N*]	Год рожд. *G*[1:*N*]	Рост *R*[1:*N*]	Вес *W*[1:*N*]	Спорт *S*[1:*N*]
...	...	...	...	...	...
N	Сидоров	1984	164	59.4	0

Приведенная табл. 12.1 состоит из шести столбцов. Каждый столбец — массив (поле). Перечислим эти массивы и запишем на алгоритмическом языке.

Первый массив — номера учеников в списке. Это натуральный ряд чисел нат N[1:N].

Второй массив — фамилии учеников — массив строковых величин стр F[1:N].

Третий и четвертый массивы — год рождения и рост — массивы целых чисел цел G[1:N] и цел R[1:N].

Пятый массив — вес учеников — массив вещественных чисел вещ W[1:N].

Последний, шестой массив — занятие спортом. Это массив логических величин лог S[1:N].

Строки таблицы занимают записи. Например, запись во второй строке таблицы следующая:

"нат N[2] стр *F*[2] цел G[2] цел R[2] вещ W[2] лог S[2]", что соответствует обычной записи:

"2 Петров 1983 162 61.5 1".

12.6. Циклические алгоритмы

Циклические алгоритмы содержат базовую структуру *цикл*. Они могут также включать участки, характерные как для линейных, так и для ветвящихся алгоритмов. От линейных и ветвящихся алгоритмов циклические алгоритмы отличаются наличием в структуре алгоритма *обратной связи*, которая позволяет некоторые действия повторять многократно, циклически. Эти действия составляют *тело цикла*.

Циклические алгоритмы делятся на алгоритмы с известным и неизвестным заранее числом повторений. Их условно называют соответственно алгоритмы с циклом типа *"для"* и алгоритмы с циклом типа *"пока"*. В обоих случаях окончание циклического процесса определяется поставленным заранее условием.

В циклических алгоритмах типа *"для"* этим условием служит явно заданное количество повторений. Блок-схемой и на алгоритмическом языке такой

тип базовой структуры "цикл" записывается в общем виде, который показан на рис. 12.8.

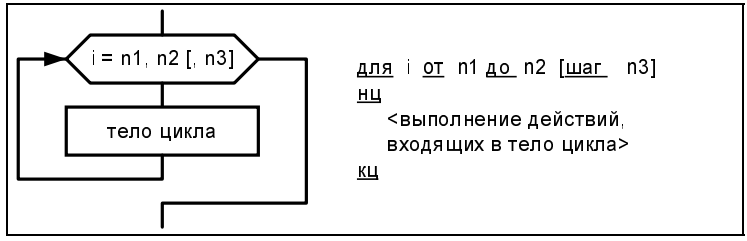


Рис. 12.8. Структура цикла типа "для"

При решении задачи определения суммы четных натуральных чисел от 2 до 100 структура цикла типа "для" может быть записана так, как показано на рис. 12.9.

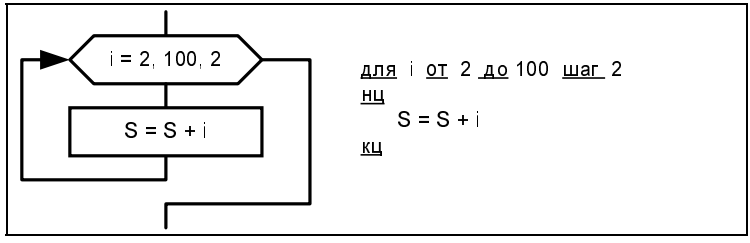


Рис. 12.9. Пример записи цикла типа "для"

В примере использован блок модификации параметра. В общем виде запись внутри блока следующая:

<параметр> = <начальное значение>, <конечное значение> [, <шаг изменения>]

Запись i = 2, 100, 2 указывает на то, что параметр i изменяется от начального значения 2 до конечного значения 100 с шагом 2, т. е. последовательно принимает четные значения 2, 4, 6, ..., 100. Заметим, что шаг, равный единице, не записывается, он принимается "по умолчанию".

В циклических алгоритмах с неизвестным заранее числом повторений (циклы типа *"пока"*) явно число повторений не задано, как, например, в задаче с такой постановкой: "Сколько нужно взять, начиная с единицы, последовательно расположенных чисел натурального ряда, чтобы их сумма превысила 1000?".

В виде блок-схемы и на алгоритмическом языке в общем виде структуру цикла типа "пока" можно записать так, как показано на рис. 12.10.

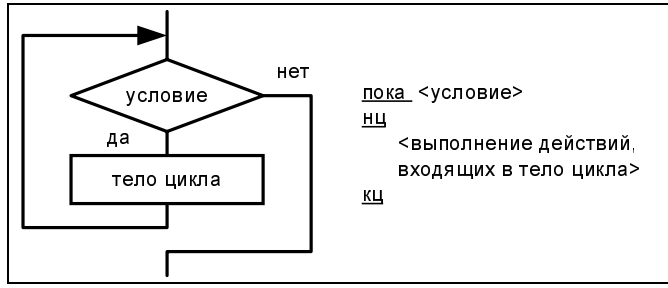


Рис. 12.10. Структура цикла типа "пока"

При решении задачи определения количества натуральных чисел начиная с единицы, сумма которых превысит 1000, структура цикла "пока" может быть записана, как показано на рис. 12.11.

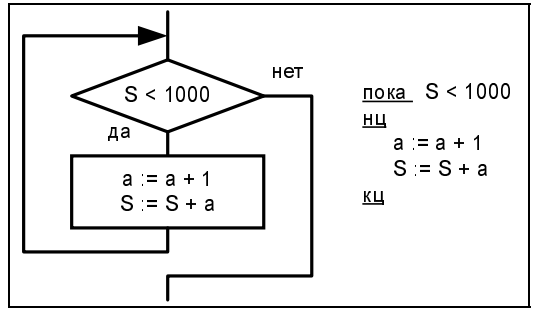


Рис. 12.11. Пример записи алгоритма типа "пока"

Характерными примерами задач, требующих для своего решения построения циклических алгоритмов, служат различные задачи обработки массивов. Счетчик циклов (счетчик повторений), работающий по формуле i := i + 1, включается в блок модификации параметра и может последовательно перебрать все индексы — номера элементов обрабатываемого массива. Перебрать все номера требуется, например, для получения суммы (S) элементов (R[i]) массива R[1:N], которая подсчитывается в цикле по формуле S := S + R[i].

Перед началом цикла сумме и другим изменяемым в процессе работы алгоритма переменным могут присваиваться начальные значения.

Рассмотрим задачу, требующую построения циклического алгоритма с известным заранее числом повторений — цикл типа "для", цикл "со счетчиком".

Пример 7

Постановка задачи

Известны показатели роста спортсменов стартовой пятерки баскетбольной команды: 195, 205, 202, 198, 192. Требуется разработать и записать алгоритм определения среднего роста спортсменов.

Формализация задачи

Для решения задачи требуется построить циклический алгоритм с заранее известным числом циклов. Это цикл типа "для", цикл "со счетчиком".

Рост спортсменов представим в виде одномерного массива целых чисел (линейной таблицы):

цел таб R[1:5] =

цел таб R[195, 205, 202, 198, 192] =

цел таб R[R[1], R[2], R[3], R[4], R[5]].

Введем вспомогательную переменную S. Это сумма элементов массива, которую после выхода из цикла разделим на число спортсменов и получим искомый результат.

В алгоритме будем использовать следующие формулы:

- S := 0 — присвоение начальных значений;
- S := S + R[i] — накапливание суммы элементов массива, в каждом цикле новая сумма равна старой сумме плюс величина очередного, i-го элемента массива;
- SR := S / 5 — определение среднего роста баскетболистов.

Решение.

Начнем с блок-схемы алгоритма (рис. 12.12).

В блок-схеме алгоритма решения задачи буквами "ос" обозначена линия обратной связи, обеспечивающей цикличность процесса вычислений. Для проверки условия окончания циклического процесса использован блок модификации параметра i.

Запись алгоритма на алгоритмическом языке следующая:

```
алг Средний рост ( цел таб R[1:5], вещ SR)
    арг R
    рез SR
нач цел S; S := 0
    для i от 1 до 5
```

```
нц
    S := S+R[i]
кц
SR := S/5
выв SR
кон
```

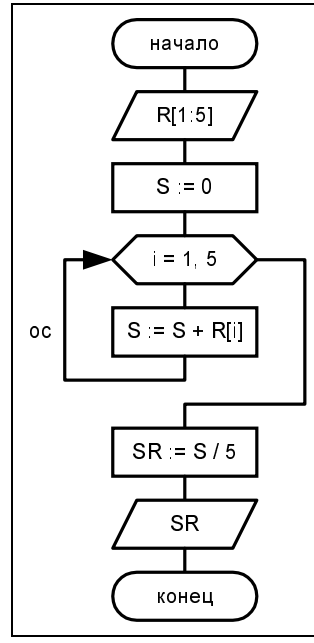


Рис. 12.12. Блок-схема циклического алгоритма "Средний рост"

Для наглядности работу алгоритма можно развернуть так, как показано на рис. 12.13.

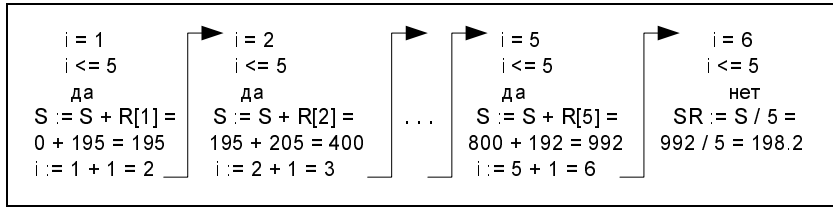


Рис. 12.13. Развернутый по циклам алгоритм "Средний рост"

В такой записи значение параметра i соответствует порядковому номеру повтора. Ниже следует проверка условия продолжения циклов. Если условие выполняется, то выполняются действия в теле цикла: накопление суммы и увеличение на единицу значения счетчика. Стрелкой показана обратная связь, обеспечивающая циклический процесс. После пятого цикла значение параметра (i = 6) обеспечивает выход из цикла, после чего вычисляется среднее значение роста баскетболистов.

Алгоритм решения следующей задачи — циклический типа "пока", с неизвестным заранее числом циклов, циклический алгоритм "без счетчика".

Пример 8

Постановка задачи

Заданы два целых числа *A* и *B*. Определить их наибольший общий делитель.

Запись алгоритма решения

Для определения наибольшего общего делителя (НОД) двух чисел используют алгоритм Евклида, один из вариантов которого можно записать следующей последовательностью пунктов.

1. Начало алгоритма.
2. Сравниваем числа *A* и *B* по величине. Если $A \neq B$, то переход к п. 3, иначе к п. 4.
3. Если $A > B$, то $A = A - B$, иначе $B = B - A$. Переход к п. 2.
4. НОД = *A*.
5. Конец алгоритма.

Работу алгоритма покажем на конкретном примере. Пусть $A = 55$, $B = 15$. Шаги работы алгоритма оформим в виде таблицы (табл. 12.2).

Таблица 12.2. *Пример работы алгоритма*

A	55	40	25	10	10	5
B	15	15	15	15	5	5

Анализ алгоритма показывает, что это циклический алгоритм с заранее неизвестным числом циклов. Выход из цикла осуществляется по условию равенства чисел *A* и *B*.

Блок-схема алгоритма показана на рис. 12.14. Запись алгоритма Евклида на алгоритмическом языке следующая:

```
алг Алгоритм Евклида (цел A, B, НОД)
    арг A, B
    рез НОД
нач
    пока A ≠ B
    нц
        если A > B
            то A = A − B
            иначе B = B − A
        все
    кц
    НОД = A
    выв НОД
кон
```

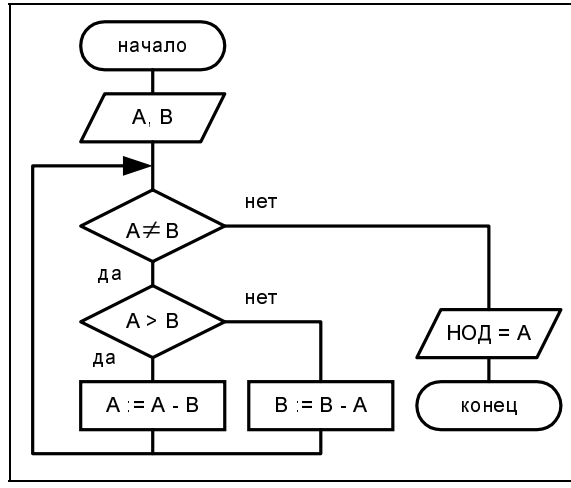


Рис. 12.14. Блок-схема алгоритма Евклида

Перед тем как предложить варианты заданий для самостоятельной работы по закреплению материала, запишем алгоритм решения одного из вариантов типовых задач. Варианты даны в конце главы.

Пример 9

Постановка задачи

Задан массив целых чисел A[0:N]. Записать алгоритм определения среднего арифметического положительных элементов и количества нулевых элементов массива.

Формализация задачи

Введем следующие обозначения:

- S и $K1$ — соответственно сумма и количество положительных чисел массива ($A[i] > 0$); вычисляются по формулам $S = S + A[i]$ и $K1 = K1 + 1$;
- $SR = S / K1$ — среднее арифметическое положительных чисел $A[i] > 0$;
- $K0$ — количество нулевых элементов массива; $K0 = K0 + 1$, если $A[i] = 0$.

Блок-схема алгоритма приведена на рис. 12.15. Запись алгоритма на алгоритмическом языке показана ниже.

```
алг Задача (цел таб A[0:N], вещ SR, цел N, K0)
    арг N, A[0:N]
    рез SR, K0
нач цел S1, цел K1, i, S1 := 0, K0 := 0, K1 := 0
    для i от 0 до N
    нц
        если A[i] > 0
            то S := S + A[i]; K1 := K1 + 1
        все
        если A[i] = 0
            то K0 := K0 + 1
        все
    кц
    SR := S / K1
    выв SR, K0
кон
```

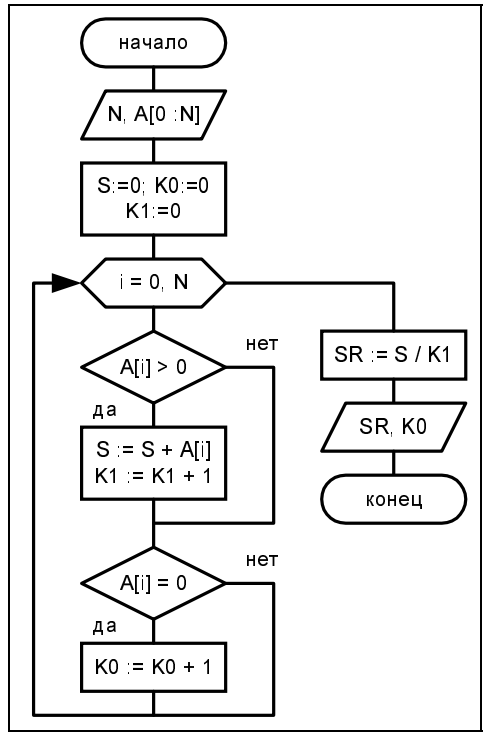


Рис. 12.15. Блок-схема алгоритма

Задания для самостоятельной работы

Составьте таблицу вариантов проверочной работы по записи циклических алгоритмов обработки массивов. Строки таблицы — номера вариантов. Столбцы соответствуют пунктам задания. В качестве примера покажем первый вариант работы (табл. 12.3).

Таблица 12.3. *Вариант задания*

№	*A*[0:*N*]	*B*[1:*M*]	*S*>0	*S*<0	*S* все	*K*>0	*K*<0	*K*=0	*K*<*C*	*SR* все	*SR*>0
1	×		×				×	×			×

Первые две колонки задания отмечены именами массивов с различным заданием размерности. Начиная с третьей колонки, записаны задания: найти сумму положительных элементов массива, сумму отрицательных, сумму всех, количество положительных, отрицательных, равных нулю, меньших заданного

числа *C*, среднее арифметическое всех, среднее положительное положительных. Задания, которые вошли в первый вариант, отмечены крестиками.

Число заданий можно увеличить, добавив, например, изменение знака отрицательных, положительных чисел, определение количества чисел в массиве больших заданного числа *C*, равных ему, определение среднего арифметического отрицательных чисел, рассмотрение в массиве элементов только с четными или нечетными номерами и т. д. Желательно в каждом варианте, кроме задания массива, иметь не более трех-пяти заданий (крестиков).

В составленной таблице выберите вариант работы и решите его. Вариант задачи, показанной в примере, может быть записан следующим образом.

В массиве целых чисел $A[0:N]$ определить сумму положительных, количество отрицательных, количество равных нулю элементов и среднее арифметическое положительных элементов массива. Алгоритм решения записать в виде блок-схемы и на алгоритмическом языке.

Глава 13

Visual Basic

13.1. Общие сведения о языках программирования

Для решения задачи на компьютере необходимо исходные данные задачи и алгоритм ее решения записать на формальном языке программирования в виде программы, ввести программу и данные в память компьютера и подать компьютеру команду на исполнение программы. По степени зависимости языков программирования от структуры конкретного компьютера их условно можно разделить на языки низкого и высокого уровня.

К языкам *низкого уровня* относят языки машинных команд конкретных компьютеров. Программа на машинном языке представляет собой последовательность команд, содержащих коды выполняемых компьютером операций и адреса участвующих в этих операциях операндов. Это *машинно-зависимые* языки.

Программа, записанная на машинном языке одного компьютера, не всегда может быть выполнена на другом компьютере ввиду различий в системах команд и структурах этих компьютеров. Такие программы отличает высокое быстродействие и малые затраты памяти. Однако программирование на машинных языках связано с большими затратами времени, поэтому в настоящее время применяется редко.

Желание облегчить программирование привело к разработке языков, в которых многие действия, затрудняющие программирование на машинном языке, переданы компьютеру. Вместо конкретных адресов и кодов операций применяется символика, используются макрокоманды, объединяющие несколько действий в одной команде. К этим языкам относят так называемые *автокоды* и *языки ассемблера* — переводчика автокодов на машинный язык. Языки ассемблера позволяют наиболее полно использовать все возможности структуры компьютера.

Переводчики программ с языка программирования на машинный язык называют *трансляторами*. Различают два вида трансляторов. *Трансляторы-интерпретаторы* переводят и исполняют программу построчно, строку за

строкой. Недостатком простых по структуре трансляторов интерпретирующего типа является необходимость выполнять свою работу с каждым запуском программы и низкое быстродействие получаемого в результате трансляции объектного кода программы.

Трансляторы-компиляторы просматривают всю программу в целом, создают оптимальный по быстродействию объектный код и сохраняют его в памяти. Трансляторы компилирующего типа сложнее интерпретаторов, но создают более быстродействующие программы.

Языки программирования *высокого уровня — машинно-независимые.* Достаточно только, чтобы компьютер имел транслятор с данного языка на свой машинный язык. Языки программирования высокого уровня делят на проблемно-ориентированные и универсальные.

Проблемно-ориентированные языки разрабатываются с целью сделать более эффективным программирование задач одного класса, например, задач экономических (Кобол), искусственного интеллекта (Лисп, Пролог), обучения (ЛОГО) и т. п. В Интернете используются язык гипертекстовой разметки HTML, язык моделирования виртуальной реальности VRML.

Универсальные языки программирования позволяют успешно решать широкий круг задач различного характера. Среди них следует назвать один из первых процедурно-ориентированных языков Фортран (1950), который и в настоящее время продолжает развиваться и совершенствоваться, Алгол-60 (1960). К ним относятся также последние версии Бейсика, Паскаль, Си и др.

Как правило, каждый из языков программирования в течение своей "жизни" пополняется, модернизируется, усиливается. Так появляются новые версии языка. Например, язык Бейсик, предложенный профессором математики Дортмутского колледжа (США) Томасом Куртцем как простой язык для начинающих, постепенно превратился в мощные универсальные языки высокого уровня: Quick Basic и Visual Basic.

С точки зрения развития *пользовательского интерфейса* (средств, обеспечивающих удобство работы пользователя), этапы развития версий языка Бейсик во многом совпадают с этапами развития системного программного обеспечения (СПО). Если у СПО этапы можно назвать "DOS — Оболочки — Windows", то у Бейсика это "Язык — Среда — Система".

На первом этапе СПО — это *DOS* (дисковая операционная система), в которой общение с компьютером сводится к набору на клавиатуре команд в командной строке почти пустого экрана. Бейсик на этом этапе — это только *язык* (компилятор или интерпретатор), без оформления в удобную для использования среду. На пустом экране только приглашение **OK** вводить строки программы. Бейсик в те времена в полной мере заслуживал название языка для начинающих и писался в виде аббревиатуры BASIC (Beginner's

ALL-purpose Symbolic Instruction Code — Всецелевой язык программирования для начинающих).

На втором этапе в СПО появились оболочки. Еще недавно популярная оболочка Norton Commander существенно упростила работу пользователя с файлами, каталогами. Это был первый большой шаг по совершенствованию пользовательского интерфейса. Бейсик — это уже *среда* со своими меню, горячими клавишами, встроенной справкой. Язык, оставаясь простым в освоении, превратился в мощный процедурный язык высокого уровня и принимается фирмой Microsoft в качестве базового. Название этого самого распространенного в мире языка стали писать — Basic (базовый).

На третьем этапе появляются сначала графические оболочки Windows 3.*x*, и затем операционная система Windows — шедевр фирмы Microsoft. Бейсик — это уже Visual Basic — популярная *система программирования*.

Одним из перспективных направлений развития технологий программирования считается создание *объектно-ориентированных* языков.

Это системы нового поколения. В качестве основного понятия систем объектно-ориентированного программирования выступает объект. В системах объектно-ориентированного программирования широко используется графический интерфейс. Процесс программирования сводится к конструированию программы с помощью мыши из готовых или вновь создаваемых объектов и записи кода, определяющего свойства и функционирование этих объектов. Принцип визуального программирования в большей степени реализован в системах объектно-ориентированного программирования Delphi (Дельфи), Visual Java ("ява" или "джава"), Visual Basic, в других системах.

Приложения, функционирующие под управлением Windows ("под Windows"), в том числе Word, Excel и Access, используют версии Бейсика в качестве языков управления, для построения макросов. Все это указывает на хорошие перспективы самого популярного в мире языка.

При обсуждении выбора начального языка программирования многие отмечают несомненные преимущества последних версий Бейсика перед другими языками. При очевидной простоте освоения они в большей степени "заставляют думать", помогают развивать воображение и чувство стиля, позволяют прививать хорошие начальные программистские навыки.

В учебнике язык Visual Basic рассматривается как средство, позволяющее продолжить цепочку "задача — алгоритм — программа — компьютер — результат". Операторы и функции языка, их синтаксис даются в объеме, необходимом для демонстрации тех или иных особенностей реализации алгоритмов.

13.2. Система программирования Visual Basic 6.0

Алфавит языка Visual Basic

Алфавит языка включает следующий набор символов:

- A—Z, a—z — прописные и строчные буквы латинского алфавита;
- 0, 1 ,2, .., 9 — арабские цифры;
- знаки арифметических операций: + — сложение, — — вычитание, * — умножение, / — деление, \ — целочисленное деление (деление нацело), ^ — возведение в степень;
- знаки операций отношения: = — равно, > — больше, < — меньше;
- знаки препинания и разделители: . — точка, , — запятая, : — двоеточие, ; — точка с запятой, — — тире, ! — восклицательный знак, ? — вопросительный знак, " " — кавычки, ‘ — апостроф;
- ( ) — круглые скобки, [ ] — квадратные скобки, " " — пробел, _ — подчеркивание;
- символы объявления типа: % — целое число одинарной точности (проценты), & — целое число двойной точности (амперсант), ! — вещественное число одинарной точности (восклицательный знак), # — вещественное число двойной точности, $ — знак строковой (символьной, текстовой) величины.

Клавиатура компьютеров позволяет также выдавать символы: @ — символ "Эт", ~ — знак эквивалентности, | — прямая черта, { } — фигурные скобки, № — знак номера, элементы псевдографики, ранее использовавшиеся для построения таблиц, и другие символы.

Бейсик также позволяет использовать при записи программ национальные алфавиты, в частности, прописные и строчные буквы русского алфавита: А—Я, а—я.

Классификация данных

Некоторые задачи имеют дело с очень большими числами (например, расстояния в космосе). Другие оперируют очень малыми величинами (школьные оценки). Отводить под их представления одинаковые, удовлетворяющие большим величинам количества числовых разрядов было бы крайне нерационально.

Поэтому в языке используют несколько различных типов числовых и нечисловых данных. Это объясняется желанием наиболее эффективно, с мень-

шими затратами памяти хранить данные и с меньшими затратами времени перемещать их от одного устройства компьютера к другому.

Синтаксис определения данных следующий:

```
{Dim | Private | Public} <данные> As  <тип>
```

где служебные слова `Dim`, `Private`, `Public` — определяют области действия данных.

Правильное определение в программе типов всех величин считается одним из правил хорошего стиля программирования. Можно привести множество примеров, когда небрежности в определении типов приводили к большим потерям. Так, в 1996 году при выводе на орбиту спутника пришлось уничтожить французскую ракету Ariane 5. Программисты отвели для горизонтальной скорости всего 2 байта (тип `Integer`), чего оказалось недостаточно. Ракета стала неуправляемой. Убытки составили 500 миллионов долларов.

Перечислим используемые в VB-6 типы данных, их диапазон и потребный размер памяти:

- `Byte` — для определения числовых битовых данных (от 0 до 255, 1 байт);
- `Integer` — целые числа одинарной точности (от −32 768 до 32 767, 2 байта). В случае присваивания типа `Byte` и `Integer` дробному значению оно округляется до целого и после этого проверяется на выход из диапазона;
- `Long` — длинные целых числа (от −2 147 483 648 до 2 147 483 648, 4 байта);
- `Single` — вещественные числа с плавающей точкой одинарной точности (4 байта). Число в форме с плавающей точкой представляется в виде $A \cdot 10^P$, где A — мантисса, P — порядок (характеристика). В формате одинарной точности пример записи числа следующий: a = 2,407153E+7. Обычная запись этого числа такая: $a = 2{,}407153 \cdot 10^7 = 24071530$. При одинаковом числе двоичных разрядов (4 байта, 32 двоичных разряда) количество представляемых чисел типа `Long` и `Single` примерно одинаково (для `Single` несколько меньше). Но диапазон представления данных типа `Single` существенно больше. Если шаг изменения целых чисел равен единице младшего разряда, то вещественные числа типа `Single` имеют неравномерный шаг, значительно увеличивающийся к краям диапазона;
- `Double` — вещественные числа с плавающей точкой двойной точности (8 байтов). Восьми байтов достаточно, чтобы иметь возможность работать с 308-значными десятичными числами. Известно, что 10^6 — миллион, 10^9 — миллиард, 10^{12} — триллион, 10^{15} — квадриллион, 10^{18} — квинтиллион, 10^{21} — секстиллион, 10^{24} — октиллион, 10^{27} — ниниллион, 10^{30} — дециллион. Физики считают, что во всей Вселенной количество элементарных частиц, из которых состоят атомы находящегося в ней вещества, не больше, чем 10^{88}. Поэтому практической необходимости пользоваться числами, большими, чем 10^{100}, нет. Для этого числа придумали название

Гугол. Компьютер даже без использования специальных подпрограмм позволяет работать с числами, имеющими в три раза больше разрядов, чем Гугол;

- `Currency` — используется при выполнении финансовых расчетов (8 байтов). Диапазон представления чисел позволяет работать с суммами свыше 922 триллиона рублей;
- `Date` — используется для определения дат и времени (8 байтов);
- `String` — используется для определения строковых переменных фиксированной длины (число байт — от 1 до 65 536 байт плюс память для хранения числа символов) При определении данных этого типа количество символов приписывается к записи типа, например: `Dim a As String * <длина>`;
- `String` — используется для определения строковых переменных переменной длины. Строки могут быть длиной от 0 до 2 миллиардов символов. Они требуют для записи память в 10 байтов плюс по байту на каждый символ;
- `Boolean` — используется для определения логического значения. (2 байта). Логическая величина может принимать только два значения: False (ложь) или True (истина);
- `Variant` — этот тип данных в процессе работы может превращаться в любой другой в зависимости от того, какая величина в действительности объявляется;
- `Object` — этот тип данных имеет дело с использованием объектов. Он содержит адрес объекта, занимающий четыре байта.

Структуры данных

Первое знакомство со структурами данных состоялось в *гл. 12*. Уточним и расширим основные понятия.

Константы

Константами называют величины, которые задаются заранее и не изменяются при выполнении программы. Различают числовые и строковые константы. Имена присваиваются константам по тем же правилам, что и имена переменных (см. ниже).

Константы объявляются служебным словом `Const`. Одновременно записывается их значение. Можно указывать и тип. Синтаксис записи объявления следующий:

```
[Public | Private] Const <имя константы> [As <тип>] = <значение>
```

Примеры

```
Const Золотое_сечение = 1.618
Private Const Золотое_сечение As Double = 1.6180339901756
Public Const Pi = 3.1415926535897932
```

Переменные

Переменными называют величины, которые в процессе выполнения программы могут менять свое значение. Различают простые переменные и переменные-массивы. Под переменной можно понимать поименованную область памяти компьютера, в которой хранится значение этой переменной.

К имени переменных предъявляются следующие требования:

- имена переменной должны начинаться с буквы;
- имена переменных не могут содержать других символов, кроме букв, цифр и символов подчеркивания;
- имена переменных не могут совпадать с ключевыми словами Visual Basic, словами, которые распознает редактор;
- длина имени не должна превышать 255 символов.

Желательно, чтобы имя имело смысловое значение, соответствующее назначению и особенностям использования переменной. Переменной присваивают результаты выполнения операций. Пока переменной в программе не присвоено значения, то числовая переменная равна нулю, строковая переменная равна пустой строке. Это освобождает от необходимости начального "обнуления" переменных в алгоритмах и программах их реализации.

Объявление переменной выполняется с помощью операторов, использующих следующий синтаксис:

```
{Dim | Private | Public} <ИмяПеременной> [As <ТипДанных>]
```

Примеры

```
Dim Limit As Long
Private Книги As Byte
Public Sum As Variant
```

Массивы

Массив представляет в программе организованную группу элементов одного типа. В качестве элементов массива выступают переменные с индексами — с номерами элементов в массиве. Индексы — целые числа. Они начинаются с нуля и заканчиваются величиной размера массива. Различают размерность массива (число измерений) и размеры размерностей массива (число элементов в измерении). Массивы делятся на статические и динамические массивы.

Статические массивы

Параметры массивов этого типа задаются в процессе разработки и не могут изменяться во время работы.

Массивы задаются своими именами. Рядом с именем приписываются в скобках значения верхней границы или нижней и верхней границы. Например, массив `w(16)` — одномерный массив. Он имеет одну размерность (одно измерение). Номера элементов массива: 0, 1, 2, ..., 16. Количество элементов в массиве 17. `w(i)` — обозначение *i*-го элемента массива. Индекс *i* может в данном примере принимать значения от 0 до 16. Такой массив можно записать еще так: `w(0 To 16)`.

Массив `v(10,10)` — двухмерный. Он имеет две размерности по 11 элементов в каждой. Такой массив можно представить в виде таблицы, имеющей 11 строк и 11 столбцов. Величина `v(i, j)` соответствует элементу такой таблицы, стоящему на пересечении *i*-й строки и *j*-го столбца. Второй вариант записи — `v(0 To 10, 0 To 10)` или `v(0 To 10,10)`.

Имеется возможность изменять нижнюю границу индекса с 0 на 1. Это делается с помощью оператора `Option Base` записью в секции кода программы `(General)(Decjarations)` строки: `Option Base 1`.

Определяются массивы как и переменные, но не могут определяться внутри процедур:

```
{Dim | Static | Public} <Имя (границы)> As Integer
```

Максимальное число размерностей в массиве — 60. Максимальное количество элементов каждой размерности — 32 767.

Динамические массивы

Массивы этого типа могут изменять границы своих индексов в процессе работы программы. Так как предельные значения индексов заранее не известны, то динамические массивы определяются с пустыми скобками:

```
Dim <ИмяМассива>( ) As Variant
```

Коррекция границ массива происходит с помощью оператора `ReDim` внутри процедуры, в которой происходит изменение числа элементов массива или его размерности. Синтаксис оператора `ReDim`:

```
ReDim [Preserve] <Имя_массива> (<границы>) [As <Тип>]
```

Ключевое слово `Preserve` исключает потерю содержимого массива при изменении его размеров или размерности.

Функции *Array*, *Lbound* и *UBound*

При работе с массивами имеется возможность задать числовой или строковый массив с помощью функции `Array(<список аргументов>)`.

Например:

```
a = Array(-2, 5, 12, 0, -4),
b = Array("Ладога", "Свирь", "Оять", "Паша").
```

Функции `Lbound(`*массив* `[, `*размерность*`])` и `Ubound(`*массив* `[, `*размерность*`])` возвращают соответственно наименьшее и наибольшее допустимое значение индекса (номера) массива. Аргумент *размерность* — это номер измерения многомерного массива. Если аргумент опущен, то функция возвращает значение индекса первого измерения.

Например, для рассмотренных в предыдущем примере массивов `a` и `b`:

```
Lbound(a)=0,
Ubound(b)=3
```

Операции и выражения

Выражением называют числовую или строковую константу, переменную или комбинацию констант, переменных и функций, соединенных знаками операций и скобками.

Различают арифметические операции, операции отношения, логические операции и функциональные операции. В соответствии с названиями операций называются и выражения. Выражение может содержать различные операции. Например, арифметическое выражение может содержать функции. Величины, сравниваемые с помощью операций отношения, могут представлять собой арифметические выражения, логические выражения и т. д.

Арифметические операции и выражения

К арифметическим операциям сложение (`a + b`), вычитание (`a – b`), умножение (`a * b`), деление (`a / b`) и возведение в степень (`a ^ b`) добавим две дополнительные.

Целочисленное деление. Знак операции — обратная наклонная черта (\). Например: `a \ b`. Перед выполнением операции операнды округляются до целого значения, а в частном отбрасывается дробная часть.

Примеры выполнения операции деления нацело:

```
19 \ 4 = 4
19.57 \ 3.32 = 20\3 = 6
```

Остаток от деления. Знак операции — слово `Mod`. Запись операции: `a Mod b`. Читается эта запись так: "*a* по модулю *b*". Результат операции — остаток от деления операндов. Перед выполнением операции операнды округляются.

Примеры выполнения операции деления по модулю:

```
19 Mod 4 = 3;
19.57 Mod 3.32 = 20 Mod 3 = 2.
```

Последовательность арифметических операций, расположенных в порядке убывания приоритета выполнения, следующая: возведение в степень, умножение и деление, деление нацело, сложение и вычитание. Действия внутри круглых скобок выполняются первыми.

В арифметических выражениях могут быть записаны числовые константы, переменные, переменные с индексами, функции. Все эти элементы выражения объединены знаками арифметических операций и круглыми скобками.

Функциональные операции

Visual Basic имеет много встроенных так называемых *стандартных* функций. Различают числовые, строковые функции, функции преобразования типов, другие функции. Кроме стандартных имеются так называемые функции пользователя, тип и вид которых имеет возможность задавать программист. Стандартные функции языка Visual Basic сведены в табл. 13.1.

Таблица 13.1. *Стандартные функции языка Visual Basic*

Название функции	Обозначение в математике	Запись на Бейсике	Примечание
Синус	sin *x*	`Sin(x)`	*x* задан в радианах
Косинус	cos *x*	`Cos(x)`	*x* задан в радианах
Тангенс	tg *x*	`Tan(x)`	*x* задан в радианах
Арктангенс	arctg *x*	`Atn(x)`	Возможно переполнение (Overflow)
Корень квадратный	$\sqrt{x}$	`Sqr(x)`	*X* >= 0
Абсолютная величина	$\lvert x \rvert$	`Abs(x)`	*x* — числовое выражение
Экспонента	e^x	`Exp(x)`	e — основание натурального логарифма, $e \approx 2{,}7$
Целая часть числа		`Int(x)`	округляет до ближайшего целого
Ближайшее целое		`Fix(x)`	отбрасывает дробную часть
Логарифм	ln *x*	`Log(x)`	натуральный логарифм, *x* > 0

Таблица 13.1 *(окончание)*

Название функции	Обозначение в математике	Запись на Бейсике	Примечание
Случайное число		`Rnd`	выдает случайное число в диапазоне от 0 до 1
Знак числа		`Sgn(x)`	выдает знак *x*, *x* — числовое выражение

Добавим к примечаниям, приведенным в таблице, следующее. В качестве аргументов функций выступают числовые выражения. Часто говорят, что программа задает аргумент, а компьютер "возвращает" значение функции. Точность возвращаемых значений совпадает с точностью задания аргумента. Перевод в тригонометрических функциях значения угла, заданного в градусах, в радианы выполняется умножением на pi/180 (`pi=3.14...`).

Примеры выполнение операции (функции) `Int` (округление до ближайшего меньшего целого):

```
Int(10.51)=10;
Int(-10.22)=-11.
```

Примеры выполнения операции `Fix` (отбрасывает дробную часть числа):

```
aFix(10.51)=10;
Fix(-10.22)=-10.
```

Для получения случайных чисел в различных диапазонах можно пользоваться следующими соотношениями:

`a=Int(Rnd*N); a` — целое положительное в диапазоне [0, $N-1$];

`a=Int(Rnd*(N+1));` — целое положительное в диапазоне [0, N];

`a=Int(Rnd*N+1); a` — целое положительное в диапазоне [1, N];

`a=Int(Rnd*1000) / 10; a` — вещественное положительное [0; 99,9];

`a=Int(Rnd*100 - Rnd*100); a` — целое в диапазоне [−99; 99];

`a=Int(Rnd*1000 - Rnd*100) / 10; a` — вещественное в диапазоне [−9,9; 99,9] и т. д.

С каждым запуском программы генератор случайных чисел будет возвращать одну и ту же последовательность. Чтобы каждый раз получать разные последовательности, нужно базу функции `Rnd` привязать, например, к таймеру компьютера. Это делается в начале программы записью оператора `Randjmize`.

Функция `Sgn(x)` возвращает знак указанного в качестве аргумента числового выражения. При положительном аргументе возвращается 1, при нулевом — 0, при отрицательном — минус 1.

Запись выражений

Запись всех элементов арифметических выражений выполняется в одну строку. Поэтому суммы и разности в числителях и знаменателях дробей, а также произведения в знаменателях необходимо заключать в скобки. Нельзя ставить два знака арифметических действий подряд. Две последовательные операции должны разделяться круглыми скобками. Например, запись `a / -b` ошибочна. Необходимо записывать это выражение так: `a / (-b)`. Ошибочен пропуск знака операции умножения. Часты ошибки при возведении в степень тригонометрических функций.

Примеры записи математических выражений, в которых начинающие пользователи могут допустить ошибки:

Выражение	Запись на Visual Basic
$\frac{a+b}{c-d}+\frac{e}{fg}$	`(a+b)/(c-d)+e/(f*g)`
$0{,}05a\sin^2 bx^3 + 2{,}5e^{2x}$	`.05*a*SIN(b*x^3)^2+2.5*EXP(2*x)`
$\lvert 12{,}6 - \sqrt{1+3\mathrm{tg}^2 x}\rvert$	`ABS(12.6-SQR(1+3*TAN(x)^2))`

Операции отношения

Операции отношения производят сравнение двух величин. Результат сравнения может быть истиной (True) или ложью (False). Этим логическим значениям присваивается, соответственно, значения 1 и 0. Перечень операций отношения приведен в табл. 13.2.

Таблица 13.2. *Операции отношения*

Знак операции	Проверяемое отношение	Пример выражения
=	равно	a=b
<>	не равно	a<>b
<	меньше	a<b
>	больше	a>b
<=	меньше или равно	a<=b
>=	больше или равно	a>=b

При объединении в одном выражении математических операций и операций отношения сначала выполняются математические операции.

Логические операции

К логическим относят следующие операции.

- *Логическое умножение* (многоместная операция И). Запись операции: `X And Y`. Результат операции принимает значение True (истина) только в случае, если все операнды имеют значение True.
- *Логическое сложение* (многоместная операция ИЛИ). Запись операции: `X Or Y`. Результат операции принимает значение False (ложь) только в случае, если все операнды имеют значение False.
- *Логическое отрицание* (одноместная операция НЕ). Запись операции: `Not X`. Логическое значение результата операции противоположно логическому значению аргумента.

Кроме этих трех несколько реже используются операции `Xor` (неравнозначность), `Eqv` (равнозначность, эквивалентность операндов), `Imp` (импликация).

Логические операции находят применение, например, при записи сложных логических условий. Напомним, что математическое выражение $a \le x \le b$ записывается так:

```
x >= a And x <= b.
```

Если функция $f(x)$ определена на двух отрезках числовой оси: $a \le x \le b$ и $c \le \le x \le d$, то это условие запишется так:

```
x <= a And x <= b Or x <= c And x <= d.
```

Строковые операции

Строковые операции включают операцию *конкатенации* (сцепления) строк и показанные в табл. 13.2 операции отношения строк. Операции отношения используются для сравнения строк. Так как кодирование букв в алфавитах идет в порядке возрастания величины кодов, то справедливы неравенства "б" > > "а", "баба" < "дед", "abcd" < "abcf", "од" & "ин" > "два" и т. д. Поэтому строковые выражения могут быть элементами условий при ветвлении алгоритмов.

Операция конкатенации (знак операции — `&`) позволяет объединять строки. Например, при желании объединить строки "21-й" и "век" нужно записать следующее выражение:

```
"21-й" & " "&"век"
```

В результате получаем строку:

```
"21-й век"
```

По сравнению с Quick Basic и QBasic в Visual Basic существенно увеличился набор встроенных строковых функций. Все функции будут рассмотрены в *гл. 16*.

13.3. Краткие сведения о среде Visual Basi

Установка Visual Basic

Перед установкой программы следует ознакомиться в файле Readme в корневом каталоге установочного диска с требованиями к компьютеру. Инсталляция приложения выполняется программой Setap. Имеется русская версия языка — MS Visual Basic 6.0 Russian Professional Edition, пользовательский интерфейс которой рассматривается в настоящем учебнике.

Кроме Visual Basic 6.0 желательно установить файлы с документацией, справочными материалами и примерами с дисков MSDN — Microsoft Developer Network. Это позволит оперативно получать справочную информацию по всем вопросам работы с Visual Basic и множество полезных профессионально выполненных примеров.

Загрузка Visual Basic

Загрузка среды Visual Basic может быть выполнена командами: **Пуск | Программы | Visual Basic**. Значок-пиктограмма Visual Basic может находиться на рабочем столе. В этом случае следует дважды щелкнуть на значке левой кнопкой мыши.

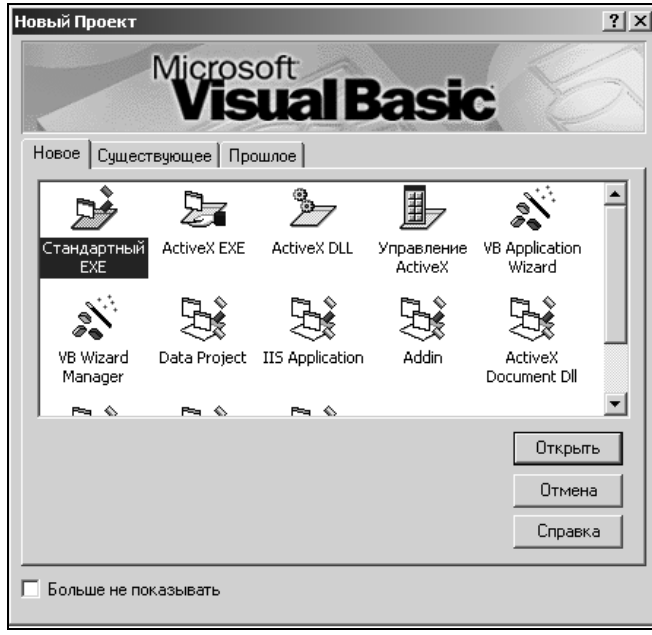


Рис. 13.1. Диалоговое окно **Новый Проект**

Загрузка завершается выводом на экран главного окна Visual Basic (рис. 13.2), перед которым в центре экрана располагается диалоговое окно **Новый Проект** (рис. 13.1).

Диалоговое окно **Новый Проект** содержит три вкладки: **Новое**, **Существующее** и **Прошлое**.

Вкладка **Новое** позволяет выбрать шаблон для создания нового приложения. Как правило, мы будем создавать стандартные EXE-проекты, начиная работу над ними двойным щелчком на значке **Стандартный EXE**.

Вкладка **Существующее** дает возможность открывать ранее созданные проекты, по умолчанию записываемые в папку VB98.

Вкладка **Прошлое** содержит список проектов, открывавшихся в числе последних.

Дважды щелкнем на значке **Стандартный EXE** и рассмотрим главное окно среды языка Visual Basic (см. рис. 13.2).

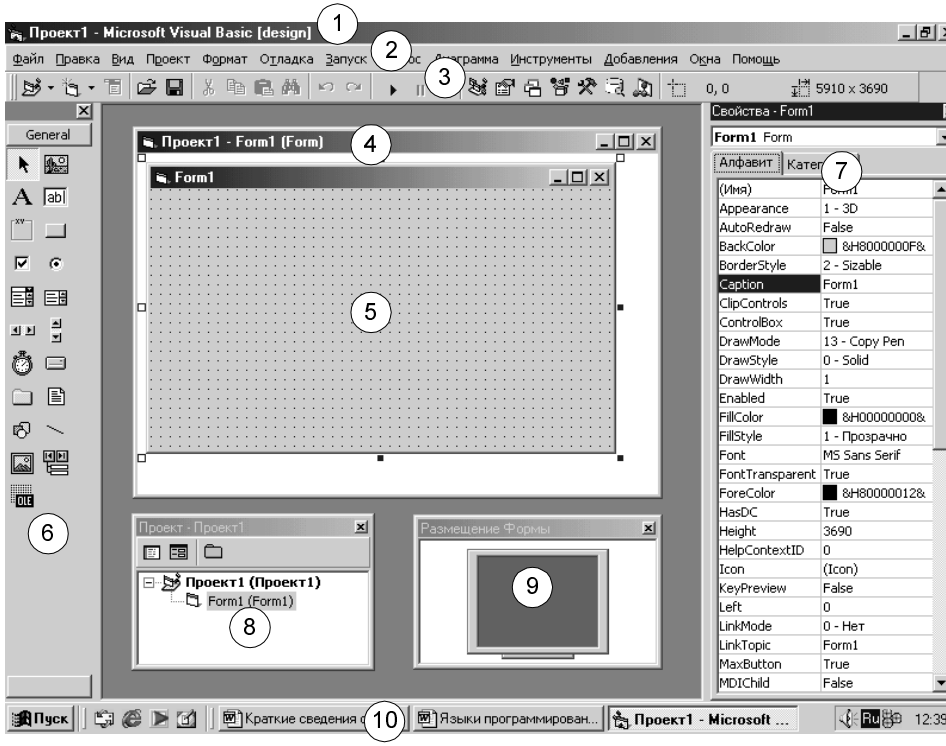


Рис. 13.2. Главное окно среды Visual Basic:
1 — заголовок; 2 — главное меню; 3 — панель инструментов; 4 — окно проектирования формы; 5 — форма; 6 — панель элементов управления; 7 — окно свойств; 8 — окно проводника проекта; 9 — окно размещения формы; 10 — панель задач

Элементы главного окна Visual Basic

Заголовок

Заголовок содержит имя проекта. В начале работы над проектом Visual Basic присваивает ему имя Проект1. Затем, перед записью в память проекту следует дать имя, связанное с его назначением и особенностями работы.

Вся работа над проектом делится на его проектирование и работу с ним. Во время проектирования в заголовке в скобках записывается design, во время работы — run.

В левой части заголовка размещен значок системного меню, в правой части — кнопки **Свернуть**, **Развернуть/Восстановить** и **Закрыть**.

Главное меню

Главное меню расположено во второй сверху строке главного окна, под его заголовком (рис. 13.3).

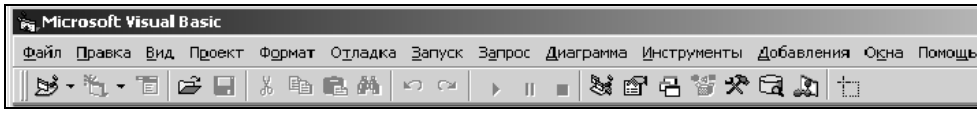


Рис. 13.3. Заголовок, главное меню и панель инструментов

Главное меню содержит следующие пункты:

- **Файл** — открывает список команд по работе с файлами проектов (**Создать**, **Открыть**, **Добавить**, **Удалить**, **Сохранить**, **Напечатать** и т. п.) и имена четырех проектов, которые вызывались последними;
- **Правка** — содержит команды работы с буфером обмена (**Вырезать**, **Копировать**, **Вставить**, **Удалить**, **Выбрать все**), команды поиска информации, команды вызова справочной информации (**Список Свойств/Методов**, **Список Констант** и т. п.), другие команды;
- **Вид** — содержит команды вывода на экран элементов главного окна среды, необходимых для работы с проектами;
- **Проект** — содержит команды добавления в проект при проектировании различных элементов и команды задания проекту свойств, в том числе и имени;
- **Формат** — открывает список команд, определяющих расположение элементов проекта;
- **Отладка** — содержит команды, используемые при отладке работы проекта;
- **Запуск** — содержит команды запуска проекта на выполнение, перезапуска, останова;

- **Запрос** — открывает список команд для работы с запросами к базе данных;
- **Диаграмма** — открывает команды для построения диаграммы для данных выбранной базы данных;
- **Инструменты** — команды этого пункта меню добавляют процедуры, вызывают средства формирования в проектах собственных меню;
- **Добавления** — открывает список команд вызова дополнительных средств, называемых надстройками, позволяет обратиться к мастерам по созданию инсталляционного пакета;
- **Окна** — содержит команды, определяющие расположение окон в главном окне (мозаичное или каскадное);
- **Помощь** — открывает команды работы со справочной системой.

Многие команды главного меню можно подавать нажатием клавиш клавиатуры или кнопок панели инструментов. Названия клавиш или их комбинаций записаны в правой части списков меню.

Панель инструментов

Панель инструментов показана на рис. 13.4. Она расположена ниже главного меню. Панель представляет собой набор кнопок, во многом соответствующих командам, спрятанным в главном меню. Вывод их на передний план упрощает работу по созданию проекта.

Познакомиться с назначением кнопок можно, если подводить к ним указатель мыши. Рядом с кнопкой в желтом прямоугольнике появляется текст, описывающий назначение выбранной кнопки.

Рис. 13.4. Панель инструментов

Рассмотрим в порядке слева направо назначение кнопок панели инструментов.

- Назначения первых двух кнопок — **Добавить Стандартный EXE Проект** и **Добавить форму** — определяют их названия. Шевроны рядом с кнопками раскрывают меню с перечнем типов добавляемых объектов.
- Третья кнопка — **Редактор меню** — открывает одноименное диалоговое окно с полным набором средств, позволяющих создать на форме собственное меню. При этом форма должна быть выделена.
- Следующие две кнопки — **Открыть проект** и **Сохранить проект** — открывают соответствующие окна для выполнения этих действий. Подробнее о структуре и сохранении проекта будет рассказано ниже.

- Следующие три кнопки используются часто и, как правило, комплексно и имеют дело с буферной памятью. Это кнопки **Вырезать** (ножницы), **Копировать** и **Вставить**. Кнопка **Вырезать** удаляет выделенный объект и отправляет его в буфер. Кнопка **Копировать** копирует выделенный объект в буфер. Кнопка **Вставить** может многократно вставлять содержимое буфера в выбранные места.
- Кнопка **Найти** (Бинокль).

Непосредственный режим

В меню **Вид** выберите пункт **Окно неотложного**. При нажатой клавише <Ctrl> переместите за заголовок мышью появившееся окно **Immediate** (рис. 13.5) на середину экрана. При желании можно изменить размеры окна, перемещая его границы. Набирайте следующие примеры и нажимайте на <Enter>. Сравнивайте получаемые результаты.

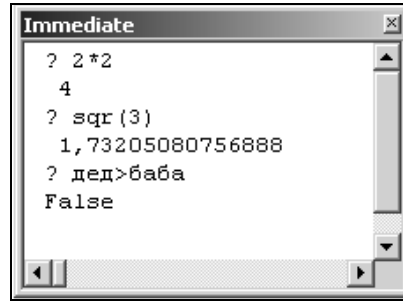


Рис. 13.5. Окно **Immediate**

Примеры (слева — запись задания, справа — результат):

```
? 2^10                1024;
? "коло"+"бок"        колобок;
? 2 + 5 > 4           True
? "2 + 5" > "4"       False
? "5 + 2" > "4"       True
? "пять"<"десять"     False
```

Оценивая результаты сравнения текстов, убеждаемся, что цифры и буквы кодируются последовательно возрастающими числами. Действительно, 2 в компьютере закодировано числом 50, 4 — 52, 5 — 53, б — 225, д — 228, п — 239. Заметим, что тексты сравниваются так, как если бы мы хотели их расположить в алфавитном порядке.

Решая предложенные задания и выдавая ответы, мы используем Visual Basic и компьютер как мощный калькулятор. Заметим, что Visual Basic позволяет создавать не только приложения-калькуляторы, ни и более сложные приложения типа текстового редактора Word.

13.4. Основные определения

Характеристики объектов

Главное отличие Visual Basic от Бейсик-систем второго поколения — способность создавать сложные Windows-приложения с проверенным годами пользовательским интерфейсом. Visual Basic — это визуализированная система программирования, работающая с объектами.

Объект — это основное понятие объектно-ориентированного программирования. Под объектами понимаются все элементы управления, которые мы видим, раскрывая окна системы или приложений Windows. Это кнопки (`CommandButton`), надписи (`Label`), поля ввода (`TextBox`), списки (`ListBox`), флажки (`CheckBox`) и т. п. Форма (`Form`), в которой размещаются эти элементы управления, — тоже является объектом. Одного типа объектов на форме может быть несколько. Их различают по именам и при совпадении имен — по номерам.

Пример: `Form1.Label1`, `Form1.Label2`.

В более строгом понимании объекты — это объединение в одном понятии как формы или элемента управления, так и связанных с ними свойств, методов, событий и программы обработки данных.

Объекты характеризуются свойствами, событиями и методами.

Свойства — это характеристики объекта, определяющие его размер, расположение, внешний вид, цвет, доступность, видимость и т. п.

Значения свойствам объектов присваиваются или изменяются:

- по умолчанию самой программой (часто оптимальные значения);
- пользователем в процессе конструирования (дизайна) приложения;
- программным путем, при выполнении кода программы.

Синтаксис:

```
<Свойство> = <значение свойства>
```

или

```
<Объект>.<Свойство> = <значение свойства>
```

Примеры:

```
Form1.FontSize = 18                      'размер шрифта на форме
Form1.Command1.Caption = "Вычислить"     'надпись на кнопке
```

События — это применяемые к объекту действия, на которые можно запрограммировать реакцию, отклик. Например, нажатие кнопки (событие), как правило, приводит в реальных приложениях к выполнению программой определенных действий (отклику на событие). Событием формы служит, например, щелчок на ней мышью (событие `Click`), на который в качестве реакции формы можно запрограммировать появление на форме текста, рисунка; окрашивание ее в другой цвет и т. п.

Пример:

В системе Windows щелчок мышью (событие) на кнопке **Пуск** открывает окно с главным меню (отклик).

Методы — это команды на выполнение с объектом некоторых действий. Например, метод формы `Cls` очищает форму, метод `Line` рисует в форме линии и прямоугольники, метод `Unload` заставляет форму выгрузиться из памяти и т. п.

Синтаксис:

```
<Объект>.<Метод>
```

Пример:

```
Form1.Show
```

Диалоговое окно *Свойства* и работа с ним

Диалоговое окно **Свойства** ("окно свойств") — одно из главных окон среды разработки (рис. 13.6). Это окно выводится на экран или кнопкой **Окно свойств** на панели инструментов или выбором в меню **Вид** пункта **Окно свойств**, или клавишей <F4>.

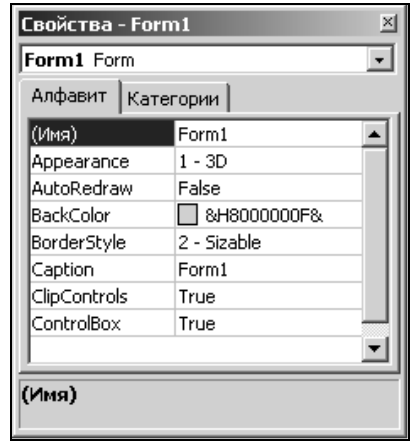


Рис. 13.6. Диалоговое окно **Свойства**

Верхняя строка окна — это его название и имя объекта, свойства которого в настоящий момент отображаются в окне. Ниже расположено текстовое окно списка объектов, используемых в данной конкретной задаче проектирования. Список раскрывается шевроном в правой части окна. Выбор в списке объекта выводит в окно **Свойства** список свойств этого объекта и их значения.

Окно свойств имеет две вкладки: **Алфавит** и **Категории**. На первой вкладке свойства расположены по алфавиту. На второй — по категориям с одинаковым назначением: **Внешний вид**, **Позиция**, **Масштаб**, **Шрифт** и т. п. Списки свойств в категориях можно сворачивать и разворачивать.

Задание свойств в диалоговом окне *Свойства*

В окно свойств вводятся и выводятся (возвращаются) значения всех свойств объектов, или задаваемые при конструировании, или присвоенные автоматически, по умолчанию. Чтобы задать объекту (форме, кнопке, рисунку и т. п.) значение некоторого его свойства, нужно этот объект выделить и внести требуемое значение в соответствующую строку окна свойств.

Полезно знать простые способы задания значений свойств в таблице.

- *Текстовые значения* свойств можно изменить следующим образом. Двойной щелчок мышью в левом столбце строки выбранного свойства выделяет старый текст. После этого можно вводить новый текст значения. С началом ввода старый текст автоматически стирается. Ввод желательно завершать нажатием клавиши <Enter> или щелчком на форме.
- *Логические свойства*, принимающие значения True и False, можно менять двойным щелчком в левом столбце выбранной строки.
- *Свойства с набором значений* можно задавать двумя способами. Во-первых, щелчок на шевроне открывает список значений. Остается только выделить требуемое значение. Во-вторых, двойные щелчки на названии свойств приводят к последовательному их перебору.
- *Свойства, определяющие цвет*, содержат две вкладки: **Система** с рекомендуемым цветом объектов и **Палитра** со стандартным набором цветов. Значение цвета выбирается его выделением на одной из вкладок. Подробнее о цвете будет рассказано ниже.
- *Файловые свойства* имеют значения, представляющие собой ссылки на файлы. Например, двойной щелчок на свойстве `Icon` вызывает окно проводника **Загрузить Иконку**, в котором требуется выбрать файл, определяющий вид значка, отображаемого при свертывании программы в режиме выполнения. Аналогично выбирается значение свойства `MouseIcon`.
- *Свойства размера и размещения* объекта — это координаты левой верхней точки объекта `X` (`Left`) и `Y` (`Top`), ширина (`Width`) и высота (`Height`) объекта. Их значения наглядно, с картинками отображаются в правой части панели инструментов главного окна. Задавать значения свойствам разме-

ра можно аналогично текстовым свойствам (двойной щелчок на названии свойства и запись нового значения). Второй способ более нагляден и удобен — значения этих свойств меняются перетаскиванием объекта и его границ.

Принципы создания и работы приложений

Процесс создания Windows-приложения делят на два этапа:

- Первый этап — это *этап конструирования* (дизайна), называемый этапом визуального программирования. С панели инструментов в форму (будущее окно приложения) перетаскиваются кнопки, переключатели, текстовые окна и другие управляющие элементы. Этим элементам задаются свойства — имена, названия, размеры, положение, цвет и т. п.
- Второй этап — *этап программирования и записи кода* программы. На языке программирования записываются алгоритмы решения задач, вывода результатов, алгоритмы работы управляющих элементов приложения.

В соответствии с этими этапами можно разделить на две части и программу приложения.

- Первая часть программы определяет состав, внешний вид окон приложения и расположенных на них элементов управления, а также свойства этих элементов. Она содержит всю информацию о пользовательском интерфейсе разработанного приложения.
- Вторая часть программы — это *код программы*, состоящий из процедур обработки событий. Эта часть определяет поведение всех элементов приложения при работе с ними пользователя.

Принцип работы программ, создаваемых в среде Visual Basic, отличается от программ в обычном процедурном языке Бейсик. Программа на Бейсике выполняется последовательно, оператор за оператором. Принцип работы программ Windows-приложений, написанных на Visual Basic, носит *событийно-управляемый* характер. Программа на Visual Basic — это набор *процедур обработки событий*.

Работой программы управляют события, возникающие в результате работы пользователя (работа с клавиатурой, мышью, воздействие на элементы управления в окнах приложений).

В общем виде процедура обработки события записывается так:

```
[{Private|Public}] Sub <объект>_<событие>([<параметры>])
        <последовательность инструкций>
End Sub
```

- `Private` — параметр, указывающий на тип процедуры, доступной только в модуле, в котором выполняется описание (присваивается по умолчанию);

- `Public` — параметр определения типа процедуры, доступной для всех модулей всех проектов;
- *`событие`* — название события, например, щелчок мышью на объекте (`Click`), двойной щелчок (`DblClick`) и т. п.;
- *`параметры`* — определяют особенности события, например, то, какая кнопка мыши была нажата (левая, средняя, правая) и т. п.

Например, процедура обработки события — щелчка на кнопке `cmdВыход` может иметь одну инструкцию — `End`, завершающую работу программы:

```
Private Sub cmdВыход _Click( )
       End
End Sub
```

13.5. Форма и ее характеристики

Формами называют главные элементы приложений. Они служат в качестве окон Windows-приложений, платформ, контейнеров для размещения элементов управления. Приложение может иметь несколько форм. Простейшие и широко распространенные приложения строятся на одной форме.

Форма представляет собой прямоугольник со строкой заголовка, на котором расположены оконное меню (слева), название формы (свойство `Caption` — в центре) и три кнопки справа: **Свернуть**, **Развернуть/Восстановить** и **Закрыть**.

Основные свойства:

`Name`, `Caption`, `BackColor`, `FontSize`, `FillColor`, `WindowState` и некоторые другие. Список свойств формы можно увидеть в окне свойств. Там же свойства можно разделить по категориям (внешний вид, размеры и т. п.).

Основные события:

`Load`, `Unload`, `DragDrop`, `MouseDown`, `MouseMove`, `Resize` и др. Список событий можно увидеть в правом списке окна кода программы.

Основные методы:

`Circle`, `Cls`, `Hide`, `Line`, `Show` и др.

Замечание

Форма и другие объекты имеют множество свойств, событий и методов. Например, форма имеет более 60 свойств, более 30 событий и почти 20 методов. Многие из них используются по умолчанию, другие применяются редко. Поэтому знакомиться с характеристиками форм и управляющих элементов будем постепенно, по необходимости, в процессе работы.

Глава 14

Линейные и ветвящиеся программы

Понятие программы в системе Visual Basic более широкое, чем, например, в QBasic и Quick Basic. Программа содержит не только запись алгоритма на языке программирования, но и большой объем информации о пользовательском интерфейсе приложения. Это сведения о используемых в приложении элементах управления (кнопках, надписях, полосах прокрутки и т. п.) и их многочисленных свойствах. Поэтому запись алгоритма работы приложения обычно называют кодом программы или просто кодом.

Материал данной главы содержит в основном вопросы алгоритмизации и программирования (создание кодов программ) и в меньшей степени вопросы конструирования (дизайна), которые рассматриваются ниже. В главе последовательно рассматривается работа с проектами, в которых программируются линейные и ветвящиеся алгоритмы. Дается синтаксис применяемых в программах операторов, их назначение и особенности применения. Приводятся примеры.

При записи синтаксиса операторов и функций используют следующие обозначения:

- **выделенным** шрифтом записываются слова, которые должны вводиться в программы без изменения;
- *курсивом* записываются переменные, аргументы и параметры, значения которых вводятся программистом;
- [ ] — в квадратные скобки помещают необязательный элемент записи оператора;
- {W1 | W2} — в фигурных скобках записывают разделенные вертикальной чертой варианты записи, из которых можно выбрать только один (W1 или W2);
- ... — многоточие указывает на возможность многократного повторения записанного перед многоточием элемента.

14.1. Наше первое приложение

Пример 1. "Мы изучаем Visual Basic!"

Постановка задачи

Создадим простое приложение, сообщающее, что "мы изучаем Visual Basic!" В качестве объекта, на который будет выведено это сообщение, будем использовать форму.

Начало работы

Запускаем Visual Basic и подаем команду **Файл | Новый проект**. В диалоговом окне **Новый проект** выбираем **Стандартный EXE** и щелкаем на кнопке **OK**.

Щелчками на кнопках **Закрыть** (в верхнем правом углу окон) убираем с рабочего поля системы все окна, кроме окна конструирования формы с расположенной в нем формой.

Запись кода

Двойной щелчок на форме выводит на экран окно кода программы (рис. 14.1) с начальной и конечной строкой процедуры загрузки формы. Эти две строки для краткости иногда называют процедурными скобками.

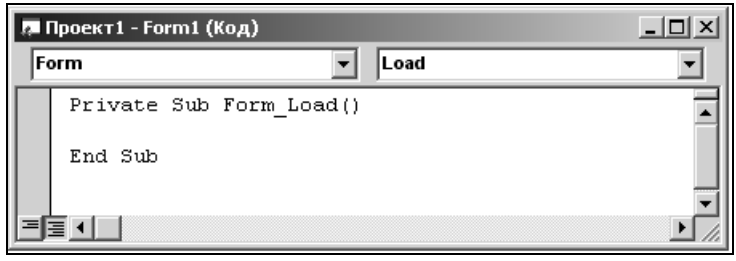


Рис. 14.1. Окно кода программы

Вписываем в окно код программы. При записи кода редактор языка Visual Basic будет усиленно помогать.

Запись следует вести строчными буквами. При переходе к следующей строке, например, при нажатии клавиши <Enter>, редактор сам сделает первые буквы служебных слов прописными. Если этого не произойдет, значит, в строке имеется ошибка, которую следует найти и исправить.

Внутренние строки процедуры записывают со сдвигом вправо. При этом достаточно сдвинуть клавишей <Tab> первую строку (как показано на рис. 14.2). При переходе к очередной строке запись будет располагаться со сдвигом.

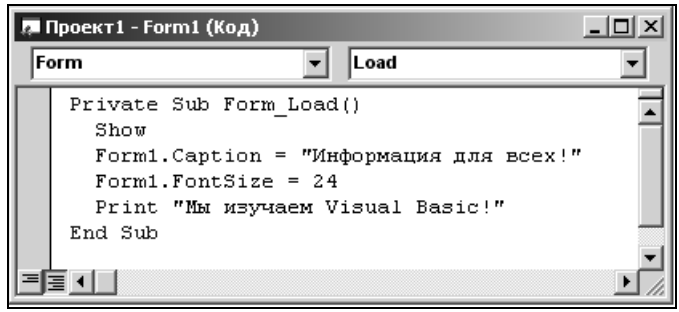


Рис. 14.2. Код решения задачи 1

Во второй строке кода запишем `Show` (рис. 14.2). Это метод формы, который сделает результат решения задачи видимым ("Шоу" — Показ).

Следующая строка кода — запись значения свойства формы `Caption` (Заголовок). Используется синтаксис:

```
Объект. Свойство = <значение свойства>
```

После запуска приложения присвоенное значение будет выведено в заглавие формы (см. рис. 14.3).

В следующей строке запишем значение свойства `FontSize` (Размер шрифта). По умолчанию он равен 8 пунктам. Записав 24 пункта, мы увеличиваем его в три раза (один пункт равен примерно одной трети миллиметра).

Предпоследняя строка — вывод нашего сообщения на форму. Для этого используется метод `Print`.

Запуск программы

Напомним, что запуск программы выполняется или командой главного меню **Запуск | Запуск**, или нажатием кнопки **Запуск** на панели инструментов, или нажатием на клавишу <F5>. После запуска на экран выводится окно (рис. 14.3) с нашим сообщением.

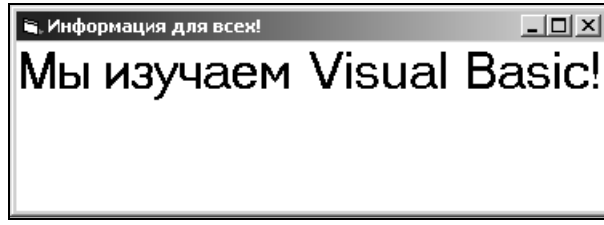


Рис. 14.3. Результат решения задачи 1

Рекомендации по записи кода

Знание приемов работы с кодами и правил их оформления позволяет существенно сокращать время набора и форматирования кода, делает код более обозримым, понятным, информативным.

Перечислим свойства редактора, которые желательно использовать при работе с кодом программы.

- Редактор сохраняет отступ в начале следующей строки.
- При написании кода немедленно сообщает о синтаксических ошибках.
- Корректирует заглавные буквы в ключевых словах, вставляет пробелы.
- Окрашивает разными цветами комментарии, строки с ошибками, выделенные слова.
- Выводит окно с быстрой информацией, подсказывающей особенности формата некоторых ключевых слов и функций Visual Basic.
- Выводит окно с подсказкой пользователю — списком доступных свойств и методов объекта, списком значений свойств.
- Способен дописывать начатые пользователем названия свойств, методов, элементов.
- При переходе на следующую строку редактор исправляет запись текста. Он увеличивает промежутки между служебными словами и делает первые буквы слов прописными. Это указывает на отсутствие в строке ошибок.

Максимальное число символов в строке кода 1023. Количество строк кода не более 65 534. Длинные строки плохо просматриваются, затруднено их редактирование. Поэтому длинные логические строки кода желательно разбивать на несколько физических строк символами подчеркивания (_).

В одной строке может быть размещено несколько операторов. При этом они разделяются двоеточиями.

Например: `Form1.Visible = True: Form1.BackColor = vbRed`

Удалить символ слева и справа от текстового курсора можно с помощью клавиш клавиатуры. Для удаления слова или большего фрагмента текста следует использовать их выделение.

Выделения используются при желании выполнить с фрагментом кода следующие действия: удалить, переместить, скопировать, вызвать справку и т. п. Имеется много приемов выделения.

- Символ, группу символов, слово, группу слов, строку, несколько строк можно выделить, проведя по ним мышью с нажатой кнопкой, или с помощью клавиш перемещения курсора клавиатуры при нажатой клавише <Shift>.

- Слово выделяется двойным щелчком на нем мышью.
- При перемещении указателя мыши к левой границе окна он превращается в стрелку. Работая стрелкой как указателем строк, можно выделять отдельные строки. Если к этому добавлять клавиши <Shift> и <Ctrl>, то можно выделять различные варианты групп строк.

Выделенные фрагменты кода можно перемещать методом Drang & Drop ("перетащить и положить"). Для этого нужно подвести мышь к выделенному фрагменту и при нажатой кнопке мыши перетащить его в новое место. При этом указатель мыши дополняется прямоугольником и вертикальной линией, которую следует подвести к выбранному месту вставки.

При записи строк кода с конструкциями, синтаксис которых предусматривает разделители в виде точек, после постановки точки редактор выводит на экран окно со списком подходящих для данной ситуации служебных слов (имен, свойств и методов). Если требуемое слово находится в окне, то двойной щелчок на нем записывает его в нужное место.

Если слово в списке предварительно выделить, то для его вставки в текст достаточно нажать на клавиши <Tab> или <Enter> (рис. 14.4). В первом случае курсор остается в этой же строке, во втором — переносится в начало следующей строки.

Если нужного слова в окне нет, то следует начать его записывать. После записи в среднем двух-трех символов слово появляется в окне. Подобные окна редактор выводит часто, помогая пользователю экономить время и исключать ошибки в записи служебных слов.

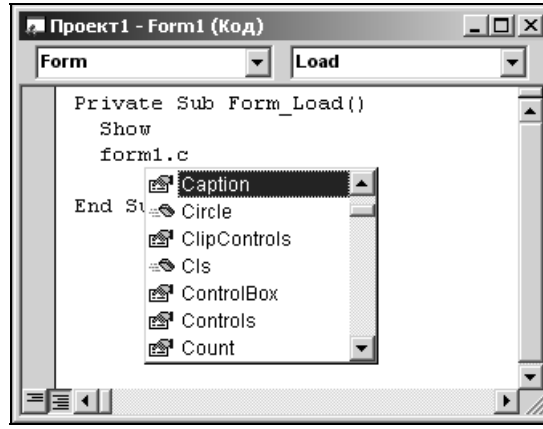


Рис. 14.4. Помощь редактора в записи свойств объекта

Редактор может дописывать в коде программы начатые знакомые ему слова. Например, если после записи в строке кода слов `form1.ca` нажать <Ctrl>+

+<Пробел>, то редактор автоматически допишет: `form1.Caption`. Это произойдет в том случае, если первых букв слова достаточно для того, чтобы редактор узнал слово.

Сдвиг вправо внутренних строк процедуры относительно первой строки можно выполнить с помощью клавиш <Tab> или <Пробел>. Такая запись называется *структурированной*. В сложных программах она делает логику работы кода более понятной.

При записи кода редактор цветом выделяет различные элементы текста: выделенные элементы (белый на синем фоне), элементы с синтаксическими ошибками (красный), строки контрольных точек (коричневый), комментарии (зеленый), ключевые слова (синий) и т. д. С цветовыми кодами можно познакомиться в диалоговом окне, вызываемом командой главного меню **Инструменты | Опции | Формат редактора**.

Сохранение созданных приложений, ввиду сложности их структуры, отличается от сохранения, например, текстовых документов и будет рассмотрено в *разд. 14.3* и *14.4*.

14.2. Наше второе приложение

Практическую работу по программированию начнем с простейшей алгоритмической структуры — следования, т. е. создадим приложение, код которого реализует линейный алгоритм решения поставленной задачи.

Пример 2. Линейный алгоритм

Постановка задачи

Создать Windows-приложение ввода двух целых чисел, вычисления и вывода на экран их суммы и среднего арифметического.

Формализация

Введем обозначения: a и b — переменные, s — сумма, sr — среднее арифметическое. Запишем формулы: $s = a + b$, $sr = s / 2$.

Начало работы

Запускаем Visual Basic и подаем команду **Файл | Новый проект**. В окне **Новый проект** выбираем **Стандартный EXE** и щелкаем на кнопке **OK**.

Щелчками на кнопках **Закрыть** (в верхнем правом углу окон) убираем все окна, кроме окна конструирования формы с расположенной в нем формой и окна **Свойства**. Окно **Свойства** обычно располагается в правой части рабочего поля.

Перемещением мышью окон и их границ задайте им удобные для работы места и размеры. В случае склеивания, например, окна **Свойства** с верхней границей рабочего поля можно воспользоваться мышью при нажатой клавише <Ctrl>.

Подготовка к записи кода программы

Одно из правил работы по созданию приложений состоит в следующем: присваивание имен всем объектам должно опережать работу по написанию кода программы.

Присвоение имени форме

Имя формы определяет свойство `Name` (Имя). Желательно, чтобы имя имело содержательный смысл, связанный с назначением формы. Имя задается в первой строке (по алфавиту) списка свойств формы.

Обычно имена снабжают префиксами. Префикс `frm` показывает, что это имя формы, а не другого объекта. Содержательный смысл имен и префиксы помогают лучше ориентироваться в сложных приложениях, имеющих большое количество различных объектов.

Как правило, имена формам, всем элементам управления в них и проекту присваиваются *до начала записи кода* программы. Это исключает расхождения в именах реальных и именах, используемых в коде. Малейшие расхождения в записи имен делают приложение неработоспособным.

В именах объектов нельзя использовать пробелы. Поэтому имена из нескольких слов или записывают слитно, выделяя слова прописными буквами (`ПримерЗаписи1`), или заменяют пробелы символами подчеркивания (`Пример_записи_2`).

В окне **Свойства** в верхней строке вкладки **Алфавит** заменим имя **Form1** на **frmЗадача_2**. Нажатие клавиши <Enter> или щелчок на форме выводит новое имя в заголовок окна конструирования формы.

Присвоение названия форме

В окне **Свойства** находим строку **Caption** и заменяем **Form1** на **Второе_приложение**. Новое название появляется в заголовке формы.

Присвоение имени проекту

В главном меню выбираем пункт **Проект**, в подменю которого выбираем **Проект1 Свойства**. В текстовом поле **Объект запуска** уже записано присвоенное форме имя: **frmЗадача_2**. В поле **Имя проекта** заменим имя **Проект1** на **Второе_приложение** и нажмем на кнопку **ОК**. Убедимся, что во все заголовки окон записались новые имена формы и проекта.

Двойной щелчок на форме выводит на экран окно кода программы (рис. 14.5). Верхняя строка окна — заголовок. В нем записывается имя проекта. Ниже заголовка расположены два раскрывающихся списка. Левый — список используемых в проекте объектов. В нашем примере в нем содержится только форма. В правом списке находятся все события, соответствующие выбранному в левом списке объекту. Среди них имеется и событие `Load` — загрузка формы из памяти.

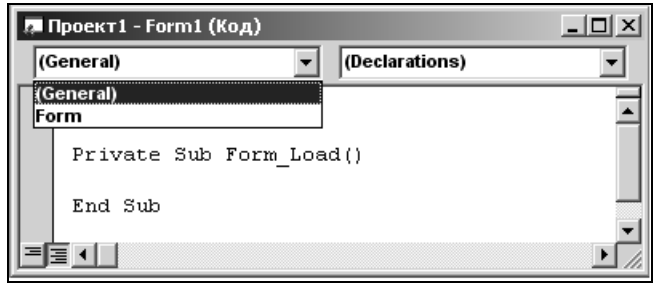


Рис. 14.5. Окно для записи кода

Ниже списков расположено поле, в котором записывается код. Щелчок на форме уже записал в нем первую и последнюю строку процедуры загрузки формы. Продолжим запись кода решения поставленной задачи.

Запись кода

Откройте левый список и выделите строку (`General`). В правом списке автоматически будет выведено (`Declarations`) и текстовый курсор встанет в начале первой свободной от записей строки. Так обычно начинается запись определений типов переменных и констант.

Рассмотрим еще один пример помощи редактора. При записи первой строки определения переменной (`dim a as integer`), после записи слова `as` и пробела на экран выводится список объектов, типов данных, других служебных слов. После нажатия на клавишу <i> список преобразуется в вид, показанный на рис. 14.6. После нажатия на клавишу <n> слово `Integer` перемещается в первую строку. Для его вставки в текст достаточно нажать на клавиши <Tab> или <Enter>. В первом случае курсор остается в этой же строке, во втором — переносится в начало следующей строки.

При записи кода программы использованы два метода формы: `Show` и `Print`. Вывод на форму результатов решения задачи выполняется с помощью метода `Print` (Печать). Метод `Show` (Показ) обеспечивает видимость напечатанного текста (см. рис. 14.10). Он занял вторую строку процедуры обработки события `Load`.

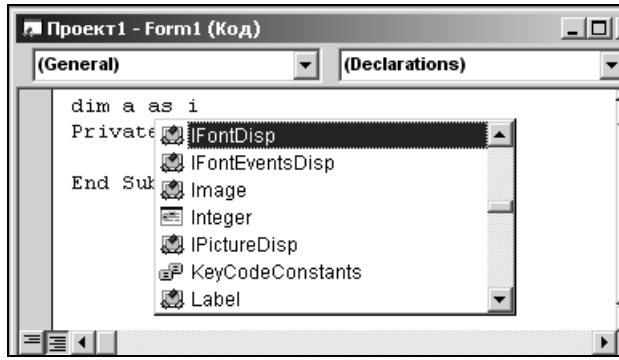


Рис. 14.6. Помощь редактора при записи типа `Integer`

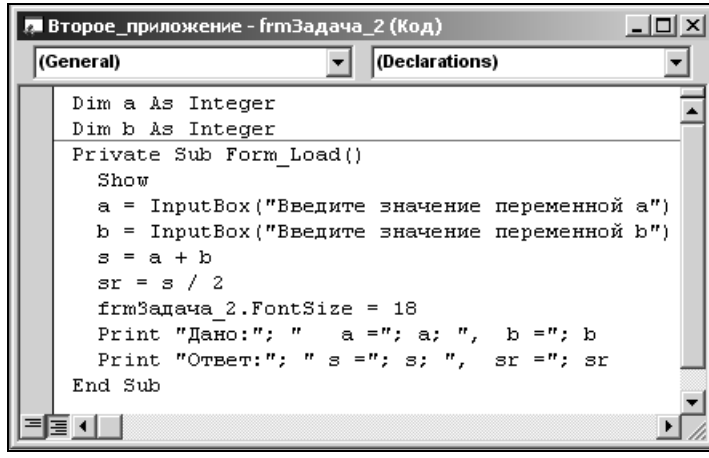


Рис. 14.7. Код программы второго проекта

Следующие две строки кода — ввод данных. С помощью двух функций `InputBox` последовательно вводятся значения переменных *a* и *b*. Каждая функция выводит на экран диалоговое окно (рис. 14.8). В нижней части окна имеется пустая строка для записи текста. В нее вводится значение затребованной переменной и нажимается кнопка **OK**.

При записи функций мы вновь встретились с помощью редактора. На этот раз он вывел на экран быструю подсказку (`QuickInfo`) с напоминанием синтаксиса функции `InputBox` (рис. 14.9), который будет рассмотрен ниже.

При желании самостоятельно вывести на экран QuickInfo для некоторого объекта нужно его выделить и подать команду **Главное меню | Правка | Быстрая информация** или нажать комбинацию клавиш <Ctrl>+<I>.

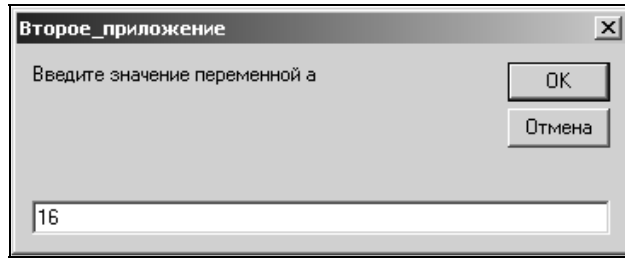


Рис. 14.8. Окно функции `InputBox` ввода переменной *a*

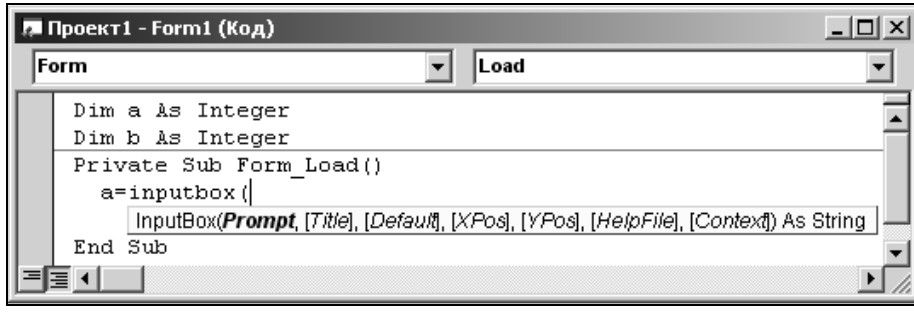


Рис. 14.9. Напоминание редактора
о синтаксисе функции `InputBox`

Следующие две строки — вычисление суммы и среднего арифметического. За ними следуют строка задания размера шрифта и две строки вывода результатов.

При запуске приложения на экран последовательно выводятся окна функций ввода переменных *a* и *b*. После ввода данных вычисляются сумма и среднее арифметическое, которые выводятся на форму (рис. 14.10).

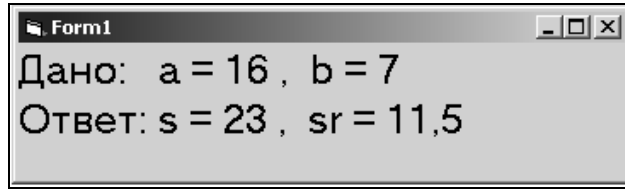


Рис. 14.10. Вывод результата решения задачи

Проверка правильности полученного результата такой простой задачи не требует большого труда.

14.3. Структура и сохранение простого проекта

Даже простой проект ввиду сложности своей структуры сохраняется в нескольких файлах. Простые проекты могут размещаться в трех файлах с расширениями vbf, vbp и vbw. При этом Visual Basic предлагает записывать только первые два файла.

- файл проекта — Visual Basic Project (расширение vbp) — служит для сохранения связей между компонентами проекта;
- файлы форм — Visual Basic Form File (расширение frm) — по числу форм в проекте сохраняют полную информацию о форме и размещенных на ней управляющих элементах;
- автоматически дописываемый файл Стили и Состояния с характеристикой рабочего пространства проекта — Visual Basic Project Workspace (расширение vbw).

В сложных проектах количество различных типов файлов может быть больше десяти. Обилие в проекте файлов с разными расширениями определяет некоторые особенности их сохранения. Оптимальный вариант — все файлы каждого проекта размещаются в отдельной папке. Это исключает путаницу, возникающую при поиске, перемещениях и копировании проектов. К тому же такое сохранение облегчает перенос создаваемых приложений на другие компьютеры.

Сохранять проект желательно начинать на начальных этапах его конструирования, на этапе записи кода программы. Обычно проект отлаживается по процедурам обработки событий. Записаны инструкции процедуры — запуск проекта; записана новая процедура — новый запуск для проверки правильности работы и т. д. Перед запусками проект желательно сохранять. Это исключит потерю проекта при возможных зависаниях системы.

При первом сохранении проекта в меню **Файл** обычно выбирается команда **Сохранить Проект как** (рис. 14.11). При последующих сохранениях подается команда **Сохранить Проект** кнопкой на панели инструментов.

Рассмотрим окно **Сохранить Файл Как**. Visual Basic предлагает записать файл с расширением frm в папку VB98. Не делайте этого! В списке **Тип файла** внизу окна выберите строку **Все Файлы (*.*)**. В окно выводятся все файлы Visual Basic, без которых он не может работать. Среди них легко найти файл VB6 — это главный файл-приложение Visual Basic. Размещать файлы своих приложений в близком соседстве с файлами Visual Basic опасно. Желательно сохранять их в отдельных папках.

В верхней части окна справа от списка **Папка** последовательно размещены кнопки: **Переход к последней рассмотренной папке**, **На один уровень вверх**,

Создание новой папки, **Меню "Вид"**. С их помощью можно перемещаться по каталогам, создавать новые папки, изменять представление их содержимого.

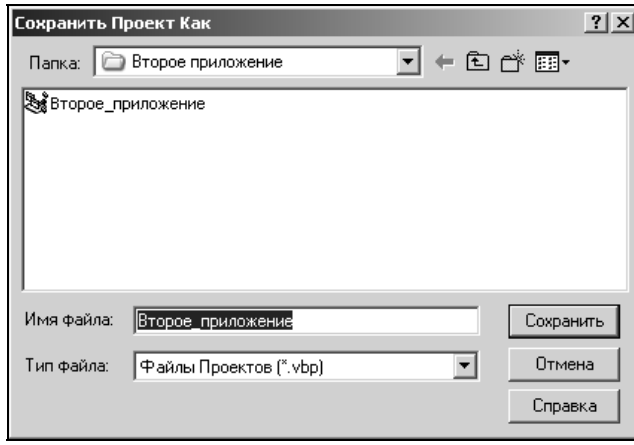


Рис. 14.11. Окно **Сохранить Файл Как** с созданной папкой

Для записи всех своих проектов на VB6 будем использовать ранее созданную папку Первые пробы. Сохраним в ней проект Второе_приложение.

Для этого сделаем следующее.

1. Создадим одноименную папку Второе приложение. Адрес этой папки такой: Мои документы\Первые пробы\Второе приложение.

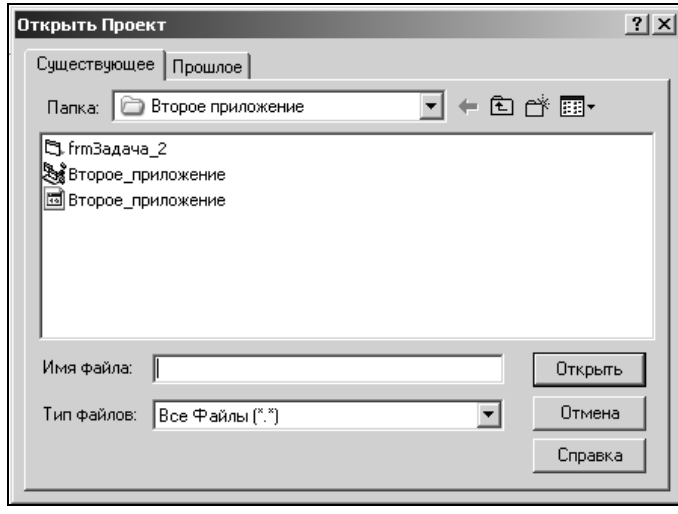


Рис 14.12. Список файлов проекта Второе приложение

2. Visual Basic сначала предлагает сохранить файл формы см. рис. 14.11). Кнопкой **Открыть** откроем папку Второе приложение. При этом кнопка **Открыть** заменится кнопкой **Сохранить**.
3. Нажимаем на кнопку **Сохранить**. В память запишется файл frmЗадача_2. При этом в поле **Имя файла** будет выведено имя второго файла: Второе_приложение, а в поле **Тип файла** — Файлы Проектов (*.vbp).
4. Нажимаем на кнопку **Сохранить**. При этом сохраняется файл проекта. Проект сохранен в отдельной папке. В такой упаковке его удобно перемещать по дискам и каталогам.

Если после записи раскрыть папку Второе приложение и выбрать все типы файлов, то увидим список, показанный на рис. 14.12.

14.4. Функция *InputBox*

Функция `InputBox` применяется для ввода данных пользователем в процессе работы приложения. Функция открывает окно диалога пользователя с приложением, предлагая ввести требуемые данные или согласиться с предложенными их значениями.

Несколько сокращенный синтаксис функции следующий:

```
<переменная>= InputBox (<"текст 1"> [, <"текст 2">] [, <"текст 3">] [, x]
[, y])
```

Все аргументы функции, кроме *текст 1*, могут быть опущены (они записаны в квадратных скобках).

- *текст 1* — может служить подсказкой, указывающей на то, что требуется ввести;
- *текст 2* — заголовок диалогового окна. Если *текст 2* опускается, то в заголовок выводится имя проекта;
- *текст 3* — предлагаемое по умолчанию значение переменной, которое пользователь может принять, нажав кнопку **OK** или клавишу <Enter>. Если опускается *текст 3*, то текстовое окно ввода очищается;

Рис. 14.13. Окно функции `InputBox`

- `x` и `y` — координаты левой верхней точки диалогового окна. При отсутствии значений координат `x` и `y` диалоговое окно располагается в центре экрана.

На рис. 14.13 показан внешний вид окна функции. При полном синтаксисе в окно добавляется кнопка **Помощь**.

14.5. Метод *Print*

Метод `Print` используется для вывода данных на форму. Для вывода данных достаточно записать слово `Print`. В сложных проектах, с элементами управления, кроме формы, использующими этот метод, следует указывать объект.

В общем виде синтаксис метода следующий:

```
[объект.] Print [<список>][{;|,}]
```

При пояснении смысла параметров метода используется пример кода, демонстрирующего работу метода (листинг 14.1).

Параметр *список* — это числовые или строковые выражения, разделенные запятыми или точками с запятыми. Строка вывода разделена на зоны с фиксированным числом знакомест.

Если разделители списка — запятые, то каждый элемент списка печатается в следующей свободной зоне (см. строку `b` кода, приведенного в листинге 14.1, и блок-схему алгоритма на рис. 14.15).

Если разделители списка — точки с запятой, то элементы списка печатаются или через пробел или подряд (см. строки `c` и `e` в листинге 14.1).

Строка кода `d` выводит строку текста для сравнения результатов вывода. Строка кода `g` с пустым *списком* выводит пустую строку.

Строка кода `h` выводит список с разными разделителями.

Следующие три строки: `i`, `j` и `k` — показывают, что, если в конце списка стоит запятая или точка с запятой, то выполнение следующего оператора `Print` начинается в этой же строке.

Строка кода `m` демонстрирует вывод списка, элементы которого — выражения. Перед выводом определяются значения всех входящих в выражение функций и выполняются все задаваемые выражением действия.

Строка кода `n` показывает вывод константы.

Запуск рассмотренного кода приводит к выводу на экран окна, показанного на рис. 14.14.

Листинг 14.1. Демонстрация работы метода Print

```
Private Sub Form_Load()
  Show
  a = 1: b = 2                        'a) задание значений двум переменным
  Print a, a, a                       'b) разделители списка — запятые
  Print a; a; a; a; a; a; a           'c) разделители — точки с запятой
  Print "11111111111111"              'd) вывод текста для сравнения
  Print "1"; "1"; "1"; "1"           'e) вывод символов
  Print                               'g) вывод пустой строки
  Print a; a, b; a                    'h) варианты с разными разделителями
  Print a,                            'i) строка с продолжением
  Print a; b;                         'j) строка с продолжением
  Print a                                 'k) последнее продолжение
  Print "a+b="; a+b, "2a+3b=";2*a+3*b      'm) вывод выражений
  Print 2004                              'n) вывод константы
End Sub
```

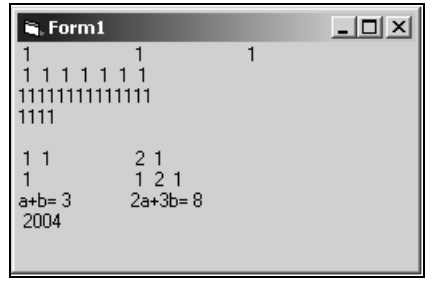


Рис. 14.14. Результат работы программы листинга 14.1

Заметим, что полный, понятный и красивый вывод результатов решения задачи является элементом хорошего стиля программирования.

14.6. Программирование ветвящихся алгоритмов

Пример 3. Оператор *If Then Else*

Постановка задачи

Создать проект, решающий следующую задачу. Вводятся два числа. Если они равны, то вычислить их сумму, иначе — вычислить их разность.

Формализация

Пусть числа имеют имена *a* и *b*, сумму чисел обозначим *S*, разность — *R*.

Запись алгоритма

Ниже приведена запись алгоритма решения задачи на алгоритмическом языке. Блок-схема алгоритма показана на рис. 14.15.

```
алг Ветвление (вещ a, b, S, R)
        арг a,b
        рез S, R
нач
        S:= a + b; R:= a − b
        если a > b
                то выв S
                иначе выв R
        все
кон
```

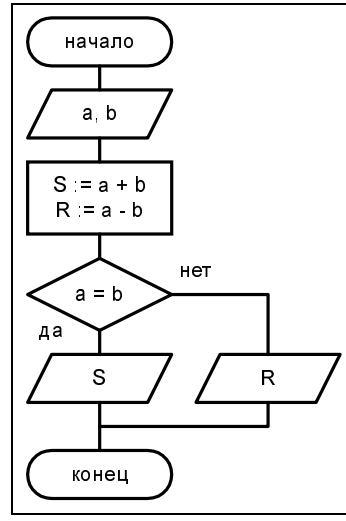


Рис. 14.15. Блок-схема алгоритма решения задачи 3

Начало работы

Запускаем Visual Basic и подаем команду **Файл | Новый проект**. В окне **Новый проект** выбираем **Стандартный EXE**. Щелчками на кнопках **Закрыть** (в

верхнем правом углу окон) убираем все окна, кроме окна конструирования формы с расположенной в нем формой.

Этап конструирования (дизайна)

На этом этапе обычно в формах размещаются элементы управления. Всем объектам присваиваются имена и выполняется первое сохранение проекта. Для решения задачи примера 3 требуется один объект — форма. Поэтому достаточно присвоить имена только форме и проекту.

Присваивание имен

Присвоим форме имя **frmЗадача_3**, а проекту — **Ветвящийся_алгоритм_1**. Кратко напомним, как это можно сделать (подробно это было изложено в *разд. 14.3*).

Откроем окно **Свойства** (кнопкой **Окно Свойств** или клавишей <F4>) и заменим заданное системой по умолчанию имя **Form1** на имя **frmЗадача_3**. После этого командой **Проект | Проект1 Свойства** открываем окно **Проект1 Свойства проекта**. На вкладке **Главное** в текстовом поле **Имя Проекта** заменяем имя **Проект1** на **Ветвящийся_алгоритм_1**.

Первое сохранение

Обычно начинают сохранять проект на начальных этапах работы с ним. Это исключает возможность потери информации при перебоях с питанием или при зависаниях компьютера. При первом сохранении выбирается место в памяти, куда будет помещена папка с файлами проекта. Порядок действий при сохранении проекта был дан в *разд. 14.3*. Вспомним его.

Подадим команду **Файл | Сохранить проект как**. В окне **Сохранить Файл Как** выберем вложенную в папку Мои документы, созданную ранее папку Первые пробы, раскроем ее и создадим в ней новую папку Ветвящийся алгоритм 1. Раскрываем эту папку и один за другим записываем в нее два файла — файл формы и файл проекта.

Последующие сохранения кнопкой **Сохранить проект** обычно выполняются перед проверкой работы проекта после записей или преобразований больших фрагментов кода. Это исключает потери информации.

Запись кода

Щелчком на форме или командой **Вид | Код** вызываем окно кода, в которое записываем следующий код решения задачи, приведенный в листинге 14.2.

Листинг 14.2. Код решения задачи примера 3

```
Dim a As Integer
Dim b As Integer
```

```
Private Sub Form_Load()
  Show
  a = InputBox("Введите число a", "Ветвящийся алгоритм", 10)
  b = InputBox("Введите число b", "Ветвящийся алгоритм", 15)
  S = a + b: R = a - b
  If a = b Then Print "a + b = "; S Else Print "a - b = "; R
End Sub
```

В записи кода все знакомо по решению предыдущих задач, кроме предпоследней строки. Рассмотрим синтаксис реализующего ветвление условного оператора `If Then Else` ("если то иначе").

Этот оператор изменяет ход выполнения программы в зависимости от результата проверки условий. Он имеет два вида синтаксиса.

Синтаксис 1 (однострочный):

If <выражение> **Then** <операторы then> [**Else** <операторы else>]

Синтаксис 2 (блоковый):

```
If  <выражение1> Then
        [<операторы1>]
[ElseIf <выражение2> Then
        [<операторы2>]
. . .
[Else
         [<операторы n>] ]
End If
```

Параметры *выражение*, *выражение1, выражение2* — логические условия. Это логические константы, переменные и выражения, принимающие значения True (истина) или False (ложь).

Параметры *операторы1*, *операторы2*, ... представляют собой один или более операторов в одной или нескольких строках. Операторы в одной строке разделяются двоеточиями.

При выполнении условного блокового оператора проверяется первое выражение и при его истинности выполняется первый блок операторов. Иначе (если выражение ложно) проверяются все логические условия, следующие за словом **ElseIf** до тех пор, пока не будет найдено условие, принимающее значение "истина". В этом случае выполняется блок операторов, следующий за словом **Then**, иначе (если условие не найдено) выполняется блок операторов, следующий за словом **Else**. Блоки **ElseIf** и **Else** необязательны.

Любые блоки в операторе могут содержать вложенные блоковые операторы `If`. Оператор `If` должен быть первым в строке программы.

Условный однострочный оператор отличается от блокового наличием операторов после слова **`Then`** в этой же строке. Однострочный оператор более подходит для кратких условий и простых действий, приводящих к записи строк небольшой длины. В коде решения задачи примера 3 применен однострочный оператор. При использовании блокового оператора `If` ветвление в коде можно было бы записать так:

```
If  a = b Then
       Print a + b = "; S
Else
       Print "a - b = "; R
End If
```

Блочный оператор `If`, хотя и требует больше строк для записи, но более нагляден, понятен, лучше показывает структуру ветвления. Поэтому он более предпочтителен на начальных этапах знакомства с программированием ветвлений.

Запуск приложения

Напомним, что запуск приложения можно выполнить кнопкой **Запуск** на панели инструментов или клавишей <F5>. После запуска начинают последовательно выполняться операторы и функции кода программы. При запросах приложением значений переменных *a* и *b* можно ввести подготовленные заранее значения *a* = 10 и *b* = 15 (см. листинг 14.2). Ввод их можно выполнять клавишей <Enter>. Эти значения могут служить для проекта простейшим проверочным тестом. На рис. 14.16 показана форма с результатом решения этого тестового задания.

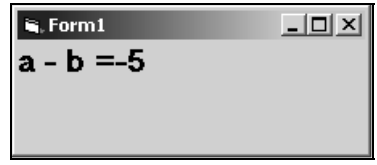


Рис. 14.16. Решение задачи примера 3

Задания для самостоятельной работы

1. Замените в коде программы однострочный оператор `If` на блочный.
2. Выберите и впишите в функции `InputBox` другие тестовые значения переменных.

3. Увеличьте размер шрифта вывода результатов.
4. Научитесь использовать буфер обмена для быстрой записи нескольких последовательно расположенных функций `InputBox`.
5. На рис. 14.17 приведен код приложения "Состояния воды". Запишите алгоритм решения задачи на алгоритмическом языке и в виде блок-схемы. Создайте приложение, проверьте его работу и сохраните в памяти.

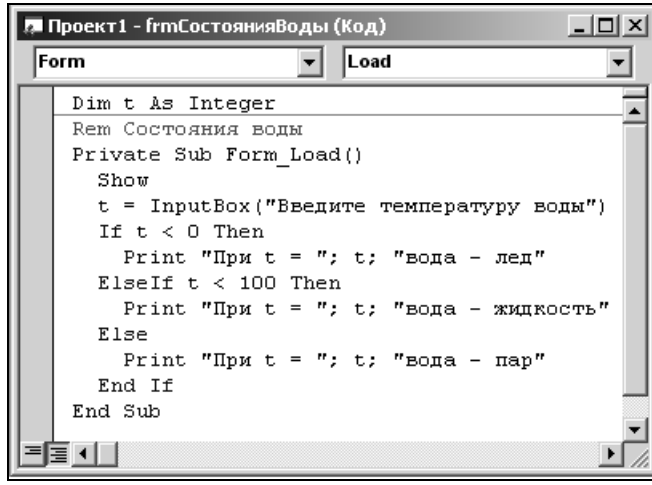


Рис. 14.17. Код приложения "Состояния воды"

Пример 4. Оператор *Select Case*

Оператор `Select Case` соответствует рассмотренным при знакомстве с алгоритмическим языком операторам "выбор" и "выбор иначе". Он позволяет упростить запись кода программ при большом числе ветвлений.

Постановка задачи

Требуется создать приложение "Школьные оценки", которое на ввод одного из чисел 1, 2, 3, 4 и 5 сообщало бы, какая это оценка. Например, на ввод числа 5 сообщало: "Отлично". При записи кода используется оператор выбора `Select Case`.

Запись алгоритма

Обозначим цифру оценки латинской буквой *a* и запишем алгоритм решения задачи на алгоритмическом языке и в виде блок-схемы (рис. 14.18).

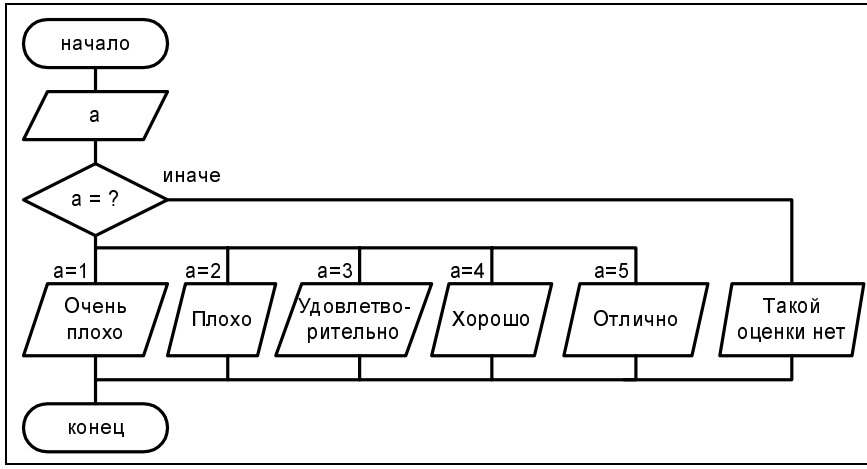


Рис. 14.18. Блок-схема алгоритма решения задачи "Школьные оценки"

```
алг Школьные оценки (а, тексты)
        арг а
        рез тексты
нач
        выбор
                при а = 1: выв "Очень плохо"
                при а = 2: выв "Плохо"
                при а = 3: выв "Удовлетворительно"
                при а = 4: выв "Хорошо"
                при а = 5: выв "Отлично"
                иначе: выв "Такой оценки нет"
        все
кон
```

Рассмотрим синтаксис оператора выбора `Select Case`. Этот оператор разветвляет процесс выполнения программы.

Синтаксис

```
Select Case <выражение для проверки>
  Case <список 1>
    <блок операторов 1>
```

```
        Case <список 2>
           <блок операторов 2>
                    ...
        Case Else
           <блок операторов n>
    End Select
```

Параметр *выражение для проверки* — любое числовое или строковое выражение, в зависимости от значения которого выполняется соответствующий *блок операторов*. Параметры *список 1*, *список 2* и т. д. — это списки выражений, записанных в следующих формах:

- *выражение [,выражение...]*, например:
  `Case 1, 3, 5, 7, 9; Case a, b, c, d`
- *выражение* `To` *выражение*, например:
  `Case 5 To 13; Case a To d`
- `Is` *операция отношения*, например:
  `Case Is<20; Case Is<>a`

В списках могут быть использованы различные формы записи, например:

```
Case 2, 5, 8 To 15, 18, 25 To 30, Is >50
```

В листинге 14.3 показан пример кода с различными вариантами записи выражений для проверки.

Листинг 14.3. Варианты записи выражений для проверки в операторе Select Case

```
k = InputBox("Введи целое число из диапазона 1...10")
Select Case k
  Case 1, 3, 5, 7, 9
    Print "Число "; k; " нечетное"
  Case 2, 4, 6, 8, 10
    Print "Число "; k; " четное"
  Case Is>10
    Print "Введено число большее, чем требуется"
  Case Is<1
    Print "Введено число меньшее, чем требуется"
End Select
```

Начало работы

Запускаем Visual Basic и подаем команду **Файл | Новый проект**. В окне **Новый проект** выбираем **Стандартный EXE**. Щелчками на кнопках **Закрыть** (в

верхнем правом углу окон) убираем все окна, кроме окна конструирования формы с расположенной в нем формой и окна **Свойства**.

Присваивание имен и сохранение проекта

В окне **Свойства** присвойте форме имя **frmШкольныеОценки**, в окне **Проект 1 Свойства проекта** дайте проекту имя **ШкольныеОценки**. Выполните первое сохранение файлов проекта во вновь созданной папке Школьные оценки, вложенной в папку Мои документы\Первые пробы.

Запись кода программы

В соответствии с построенным алгоритмом решения задачи код программы можно записать так, как показано на рис. 14.19.

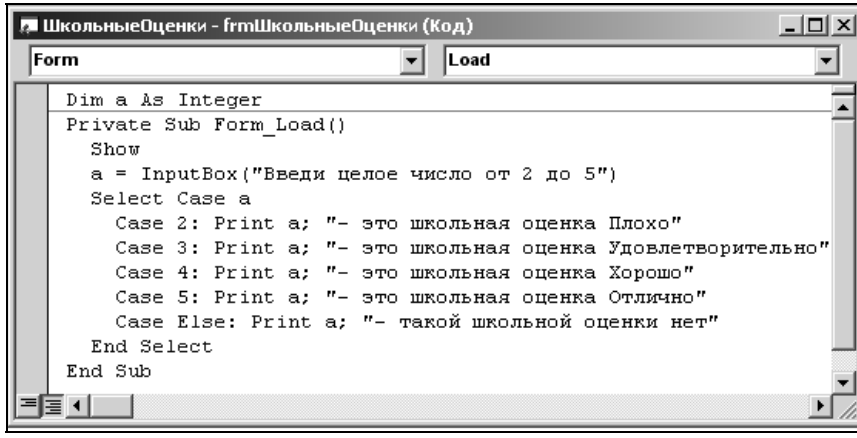


Рис. 14.19. Код программы решения задачи "Школьные оценки"

Запуск и проверка работы приложения

Кнопкой **Запуск** или клавишей <F5> запустите проект. Проверьте работу приложения при различных значений переменной *a* и сохраните проект. На рис. 14.20 показан вывод результата при *a* = 5.

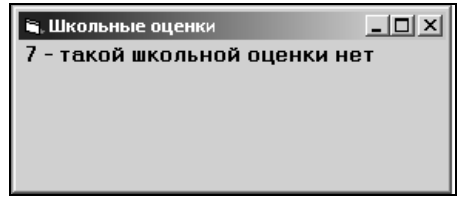


Рис. 14.20. Вывод результата

Задания для самостоятельной работы

1. Заданы два натуральных числа. Записать алгоритм процесса определения того, является ли их среднее арифметическое целым числом.
2. Записать алгоритм-код программы, которая в ответ на ввод времени суток выводит соответствующее пожелание доброго утра, доброго дня, доброго вечера и спокойной ночи.
3. Возраст человека условно разделим на детский, юношеский, возраст взрослого человека и возраст пожилого человека. Записать алгоритм и код программы, отвечающей на ввод с клавиатуры возраста 15 лет — "Вы юноша", возраста 65 лет — "Вы пожилой человек" и т. д. Градации возраста определить самостоятельно.
4. На ввод названия единицы длины приложение отвечает выводом длины этой единицы в сантиметрах: "метр" — 100 см, "аршин" — 71,12 см, "фут" — 30 см, "вершок" — 4,45 см, "дюйм" — 2,54 см, "сажень" — 213,36 см. На ввод другой информации приложение отвечает "Такая единица длины в программе не определена". Ценность такого приложения увеличится, если вы добавите известные вам единицы: "лье", "кабельтов" и др.
5. На ввод номера месяца программа должна печатать, к какому времени года этот месяц относится.

Глава 15

Программирование циклических алгоритмов

15.1. Операторы цикла

Циклические алгоритмы с заранее известным (цикл типа "для") и неизвестным (цикл типа "пока") числом повторений при записи кодов программ используют, как правило, разные операторы цикла. Рассмотрим задачи на применение этих операторов.

При решении простых задач для упрощения работы не всегда будем присваивать имена форме и проекту и сохранять проект. Главное — убедиться в работоспособности написанного кода программы.

Оператор *For Next*

Оператор `For Next` организует в программе выполнение группы операторов заданное число раз.

Его синтаксис:

```
For счетчик = начало To конец [Step шаг]
   [операторы] [Exit For]
Next [счетчик][, счетчик...]
```

Параметр *счетчик* — числовая переменная. Его так и называют *параметром цикла*. Параметры *начало* и *конец* равны соответственно начальному и конечному значениям счетчика. Значение счетчика при прохождении цикла каждый раз увеличивается на величину параметра *шаг*. Если параметр *шаг* не указан, то по умолчанию он принимается равным единице. Шаг может быть отрицательным числом. В этом случае начальное значение счетчика должно превышать конечное. При выходе из цикла параметр *счетчик* имеет значение параметра *конец*, увеличенное (уменьшенное) на величину параметра *шаг*.

Оператор `Exit For` позволяет выйти из цикла при выполнении заданного условия.

Пример 1. Цикл с оператором *For Next*

Постановка задачи

Вычислить сумму целых нечетных чисел от 1 до *n*.

Количество повторений заранее известно. Поэтому для решения задачи требуется построить алгоритм с циклом типа "для".

Формализация

Обозначим: *s* — искомый результат, переменная *a* — это целые нечетные числа (*a* последовательно принимает значения 1, 3, 5, ..., *n*). Будем использовать переменную a в качестве параметра цикла. Для накопления суммы применим формулу $s = s + a$.

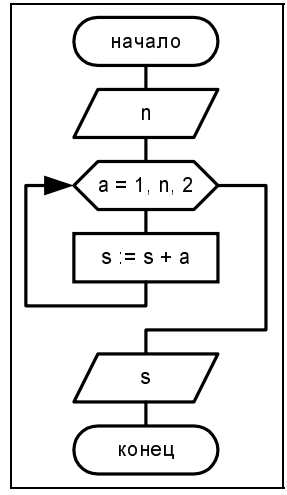


Рис. 15.1. Блок-схема алгоритма к примеру 1

Блок-схема алгоритма решения задачи показана на рис. 15.1. Запись алгоритма на алгоритмическом языке следующая:

алг Сумма нечетных от 1 до n (цел n, s)

арг n

рез s

нач цел a; s:=0
  для a от 1 до n шаг 2
  нц
    s = s + a
  кц
  выв s
кон

При решении задачи используем оператор цикла `For Next`.

Запись кода программы решения задачи

При определении в коде переменных для параметра выбран тип `Integer` — целое число одинарной точности, для суммы выбран тип `Long` — целое двойной точности (рис. 15.2).

Для проверки правильности работы программы желательно иметь хотя бы одно тестовое значение. Его можно подсчитать с помощью калькулятора. Например, при $n = 10$ имеем $s = 25$.

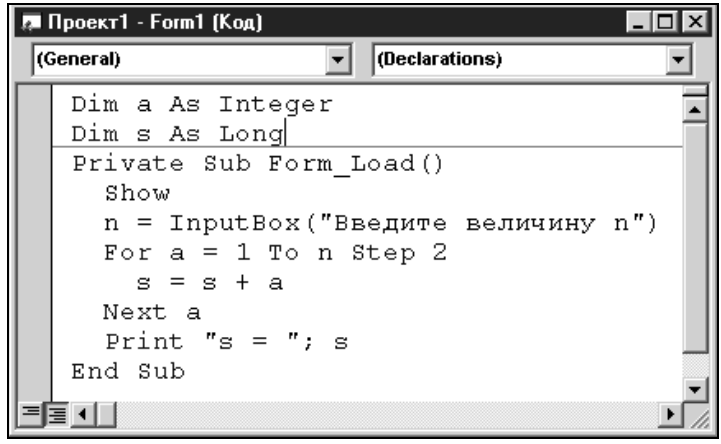


Рис. 15.2. Код решения примера 1

Оператор *Do Loop*

Оператор `Do Loop` организует циклический процесс без счетчика. Такой цикл управляет ходом выполнения программы по указанным условиям.

Синтаксис 1 (цикл с предусловием):

```
Do [{While | Until} выражение]
   [операторы] [Exit Do]
Loop
```

Синтаксис 2 (цикл с постусловием):

```
Do
   [операторы] [Exit Do]
Loop[{While | Until} выражение]
```

Параметр *выражение* представляет собой логическое условие, которое может быть истинным и ложным. Блок операторов, заключенный между ключевыми словами `Do` и `Loop`, является телом цикла.

Блок *операторов* циклически выполняется до тех пор, пока условие, определяемое параметром *выражение*, истинно (при использовании ключевого слова `While`) или до тех пор, пока условие не станет истинным (при использовании `Until`). Другими словами это можно записать так: "Делай" (`Do`) "петлю" (`Loop`) до тех пор, "пока" (`While`) условие выполняется, или "пока" (`Until`) условие не будет выполнено.

Кроме операторов `For Next` и `Do Loop` Visual Basic имеет еще два оператора цикла. Оператор `For Each Next` используется для работы с элементами некоторой заданной группы объектов. Он перебирает все элементы группы и повторяет набор операторов для каждого из элементов. Оператор `While Wend` по своим возможностям несколько уступает оператору `Do Loop`. Подробнее рассматривать эти операторы не будем.

Пример 2. Цикл с оператором *Do Loop*

Постановка задачи

Определить наименьшее количество последовательно расположенных четных целых чисел, начиная с числа 2, сумма которых превысит число *r*.

Число повторений заранее не известно. Это алгоритмом типа "пока".

Формализация и построение алгоритма

Введем обозначения: *s* — накапливаемая сумма, *k* — искомое количество чисел.

Алгоритм простой. В цикле последовательно формируем четные целые, используя формулу a := a + 2, что в результате дает числа 2, 4, 6, 8, ... и т. д. Сумму получаемых чисел накапливаем по формуле s := s + a и подсчитываем, сколько чисел прибавлено с помощью счетчика k := k + 1. Выход из цикла по условию $s \geq r$. Пока $s < r$, следует продолжать суммирование.

Блок-схема алгоритма решения показана на рис. 15.3. Запись на алгоритмическом языке следующая:

```
алг Цикл "пока" (цел r, k)
    арг r
    рез k
нач цел a, s
    пока s < r
    нц
      a := a + 2
      s := s + a
      k := k + 1
    кц
    выв "k = "; k, "s = ";s
кон
```

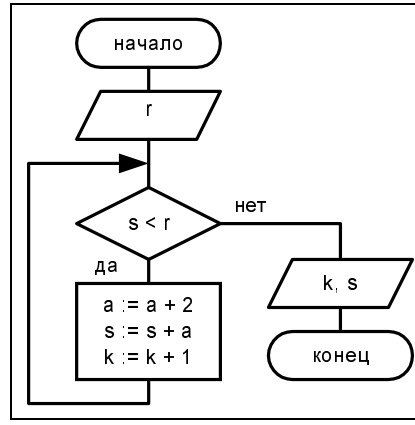


Рис. 15.3. Блок-схема алгоритма для примера 2

Количество повторений неизвестно. Оператор цикла `For Next` использовать сложно. Для решения задач с такой постановкой удобно применять оператор цикла без счетчика `Do Loop`. Код программы показан на рис. 15.4.

Синтаксис основного оператора цикла без счетчика предусматривает возможность проверки условия выхода из цикла как до тела цикла (цикл с предусловием), так и после него (цикл с постусловием). В программе использован первый вариант.

Если развернуть этот вариант по циклам для r = 10, то получим следующее:

s < r	0 < 10	2 < 10	6 < 10	12 < 10
проверка	да	да	да	нет
a = a + 2	a = 2	a = 4	a = 6	
s = s + a	s = 2	s = 6	s = 12	Результат: s = 12
k = k + 1	k = 1	k = 2	k = 3	k = 3

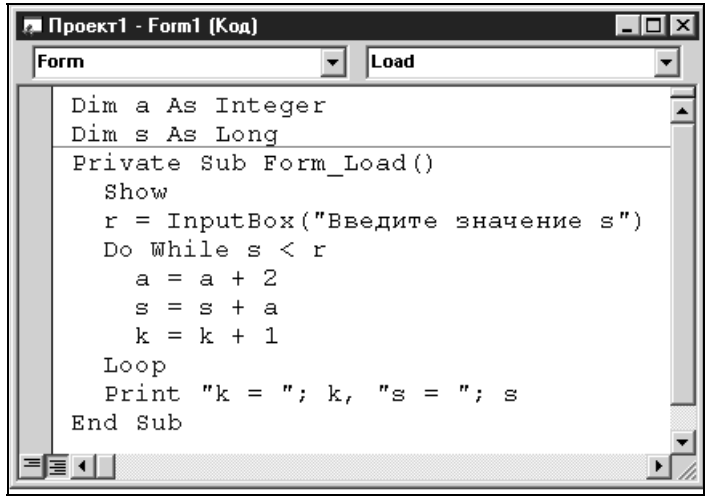


Рис. 15.4. Код программы решения примера 2

Использование операторов *Do Loop* и *End*

При работе с рассмотренными выше проектами, использующими оператор `Select Case`, для получения ответа на новый запрос требовалось вновь запускать приложение кнопкой **Запуск**. Исключить лишние затраты времени позволяет оператор `Do Loop`.

Если функцию `InputBox` и строки оператора `Select Case` записать в качестве тела цикла оператора `Do Loop`, то можно непрерывно и бесконечно задавать приложению вопросы и получать ответы. В этом случае необходимо обеспечить команду окончания работы с приложением. Это можно сделать, например, с помощью оператора `End`, закрывающего приложение. Можно также использовать комбинацию клавиш <Ctrl>+<Break>, которая помогает прерывать бесконечные вычисления.

На рис. 15.5 показан код программы, реализующий непрерывность запросов и ответов. Значения вводимой при запросе температуры — вся числовая ось (числа положительные, отрицательные, целые и дробные). Для сокращения записей в коде вводятся две строковых переменных *a* и *b*.

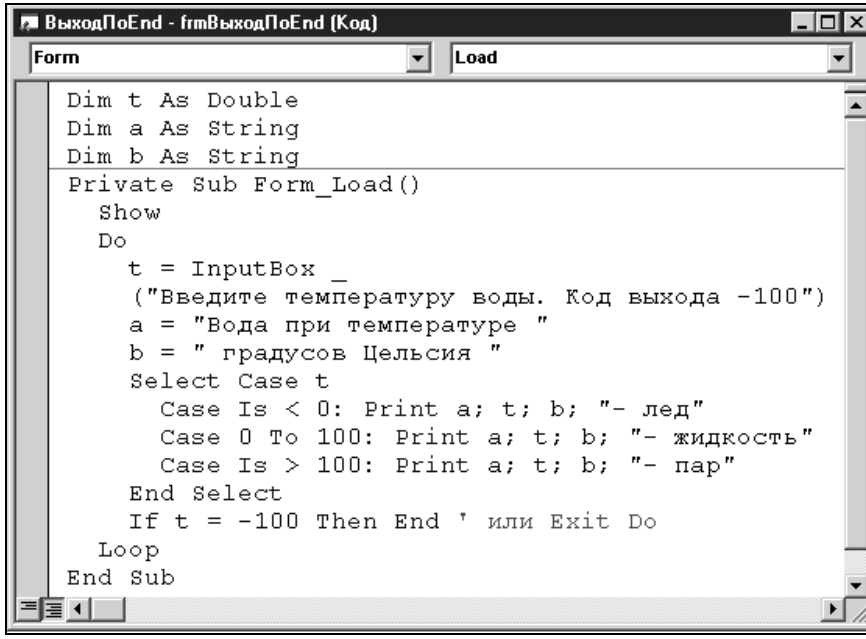


Рис. 15.5. Код программы
с завершением работы приложения по условию

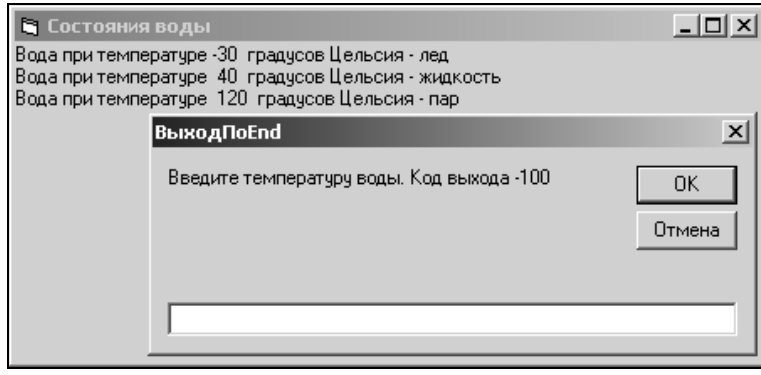


Рис 15.6. Окна формы и функции `InputBox`

В тексте функции `InputBox` добавлена информация о коде выхода из цикла и завершения работы приложения. Для уменьшения длины логической строки она символом подчеркивания разбита на две физических строки.

Беспрерывный процесс ввода вопросов и вывода ответов позволяет накапливать ответы на форме (рис. 15.6). Исключить накопление можно записанным в начале тела цикла методом `Cls`.

15.2. Вложенные циклы

Циклы, как `For Next`, так и `Do Loop` могут быть вложенными. Вложенным называют цикл, входящий в тело другого цикла. Вложенность может быть многократная. Рассмотрим алгоритм и программу создания таблицы умножения.

Таблица умножения

Пример 3. Выведение на экран таблицы умножения

Постановка задачи

Вывести на экран таблицу умножения.

Запись на алгоритмическом языке будет такой:

```
алг Таблица умножения (цел i, j, цел таб U[1:9;1:9])
   арг i, j
   рез U[1:9;1:9]
нач
   для i от 1 до 9
   нц
      для j от 1 до 9
      нц
         выв i*j
      кц
   кц
кон
```

Блок-схема алгоритма показана на рис. 5.7. Внешний цикл с параметром *i* перебирает все номера строк таблицы умножения от 1 до 9. Для каждого номера строки во внутреннем цикле перебираются все номера столбцов *j* таблицы и печатается произведение номера строки на номер столбца. Код программы приведен в листинге 15.1.

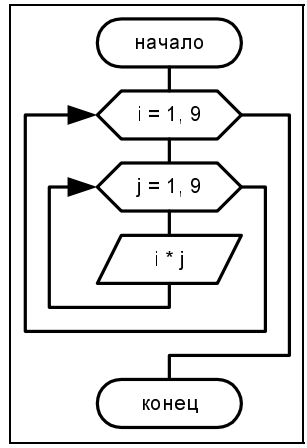


Рис. 15.7. Вложенные циклы. Таблица умножения

Листинг 15.1. Код программы "Таблица умножения"

```
Private Sub Form_Load()
  Show
  PRINT "Таблица умножения"
  For i = 1 To 9              'внешний цикл, номера строк таблицы
    For j = 1 To 9            'внутренний цикл, номера столбцов таблицы
      Print Format(i * j, "0#"); " ";      'печать произведения
    Next j
    Print                     'переход к печати на следующей строке
  Next i
End Sub
```

Функция *Format*

В программе используется функция `Format` форматирования данных при их выводе оператором `Print`. Функция `Format` представляет данные в виде, который выбирает или задает пользователь. С функцией `Format` можно познакомиться в *приложении 2*. Для вывода квадратной таблицы умножения в коде программы результат форматируется таким вызовом функции:

```
Format(i*j, "0#").
```

Синтаксис функции:

Format (*выражение*, "*строка символов*")

При записи функции `Format` используются символы:

- `0` (ноль) — резервирует место для цифр числа, при отсутствии цифры отображается ноль;
- `#` — резервирует место для цифр числа, при отсутствии цифры ничего не выводится;
- `%` — резервирует процентное изображение числа;
- `$` — выводит перед числом знак `$`.

Задания для самостоятельной работы

1. Записать и отладить код программы вывода таблицы квадратов и кубов чисел от 1 до 10. Столбцы таблицы должны иметь обозначения, например, `a`, `a^2`, `a^3`.
2. Записать и отладить код программы, определяющей для заданного натурального числа все его делители.
3. Проверить, существует ли натуральное число $a < 100$, которое обладает следующими свойствами: `a Mod 3 = 1`, `a Mod 4 = 2`, `a Mod 5 = 3`, `a Mod 6 = 4`. Сколько таких чисел?
4. Создать приложение, выводящее на форму таблицу значений температуры по Цельсию от 0 до 100 градусов с дискретностью в 5 градусов и их эквивалентов по шкале Фаренгейта, используя формулу $T_ф = 9T_ц / 5 + 32$.
5. Создать приложение, определяющее для натуральных чисел n ($n < 10$) тройки таких натуральных чисел x, y, z, что $n = x_2 + y_2 + z_2$.
6. Создать приложение, определяющее, является ли задаваемое с клавиатуры число простым. К простым относятся числа, большие единицы и не имеющие других делителей, кроме единицы и самого себя.
7. Запишите две программы определения суммы первых двадцати натуральных чисел. В первой программе используйте оператор `For Next`, во второй — `Do Loop`. Сравните программы и сделайте выводы.

Замечание

Результаты трех следующих задач могут удивить и озадачить. Тысячелетиями человечество, выполняя различные вычисления, находило среди чисел и результатов операций над ними интересные закономерности. Некоторым числам, например, 3, 7, 13, 666 и т. п. придавалось мистическое значение. В наше время можно целенаправленно заниматься поиском различных "фокусов" с числами. Как правило, такой поиск требует значительных переборов вариантов и по силам только компьютеру.

8. Создайте приложение, которое печатает в столбец произведения чисел $a = 143$, $b = 777$ и числа c, последовательно принимающего значения 1, 2, 3, ..., 9. Используйте цикл со счетчиком.
9. Используя цикл со счетчиком, создайте приложение, которое печатает в столбец произведения числа $a = 123\,456\,789$ на числа 9, 18, 27, ..., 81.
10. Замечены следующие закономерности:

$1 \cdot 9 + 2 = 11$; $9 \cdot 9 + 7 = 88$; $\underline{6}^2 = 3\underline{6}$;

$12 \cdot 9 + 3 = 111$; $98 \cdot 9 + 6 = 888$; $\underline{76}^2 = 57\underline{76}$;

$123 \cdot 9 + 4 = 1111$; $987 \cdot 9 + 5 = 8888$; $\underline{376}^2 = 141\,\underline{376}$;

Проверьте и продолжите эти последовательности.

15.3. Табуляция функций

Часто по формуле функции трудно сказать о ее поведении при изменении аргументов. В таких случаях функцию табулируют — строят таблицу значений функции при различных значениях ее аргументов. Примером таких таблиц могут служить таблицы Брадиса.

Например, легко выполнить табуляцию функции $F(x) = 2x^2 - 5x$. Выберем диапазон изменения аргумента x от -5 до 5 с шагом 1. Подставляем значения аргумента в формулу функции и вычисляем значение функции. Заполняем таблицу:

x	−5	−4	−3	−2	−1	0	1	2	3	4	5
$F(x)$	75	52	33	18	7	0	−3	−2	3	12	25

Табуляция эффективно выполняется с помощью компьютера. Простейшие программы табуляции — это циклические программы, в которых в качестве параметров циклов выступают аргументы табулируемой функции. Программа табуляции функции одного аргумента $F(x)$ выводит на экран дисплея или печать таблицу из двух колонок: задаваемые значения x и вычисленные значения $F(x)$.

При исследовании поведения функции возникает задача выбора диапазона и шага изменения аргумента. Это в большой степени зависит от вида функции и задачи анализа. Например, функция может быть периодической (все тригонометрические функции и подобные им). В этом случае в качестве диапазона достаточно взять один-два периода. При табуляции непериодических функций диапазон изменения аргумента нередко подбирают эмпирически, опытным путем.

При выборе шага изменения аргумента также исходят из вида анализируемой функции. Слишком большой шаг может привести к пропуску некоторых особых точек функции (максимальное, минимальное значения), не позволит увидеть, например, ее периодичность. Малый шаг изменения приведет к излишне большому числу значений, что затруднит анализ функции и увеличит время табуляции.

Рассмотрим некоторые из программ табуляции. При небольшом числе значений аргумента их можно вводить с помощью функции `InputBox`. При равномерном шаге изменения значений аргумента их удобно вырабатывать в качестве параметра цикла.

Пример 4. Табуляция функций

Постановка задачи

Табулировать функцию $f(x) = 3x^2 + 2x$ в диапазоне значений аргумента x от -5 до 10 с шагом 0,5.

Код программы, решающей задачу примера 4, приведен в листинге 15.2.

Листинг 15.2. Код программы табуляции функции из примера 4

```
Dim x As Single
Dim f As Single
Private Sub Form_Load()
  Show
  WindowState = 2              'Форма развернута
  For x = -5 To 10 Step 0.5
    f = 3 * x ^ 2 + 2 * x
    Print " При x = "; x, " f(x) = "; f
  Next
End Sub
```

Для исключения прерываний работы программ табуляции функций, имеющих сложные формулы, необходимо производить их анализ на особые точки: деление на ноль или нулевой результат возводится в отрицательную степень, отрицательное подкоренное выражение — отрицательный аргумент функции `Sqr`, отрицательное или нулевое значение аргумента функции `Log`.

Следующая программа табуляции имеет такую проверку функции на отрицательное подкоренное выражение и деление на ноль.

Пример 5. Проверка функции на особые точки

Постановка задачи

Разработать алгоритм, записать код программы, ввести и отладить проект табуляции функции $f(x) = \dfrac{1 - \sqrt{4 - x}}{2 - x}$ в диапазоне значений аргумента x от -5 до 5 с шагом 1 и проверкой на особые точки.

Код программы, решающей задачу примера 5, приведен в листинге 15.3.

Листинг 15.3. Табуляция с проверкой на особые точки функции из примера 5

```
Dim x As Single
Dim f As Single
Private Sub Form_Load()
  Show
  For x = -5 To 5
    If 4 - x < 0 Or 2 - x = 0 Then
      Print "   При x = "; x; "   решения нет"
    Else
      f = (1 - Sqr(4 - x)) / (2 - x)
      Print "   При x = "; x, "f(x) = "; f
    End If
  Next
End Sub
```

Программе соответствует блок-схема, показанная на рис.15.8.

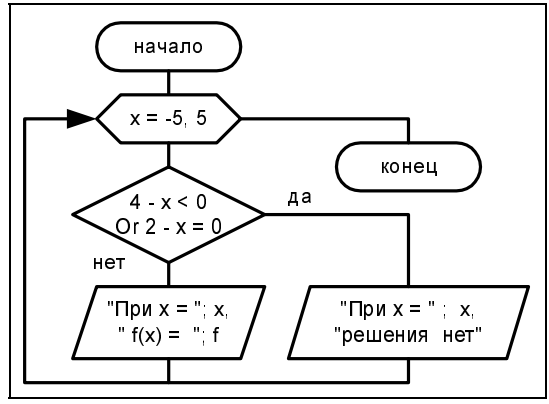


Рис. 15.8. Блок-схема алгоритма табуляции функции

Задания для самостоятельной работы

Записать и отладить программы табуляции следующих функций:

1. $f1(x) = 2ae^{2x} - \dfrac{\sqrt{1 - 1{,}5\sin^2 x}}{0{,}3ab - x}$;

   $a = 0{,}2$; $b = 1{,}8$; x принимает значения из диапазона [−2, 2] с шагом 0,2.

2. $f2(x) = 5\sqrt{\dfrac{|\,2\cos^3 2x^2 - 3\sin^2 3x^3\,|}{0{,}2 - x}}$;

   аргумент x принимает значения из диапазона [−1, 1] с шагом 0,1.

3. $f3(x) = \dfrac{2 + 0{,}5a}{ab^2 x} + 4{,}7\sqrt{1 - 1{,}6\cos x}$;

   $a = 3$; $b = 0{,}8$; аргумент x принимает значения из диапазона [−2, 2] с шагом 0,2.

4. $f4(x) = 3ae^{x} + \dfrac{3a - 4tg^2 x}{\sqrt{0{,}8 - \sin x}}$;

   $a = 12{,}6$; x принимает значения 0; 0,2; 0,4; 0,7, 1,1; 1,6; 2,2; 2,9; 3,7.

15.4. Программирование задач на прогрессии

Прогрессии — это последовательности. Последовательностями называют упорядоченные множества определенным образом полученных величин. Числовой последовательностью называют множество чисел, каждое из которых снабжено своим номером. Последовательности делятся на конечные и бесконечные, возрастающие и убывающие. Примерами могут служить последовательности простых чисел: 2, 3, 5, 7, 11, 13, ... (бесконечная), натуральных чисел: 1, 2, 3, ..., 100 (конечная, возрастающая); 1, 2, 3, ... (бесконечная). В общем виде последовательность можно записать так: a_1, a_2, a_3,... , a_n, ...

Рассмотрим построение последовательности чисел Фибоначчи. Итальянский математик Леонардо Пизанский (Fibonacci) пришел к этой последовательности, решая задачу о размножении кроликов. Первый и второй элементы этой последовательности равны единице. Каждый из последующих элементов равен сумме двух предыдущих: 1, 1, 2, 3, 5, 8, 13, ... Формирование элементов последовательности чисел Фибоначчи, начиная с третьего, выполняется по формуле:

$$a_n = a_{n-1} + a_{n-2}\,.$$

Такие формулы называют рекуррентными (от лат. *recurrere* — возвращаться). Рекуррентные соотношения связывают последующий элемент последова-

тельности с предыдущими элементами. Из простейших последовательностей выделяют арифметические и геометрические прогрессии.

В арифметической прогрессии каждый последующий элемент отличается от предыдущего на некоторое постоянное для данной прогрессии число, называемое *разностью*. К геометрическим прогрессиям относятся такие последовательности чисел, в которых каждый последующий член отличается от предыдущего на некоторый постоянный множитель, называемый *знаменателем*.

Далее будем пользоваться следующими обозначениями:

- a — начальный, первый член прогрессии (арифметической и геометрической);
- x — любой член прогрессии;
- d — разность в арифметической прогрессии;
- q — знаменатель в геометрической прогрессии;
- n — номер члена прогрессии;
- k — количество членов прогрессии;
- p — шаг изменения номеров членов прогрессии.

Пусть $a = 2$, $d = 3$. Получаем следующую арифметическую прогрессию: 2, 5, 8, 11, 14, 17, ... Рекуррентная формула вычисления n-го члена прогрессии следующая:

$a_n = a_{n-1} + d$.

Значение n-го члена прогрессии также можно определить по формуле:

$x = a + d \cdot (n - 1)$.

Заметим, что при $n = 1$ получаем начальный, первый член прогрессии, при $n = 2$, получаем второй член и т. д.

Пусть $a = 1$, $q = 2$. Получаем следующую геометрическую прогрессию: 1, 2, 4, 8, 16, 32,... . Рекуррентная формула, связывающая n-й член прогрессии с предыдущим, следующая:

$a_n = a_{n-1} \cdot q$.

Значение n-го члена геометрической прогрессии, как функции начального члена и знаменателя, можно определить по формуле:

$a_n = a \cdot q^{n-1}$.

В отношении числовых последовательностей, в том числе и прогрессий, возможна постановка, например, таких задач:

- определить сумму или произведение заданного числа элементов последовательности, возможно, с ограничениями на номера элементов — например, учитывать только элементы с четными или нечетными номерами, с номерами, кратными некоторому целому числу, и т. п.;

- определить количество элементов последовательности, которые нужно просуммировать (перемножить), чтобы их сумма (произведение) превысили заданную величину;
- определить величину и номер первого элемента возрастающей (убывающей) последовательности большего (меньшего) заданного числа;
- определить величину элемента последовательности с заданным номером и т. д.

Приведем примеры программ, решающих некоторые из перечисленных задач.

Пример 6. Сумма членов прогрессии

Постановка задачи

Разработать алгоритм, записать код программы, ввести и отладить проект определения суммы *k* членов арифметической прогрессии.

Код программы, решающей задачу примера 6, приведен в листинге 15.4.

Листинг 15.4. Вычисоение суммы членов арифметической прогрессии

```
Dim d As Single: Dim k As Integer
Private Sub Form_Load()
  Show
  a = InputBox("Введите первый член прогрессии")
  d = InputBox("Введите разность прогрессии")
  k = InputBox("Введите число членов суммы")
  For n = 1 To k
    x = a + d * (n - 1)   'формула n-го члена
    s = s + x
  Next
  Print "при a = "; a, " d = "; d, " k= "; k, " s= "; s
End Sub
```

Тело цикла приведенной выше программы с использованием рекуррентной формулы можно было записать так: s = s + a: a = a + d.

Пример 7. Номер члена геометрической прогрессии

Постановка задачи

Разработать алгоритм, записать код программы, ввести и отладить проект определения величины и номера члена геометрической прогрессии, первым превысившего заданное число *z*.

Код программы, решающей задачу примера 7, приведен в листинге 15.5.

Листинг 15.5. Определение величины и номера члена геометрической прогрессии

```
Dim a As Single: Dim q As Single: Dim z As Single
Private Sub Form_Load()
  Show
  a = InputBox("Введите первый член прогрессии")
  q = InputBox("Введите знаменатель прогрессии")
  z = InputBox("Введите число z")
  Do
    x = a * q ^ n
    n = n + 1
  Loop While x < z
  Print "При a= "; a, "q="; q, "z= "; z
  Print "x= "; x, "n= "; n
End Sub
```

Для большей убедительности результатов можно в цикле распечатывать всю последовательность членов прогрессии. Это можно сделать последней строкой тела цикла `Print x`.

Задания для самостоятельной работы

1. Заданы начальный член и разность арифметической прогрессии. Записать алгоритм и программу определения величины члена этой прогрессии с номером k.
2. Сколько нужно взять членов арифметической прогрессии с заданными первым членом и разностью, чтобы их сумма превысила заданное число q?
3. Разработать алгоритмы и составить программы вычисления заданного числа членов арифметической прогрессии: а) по любым двум ее членам, номера которых известны; б) по любому члену прогрессии, номер которого известен, и разности прогрессии.
4. Записать алгоритм и программу проверки: являются ли вводимые с клавиатуры числа a, b, c членами какой-либо прогрессии (арифметической или геометрической).
5. Числовая последовательность образуется следующим образом. Первый элемент последовательности — натуральное произвольное число, кратное трем. Каждый последующий элемент последовательности равен сумме ку-

бов всех цифр предыдущего элемента. Записать программу, которая покажет, что, начиная с некоторого элемента, такая последовательность становится постоянной и равной некоторому числу. Чему равно это число?

6. Построим последовательность натуральных чисел. Начальный элемент — натуральное число с четырьмя цифрами, которые не все равны между собой. Переход от предыдущего элемента последовательности к последующему выполняется по такому правилу. Цифры предыдущего числа записываются в убывающем порядке. Из полученного таким образом числа вычитается число, полученное записью цифр предыдущего числа в возрастающем порядке. Результат этого вычитания является следующим членом последовательности. Запишите программу, которая докажет, что для любого числа, удовлетворяющего заданному ограничению, формируемая таким образом последовательность становится постоянной и равной некоторому числу. Какому числу?

Замечание

Комментарии служат обязательным элементом хорошего стиля записи программ. Общепринято мнение, что хорошая программа, как правило, снабжается подробными комментариями. Назначение программы, принятые ограничения, особенности ее построения и работы со временем забываются и восстанавливаются с большим трудом. Комментарии в программу записываются с помощью оператора `Rem` (ремарка), который имеет второй вид записи в виде одиночной кавычки ('). Например: `Rem Задание начальных значений`

15.5. Работа с массивами

Работа с массивами — одна из наиболее часто встречающихся операций обработки информации. Напомним, что массивы представляют в программах организованные группы элементов одного типа. В качестве элементов массива выступают переменные с индексами — номерами элементов в массиве. Индексы — целые числа. По умолчанию они начинаются с нуля. Различают размерность массива (число измерений) и размеры размерностей массива (число элементов в каждом измерении). Массивы делятся на статические и динамические. Примером массива может служить список фамилий учеников в классном журнале. Это одномерный массив строкового типа. Таблица умножения — двухмерный массив целого типа.

Статические и динамические массивы

Статические массивы

Параметры массивов этого типа задаются в процессе проектирования и не могут изменяться во время выполнения программы.

При задании массивов им присваиваются имена. Рядом с именем в скобках указывается значение наибольшего индекса элементов массива. Например, массив `w(16)` — одномерный массив, имеет одно измерение. Номера элементов массива: 0, 1, 2, ..., 16. Всего в этом массиве 17 элементов. `w(i)` — обозначение *i*-го элемента массива. Индекс *i* может принимать в данном примере значения от 0 до 16. Граничные значения индексов (номеров) такого массива можно записать еще так: `w(0 To 16)`.

Массив `v(10,10)` двухмерный. Он имеет две размерности по 11 элементов в каждой. Двухмерный массив `z(n, m)` можно представить в виде таблицы, имеющей $m + 1$ строк и $n + 1$ столбцов. Элемент `z(i, j)` в такой таблице находится на пересечении *i*-ой строки и *j*-го столбца. Допустимы такие варианты записи границ массива: `v(0 To 10, 0 To 10)` и `v(0 To 10, 10)`.

Определяются массивы так же, как и переменные, но не могут определяться внутри процедур.

Синтаксис:

{Dim | Static | Public} <*Имя (границы)*> As <*тип*>

Максимальное число размерностей в массиве — 60. Максимальный размер (число элементов) каждой размерности — 32 767.

Динамические массивы

Массивы этого типа могут изменять границы своих индексов в процессе работы программы. Так как предельные значения индексов заранее не известны, то динамические массивы определяются с пустыми скобками, например:

```
Dim <ИмяМассива> ( ) As String
```

Коррекция границ массива производится с помощью оператора `ReDim` внутри процедуры, в которой требуется изменить число элементов массива или его размерность. Если начальное определение массива определяет его тип как `Variant`, то оператором `ReDim` можно задать и тип массива. Чтобы при изменении размеров или размерности массива исключить потери информации, в строку определения после оператора `ReDim` записывают ключевое слово `Preserve`. Один оператор `ReDim` может доопределить несколько массивов:

ReDim [**Preserve**] *Имя1*(*границы*) [**As** type] [, *Имя2*(*границы*) [**As** type]] ...

Ввод массивов в программу

Способов ввода массивов в программу множество. Применение того или иного способа зависит от многих факторов. Во-первых, от того, служат ли данные массива для получения рабочего результата или только для отладки программы. Во-вторых, от того, должен ли массив обладать некоторыми

заранее заданными свойствами или требования к нему не определены. В-третьих, от того, задана ли размерность массива или написана для общего случая. И т. д. Выбор способа ввода позволяет экономить время (например, устраняет необходимость ввода данных при каждом запуске программы при ее отладке), может сделать ввод более надежным, контролируемым.

Рассмотрим некоторые из способов ввода статических и динамических массивов. Для контроля правильности ввода в кодах программ будет предусмотрен вывод массивов.

Ввод массива присваиванием значений

Простой способ ввода элементов массива при небольшом его размере можно выполнить, используя операцию присваивания. В этом случае для изменения значений элементов приходится работать с кодом программы. В листинге 15.6 показан код программы, в которой определяется массив из пяти целых чисел. Ввод массива выполнен присваиванием значений элементам массива. Вывод массива производится в одну колонку (рис. 15.9). Выводится имя элемента с номером и его значение.

Листинг 15.6. Вариант ввода и вывода массива

```
Dim a(4) As Integer
Private Sub Form_Load()
  Show
  a(0) = 7: a(1) = 5: a(2) = 8: a(3) = 12: a(4) = 3
  For i = 0 To 4
    Print "a("; i; ") = "; a(i)
  Next
End Sub
```

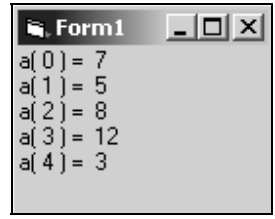


Рис. 15.9. Вариант вывода массива в столбец

Ввод массива и функция *InputBox*

Приведенный в листинге 15.7 код программы определяет массив фиксированной длины с явно заданной нижней границей номеров элементов. Все

пять элементов массива строкового типа. Они вводятся в цикле по запросам функции `InputBox`. Вывод массива организован в одну строку (рис. 15.10).

Листинг 15.7. Ввод массива с помощью функции `InputBox`

```
Dim b(1 To 5) As String
Private Sub Form_Load()
  Show
  For i = 1 To 5
    b(i) = InputBox ("Введите элемент массива с номером  " & i )
  Next
  For i = 1 To 5: Print " b("; i; ")="; b(i),: Next
End Sub
```

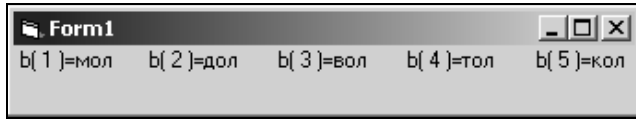


Рис. 15.10. Вариант вывода массива в одну строку

Ввод динамического массива

С помощью функций `InputBox` вводятся размер массива и значения элементов. Как правило, программы пишут для наиболее общих случаев. Это согласуется с возможностью ввода размера массива с клавиатуры.

В листинге 15.8 приведен код программы ввода динамического массива.

Листинг 15.8. Ввод динамического массива

```
Dim c() As Integer
Private Sub Form_Load()
  Show
  n = InputBox("Введите размер массива")
  ReDim c(n)
  For i = 0 To n: c(i) = InputBox("Введите значение элемента " & i): Next
  For i = 0 To n: Print "c("; i; ") = "; c(i): Next
End Sub
```

Использование функции *Array*

Функция `Array(<список аргументов>)` позволяет задавать список, состоящий из величин (не обязательно одного типа), с которыми можно выполнять различные действия. В этом случае по умолчанию элементам массива присваивается тип `Variant`.

В листинге 15.9 приведен код программы ввода массива с использованием функции `Array`.

Листинг 15.9. Ввод массива и функция `Array`

```
Private Sub Form_Load()
  Show
  a = Array("CO", 12, "LOR", 25)
  b = a(0) + a(2): Print "b="; b                    'печатается b = COLOR
  c = a(1) + a(3): Print "c="; c                    'печатается c = 37
End Sub
```

Функцию `Array` удобно использовать для задания тестов при проверках функционирования программ обработки массивов, как показано в листинге 15.10.

Листинг 15.10. Ввод функцией `Array` тестовой последовательности

```
Private Sub Form_Load()
  Show
  a = Array(5, -7, 0, 4, 0, 9, -6, 4)
  For i = 0 To 7
    Print a(i);
    s = s + a(i)
  Next
  Print: Print "s="; s
End Sub
```

Использование случайных чисел

Случайными называют числа, значение которых заранее неизвестно. Случайные числа находят широкое применение в игровых программах, различных прикладных программах. При отладке программ удобен ввод массивов случайных чисел, генерируемых с помощью функции `Rnd`. Такие массивы служат простейшими тестами, проверяющими работоспособность создаваемых программ решения задач.

Для того чтобы функция `Rnd` могла генерировать с каждым запуском различные последовательности чисел, она используется совместно с оператором `Randomize`. В листинге 15.11 показан код программы создания и вывода массива из семи случайных целых чисел.

Листинг 15.11. Функция `Rnd`. Ввод массива случайных чисел

```
Dim d(1 To 7) As Integer
Private Sub Form_Load()
  Show
  Randomize
  For i = 1 To 7
    d(i) = Int((Rnd * 10) + 1)
    Print d(i);
  Next
End Sub
```

В программах элементы массивов носят различную смысловую нагрузку: вес и рост обучаемых в программах учета данных диспансеризации, температура воздуха и атмосферное давление в программах моделирования погоды, другие применения. Желательно, чтобы интервалы изменения случайных чисел, присваиваемых элементам этих массивов, были близки к реальным значениям. Ниже приведены варианты присваивания элементам массива различных последовательностей случайных чисел.

```
a(i)=INT(RND*100+1)                'натуральные числа [1, 100]
b(i)=INT(RND*100 - 50)             'целые числа из интервала [-50, 49]
c(i)=INT(RND*100 - 50) / 10        'дроби из интервала [-5.0, 4.9]
d(i)=INT(RND*100 - 30)             'целые, больше положительных
e(i)=INT(RND*180 + 50)             'рост человека
f(i)=INT(RND*40 + 740)             'атмосферное давление
```

Решение задач, связанных с обработкой массивов

К простейшим алгоритмам обработки массивов можно отнести определение в заданном числовом массиве количества, суммы и среднего арифметического заданных элементов (положительных, отрицательных, равных нулю или заданному числу, превышающих заданную величину и т. п.). Далее по нарастанию сложности идут алгоритмы определения максимальных и ми-

нимальных по величине элементов и алгоритмы сортировки (упорядочивания) массивов.

Пример решения простой задачи

В *гл. 12* (пример 9) была показана разработка алгоритма решения следующей задачи.

Динамическому массиву целых чисел `a()` задается размер и он формируется с помощью функции `Rnd`. Требуется записать программу определения среднего арифметического положительных элементов и количества нулевых элементов массива.

Программное решение этой задачи приведено в листинге 15.12.

Листинг 15.12. Решение примера 9 из *главы 12*

```
Dim a() As Integer: Dim s As Integer: Dim k0 As Integer
Dim k1 As Integer: Dim sr As Single
Private Sub Form_Load()
  Randomize: Show
  n = InputBox("Введите размер массива", , 10)
  ReDim a(n)
  For i = 0 To n
    a(i) = Int((Rnd * 10) - 5): Print a(i);
  Next
  For i = 1 To n
    If a(i) > 0 Then s = s + a(i): k1 = k1 + 1
    If a(i) = 0 Then k0 = k0 + 1
  Next
  sr = s / k1: Print
  Print "sr="; sr, "k0="; k0
End Sub
```

Определение максимального по величине числа

Алгоритм поиска максимума среди элементов массива заключается в следующем. Переменной `MAX` присваивается значение первого элемента массива. В цикле просматриваются все элементы массива и последовательно сравниваются со значением переменной `MAX`. Если при сравнении очередной элемент превышает значение переменной `MAX`, то переменной `MAX` присваи-

вается значение этого элемента. По окончании просмотра значение переменной MAX равно наибольшему по величине элементу массива.

Аналогично строится алгоритм определения в массиве минимального по величине элемента: переменной MIN присваивается величина первого элемента и значения переменной MIN последовательно сравнивается с элементами массива. Если очередной элемент меньше MIN, то MIN присваивается значение этого элемента.

Код программы поиска в массиве максимального элемента приведен в листинге 15.13.

Листинг 15.13. Поиск в массиве максимального элемента

```
Dim b() As Integer: Dim max As Integer
Private Sub Form_Load()
  Randomize: Show
  n = InputBox("Введите размер массива", , 12)
  ReDim b(n)
  For i = 0 To n
    b(i) = Int(Rnd * 100 + 1): Print b(i);
  Next
  max = b(0)
  For i = 1 To n
    If b(i) > max Then max = b(i)
  Next
  Print: Print "max = "; max
End Sub
```

Работа с динамическими массивами

Пусть требуется сформировать два массива. Первый массив — массив случайных целых чисел. Второй — состоит из отрицательных элементов первого массива.

Показанный в листинге 15.14 код программы определяет два динамических массива. После доопределения размера первый массив формируется в цикле с помощью функции Rnd и выводится в форму.

В следующем цикле формируется второй массив из отрицательных элементов первого массива. Размер второго массива заранее неизвестен. Поэтому его размер доопределяется при добавлении каждого нового отрицательного числа. В третьем цикле второй массив выводится на форму.

Листинг 15.14. Работа с двумя динамическими массивами

```
Dim a() As Integer: Dim b() As Integer: Dim n As Integer: Dim m As Inte-
ger
Private Sub Form_Load()
  Randomize: Show
  n = InputBox("Введите размер массива", , 20)
  ReDim a(n)
  For i = 0 To n: a(i) = Int((Rnd * 100) - 50): Print a(i);: Next: Print
  For i = 0 To n
    If a(i) < 0 Then
      m = m + 1: ReDim Preserve b(m): k = m: b(k) = a(i)
    End If
  Next
  For i = 1 To m: Print b(i);: Next
End Sub
```

Двухмерные массивы

Для формирования и вывода двухмерных массивов используются вложенные циклы. Показанный в листинге 15.15 код программы решает следующую задачу.

Создается двухмерный массив целых чисел `w(1 To n, n)`. В форму выводится в виде таблицы массив и сумма элементов главной диагонали таблицы (`i = j`).

Листинг 15.15. Двухмерные массивы

```
Dim w() As Integer: Dim n As Integer: Dim s As Integer
Private Sub Form_Load()
  Randomize: Show
  n = InputBox("", , 4): ReDim w(1 To n, n)
  For i = 1 To n
    For j = 1 To n
      w(i, j) = Int(Rnd * 10)
      If i = j Then s = s + w(i, j)
      Print w(i, j);
    Next j: Print
  Next i
  Print "s="; s
End Sub
```

Задания для самостоятельной работы

Замечание

Как правило, одну и ту же задачу можно решить различными способами. Наиболее привлекательны при этом более простые, оригинальные, быстрые алгоритмы решения. При реализации алгоритмов в виде кодов программам приходится решать задачу выбора операторов, оформления ввода данных, вывода результатов, записи комментариев и т. д. Считают, что лучшей из решающих задачу программ будет та, которая при прочих равных условиях имеет в своем составе меньше операторов. Такую программу быстрее вводить, легче исправлять и отлаживать.

1. Записать алгоритм и программу определения разности между средним арифметическим всех элементов массива и средним арифметическим его максимального и минимального элемента. Массив целых чисел задать с помощью генератора случайных чисел.
2. Записать алгоритм и программу определения в массиве целых чисел максимального и минимального по величине элементов и их номеров в массиве. Массив задать с помощью генератора случайных чисел.
3. Записать алгоритм и программу определения расстояния в массиве (разности номеров) между максимальным и минимальным элементами. Придумать массив-тест и ввести его с помощью функции `InputBox`.
4. Сформировать двухмерный массив случайных целых чисел `a(1 TO n,n)` и определить разность между средним арифметическим элементов главной диагонали (`i = j`) и максимальным из ее элементов.
5. Вывести на экран таблицу `w(7,7)`, у которой элементы главной диагонали (`i = j`) равны 1, элементы выше главной диагонали равны сумме индексов (`i + j`), а элементы ниже ее — разности индексов (`i - j`).
6. Для формирования денежной суммы имеются купюры и монеты достоинством в 1000, 500, 100, 10, 5, 2 и 1 рубль. Разработать алгоритм и записать программу, которая по вводимой с клавиатуры сумме выдает наименьшее количество составляющих ее купюр и монет.

   Выполните анализ программы, приведенной в листинге 15.16. Объясните назначение и работу каждого оператора и каждой строки программы. Убедитесь в ее работоспособности. Организуйте вывод в виде суммы произведений с записью в одну строку. Усложните задачу добавлением номиналов копеек.

Листинг 15.16. Формирование сдачи

```
Dim a(8) As Integer
Private Sub Form_Load()
  Show
  a(1) = 1000: a(2) = 500: a(3) = 100: a(4) = 50
  a(5) = 10: a(6) = 5: a(7) = 2: a(8) = 1
  Sum = InputBox("Введите сумму")
  Print "Сумма "; Sum
  For i = 1 To 8
    k = Sum \ a(i)
    Sum = Sum Mod a(i)
    Print k; "*"; a(i)
  Next
End Sub
```

Глава 16

Строковые операторы и функции

16.1. Общие сведения. Операция конкатенация

Все системные и прикладные программы, с которыми приходится работать пользователю, способны выполнять с текстами множество различных операций. К ним относятся: поиск слов в тексте, сортировка слов по алфавиту, поиск синонимов, проверка правописания и многие другие операции, которые включены во многие программы, имеющие дело с текстовой информацией.

Строкой символов или просто строкой называют ограниченный двойными кавычками набор вводимых с клавиатуры или формируемых компьютером различных символов. Строка может быть пустой и между кавычек не иметь ни одного символа. В некоторых случаях строки символов допускают запись без ограничивающих кавычек.

С символьными переменными и константами можно выполнять операцию их объединения — конкатенацию. Знак операции конкатенация: + или &. Второй знак более предпочтителен. Следующая строка кода объединяет три слова в одно предложение:

```
a = "береги": b = "  честь  ": c = "смолоду": Print c + b + a, a & b & c
```

На экран монитора будет выведено: "смолоду честь береги" и "береги честь смолоду". Заметим, что слово "честь" в примере обрамлено лидирующим и замыкающим пробелами. Иначе слова в предложении были бы отпечатаны слитно, без разделения пробелом.

16.2. Кодирование символов и команд в компьютере

Все символы и подаваемые с клавиатуры компьютера команды кодируются в стандартном коде ASCII (American Standard Coding for Information

Interchange — Американский Стандартный Код для Обмена Информацией). Один байт (восьмибитовое двоичное слово) позволяет закодировать до 256 символов и команд. Заметим, что байт является наименьшей имеющей свой адрес единицей памяти компьютера. Новый международный стандарт Unicode отводит для кодирования символов и команд два байта — 16 бит. Это позволяет кодировать до 65 536 объектов.

Числами от 1 до 31 закодированы управляющие команды: например, числом 9 кодируется сигнал, подаваемый клавишей <Tab>, числом 27 — клавишей <Esc>, число 32 — код пробела.

Числа от 48 до 57 — коды цифр от 0 до 9.

Числа от 65 до 90 — коды прописных букв латинского алфавита (A — 65, B — 66, C — 67, D — 68 и т. д.).

Числа от 97 до 122 — коды строчных латинских букв.

Числа от 176 до 223 — коды символов псевдографики.

Числами от 32 до 63, от 58 до 63, от 91 до 95, от 123 до 127 и от 240 до 254 кодируются знаки препинания и другие символы.

При записи программ разрешается использовать буквы национальных алфавитов: числами от 128 до 159 кодируются прописные буквы русского алфавита; числами от 160 до 175 и от 224 до 239 — строчные буквы русского алфавита.

Перечислим с краткими примерами строковые операторы и функции Visual Basic 6.0. После этого некоторые функции рассмотрим более подробно с примерами их использования. Затем остановимся на задачах по преобразованию текстов.

16.3. Операторы и функции действий над строками

В VB-6 существуют следующие операторы и функции, реализующие действия над строками.

- **Asc** — возвращает ASCII-код первого символа строки.

  Пример:

  ```
  Asc("A") = 65;
  Asc("BIT") = 66.
  ```

- **Chr** — возвращает символ, соответствующий ASCII-коду.

  Пример:

  ```
  Chr(67) = C;
  Chr(68) = D.
  ```

- **Hex** — переводит число в шестнадцатеричную систему.

  Пример:

  ```
  Hex(17) = 11;
  Hex(30) = 1E.
  ```

- **Oct** — переводит число в восьмиричную систему.

  Пример:

  ```
  Oct(17) = 21;
  Oct(50) = 62.
  ```

- **Str** — переводит число в строку.

  Пример:

  ```
  Str(5) = "5";
  Str(25) = "25".
  ```

- **Val** — переводит строку в число.

  Пример:

  ```
  Val("15 bit") = 15;
  Val("ABC 5") = 0.
  ```

- **Lcase** — все буквы преобразует в строчные.

  Пример:

  ```
  Lcase("Бим-Бом") = "бим-бом".
  ```

- **Ucase** — все буквы преобразует в прописные.

  Пример:

  ```
  Ucase("Бим-Бом") = "БИМ-БОМ".
  ```

- **StrConv** — изменяет регистр букв строки.

  Пример:

  ```
  StrConv("Bit", 1) = "BIT";
  StrConv("Bit", 2) = "bit";
  StrConv("bit", 3) = "Bit".
  ```

- **Len** — возвращает длину строки.

  Пример:

  ```
  Len("дискета") = 7;
  Len ("экран") = 5.
  ```

- **Left** — возвращает подстроку с начала строки.

  Пример:

  ```
  Left("дискета", 4) = "диск".
  ```

- **Right** — возвращает подстроку с конца строки.

  Пример:

  ```
  Right("пароход", 3) = "ход".
  ```

- **Mid** — функция, возвращает часть строки.

  Пример:

  ```
  Mid("сокол", 2, 3) = "око";
  Mid("сокол", 4) = "ол".
  ```

- **Mid** — оператор, заменяет часть строки подстрокой.

  Пример:

  ```
  a = "класс"; b = "оло";
  Mid (a, 2, 3) = b; a = "колос".
  ```

- **InStr** — выполняет поиск в строке подстроки.

  Пример:

  ```
  a = "клавиатура", b = "тур";
  InStr (a, b) = 7;InStr (4, a, "a") = 6.
  ```

- **InStrRev** — выполняет поиск в строке подстроки, начиная с конца.

  Пример:

  ```
  a = "клавиатура", b = "тур";
  InStrRev (a, b) = 7.
  ```

- **Ltrim**, **Rtrim**, **Trim** — удаляют пробелы соответственно с начала, с конца и с обеих сторон строки.

- **StrReverse** — изменяет порядок следования символов в строке на обратный.

  Пример:

  ```
  StrReverse ("нофелет") = "телефон".
  ```

- **Space** — возвращает строку, состоящую из указанного числа пробелов.

  Пример:

  ```
  a = Space (3); a = "   ".
  ```

- **String** — возвращает строку, состоящую из указанного числа заданного символа.

  Пример:

  ```
  a = String (5, "&"); a = "&&&&&".
  ```

- **StrComp** — возвращает результат сравнения двух строк.

  Синтаксис:

  ```
  StrComp (<a1>, <a2>, <способ сравнения>)
  ```

  <способ сравнения>: 0 — двоичное сравнение, 1 — посимвольное без учета регистра.

  Результат: если a1 < a2, то −1, если a1 = a2, то 0, если a1 > a2, то 1.

  Примеры:

  ```
  StrComp ("бал", "бак", 1) = 1;
  StrComp("17", "32", 1) = -1.
  ```

- **Replace** — находит и заменяет в строке подстроку другой подстрокой.

  Синтаксис:

  ```
  Replace (a, b, c, n, m, p),
  ```

  где a — строка, в которой происходит замена; b — заменяемая подстрока строки a; c — заменяющая подстрока; n — позиция в строке a, начиная с которой ищется подстрока b; m — число строк, которые нужно заменить (если m опущен, то выполняются все замены); p — способ сравнения (0 — двоичное сравнение, 1 —посимвольное сравнение).

  Пример:

  ```
  Replace("asdaf", "a", "b", , , 1) = "bsdbf".
  ```

- **Split** — преобразует строку в одномерный массив.

  Синтаксис:

  ```
  Split (<строка>, <разделитель>, < элементов в массиве>, <критерий отбора>)
  ```

  <строка> — строковое выражение;

  <разделитель> — по умолчанию пробел;

  <число элементов> — если −1, то без ограничений;

  <критерий> — целое число.

  В листинге 16.1 приведен пример программы, использующей функцию Split.

Листинг 16.1. Пример использования функции Split

```
Private Sub Command1_Click()
  строка = "место встречи изменить нельзя"
  массив = Split(строка)
```

```
  For i = 0 To UBound(массив)  'функция Ubound возвращает
↳наибольшее допустимое значение индекса массива
    Print массив(i)    'вывод в столбец элементов массива
  Next
End Sub
```

- **Join** — преобразует массив в строку.

  Синтаксис:

  `Join (<массив>, <pразделитель>)`

  В листинге 16.1 приведен пример программы, использующей функцию `Join`.

Листинг 16.2. Пример использования функции Join

```
Private Sub Command1_Click()
  массив = Array("место", "встречи", "изменить", "нельзя")
  строка = Join(массив, ",  ")
  Print строка            'будет выведено "место, встречи, изменить,
нельзя"
End Sub
```

Синтаксис основных функций

Дадим описание синтаксиса основных функций и операторов VB-6, перечисленных выше.

- **Asc** — строковая функция.

  Переводит первый символ символьной строки в число, которым этот символ кодируется в коде ASCII.

  Синтаксис:

  `x = Asc (`*`строковое выражение`*`)`,

  где `x` — переменная целого типа.

  Под *строковым выражением* понимается или одиночный символ, или строка символов, или конкатенация символьных строк, или идентификатор символьной переменной.

  Примеры:

  `x = Asc("!")`, результат `x = 33`;

  `y = Asc("Фото")`, результат `y = 244`;

  `z = Asc(",")`, результат `z = 44`.

С помощью кода, привеенного в листинге 16.3, можно познакомиться с кодированием символов в компьютере. Достаточно после запуска проекта отвечать на приглашения функции `InputBox`.

Листинг 16.3. Задаем символ — получаем его ASCII-код

```
Dim a As String: Dim x As Integer
Private Sub Form_Load()
  Show
  Do
    a = InputBox("Введи символ", , , 4000, 3000)
    x = Asc(a): Print "Символ "; a; " кодируется числом "; x
  Loop
End Sub
```

Такой код будет бесконечно приглашать вводить символ. Для остановки выполнения программы следует нажать комбинацию клавиш <Ctrl>+ +<Break>.

- **`Chr`** — строковая функция.

  Переводит целое число в символ, который этим числом кодируется в компьютере.

  Синтаксис:

  `x = Chr(`*`целое выражение`*`)`,

  где `x` — переменная строкового типа,

  `целое выражение` — целое число или выражение, приводимое к целому типу.

  Примеры:

  `x = Chr(68)`, результат `x = "D"`;

  `x = Chr(33)`, результат `x = "!"`;

  `x = Chr (42)`, результат `x = "*"`.

  Функции `Asc` и `Chr` являются обратными друг другу и связаны соотношениями: `x = Asc(Chr(x))` и `x = Chr(Asc(x$))`. Код программы, приведенный в листинге 16.4, решает задачу, обратную той, которая решалась в предыдущем коде.

Листинг 16.4. Задаем ASCII-код — получаем символ

```
Dim a As String: Dim x As Integer
Private Sub Form_Load()
```

```
  Show
  Do
    x = InputBox("Введи целое от 32 до 255", , , 4000, 3000)
    a = Chr(x): Print " Числом "; x; " кодируется символ  "; a
  Loop
End Sub
```

❐ **Mid** — строковые оператор и функция.

- Функция `Mid` выделяет из заданной строки подстроку заданной длины, начиная с заданного символа.

  Синтаксис:

  `x$ = (`*`строка`*`, `*`начало`*` [, `*`длина`*`])`

  Параметр *`строка`* — заданная строка символов. Параметры *начало* и *`длина`* — целые числа, соответствующие номеру символа, с которого начинается выделение, и количеству выделяемых символов. Если параметр *`длина`* опущен, то выделение выполняется до конца заданной строки.

  Примеры:

  `a$="информатика", b$=Mid (a$,3,5)`, результат `b$="форма"`;

  `a$="паровоз", b$=Mid (a$,5)`, результат `b$="воз"`;

- Оператор `Mid` заменяет подстроку одной строки подстрокой другой строки.

  Синтаксис:

  `Mid(`*`строка 1`*`, `*`начало`*` [, `*`длина`*`]) = `*`строка 2`*

  Параметр *`строка 1`* — это символьная строка, подстрока которой должна быть заменена подстрокой *`строка 2`*. Параметры *`начало`* и *`длина`* — целые числа соответственно: номер символа в исходной строке, начиная с которого нужно произвести замену, и количество заменяемых символов.

  Пример программы с использованием оператора `Mid` приведен в листинге 16.5.

Листинг 16.5. Работа оператора `Mid`

```
Private Sub Form_Load()
  Show
  a = "Информатика": b = "дисплей": Print a, b
  Mid(a, 3, 4) = b: Print a
End Sub
```

Результатом работы программы, приведенной в листинге 16.5, будет вывод на форму слова "Индиспатика".

Рассмотрим еще два примера. Пусть `x = "класс"`, а `y = "золото"`. Выражение `Mid (x, 2) = Mid (y, 2, 3)` преобразует переменную `x = "класс"` в переменную `x = "колос"`. Если присвоить переменной а значение `"летят утки"`, то строка кода `Print Mid(a, 7) & " " & Mid(a, 1, 5)` выведет на форму строку "утки летят".

- **Str** — строковая функция.

  Превращает численные данные в символьную строку.

  Синтаксис:

  `x = STR(`*целое выражение*`)`,

  где `x` — переменная строкового типа.

  Параметр *выражение* — число или выражение целого типа. Если число положительное, то функция превращает его в символьную строку с пробелом в начале строки (лидирующим пробелом). Полезность этой функции заключается в возможности оперировать с цифрами числа как с символами. Их можно вырезать, вставлять, переставлять и т. д., пользуясь простыми строковыми операторами и функциями.

  Пример:

```
x = 1945   'исходное число
x = Str(x)  'переход от числа к строке x="  1945" (с лидирующим пробелом)
n = Len(x)                    'длина символьной строки
y = Mid(x, 3, 2)              'вырезка из строки двух символов
Print "n="; n, "y=";y          'с учетом лидирующего пробела n= 5, y=94
```

- **Val** — строковая функция

  Переводит символьную строку в число.

  Синтаксис:

  `x = Val(`*строка*`)`,

  где `x` — переменная целого типа.

  Параметр *строка* просматривается с начала до первого символа, не являющнгося цифрой. Функция опускает все лидирующие пробелы и возвращает в качестве результата число. Если первый символ строки не числовой, то результат функции равен нулю.

  Примеры:

  `x = Val("z12")`, результат `x=0`;

  `x = Val ("23z12")`, результат `x=23`.

Функция `Val` обратна функции `Str`. Можно записать следующие соотношения:

`x=Val(Str(x))` и `x=Str(Val(x))`.

- **`InStr`, `InStrRev`** — строковые функции.

  Начиная с заданной позиции, выполняют поиск первого включения одного текста в другой текст и возвращают номер позиции начала включения. Функции различаются тем, что первая проводит поиск с начала строки, вторая — с конца.

  Синтаксис:

  `x = InStr([`*`начало`*`,] `*`строка 1`*`, `*`строка 2`*`)`,

  где `x` — переменная целого типа.

  Параметр *`начало`* имеет целый тип и вводится при необходимости задать номер позиции, с которой требуется начать поиск вхождения текста *`строка 2`* в текст *`строка 1`*. Он может принимать значения от 1 до 32 767. Функция возвращает значение 0, если текст *`строка 1`* пуст или текст *`строка 2`* в тексте *`строка 1`* не найден.

  Примеры:

  `x = InStr ("практика", "акт")`, результат `x=3`;

  `x = InStr (4, "практика", "акт")`, результат `x=0`;

  `x = InStr ("практика", "к")`, результат `x=4`;

  `x = InStr (5, "практика", "к")`, результат `x=7`.

  Замечание

  Последовательно увеличивающиеся коды ASCII букв алфавитов дают возможность выполнять сравнение букв по величине их порядкового номера в алфавите. Это позволяет применять сравнение строк при записи условий. Например, истинны выражения: "д"<"ж", "аб"<"абв", "z"<"ж" и т. д.

16.4. Решение типовых задач

Пример 1. Количество заданных символов в заданном тексте

Определить. Алгоритм решения задачи простой. Он совпадает с алгоритмом решения задачи вручную, который сводится к следующему. Просматриваем заданный текст посимвольно с начала до конца и с появлением заданного символа добавляем в "счетчик" единицу (ставим штрих на листке бумаги). В итоге получаем искомую величину — количество заданных символов.

Листинг 16.6. Количество заданных символов в тексте

```
Dim a As String: Dim b As String: Dim k As Integer
Private Sub Form_Load()
  Show
  a = InputBox("Введите текст")
  b = InputBox("Введите символ")
  For i = 1 To Len(a)
    If Mid(a, i, 1) = b Then k = k + 1
  Next
  Print "k = "; k
End Sub
```

Пример 2. Проверить возможность записи текста *a* символами текста *b*

Рассмотрим модель задачи. Возьмем два слова: `a = "форточка"` и `b = "информатика"`.

Алгоритм решения может быть следующим. Берем одну за другой буквы слова "форточка" и проверяем их наличие в слове "информатика". Если все буквы есть, то записать слово "форточка" буквами слова "информатика" можно, иначе — нельзя. В нашем примере ответ отрицательный, т. к. в слове "информатика" нет буквы "ч".

Ниже приведен код программы, реализующий рассмотренный алгоритм. Он интересен тем, что в нем используется принцип "флага", который "поднимается" (`f = 1`), если в теле цикла произошло нужное событие — в тексте `b` имеется буква текста `a`. Если флаг остается опущенным, то нужной буквы в тексте `b` нет и записать текст `a` буквами текста `b` нельзя.

Листинг 16.7. Записать текст символами другого текста

```
Dim a As String: Dim b As String: Dim f As Byte
Private Sub Form_Load()
  Show
  a = InputBox("Введи текст 1")
  b = InputBox("Введи текст 2")
  For i = 1 To Len(a)
    f = 0
    For j = 1 To Len(b)
      If Mid(a, i, 1) = Mid(b, j, 1) Then f = 1          'f - флаг
```

```
    Next j
    If f = 0 Then
      Print "Текст 1 нельзя записать символами текста 2"
      Exit For
    End If
  Next i
  If f = 1 Then
    Print "Текст 1 можно записать символами текста 2"
  End If
End Sub
```

16.5. Строковые функции и создание макросов

Макросы в приложениях Microsoft Office создаются на языке VBA (Visual Basic Application), набор функций которого в основном совпадает с набором функций Visual Basic 6.0. Поэтому при построении многих макросов необходимы знания строковых операций и функций.

Создание макросов покажем на примере простого макроса в текстовом редакторе Word. Пусть требуется определить количество пробелов в выделенном текстовом документе. Проверить работу макроса можно будет, сравнив результат с данными, которые можно получить в редакторе Word командой **Сервис | Статистика**.

Алгоритм создания макроса *Число_пробелов*

Подадим команду **Сервис | Макрос | Макросы**. Открывается диалоговое окно **Макрос**.

В текстовое поле **Имя** впишем имя макроса, задаваемое по всем правилам Visual Basic — `Число_пробелов`, и нажмем кнопку **Создать**.

Открывается окно **Microsoft Visual Basic — Normal** с документом **Normal — New Macros (Code)**. В модуле подготовлено место для записи кода процедуры нового макроса `Число_пробелов`.

Вписываем код макроса (листинг 16.8).

Листинг 16.8. Код макроса `Число_пробелов`

```
Sub Число_пробелов()
'оставляем две строки сообщения об имени,
'сроке создания и авторе макроса
```

```
  z$ = Selection.Text
  y$ = ""
  For i = 1 To Len(z$)
    d$ = Mid(z$, i, 1)
    y$ = y$ & d$
    If Asc(d$) = 32 Then k% = k% + 1
  Next
  Selection.Text = y$ & Str(k%)
End Sub
```

Закрываем окно с кодом созданного макроса.

Подаем команду **Сервис | Настройка**. В открывшемся окне **Настройка** выбираем вкладку **Команды**. В списке **Категории** выбираем строку **Макросы**.

В правом списке **Команды** находим имя макроса `Число_пробелов` и перетягиваем его мышью на панель инструментов редактора Word.

Не закрывая окно **Настройка**, щелчком правой кнопкой мыши на кнопке макроса на панели инструментов открываем окно со средствами дизайна кнопки. В нем выбираем, например, сначала строку **Основной стиль**, а затем строку **Выбрать значок для кнопки**, после выбора которой появляется палитра значков. Выбираем в ней щелчком внешний вид кнопки.

Проверяем правильность работы макроса. Выделяем фрагмент текста и нажимаем на кнопку макроса на панели инструментов. Сравниваем отпечатанное после выделенного фрагмента число пробелов с фактическим количеством.

Макрос *Один_пробел*

Создадим макрос для текстового редактора Word, оставляющий в выделенном тексте между словами только по одному пробелу. Необходимость в этом возникает при подготовке рукописи к публикации. Именно по этой причине рекомендуется набор документов выполнять с включенной на панели инструментов кнопкой **Непечатаемые знаки**, разделяющей слова приподнятыми точками.

Пододим команду **Сервис | Макрос | Макросы**. Открывается диалоговое окно **Макросы**. В текстовое поле **Имя** впишем имя макроса `Один_пробел` и нажмем кнопку **Создать**.

Открывается окно **Microsoft Visual Basic – Normal** с документом **Normal – New Macros (Code)**. Это наглядный пример MDI-приложения: родительская форма с дочерней, в качестве которой выступает модуль с кодами процедур макросов. В модуле подготовлено место для записи кода процедуры нового макроса `Один_пробел`.

Вписываем код макроса. Он приведен в листинге 16.9.

Листинг 16.9. Код макроса `Один_пробел`

```
Sub Один_пробел()
'оставляем две строки сообщения об имени,
'сроке создания и авторе макроса
  x = Selection.Text
  z = ""
  For I = 1 To Len (x)
    y = Mid (x, I, 1)
    If Asc (y) < > 32 Then
      f = 0
      z = z & y
    ElseIf f = 0 Then
      f = 1
      z = z & y
    End If
  Next
  Selection. Text = z
End Sub
```

Закрываем окно с кодом созданного макроса и выполняем действия, рассмотренные в предыдущем примере.

Макрос *Латиница_Кириллица*

Обычно макросы создаются для решения встречающихся при работе с приложениями вопросов, которые не решаются средствами приложения. Так и с макросом `Латиница_Кириллица`, который помогает пользователям экономить время, затрачиваемое на переписывание текста, случайно введенного не в том регистре клавиатуры [20].

Листинг 16.10. Код макроса `Латиница_Кириллица`

```
Sub Латиница_Кириллица()
' Латиница_Кириллица Макрос создан 30.01.2004
  Lat$   =   "qwertyuiop[]asdfghjkl;'zxcvbnm,./QWERTYUIOP[]ASDFGHJKL:"   &
Chr$(34) & "ZXCVBNM<>"
Cyr$="йцукенгшщзхъфывапролджэячсмитбю.ЙЦУКЕНГШЩЗХЪФЫВАПРОЛДЖЭЯЧСМИТЬБЮ"
  nCyr = Len(Cyr$)
```

```
  sel$ = Selection.Text
  Change$ = ""
  For i = 1 To Len(sel$)
    z = Asc(Mid(sel$, i, 1))
    Flag = 0
    For j = 1 To nCyr
      c1 = Asc(Mid(Lat$, j, 1))
      c2 = Asc(Mid(Cyr$, j, 1))
      If c1 = z Then z = c2: Exit For
      If c2 = z Then z = c1: Exit For
    Next j
    Change$ = Change$ & Chr$(z)
  Next i
  Selection.Text = Change$
End Sub
```

Задания для самостоятельной работы

1. Задан текст: a = "В 1703 году основан Санкт-Петербург". Определить значения:

   ```
   n = Len (a)
   b = Mid (a, 21)
   c = Left (a,11)
   d = Right (a, 9)
   e = Upcase (Mid (a, 21))
   f = Len (Mid (a, 8, 19)
   g = Mid (a, 21)+Mid (a, 13, 7)+Lef t(a, 11)
   h = String (Mid (a, 4, 1))
   k = InStr (a, "год")
   s = Space (InStr (14 ,a, "о")).
   ```

2. Из символов одного текста с помощью строковых функций и операции конкатенация получить другой текст.

   Из "информатика" получить "форма" и "рифма".

   Из "индивидуальность" получить "диво" и "лавина".

   Из "алгоритм" получить "гора" и "литр".

   Из "структура" получить "рак" и "трут".

   Из "монография" получить "мафия" и "графин".

3. Преобразовать тексты:

   Из текста "береги честь смолоду" получить текст "смолоду честь береги".

   Из текста "каков поп, таков и приход" получить текст "каков приход, таков и поп".

   Из текста "у семи нянек дитя без глаза" получить текст "дитя без глаза у семи нянек".

   Из текста "один с сошкой, а семеро с ложкой" получить текст "семеро с сошкой, а один с ложкой".

   Из текста "мал золотник, да дорог" получить текст "дорог золотник, да мал".

4. Отпечатать заданный текст с заменой одного из заданных символов другим символом.

5. Определить в заданном тексте количество пар одинаковых рядом стоящих символов: нн, сс и т. п.

6. Выполните анализ следующего фрагмента кода.

```
z = InputBox ("Введите шестизначное натуральное число")
Print Val (Mid (Str (z), 5)+Mid (Str (z), 2, 3))
```

   Определите его назначение. Решая задачу, следует помнить, что функция `Str` добавляет к цифрам числа лидирующий пробел.

7. Выполните анализ кода, приведенного в листинге 16.11. Определите, что выводится в последней строке процедуры.

Листинг 16.11. Задание на анализ кода программы

```
Dim x As Long: Dim z As String: Dim y As String
Private Sub Form_Load()
  Show
  x = InputBox("Введите целое число")
  z = "0"
  For i = 2 To Len(Str(x))
    y = Mid(Str(x), i, 1)
    If y > z Then z = y
  Next
  Print "z = "; z
End Sub
```

Глава 17

Графика и Visual Basic

По умолчанию для задания координат и размеров объектов принимается аппаратно-независимая единица измерения *твип*. При создании проектов имеется возможность принять для использования другие единицы: point (пункт, в дюйме 72 пункта), pixel (пиксел), character (символ), дюйм (в дюйме 1440 твипов, как минут в сутках), миллиметр, сантиметр (в сантиметре 567 твипов) и единицу измерения, задаваемую пользователем.

Изменить единицы измерения можно в списке значений свойства `ScaleMode` формы в окне **Свойства**. Заметим, что все примеры в учебнике используют предлагаемую по умолчанию единицу — твип.

Если кнопкой **Развернуть** придать форме максимальные размеры, то в правой части панели инструментов окна Visual Basic можно увидеть, что эти размеры по координатам *x* и *y* равны 12 000×9000 твипов.

17.1. Графические методы

Графические методы Visual Basic 6.0 включают методы: `Cls` — очистка, `Pset` — рисует точку, `Line` — рисует линии и прямоугольники, `Circle` — рисует окружность, `Point` — возвращает цвет точки, `Print` — выводит текст, `Show` — обеспечивает видимость текста и графики, `PrintPicture` — выводит рисунок.

В общем виде синтаксис метода предусматривает необязательное во многих случаях указание на то, к какому объекту метод применяется, и имеет вид:

`[`*`объект`*`.] `*`метод`*

Если параметр *`объект`* не указывается, то по умолчанию в качестве объекта принимается форма. Для упрощения записей параметр *`объект`* будем чаще всего опускать.

Графические методы связаны с такими свойствами формы и других элементов управления, как `DrawStyle` и `DrawWidth`. Свойство `DrawStyle` определяет стиль линий фигур: сплошные, невидимые, штриховые, пунктирные и т. п.

Свойство `DrawWidth` определяет ширину линий. О свойствах, связанных с цветом, рассказывается в следующем разделе.

О графических методах `Show` и `Print` было рассказано в предыдущих главах. Рассмотрим оставшиеся методы и примеры их использования.

- **`Cls`** — очищает поверхность объектов (формы и элементов управления) от графики и текста.

  Синтаксис:

  ```
  Cls
  ```

- **`Pset`** — рисует точку на экране монитора.

  Синтаксис:

  ```
  Pset [Step] (x, y) [, цвет]
  ```

  Параметры *x* и *y* — координаты точки. Параметр *цвет* — цвет точки. Ключевое слово `Step` указывает на то, что координаты x и y являются приращениями к координатам текущей точки — последней точки, установленной графическими методами при выполнении кода программы.

- **`Line`** — графический метод. Строит линии и прямоугольники.

  Синтаксис:

  ```
  Line [[Step] (x1, y1)] - [Step] (x2, y2) [, [цвет][, B[F]]]
  ```

  Параметры `(x1, y1)` и `(x2, y2)` — координаты концов отрезка прямой линии или координаты противоположных углов прямоугольника. Параметр *цвет* указывает на цвет рисованной фигуры.

  Параметр-константа `B` (от англ. *box* — прямоугольник) устанавливает режим рисования прямоугольника. Параметр-константа `BF` (*filled box* — закрашенный прямоугольник) обеспечивает закраску прямоугольника внутри.

Пример 1. Метод `Line`

Пример программы с использованием метода `Line` приведен в листинге 17.1.

Листинг 17.1. Пример использования метода `Line`

```
Private Sub Form_Load()
  Show
  DrawWidth = 2
1   Line (500, 1500)-(1000, 500)
2   Line -(1500, 1500)
3   Line Step(-1000, 500)-Step(1000, 1000)
```

```
4   Line (2500, 500)-(4000, 1500), , B
5   Line (2500, 2000)-(4000, 3000), , BF
  CurrentX = 440: CurrentY = 700: Print 1
  CurrentX = 1250: CurrentY = 700: Print 2
  CurrentX = 800: CurrentY = 2000: Print 3
  CurrentX = 2100: CurrentY = 700: Print 4
  CurrentX = 2100: CurrentY = 2000: Print 5
End Sub
```

Приведенная программа рисует в форме показанные на рис. 17.1 пять фигур. Номера фигур соответствуют номерам строк кода. Отрезок 2 в качестве первой точки использует вторую точку отрезка 1. Координаты крайних точек отрезка 3 формируются как приращения к последней точке отрезка 2 и к первой своей точке. Абсолютные координаты отрезка 3 можно записать так:

`3 Line (1500 - 1000, 1500 + 500) - Step (1000, 1000)`, или короче:

```
  Line (500, 2000) - (1500, 3000).
```

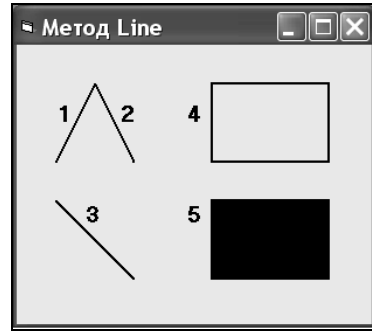


Рис. 17.1. Рисунки, полученные с помощью графического метода `Line`

- **`CIRCLE`** — рисует окружности, эллипсы и дуги.

  Синтаксис:

  `Circle [Step](`*`x`*`, `*`y`*`), `*`радиус`*` [,[`*`цвет`*`] [, [`*`начало дуги`*`], [`*`конец дуги`*`] [`*`отношение`*`]]`

  Параметры *`x`* и *`y`* — координаты центра фигуры. Параметр *`радиус`* — радиус окружности. Параметр *`цвет`* — цвет фигуры. Параметры *`начало дуги`* и *`конец дуги`* указывают на начальный и конечный углы рисования дуги в радианах. Параметром *`отношение`* задается отношение длин полуосей эллипса. Служебное слово `Step` служит указателем на относительность ко-

ординат центра окружности, которые являются в этом случае приращениями к координатам текущей точки.

Пример 2. Метод `Circle`

Пример программы с использованием метода `Circle` приведен в листинге 17.2.

Листинг 17.2. Рисунки графическим методом `Circle`

```
Private Sub Form_Load()
  Show
  DrawWidth = 2
1 Circle (1000, 1000), 500
2 Circle Step(0, 1500), 750
3 Circle (2500, 1000), 500, , , , 0.3
4 Circle (2500, 2500), 750, , , , 3
5 Circle (4000, 2000), 1000, , 0.8, 2.4
6 Circle (4000, 2000), 1000, , -4, -5.5
  CurrentX = 300: CurrentY = 300: Print 1
  CurrentX = 300: CurrentY = 1500: Print 2
  CurrentX = 1900: CurrentY = 300: Print 3
  CurrentX = 1900: CurrentY = 1500: Print 4
  CurrentX = 3500: CurrentY = 300: Print 5
  CurrentX = 3500: CurrentY = 1500: Print 6
End Sub
```

Результат работы программы показан на рис. 17.2. Номера фигур на рисунке соответствуют номерам строк кода.

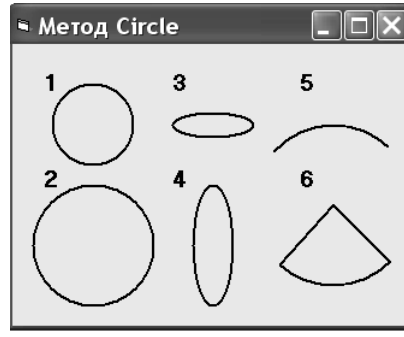


Рис. 17.2. Графический метод `Circle`

Координаты центра окружности в строке 2 кода программы относительны к координатам центра в строке 1. Это приращения к координатам центра в строке 1. Операторы в строках 3 и 4 рисуют эллипсы. Если отношение полуосей эллипса меньше единицы (строка 3), то радиус задается в точках экрана по оси *x*, иначе — по оси *y* (строка 4). Начало и конец дуги отсчитываются в радианах против часовой стрелки (строка 5). Если перед их значениями поставить знак минус, то от концов дуги к центру будут проведены радиусы (строка 6).

- **`Point`** — возвращает цвет точки с заданными координатами.

  Синтаксис:

  ```
  Point (x, y),
  ```

  где параметры `x`, `y` — координаты точки.

17.2. Цвет в Visual Basic 6.0

Значение цвета рисуемой фигуры при записи кода программы можно задавать различными способами.

- Простейший способ задания — это использование констант. Соответствие констант восьми основным цветам дано в табл. 17.1. Красную точку на форме поставит такая запись:

  ```
  Pset (100, 100), vbRed.
  ```

- Более шестнадцати миллионов цветов и оттенков можно получить, используя функцию **`RGB`**.

  Ее синтаксис:

  ```
  RGB(Red, Green, Blue),
  ```

  где каждый аргумент функции определяет интенсивность основного цвета и может принимать значения от 0 до 255. Для получения восьми основных цветов аргументы функции `RGB` должны принимать значения, указанные в табл. 17.1.

- Использование функции **`Color`** знакомо работавшим с графикой языков QBasic и Quick Basic. Синтаксис этой функции:

  ```
  QBColor (color),
  ```

  где *`color`* — целое число из диапазона от 0 до 15. Названия и значения цветов приведены в табл. 17.2. Квадрат светло-красного цвета создаст на форме такая запись:

  ```
  Line (100, 100) - (500,500), QBColor (12).
  ```

- Можно использовать шестнадцатеричное представление кода цвета. Такую запись можно наблюдать в окне **Свойства** в правой колонке строки свойств, определяющих цвет. Запись обрамлена лидирующим и замы-

кающим символом амперсенда (&) и начинается с буквы H. Значения основных цветов приведены в табл. 17.1. Значения цветов палитры можно выписать из окна **Свойства**.

Таблица 17.1. *Задание цвета*

Цвет	Константа	Значение в шестандцатеричной системе	Функция RGB		
			R	G	B
Черный	VbBlack	&H0	0	0	0
Красный	VbRed	&HFF	255	0	0
Зеленый	VbGreen	&HFF00	0	255	0
Желтый	VbYellow	&HFFFF	255	255	0
Синий	VbBlue	&HFF0000	0	0	255
Розовый	VbMagenta	&HFF00FF	255	0	255
Голубой	VbCyan	&HFFFF00	0	255	255
Белый	VbWhite	&HFFFFFF	255	255	255

Таблица 17.2. *Значения параметра функции* QBColor

Цвет	Значение	Цвет	Значение
Черный	0	Серый	8
Синий	1	Светло-синий	9
Зеленый	2	Светло-зеленый	10
Голубой	3	Светло-голубой	11
Красный	4	Светло-красный	12
Розовый	5	Светло-розовый	13
Желтый	6	Светло-желтый	14
Белый	7	Насыщенный белый	15

Пример 3. Способы задания цвета

В листинге 17.3 приведен код приложения, в котором при загрузке формы с помощью графического метода Line на форме отображаются четыре закрашенных разными способами квадрата.

Листинг 17.3. Способы закрашивания фигур

```
Private Sub Form_Load()
  Show
  Line (500,500)-(1000,1000), vbBlue, BF
  Line (1500, 500)-(2000, 1000), RGB(255, 255, 255), BF
  Line (2500, 500)-(3000, 1000), QBColor(13), BF
  Line (3500,500)-(4000, 1000), &HFC00&, BF
End Sub
```

В первом квадрате для закраски квадрата использована константа из табл. 17.1.

Второй квадрат закрашен с помощью функции `RGB`. Изменяя значения параметров функции `RGB` (`Red`, `Green`, `Blue`), можно получать квадраты разного цвета (в программе он белый).

Третий квадрат закрашен с помощью функции `QBColor` (номер цвета). Номера цветов приведены в табл. 17.2.

В четвертом квадрате использован способ непосредственной установки цвета с помощью шестнадцатеричного номера цвета. Квадрат закрашен светло-зеленым цветом с номером &HFC00& (цвет взят из графического редактора Paint, его десятичный номер — 64 512 — был получен с помощью функции `Point`).

Свойства *BackColor*, *FillColor* и *ForeColor*

В окне **Свойства** в списке свойств формы имеются три свойства, определяющие цвет: `BackColor` — определяет цвет формы, `FillColor` — определяет цвет внутренней заливки фигур, созданных графическими методами, и `ForeColor` — задает цвет линий и контуров фигур.

Пример 4. Определяющие цвет свойства объектов

В листинге 17.4 показан код программы, рисующей на желтой форме синюю окружность, залитую красным цветом. Синий цвет (`vbBlue`), заданный вместе с методом `Circle`, победил зеленый цвет (`vbGreen`), заданный в коде свойством `ForeColor`. В свою очередь, заданный в коде зеленый цвет побеждает цвет, который может быть задан свойством `ForeColor` в окне свойств.

Листинг 17.4. Свойства объектов и цвет

```
Private Sub Form_Load()
  Show
```

```
  DrawWidth = 5
  BackColor = vbBlue                'синий цвет формы
  FillStyle = 0                     'заливка круга
  FillColor = vbRed                 'красный цвет заливки
  Circle (1000, 1000), 500, vbCyan  'окружность голубого цвета
  ForeColor = QBColor(15)           'белый цвет надписи
  FontSize = 12: Print Spc(5); "Задание цвета"
End Sub
```

Пример 5. Демонстрация палитры из 16 цветов

В листинге 17.5 приведен код программы, которая рисует в форме 16 кругов, закрашенных разным цветом, задаваемым функцией `QBColor (`*`номер цвета`*`)`.

Листинг 17.5. Цветные круги

```
Dim x As Integer: Dim y As Integer: Dim c As Integer
Private Sub Form_Load()
  Show
  WindowState = 2
          'форма развернута
  For x = 1000 To 8000 Step 2000    'центр окружности по горизонтали
    For y = 1000 To 6000 Step 1500    'центр окружности по вертикали
      FillColor = QBColor(c): FillStyle = 0       'цвет и стиль заливки
      Circle (x, y), 400, QBColor(c)       'вывод в форму окружности
      Print String(12, 32); c       'выбор места и вывод номера цвета
      c = c + 1
          'перебор номеров цвета от 0 до 15
  Next y, x
End Sub
```

Пример 6. Коврик

Программа, код которой приведен в листинге 17.6, кроме рисунка, демонстрирует принцип событийного управления. Используется событийная процедура, запускаемая щелчками мыши на форме (`Form_Click`). В качестве упражнения можно закрасить форму и окружности орнамента.

Листинг 17.6. Рисунок коврика с управляемым орнаментом

```
Dim r As Integer: Dim x As Integer: Dim y As Integer
Private Sub Form_Click()
  Static r
```

```
  Form1.WindowState = 2
  Cls
  r = ((r + 20) Mod 600)
  For x = 1000 To 10000 Step 400
    For y = 1000 To 7000 Step 400
    Circle (x, y), r
  Next y, x
End Sub
```

17.3. Построение графиков функций

Задача построения графиков функций ставится при исследовании функций и демонстрации их поведения. Построение графиков тесно связано с задачей табуляции функций. Построение графика функции $y = f(x)$ сводится к вычислению значения функции y при различных значениях аргумента x, после чего на экране ставятся точки с координатами (x, y). Множество точек, получаемых, например, с помощью оператора `PSET(x, y)`, образуют график функции.

Итак, имеем:

```
Pset (x, y)
```

При построении графиков приходится осуществлять выбор параметров. Рассмотрим их.

Выбор начала координат

График функции может находиться в одном из квадрантов декартовой системы координат или в их совокупности. Если заранее, хотя бы ориентировочно, известно поведение функции, то выбор начала координат не вызывает затруднений. Если же поведение функции неизвестно, то вначале естественно выбрать начало координат в центре экрана, а затем, увидев график функции, выполнить коррекцию.

Смещение начала координат по координате x обозначим величиной `A`, по координате y — величиной `B` (рис. 17.3). В результате получаем оператор:

```
Pset (x+A, y+B).
```

Изменение масштаба отображения графика функции

Рассмотрим такой пример. Пусть требуется построить график функции $y = \sin(x)$ на интервале изменения аргумента x от $-6{,}28$ до $6{,}28$. Без измене-

ния масштаба длина графика (по оси *x*) будет равна 12 твипам, а высота графика над осью *x* будет равна одному твипу (максимальное значение функции sin(*x*)). Такой график на экране трудно заметить. Для расширения или сужения графика по координате *x* и увеличения или уменьшения его амплитуды по координате *y* введем масштабные множители `C` и `D` (см. рис. 17.3).

В итоге имеем выражение:

```
Pset (x*C+A, y*D+B).
```

Пример 7. График функции *y* = Sin(*x*)

Код программы построения графика функции *y* = Sin(*x*) приведен в листинге 17.7.

Листинг 17.7. График функции `Sin(x)`

```
Private Sub Form_Load()
  Show
  WindowState = 2
  Drawwidth = 2
  Line (2500, 3000)-(9500, 3000)
  Print "  X"
  Line (6000, 5000)-(6000, 800)
  Print "  Y"
  For x = -6.28 To 6.28 Step 0.05
    Pset (x * 500 + 6000, -Sin(x) * 1500 + 3000)
  Next
End Sub
```

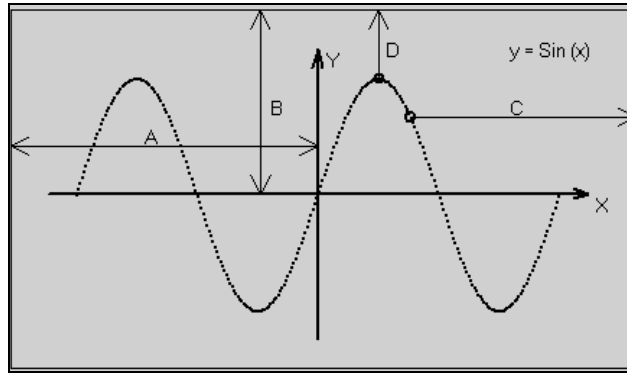


Рис. 17.3. Смещения (`A` и `B`) и масштабные коэффициенты (`C` и `D`) для графика функции `y = Sin (x)`

Выбор способа отображения и шага изменения аргумента

Наглядность графика зависит от шага изменения аргумента `x`, что влияет на плотность точек на графике. Шаг изменения аргумента можно подобрать опытным путем. Кроме оператора `Pset`, можно использовать оператор `Line - (x, y)`. В этом случае график выглядит в виде ломаной линии.

Оформление графика функции

На экране желательно показать координатные оси с обозначениями. Хорошо смотрится и помогает в анализе функции координатная сетка, нарисованная одним из неярких цветов. График функции также можно закрасить, но более ярким цветом. В правом верхнем углу экрана печатается формула функции. Там же можно отпечатать диапазон изменения аргумента.

Обеспечение работоспособности программы

При написании программ построения графиков функций следует пользоваться теми же приемами анализа функций на особые точки, как и при табуляции функций.

Задания для самостоятельной работы

1. Создайте приложение, демонстрирующее график функции $y = \sin(x)$ (см. листинг 17.7). Замените функцию $y = \sin(x)$ на $y = \cos(x)$, $y = \tan(x)$, $y = 1/x$.
2. Строку листинга 17.7 `Pset (x * 500 + 6000, -Sin(x) * 1500 + 3000)` замените на `Pset Sin(x) * 1000 + 6000, -Cos(x) * 1000 + 3000)`. Такая строка нарисует круг. Умножьте аргумент функции `Sin(x)` на 2, т. е. запишите: `Sin(2*x)`. График при этом выглядит в виде цифры 8. Если умножать аргументы обеих функций на целые числа (2 и 3, 3 и 5 и т. д.), то будете получать так называемые фигуры Лиссажу. При этом желательно уменьшить шаг изменения аргумента `x` с `0.05` до `0.005`.
3. На развернутую форму выведите 4 комплекта осей координат. Это можно сделать, например, с помощью кода, приведенного в листинге 17.8.

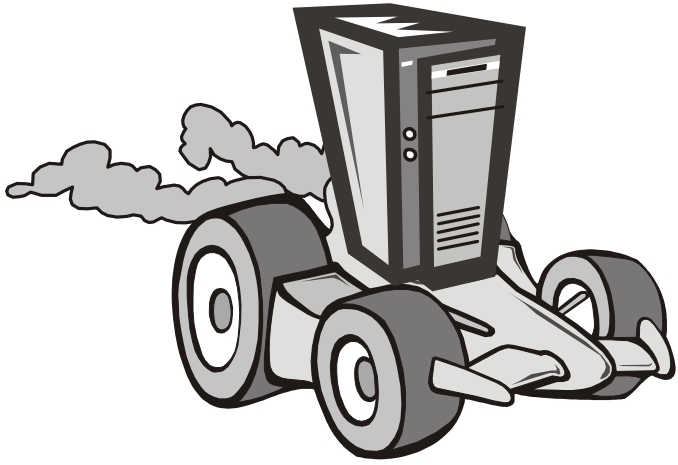
Листинг 17.8. Оси координат

```
Private Sub Form_Load()
  Show
  WindowState = 2
```

```
  DrawWidth = 2
  For x = 3000 To 9000 Step 6000
    For y = 2000 To 6000 Step 4000
      Line (x, y + 1500)-(x, y - 1500): Print "  Y"
      Line (x - 2500, y)-(x + 2500, y): Print "  X"
  Next y, x
End Sub
```

Разместите на осях координат графики различных функций.

4. Создайте в форме три или более осей координаты `x`. Разместите на них графики функций (сверху вниз) `Sin(x)`, `Cos(x)` и `Sin(x)*Cos(x)`. Убедитесь в наглядности демонстрации графиков функций двойного, половинного угла, других функций.



ЧАСТЬ II

Дополнительный материал для углубленного изучения

Глава 18

Управляющие элементы (элементы управления)

Под управляющими элементами (элементами управления) в Visual Basic понимаются объекты, обеспечивающие комфортную работу пользователя с создаваемыми приложениями. К ним относятся кнопки, текстовые и графические поля, списки, флажки, полосы прокрутки и многие другие элементы, которые мы видим в окнах системы Windows и работающих в ней приложений.

При знакомстве с элементами управления будем создавать приложения с демонстрациями их свойств, методов и событий. Желательно папки всех приложений записывать в отдельную папку, назвав ее, например, Управляющие элементы.

18.1. Общие сведения об элементах управления

Панель встроенных элементов управления

Окно панели встроенных управляющих элементов обычно размещается слева от рабочего поля конструктора форм. На рис. 18.1 показана панель и приведены названия расположенных на ней управляющих элементов.

Перемещение элементов управления в форму

В режиме конструирования приложения решается задача размещения в формах элементов управления. Перенести элемент управления на форму можно различными способами.

Во-первых, при двойном щелчке на кнопке выбранного элемента на панели он появляется в середине формы. После этого можно менять его размеры и перемещать по форме. Это самый простой и удобный способ размещения элементов.

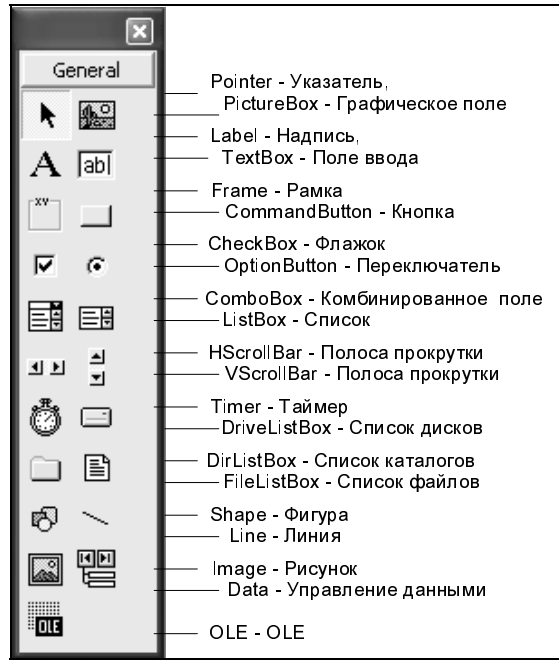


Рис. 18.1. Панель элементов управления

Во-вторых, можно подвести указатель мыши на кнопку выбранного элемента и при нажатой кнопке перетащить указатель на поверхность формы. При этом значок указателя изменится, а при отпускании кнопки мыши он превратится в крестик. Далее элемент "рисуется" на форме перетаскиванием указателя мыши.

Форма имеет три графических слоя: верхний, средний и нижний. В верхнем слое размещаются все неграфические элементы, кроме надписей (например, кнопки, рамки). Средний слой — это все графические элементы и надписи. В нижнем слое размещаются результаты применения методов (`Line`, `Circle` и т. п.). Видимость элементов в одном слое зависит также от последовательности во времени их размещения в форме. Перекрытие элементов в одном слое или разных слоях можно проследить, размещая их в форме с частичным совмещением площадей.

Имена элементов управления

По умолчанию элементы управления имеют различные имена. Например, при выборе двойным щелчком подряд пяти кнопок они будут штабелем лежать в середине формы. Сверху будет находиться кнопка `Command 5`, ниже — `Command 4` и т. д. до кнопки `Command 1`. Из штабеля их можно растащить по

своим местам. Заметим, что последовательность выбора элементов влияет на особенность работы с ними, например на последовательность выделения клавишей <Tab> и значение свойства `TabIndex`.

В процессе переименования элементам управления также присваиваются, как правило, разные имена. Желательно, чтобы имена имели содержательный смысл, связанный с их назначением и особенностями работы.

Имена элементов полезно начинать с префикса: формы (`Form`) — `frm`, кнопки (`CommandButton`) — `cmd`, метки (`Label`) — `lbl` и т. п. В сложных приложениях префиксы помогают лучше ориентироваться среди большого количества различного рода элементов. Префиксы сведены в табл. 18.1.

Таблица 18.1. *Префиксы имен управляющих элементов*

Управляющий элемент	Префикс	Управляющий элемент	Префикс
CheckBox	chk	Image	img
ComboBox	cbo	Label	lbl
CommandButton	cmd	Line	lin
Data	dat	ListBox	lst
DirListBox	dir	OptionButton	opt
DriveListBox	drv	PictureBox	pic
FileListBox	fil	Shape	shp
Form	frm	TextBox	txt
Frame	fra	Timer	tmr
HScrollBar	hsb	VScrollBar	vsb

Упражнение

Создайте новый проект. Разместите в форме различные элементы управления. Перемещайте их по форме. Изменяйте их размеры. Выделяйте и удаляйте.

Напомним, что выделенный объект обрамляется маркерами изменения размеров. Последовательное выделение объектов выполняется клавишей <Tab>. Все объекты на форме можно выделить командой **Правка | Выбрать все**.

Установите указатель мыши на каком-нибудь элементе управления и нажмите ее левую кнопку. Через секунду рядом с указателем появится желтое поле с координатами *X* и *Y* левого верхнего угла элемента. Если начать перемещать элемент управления по форме, то соответственно будут изменяться и его координаты. Значения этих координат и размеров выделенного

элемента управления можно увидеть в правой части панели инструментов главного окна.

18.2. Кнопка *(Command Button)*

Кнопка — это один из простейших и часто используемых элементов управления. Ее можно увидеть почти в каждом окне Windows-приложений.

Из свойств, с которыми начнем встречаться в ближайших приложениях, выделим следующие:

- `Name` — имя кнопки, будем записывать с префиксом `cmd`, например, `cmdВыход`;
- `Caption` — текст на кнопке, ее название;
- `Enabled` — доступность кнопки, готовность ее к нажатию;
- `Visible` — видимость кнопки.

Из событий чаще всего используется событие `Click` — щелчок на кнопке мышью или нажатие ее с помощью клавиш клавиатуры.

С другими свойствами, событиями и методами кнопок познакомимся позднее, по мере необходимости.

Пример 1. Событие кнопки *Click*. Цвет формы

Постановка задачи

Создадим проект, в окне которого размещены три кнопки. Нажатие первых двух кнопок должно приводить к окрашиванию формы в различные цвета (нажатие первой кнопки — в красный цвет, нажатие второй — в синий). При этом в окне должна появляться надпись с названием цвета, например: "Это красный цвет". Нажатие третьей кнопки завершает работу программы.

Решение

Создадим новый проект типа **Стандартный EXE**. Выведем на экран диалоговое окно **Свойства** и панель элементов управления. Не помешает и диалоговое окно **Проводник Проекта**, кнопки которого позволяют быстро выводить на передний план форму или окно кода программы. Вывести перечисленные окна можно командами меню **Вид** или кнопками на главной панели инструментов.

Придадим форме размеры примерно 4800 на 3400 твипов. Размеры можно контролировать в правой части панели инструментов системы. Разместим в форме три кнопки, как показано на рис. 18.2. Последовательно выделяя форму и кнопки, присвоим их свойствам в окне **Свойства** значения, показанные в табл. 18.2.

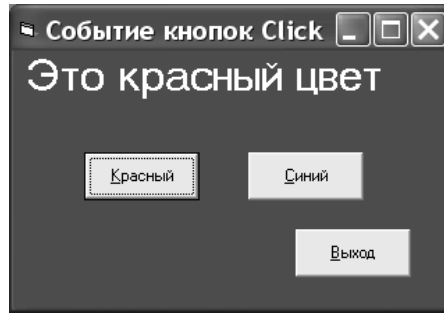


Рис. 18.2. Окно приложения к задаче о кнопках

Таблица 18.2. *Значения свойств объектов проекта*

Элемент	Свойство — Значение	Свойство — Значение
Form1	Name — frmЦвет	Caption — Событие кнопок Click
Command1	Name — cmdRed	Caption — &Красный
Command2	Name — cmdBlue	Caption — &Синий
Command3	Name — cmdВыход	Caption — &Выход

В запись названия кнопки (свойство Caption) добавлен знак амперсанда (&). Буква, перед которой он записывается, в названии кнопки подчеркивается. Одновременное нажатие клавиши <Alt> и клавиши с этой буквой (быстрой клавиши) равноценно нажатию кнопки мышью. Добавление в название амперсанда необязательно, но желательно. Особенно ценят возможность меньше переключаться на работу с мышью пользователи, которые много работают с клавиатурой и экономят время.

Щелчок на форме выводит на экран окно кода с процедурой загрузки формы. Щелчки на кнопках выводят в окно кода процедуры обработки событий Click. Заготовки процедур можно также получать выбором объекта в левом списке окна кода и выбором для него события в правом списке окна кода (в нашем случае для формы — Load, для кнопок — Click). Запишем тексты процедур так, как показано на рис. 18.3.

Запустив программу на выполнение, убедимся в правильности работы приложения. При появлении окна с сообщением об явных ошибках, например "For without Next" ("For без Next"), устраняйте их. При неявных ошибках

нажимайте в окне сообщения кнопку **Отладка** и устраняйте ошибки, на которые указывают окрашенные строки кода.

Присвоим проекту имя **Кнопки_1** и сохраним его в отдельной папке Кнопки 1.

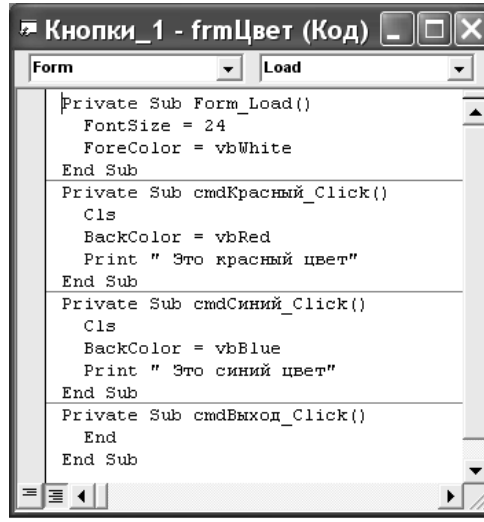


Рис. 18.3. Код программы решения задачи о кнопках

Упражнения

1. Убедимся, что при запуске программы в фокусе находится кнопка с наименьшим номером. Ее внешний вид отличается от вида остальных кнопок. По кнопкам фокус передается с их нажатием. Последовательно от кнопки к кнопке фокус можно передавать клавишей <Tab> или клавишами управления курсором клавиатуры. Если кнопка в фокусе, то нажатие клавиши <Пробел> или <Enter> равноценно щелчку мышью на этой кнопке. Проверьте эту возможность.
2. В режиме конструирования выделим в форме кнопку `cmdВыход` и в окне **Свойства** присвоим ее свойству `Cancel` (Отмена) значение True. Так создается *кнопка отмены*. Независимо от того, какой элемент имеет в данный момент фокус, нажатие клавиши <Esc> равноценно щелчку на кнопке **Выход**. Убедимся в этом.
3. Проверим работу быстрых клавиш, соответствующих подчеркнутым буквам в названиях кнопок. Щелчок мыши на кнопке заменяет нажатие клавиши <Alt> и быстрой клавиши этой кнопки. Например, для кнопки **Выход** это соответствует комбинации <Alt>+<В>.

Пример 2. Свойства кнопок *Enabled* и *Visible*

Постановка задачи

Создать приложение "Свойства кнопок". На форме расположены три кнопки, как показано на рис. 18.4. При запуске приложения видны только первые две кнопки. Из них доступна только первая. При нажатии на первую кнопку в форму выводится текст и делается доступной вторая кнопка. При нажатии второй кнопки форма очищается от текста и становится видимой третья кнопка. Нажатие на третью кнопку закрывает приложение.

Конструирование (визуальное программирование)

Создайте форму, разместите на ней три кнопки. На рис. 18.4 показан внешний вид окна приложения в процессе работы. Последовательно выделяйте форму, кнопки и присваивайте их свойствам значения, показанные в табл. 18.3. Дайте проекту имя **Кнопки_2**.

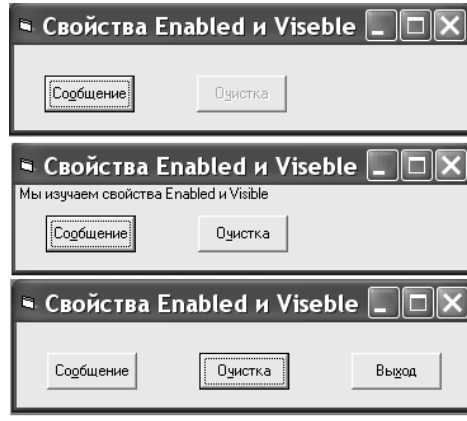


Рис. 18.4. Работа приложения при нажатии кнопок

Замечание

До присвоения всем элементам проекта имен не начинайте работу над кодом программы! Этим вы упростите отладку проекта и сохраните много времени.

Таблица 18.3. *Свойства объектов приложения*

Элемент	Свойство — Значение	Свойство — Значение
`Form1`	`Name` — `frmСвойстваКнопок`	`Caption` — Свойства `Enabled` и `Visible`

Таблица 18.3 *(окончание)*

Элемент	Свойство — Значение	Свойство — Значение
Command1	Name — cmdСообщение	Caption — Со&общение
Command2	Name — cmdОчистка	Caption — О&чистка
Command3	Name — cmdВыход	Caption — Вы&ход

Запись кода программы

Код программы состоит из четырех процедур обработки событий. Двойной щелчок на форме и на кнопках записывает в код заготовки этих процедур, которые следует заполнить инструкциями, определяющими логику работы приложения. В листинге 18.1 приведен код программы приложения.

Листинг 18.1. Свойства кнопок `Enabled` и `Visible`

```
Private Sub Form_Load()
  cmdОчистка.Enabled = False        'кнопка недоступна
  cmdВыход.Visible = False          'кнопка невидима
  cmdВыход.Cancel = True            'кнопку можно нажимать клавишей <Esc>
End Sub
Private Sub cmdСообщение_Click()
  Print " Мы изучаем свойства Enabled и Visible"
  cmdОчистка.Enabled = True         'кнопка стала доступной
End Sub
Private Sub cmdОчистка_Click()
  Cls
                                    'очистка формы
  cmdВыход.Visible = True           'кнопка видима
End Sub
Private Sub cmdВыход_Click()
  End
                                    'завершение работы
End Sub
```

Отладьте и сохраните созданное приложение под именем **Кнопки_2** в папке Кнопки 2.

18.3. Метка *(Label)*

Метки (надписи) выполняют важную роль при оформлении создаваемых проектов. Они могут использоваться самостоятельно или давать поясняющую информацию, например, по назначению и принципам работы других элементов управления. Текст названий при конструировании присваивается свойству `Caption` и далее может изменяться программным путем.

Перечислим некоторые из важных свойств меток.

- `Name` (имя) — записывается с префиксом `lbl`, например, `lblСообщение`.
- `Caption` (заголовок) — важнейшее свойство меток, определяющее содержание текста.
- `Alignment` (выравнивание) — позволяет выравнивать текст в надписи по левому краю, правому краю, по центру метки.
- `AutoSize` (авторазмеры) — логическое свойство, при значении True расширяет размеры метки в горизонтальном направлении, чтобы в ней поместился весь текст.
- `WordWrap` (перенос слов) — логическое свойство, при значении True обеспечивает перенос текста на следующую строку и расширение размеров метки по вертикали.
- `Appearance` (вид) — позволяет выбирать плоское или объемное изображения метки. При установке объемного изображения (3D) свойство `BorderStyle` требуется установить в 1 — Фиксировано один.

Меткам соответствуют многие стандартные события. В частности, они поддерживают события `Click`, `DblClick`, `Change`, `MouseDown`, `MouseUp`. Методы меток используются редко.

Пример 3. Свойства меток *AutoSize* и *WordWrap*

Создадим новый проект. Зададим форме имя `frmМетки`, свойству формы `Caption` присвоим значение `Свойства меток`.

Разместим на форме слева друг под другом две метки `Label1` и `Label2`. Работая с маркерами перемещения границ меток, зададим меткам одинаковые размеры (ширина и высота), примерно 800×500 твипов. Размеры можно контролировать по сведениям, отображаемым в правой части панели инструментов главного окна.

На рис. 18.5 показано окно приложения в процессе работы. В начале работы одинаковые по размерам метки трансформировались в соответствии со значениями свойств `AutoSize` и `WordWrap`.

Меткам Label1 и Label2 оставим имена по умолчанию, а их свойствам Caption присвоим соответственно тексты "Демонстрация свойства метки AutoSize" и "Демонстрация свойства метки WordWrap".

В окне свойств установим для обеих меток значение свойства BorderStyle, равное 1. При значении свойства Appearance, равном 3D, метка приобретает четко очерченный объемный вид.

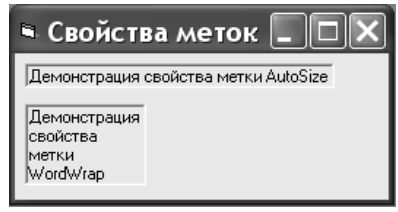


Рис. 18.5. Окно приложения "Свойства меток"

Выделим первую метку. В окне **Свойства** дважды щелкнем на ее свойстве AutoSize. Свойство примет значение True. При этом длина метки автоматически увеличивается до размера всего текста.

Выделим вторую метку. Придадим свойству AutoSize значение False. Вручную установим исходные размеры метки. Последовательно зададим сначала свойству WordWrap, затем AutoSize значения True. В результате увеличится высота метки. Текст окажется записанным полностью в нескольких строках метки.

Упражнение

Создайте приложение, демонстрирующее события метки Click и DblClick. Код приложения приведен в листинге 18.2. Свойствами метки BorderStyle и Font в окне **Свойства** сделайте объемное изображение метки и увеличьте размер шрифта.

Листинг 18.2. Щелчки на надписи

```
Private Sub lblНадпись_Click()
  lblНадпись.Caption = "Сделан одиночный щелчок"
  lblНадпись.AutoSize = True
End Sub
Private Sub lblНадпись_DblClick()
  lblНадпись.Caption = "Сделан двойной щелчок"
  lblНадпись.AutoSize = True
End Sub
```

18.4. Текстовое поле (TextBox)

Элемент управления TextBox используется для ввода и отображения информации, как во время проектирования, так и во время выполнения программы.

Свойства элемента TextBox следующие.

- Name — записывается с префиксом txt, например, txtПример.
- Text — это свойство заменило свойство Caption и определяет содержимое строки списка.
- Multiline — позволяет записывать текст в нескольких строках.
- PasswordChar — определяет символ, который заменяет вводимые пользователем символы пароля на некоторый выбранный символ, например *.

Основные события: Change, Click и MouseDown. Событие Change происходит каждый раз при вставке, замене или удалении символов в текстовом поле. Это можно использовать, например, для ввода скрытного текста с заменой символов.

Методы элемента TextBox: SetFocus и др.

Пример 4. Вычисление НДС

Создать приложение расчета НДС (налог на добавленную стоимость). Приложение использует метку, текстовое поле и три кнопки.

Этап конструирования (дизайн)

Разместите в форме управляющие элементы Label (метка), TextBox (текстовое поле), три кнопки (рис. 18.6).

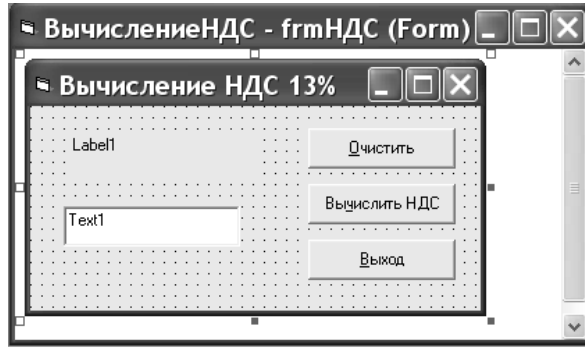


Рис. 18.6. Окно приложения "Вычисление НДС" в режиме конструирования

На рис. 18.6 форма расположена в окне конструирования формы. Заголовок окна содержит имя проекта (**ВычислениеНДС**) и имя формы (**frmНДС**). Заголовок формы (свойство `Caption`) — **Вычисление НДС 13 %**.

Значение свойства `Caption` метки `Label1` над текстовым окном будет меняться в процессе работы приложения. В табл. 18.4 приведены присвоенные значения свойств элементов приложения.

Таблица 18.4. *Свойства объектов приложения*

Элемент	Свойство — Значение	Свойство — Значение
`Form1`	`Name —frmНДС`	`Caption — Свойства Enabled и Visible`
`Label1`	`Name — lblНадпись`	`Caption — Введите сумму` `Caption — Размер НДС`
`TextBox`	`Name — txtНДС`	
`Command1`	`Name — cmdОчистить`	`Caption — &Очистить`
`Command2`	`Name — cmdВычислитьНДС`	`Caption — Вы&числитьНДС`
`Command3`	`Name — cmdВыход`	`Caption — &Выход`

Этап записи кода программы

В текстовое поле записывается текстовая величина. Поэтому для вычисления НДС требуется преобразование строковой величины в число, и обратно. Это можно делать, как показано в коде листинга 18.3, с помощью строковых функций `Val` и `Str` или с помощью функций преобразования `CDbl` и `CStr`.

Листинг 18.3. Код приложения “Вычисление НДС”

```
Private Sub Form_Load()
  txtНДС = ""                                'очистка текстового поля
  lblНадпись.Caption = "Введите сумму"
End Sub
Private Sub cmdВычислитьНДС_Click()
  lblНадпись.Caption = "Размер НДС"
  Sum = txtНДС.Text
  x = Val(Sum) * 13 / 100
  txtНДС.Text = Str(x)
End Sub
```

```
Private Sub cmdОчистить_Click()
  txtНДС.Text = ""
  txtНДС.SetFocus                    'установка в текстовое поле курсора
  lblНадпись.Caption = "Введите сумму"
End Sub
Private Sub cmdВыход_Click()
  End
End Sub
```

Запустите и при необходимости отладьте код приложения.

Упражнение

Приложение получилось "жестким", рассчитанным только на 13 процентов НДС. Желательно создавать приложения, способные легко перестраивать свои функции. В данном примере можно добавить еще одно тестовое окно, в которое можно будет записывать значение процентов НДС. Модернизируйте проект. Отладьте и сохраните его.

18.5. Таймер *(Timer)*

Таймер связан с системными часами компьютера и применяется для отсчета интервалов времени, задаваемых программно или при проектировании. Кроме свойств `Name` и `Enabled` важнейшим свойством таймера служит свойство `Interval`. Таймер в режиме выполнения не отображается на экране, что позволяет размещать его в любом месте формы.

Таймеры не обладают методами. Событие у них только одно — `Timer`. Оно наступает с частотой, определяемой свойством `Interval`. Значение свойства `Interval` задается в миллисекундах. Поэтому, если `Interval = 1000`, то событие `Timer` наступает каждую секунду.

Пример 5. Использование таймера. Бегущая строка

Создадим приложение "Бегущая строка". Используем форму, текстовое окно и таймер. Ниже показаны окна проекта на этапах конструирования (рис. 18.7) и выполнения программы (рис. 18.8).

Присвойте форме имя `frmБегущаяСтрока` и название `Бегущая строка`. Это же название "Бегущая строка" дайте проекту. По завершении отладки сохраните проект. Оставим по умолчанию имена таймера (`Timer1`) и текстового окна (`Text1`).

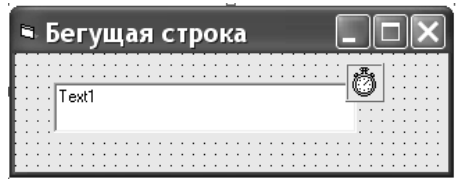


Рис. 18.7. Окно в режиме конструирования

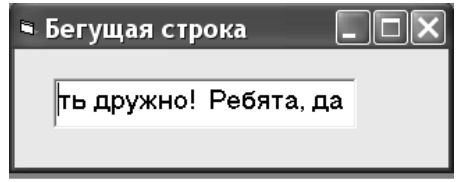


Рис. 18.8. Окно в режиме работы

Принцип получения эффекта бегущей строки заключается в замене с некоторой выбранной частотой текста в текстовом окне на текст, сдвинутый на один или большее количество символов. При этом, например, слово "роза " (с пробелом в конце) будет последовательно отображаться, как "оза р", "за ро", "а роз", "роза " и т. д. Вырезку частей текста и их склеивание выполняют в процедуре обработки события `Timer` строковая функция `Mid` и операция конкатенация.

В листинге 18.4 приведен код приложения. При загрузке формы устанавливаются значение свойства `Interval`, размер шрифта и текст текстового окна.

Листинг 18.4. Код приложения "Бегущая строка"

```
Dim a As String
Private Sub Form_Load()
  Timer1.Interval = 100
  Text1.Alignment = 2
  Text1.FontSize = 16
  Text1.Text = "Ребята, давайте жить дружно!  "
End Sub
Private Sub Timer1_Timer()
  a = Text1.Text
  a = Mid(a, 2, Len(a)) & Mid(a, 1, 1)
  Text1.Text = a
End Sub
```

Упражнение

Добавьте в форму две кнопки **Пуск** и **Стоп**, запускающую и останавливающую бегущую строку. Щелчки на кнопках в режиме конструирования добавят в код две процедуры события `Click`, в которые соответственно с кнопками впишите инструкции: `Timer1.Interval = 200` и `Timer1.Interval = 0`. Проверьте работу модернизированного приложения.

18.6. Рамка *(Frame)*

Рамку применяют при желании объединить несколько, как правило, одноименных, элементов управления в группу, выделить эту группу рамкой и дать ей название. Чаще всего рамку используют для выделения групп флажков и переключателей (рис. 18.9).

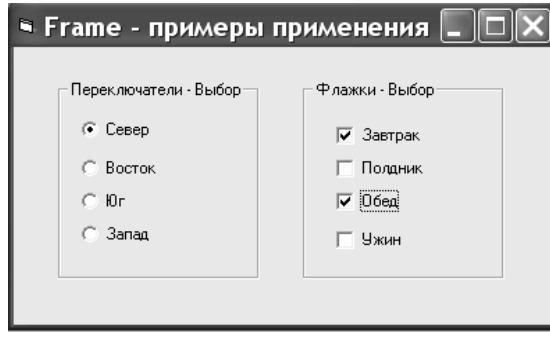


Рис. 18.9. Примеры использования рамки

Желательно обеспечить видимость элементов в рамке и возможность перемещать рамку в форме вместе с элементами. Для этого сначала на форму добавляется рамка, а затем в ней рисуются объединяемые элементы. Если элемент выбран на панели инструментов двойным щелчком или перемещен в рамку со стороны, то элемент и рамка будут перемещаться отдельно.

18.7. Флажок *(Check Box)*

Флажок представляет собой "галочку" ("птичку"), которая ставится в маленьком квдратном окне. Группа таких квадратов соответствует некоторому множеству вариантов, из которых установкой флажков (проставлением "галочек") выбираются нужные комбинации.

По логике работы установка или снятие одного флажка соответствует выбору из двух вариантов, решению альтернативы: "Да" или "Нет". Флажки работают независимо друг от друга.

Свойства флажков: `Name`, `Caption`, `Enabled`, `Value`, `Style` и др.

Основное событие: `Click`.

Методы почти не используются.

Состояния флажка определяются свойством `Value` (значение). Флажок может быть снят (сброшен) (значение 0), установлен (значение 1) и недоступен (значение 2). По умолчанию флажки сброшены. Свойство `Value` используется при необходимости или установить некоторые флажки заранее, или определить состояние флажка, или сделать его при определенных обстоятельствах недоступным.

Внешний вид флажка определяется свойством `Style` (стиль). Принимает два значения: 0 — `Standard` и 1 — `Graphical`. При втором значении флажок принимает вид отпущенной или нажатой кнопки.

На рис. 18.10 показано окно **Свойства флажков**, с демонстрацией внешнего вида флажков, имеющих различные значения свойств `Value` и `Style`.

Рис. 18.10. Демонстрационное окно **Свойства флажков**

Пример 6. Работа с флажками. Вид шрифта

На рис. 18.11 показано окно приложения "Работа с флажками". Два флажка позволяют задать четыре варианта написания шрифта надписи. Реализуйте этот проект приложения и проверьте его работу.

Код программы состоит из трех процедур. Первая процедура — это процедура обработки события `Click` — смены состояния флажка **Жирный** (`Bold`). Вторая процедура — смена состояния флажка **Курсив** (`Italic`).

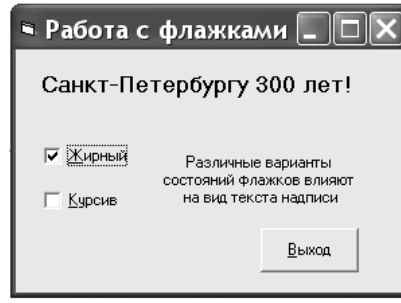


Рис. 18.11. Окно приложения "Работа с флажками"

Заметим, что свойства `FontBold`, `FontItalic`, `FontStrikethru` (перечеркнутый) и `FontUnderline` (подчеркнутый) являются свойствами текстовых полей и меток.

Листинг 18.5. Код приложения "Работа с флажками"

```
Private Sub chkBold_Click()
  If chkBold.Value = 1 Then
    lblМетка.FontBold = True
  Else
    lblМетка.FontBold = False
  End If
End Sub
Private Sub chkItalic_Click()
  If chkItalic.Value = 1 Then
    lblМетка.FontItalic = True
  Else
    lblМетка.FontItalic = False
  End If
End Sub
Private Sub cmdВыход_Click()
  End
End Sub
```

Упражнение

Добавьте на форму еще два флажка: **Подчеркнутый** (`FontUnderline`) и **Перечеркнутый** (`FontStrikethru`). Допишите код программы. Проверьте работу всех шестнадцати вариантов записи текста.

18.8. Переключатель *(Option Button)*

Переключатели используются при выборе единственного варианта из нескольких. Если в группе можно установить одновременно несколько флажков, то при выборе некоторого переключателя все остальные переключатели группы автоматически отключаются. Группы переключателей можно создать, нарисовав их внутри некоторого контейнера или в форме. В качестве контейнера можно использовать графическое окно или рамку.

Свойства, события и методы переключателей во многом совпадают со свойствами, событиями и методами флажков. Название переключателя определяется значением свойства `Caption`, располагается на форме справа от переключателя и указывает пользователю на его назначение.

Состояние переключателя определяется логическим свойством `Value`. По умолчанию это свойство имеет значение False. Если при конструировании задать ему значение True, то при открытии формы соответствующий переключатель будет включен. Если программным путем задать свойству `Value` значение True у другого переключателя, то предыдущий переключатель автоматически выключатся.

Внешний вид переключателя определяется свойством `Style`, которое работает аналогично свойству `Style` у флажков. Свойство можно использовать, если возникает желание установить несколько кнопок, из которых одновременно можно выбрать только одну.

Пример 7. Использование переключателей

В качестве примера использования переключателей реализуйте простое приложение "Работа переключателей", окно которого показано на рис. 18.12.

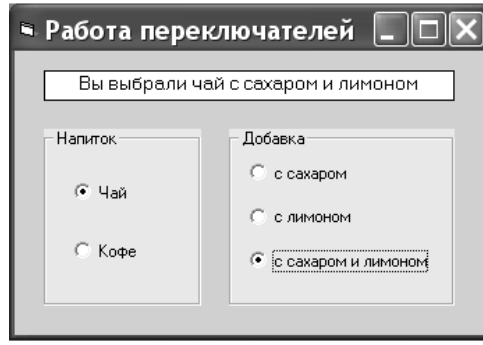


Рис. 18.12. Окно приложения "Работа переключателей"

Окно имеет метку `lblТекст` со свойствами `Appearance = 0` и `BorderStyle = 1`. Это придает метке вид текстового окна. Ниже метки размещены две группы переключателей. Обе группы размещены в рамках. Обе группы не зависят друг от друга.

Код программы состоит из пяти процедур обработки события `Click` пяти переключателей приложения. В каждой из этих процедур строковым переменным `strНапиток` или `strДобавка` присваиваются соответствующие значения. После этого управление передается закрытой общей процедуре `Заказ`, где с помощью операции конкатенации формируется значение свойства `Caption` метки `lblЗаказ`. Подробнее о работе с общими процедурами будет рассказано в *гл. 19*.

Можно было обойтись без передачи управления процедуре `Заказ` и формировать текст в каждой процедуре обработки события `Click` переключателей, анализируя значения свойства `Value`. Например, так было сделано в программе листинга 18.5, где рассматривается работа флажков. Однако использование общих процедур позволяет, как правило, существенно упрощать код программы.

Листинг 18.6. Код приложения "Работа переключателей"

```
Dim Напиток As String: Dim Добавка As String
Private Sub optЧай_Click()
  Напиток = "чай ": Call Заказ
End Sub
Private Sub optКофе_Click()
  Напиток = "кофе ": Call Заказ
End Sub
Private Sub optС_лимоном_Click()
  Добавка = "с лимоном": Call Заказ
End Sub
Private Sub optС_сахаром_Click()
  Добавка = "с сахаром": Call Заказ
End Sub
Private Sub optС_сахаром_и_лимоном_Click()
  Добавка = "с сахаром и лимоном": Call Заказ
End Sub
Sub Заказ()
  lblЗаказ.Caption = "Вы выбрали " & Напиток & Добавка
End Sub
```

18.9. Объединение элементов управления в массив

В рассмотренном выше примере обращает на себя внимание большое количество процедур (по числу переключателей), включающих небольшое число инструкций. Если одноименное событие связано с однотипными элементами управления (флажки, переключатели, кнопки), то их обычно объединяют в массив. К примеру, удобно объединить в массив кнопки ввода цифр в калькуляторе.

Пример 8. Массив переключателей. Приложение "Выбор цветка"

Показанное на рис. 18.13 окно проекта содержит две метки (`Label1` и `Label2`) и пять переключателей.

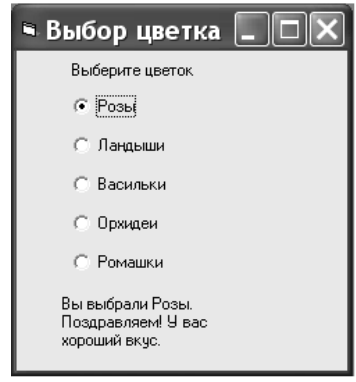


Рис. 18.13. Массив переключателей (окно приложения "Выбор цветка")

Для создания массива (группы) переключателей следует перенести в форму первый переключатель, выделить его и нажать кнопку **Копировать** главного меню. Этим мы отправляем переключатель в буферную память. При нажатии кнопки **Вставить** на экран выводится окно с вопросом — действительно ли мы желаем создать массив (`array`) этих элементов управления. Нажмем кнопку **Да**. После этого вставляем из буфера нужное число (пять) переключателей, каждый раз сбрасывая выделение очередного элемента щелчком на форме.

Все переключатели созданного массива имеют одно имя `Option1` и различаются только значением свойства `Index`, которое изменяется от 0 до 4. Поэтому запись кода программы с массивом элементов управления получается

более компактной и укладывается в одну процедуру, как показано в коде листинга 18.7.

Листинг 18.7. Код приложения с массивом переключателей "Выбор цветка"

```
Private Sub Option1_Click(Index As Integer)
  Label1.Caption = "Выберите цветок"
  Select Case Index
    Case 0: strA = "Розы.  "
    Case 1: strA = "Ландыши.  "
    Case 2: strA = "Васильки.  "
    Case 3: strA = "Орхидеи.  "
    Case 4: strA = "Ромашки.  "
  End Select
  Label2.Caption = "Вы выбрали " & strA
  ↳"Поздравляем! У вас хороший вкус."
End Sub
```

18.10. Список *(ListBox)*

Элемент `ListBox` (рис. 18.4) хранит список элементов, из которых можно выбирать один или несколько элементов. Элементы в списке можно читать, добавлять, удалять, сортировать, изменять их названия, выполнять с ними другие действия.

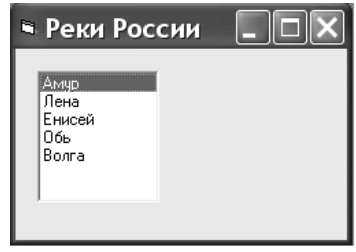


Рис. 18.14. Пример списка **Реки России**

Свойства, методы, события элемента *ListBox*

Свойства элемента `ListBox` следующие.

- `Name` — записывается обычно с префиксом `lst`, например, `lstРеки_России`.

- `List` — свойство, значение которого представляет собой массив строк списка. Оно задает или возвращает содержимое строки списка. Для этого требуется указывать номер (`Index`) строк списка, которые нумеруются в последовательности: 0, 1, 2.

  Синтаксис:

  `<имя списка>.List (Index).`

  `Пример`:

  строка кода `Print lstРеки_России.List (2)` возвращает слово Енисей (см. рис. 18.14).

- `ListCount` — возвращает число элементов списка.

  Пример:

  строка кода `Print lstРеки_России.ListCount` выведет в форму число 5.

- `NewIndex` — возвращает номер последней строки (значение `ListCoumnt` равно значению `NewIndex` плюс один).

- `ListIndex` — возвращает или устанавливает номер текущей выделенной строки списка.

- `Text` — хранит содержимое выделенной строки списка.

- Selected — выделяет строку списка.

  Пример:

  строка кода `lstРеки_России.Selected (4)` выделяет слово Волга.

- `MultiSelect` — позволяет выделить сразу несколько строк списка подряд (с клавишей <Shift>) или выборочно (с клавишей <Ctrl>).

- `Sorted` — обеспечивает сортировку списка в алфавитном порядке без учета регистра символов. Значение свойства задается только в режиме конструирования.

Методы элемента **`ListBox`** следующие.

- `Clear` — очищает список от всех элементов.

  Пример:

  `lstРеки_России.Clear.`

- `AddItem` — добавляет элемент списка.

  Пример:

  строка кода `lstРеки_России. AddItem "Ангара"` — добавит реку Ангара в конец списка, а строка кода `lstРеки_России. AddItem "Нева", 0` — поставит реку Нева в начало списка.

- `RemoveItem` — удаляет элементы списка.

  Пример:

  строка кода `List1.RemoveItem (0)` удаляет начальный элемент списка.

Из событий элемента `ListBox` чаще других используются события `DblClick` и `Click`.

Заполнение списка в окне *Свойства*

Для заполнения списка во время разработки можно воспользоваться свойством `List`.

Упражнение

Поместите в форму элемент управления `ListBox` и дважды щелкните мышью по свойству `List`. Открывается поле списка, в которое введите элементы списка **Реки России**.

Переход на новую строку при заполнении списка выполняется командой <Ctrl>+<Enter>. Эта команда также позволяет освободить место в любом месте списка для вставки нового элемента. После заполнения списка в окне свойств нажмите на клавишу <Enter>. Это перенесет список в окно элемента `ListBox`.

Заполнение списка программным путем

В листинге 18.8 приводится код процедуры загрузки формы и заполнения списка `List1`.

Листинг 18.8. Процедура заполнения списка

```
Private Sub Form_Load()
  With List1
    .AddItem "Амур"
    .AddItem "Лена"
    .AddItem "Енисей"
    .AddItem "Обь"
    .AddItem "Волга"
  End With
End Sub
```

В процедуре заполнения использован оператор `WithEnd With`, который часто упрощает запись кода, делая его более наглядным.

Пример 9. Свойства и методы списков. Приложение "Реки Европы и Азии"

Создадим проект-приложение, демонстрирующий свойства и методы элемента управления `ListBox`. Внешний вид окна приложения после загрузки формы показан на рис. 18.15.

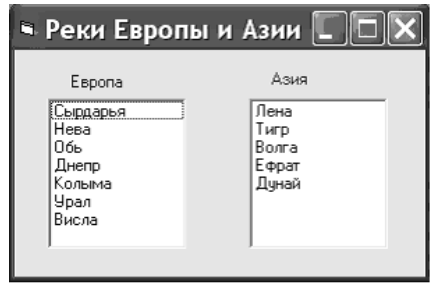


Рис. 18.15. Окно приложения "Реки Европы и Азии"

Проект содержит:

- одну форму (`Name` — `frmДваСписка`, `Caption` — `Реки Европы и Азии`);
- две метки (`Label1`, `Caption` — `Европа` и `Label2`, `Caption` — `Азия`);
- два элемента `ListBox` (имена — `lstЕвропа` и `lstАзия`).

После присвоения имен запишите показанный в листинге 18.9 код трех процедур обработки событий: загрузки формы с заполнением списков и двойных щелчков на элементах списков. Двойные щелчки отправляют выбранные строки в соседний список. Исправьте заполнение списков в соответствии с истинным расположением рек.

Листинг 18.9. Код приложения "Реки Европы и Азии"

```
Private Sub Form_Load()
  lstЕвропа.AddItem "Сырдарья": lstЕвропа.AddItem "Нева"
  lstЕвропа.AddItem "Обь":  lstЕвропа.AddItem "Днепр"
  lstЕвропа.AddItem "Колыма":  lstЕвропа.AddItem "Урал"
  lstЕвропа.AddItem "Висла":  lstАзия.AddItem "Лена"
  lstАзия.AddItem "Тигр":  lstАзия.AddItem "Волга"
  lstАзия.AddItem "Ефрат":  lstАзия.AddItem "Дунай"
End Sub
```

```
Private Sub lstАзия_DblClick()
  lstЕвропа.AddItem  lstАзия.Text
  lstАзия.RemoveItem  lstАзия.ListIndex
End Sub
Private Sub lstЕвропа_DblClick()
  lstАзия.AddItem  lstЕвропа.Text
  lstЕвропа.RemoveItem  lstЕвропа.ListIndex
End Sub
```

18.11. Комбинированное поле — поле со списком *(ComboBox)*

Элемент управления `ComboBox` сочетает в себе возможности текстового поля (`TextBox`) и списка (`ListBox`).

Свойства элемента `ComboBox` во многом совпадают со свойствами элемента `ListBox`. Это `List`, `ListCount`, `ListIndex`, `NewIndex`, `Sorted` и др. Рассмотрим имеющиеся отличия.

`Name` (имя) — записывается с префиксом `cbo`, например, `cboГорыРоссии`.

`Style` — может принимать три значения (0, 1 и 2), определяющие внешний вид и поведение элемента:

- раскрывающееся комбинированное поле. Имеет вид текстового поля со стрелкой (шевроном) в правой части. Нажатие на шеврон раскрывает список. В списке можно выбрать строку, которая переместится в текстовое поле, или добавить в поле новый элемент;
- простое комбинированное поле. Отличается постоянно открытым списком;
- раскрывающийся список.

Методы комбинированных полей: `Clear`, `AddItem`, `RemoveItem` и др.

События комбинированных полей: `Click`, `DblClick` и др.

Пример 10. Поле со списком. Приложение "Горы России"

Создадим приложение, демонстрирующее работу некоторых свойств и методов комбинированного поля.

Приложение содержит форму, комбинированное поле и четыре кнопки. Имена и значения свойств элементов собраны в табл. 18.5.

Таблица 18.5. *Свойства объектов приложения*

Элемент	Свойство — Значение	Свойство — Значение
Form1	Name —frmПолеСоСписком	Caption = Комбинированное поле
ComboBox	Name — cboГорыРоссии	Style = 0 или Style = 1
Command1	Name — cmdДобавить	Caption = &Добавить
Command2	Name — cmdУдалить	Caption = &Удалить
Command3	Name — cmdОчистить	Caption = &Очистить
Command4	Name — cmdВыход	Caption = &Выход

На рис. 18.16 показано окно приложения после запуска.

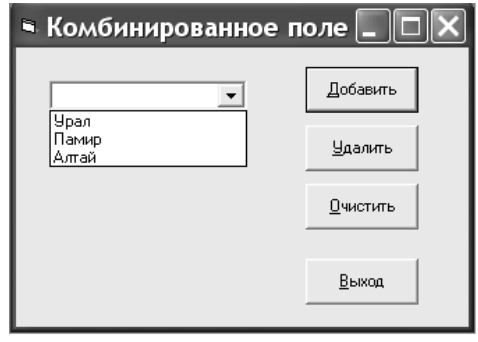


Рис. 18.16. Окно приложения "Горы России" с элементом ComboBox

В листинге 18.10 приведен код программы. При загрузке формы список заполняется тремя названиями гор. Список можно дополнить, исправить, очистить. В списке можно удалять отдельные строки. Проверьте эти возможности.

Листинг 18.10. Код приложения "Горы России"

```
Private Sub Form_Load()
  cboГорыРоссии.Clear
  cmdДобавить.Default = True
  cboГорыРоссии.AddItem "Урал"
  cboГорыРоссии.AddItem "Памир"
```

```
  cboГорыРоссии.AddItem "Алтай"
End Sub
Private Sub cmdДобавить_Click()
  cboГорыРоссии.AddItem cboГорыРоссии.Text
End Sub
Private Sub cmdОчистить_Click()
  cboГорыРоссии.Clear
End Sub
Private Sub cmdУдалить_Click()
  cboГорыРоссии.RemoveItem cboГорыРоссии.ListIndex
End Sub
Private Sub cmdВыход_Click()
  End
End Sub
```

18.12. Изображение, рисунок *(Image)*

Элемент управления `Image` используется для отображения графики. Он позволяет размещать в окне приложения рисунки, задаваемые файлами с расширениями bmp (растровые файлы), ico (значки), wmf (метафайлы), cur (курсоры) и др. Для помещения элемента в форму используется расположенная на панели элементов управления кнопка **Image**.

Элемент имеет свойства: `Name`, `Picture`, `Stretch`, `BorderStyle`, `Index` и др.

Методы: `Drag`, `Move` и др.

События: `Click`, `DragDrop` и др.

Загрузить изображения в элемент `Image` можно несколькими способами.

- Щелчок на свойстве `Picture` вызывает диалоговое окно **Загрузить рисунок**, в котором предлагается выбрать файл, содержащий требуемый рисунок. После выбора файла нажатие кнопки **Открыть** вставляет файл в поле элемента `Image`.
- Во время разработки вставка изображения может выполняться также посредством буфера обмена. Для этого нужно в приложении (Word, Paint и др.) отправить выбранный рисунок в буфер обмена и вставить его в выделенный элемент `Image` кнопкой **Вставить** или клавишами <Ctrl>+<V> или <Shift>+<Insert>.
- Во время выполнения программы рисунок можно вставить, используя синтаксис:

```
Image1.Picture = LoadPicture("полное имя файла").
```

Одним из главных свойств элемента `Image` является свойство `Stretch` (растягивать). При значении этого свойства False (задается по умолчанию) независимо от размеров элемента `Image` рисунок вставляется без изменения размеров. При этом размеры элемента автоматически подгоняются под размер рисунка. При изменении размеров в дальнейшем изображение может обрезаться и окружаться пустотами.

При значении свойства `Stretch`, равном True, рисунок трансформируется в границы заданного элемента, размеры которого при вставке не изменяются. При этом рисунок можно растягивать, увеличивая его размеры. При растягивании рисунка его качество может ухудшаться.

Пример 11. Свойство *Stretch* элемента *Image*

На форме расположены два элемента `Image` (рис. 18.17) одинакового размера (примерно 1200×1200 твипов). У первого элемента значение свойства `Stretch` установлено в True, у второго — в False.

В текстовом редакторе Word в меню **Вставка | Рисунок | Картинки | Животные** был выбран рисунок и через буфер обмена вставлен в поле элементов `Image`. Перемещая границы элементов, можно убедиться в различном поведении вставленных рисунков.

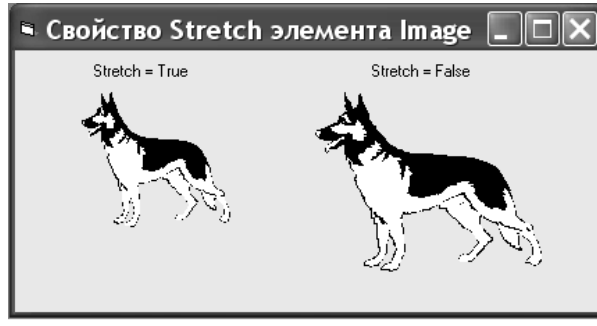


Рис. 18.17. Демонстрация свойства `Stretch`

Пример 12. Импорт рисунка в режиме выполнения

Создано приложение с импортом рисунка с помощью кода в процессе выполнения приложения. Приложение содержит два элемента `Image` одинакового размера.

В текстовом редакторе Word в меню **Вставка | Рисунок | Картинки | Животные** был выбран рисунок собаки, скопирован через буфер обмена в графический редактор Paint, несколько увеличен и записан в память по адресу C:\Мои документы\Мои рисунки\Собака.bmp. Программа, приведенная в листинге 18.11, вставляет рисунок в оба элемента `Image`.

Листинг 18.11. Вставка рисунка при работе приложения

```
Private Sub Form_Load()
  Image1.Stretch = True
  Image1.Picture = LoadPicture("C:\Мои документы\Мои рисунки\Собака.bmp")
  Image2.Stretch = False
  Image2.Picture = LoadPicture("C:\Мои документы\Мои рисунки\Собака.bmp")
End Sub
```

Пример 13. Приложение "Геометрические фигуры"

Постановка задачи

Создать показанное на рисунке приложение. Приложение содержит 10 элементов `Image` и два элемента `Label`.

Этап конструирования

Создайте новый проект и разместите на форме элементы управления. Из элементов `Image` создайте массив элементов. Свойство `Stretch` элементов `Image` установите в состояние True.

Рис. 18.18. Окно приложения "Геометрические фигуры"

В текстовом редакторе Word создайте документ и, используя на панели инструментов **Рисование** раздел **Автофигуры | Основные фигуры**, разместите в документе показанные на рисунке 18.18 фигуры. Используйте второй способ вставки изображений в элемент `Image` — через буфер обмена.

Запись кода программы

Щелчок на одном из элементов `Image` выводит на экран окно кода с заготовкой процедуры обработки события. Заполните эту процедуру инструкциями.

Листинг 18.12. Код приложения "Геометрические фигуры"

```
Private Sub Image1_Click(Index As Integer)
  Select Case Index
    Case 0: Label2.Caption = "Это круг"
    Case 1: Label2.Caption = "Это квадрат"
    Case 2: Label2.Caption = "Это треугольник"
    Case 3: Label2.Caption = "Это цилиндр"
    Case 4: Label2.Caption = "Это куб"
    Case 5: Label2.Caption = "Это параллелограмм"
    Case 6: Label2.Caption = "Это пятиугольник"
    Case 7: Label2.Caption = "Это ромб"
    Case 8: Label2.Caption = "Это трапеция"
    Case 9: Label2.Caption = "Это овал (эллипс)"
  End Select
End Sub
```

18.13. Графическое поле *(PictureBox)*

Элемент управления `PictureBox` (графическое поле), как и элемент `Image`, служит для отображения графики. Он позволяет размещать на своем поле рисунки графических файлов, текст и рисунки, создаваемые разными методами. Кроме того, элемент `PictureBox` используют в качестве контейнера для размещения других элементов.

Перечислим наиболее используемые свойства элемента `PictureBox` (всего их более 66).

- `Name` — записывается с префиксом `pic`.
- `Align` — выравнивает границы элемента по границам формы.

- `AutoSize` — подгоняет графическое поле под размер изображения.

  Синтаксис:

  `Picture1.AutoSize = {False|True}.`

- `Picture` — определяет содержимое графического поля.

  Например, код

  `Picture1.Picture = LoadPicture("<полное имя файла>")` — выполяет загрузку рисунка,

  а код

  `Picture1.Picture = LoadPicture()` — очистку графического поля.

- `Image` — копирует рисунок одного графического поля в другое.

  Синтаксис:

  `Picture2.Picture = Picture1.Image.`

- `Scale` — задает систему координат.

Методы: `PaintPicture`, `Cls`, `Circle`, `Line`, `Pset` и др.

События: `Click`, `DragDrop`, `Paint`, `MouseMove` и др.

Пример 14. Свойство Графического окна *AutoSize*

На рис. 18.19 показаны результаты вставки изображения в два одинаковых по размерам элемента `Picture` (примерно 1400×1400 твипов). У левого элемента свойство `AutoSize` = False, у правого — `AutoSize` = True.

В отличие от элемента `Image` размер вставляемого изображения не изменяется. При `AutoSize` = False рисунок обрезается по размеру элемента, при `AutoSize` = True — размеры элемента подгоняются под размер рисунка.

Рис. 18.19. Элемент `PictureBox`

18.14. Полосы прокрутки (ScrollBar)

Элементы ScrollBar используются для установки в заданном диапазоне численных значений некоторых параметров. Например, с их помощью можно плавно изменять и задавать интенсивность цвета, громкость звука, значения температуры, скорости и т. п. Горизонтальные и вертикальные полосы прокрутки различаются только ориентацией.

Элементы ScrollBar — это отдельные элементы, в отличие от полос прокрутки, которые автоматически вставляются, например, в списки. Они по умолчанию отсутствуют на панели встроенных управляющих элементов, относятся к группе стандартных элементов Windows и выводятся на панель командой главного меню **Проект | Компоненты |** вкладка **Управления** и установкой флажка **Microsoft Windows Common Controls 6.0** с последующим нажатием кнопки **OK**.

Свойства элемента ScrollBar следующие.

- Name (имя) — записывается с префиксом hsb (горизонтальный) или vsb (вертикальный), например, hsbЯркость.
- Value — это целочисленная величина, текущее значение полосы прокрутки.
- Min, Max — граничные значения полосы прокрутки.
- SmallChange — малое смещение, определяет величину, на которую происходит смещение бегунка при щелчке по стрелке полосы прокрутки.
- LargeChange — большое смещение, определяет величину, на которую смещается бегунок при щелчке на полосе прокрутки.

События элемента ScrollBar следующие.

- Change — происходит при изменении свойства Value с окончанием перетаскивания бегунка или после щелчка по стрелке или полосе прокрутки.
- Scroll — происходит во время перетаскивания бегунка.

Методы элемента ScrollBar используются редко.

Пример 15. Элемент прокрутки в приложении "Температура воздуха"

Создайте EXE-проект демонстрации свойств элемента ScrollBar и его события Change. Для этого потребуется одна форма и один элемент горизонтальной прокрутки (рис. 18.20).

Форме присвойте имя frmТемпноатураВоздуха и Caption — Температура воздуха. Элементу HscrollBar дайте имя hsbТемпература. Значения основ-

ных свойств элемента прокрутки будем присваивать в процедуре загрузки формы.

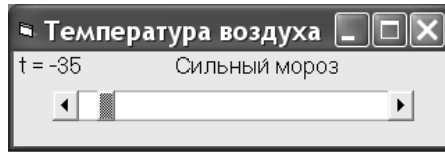


Рис. 18.20. Использование элемента `ScrollBar`

Запишите показанный в листинге 18.13 код программы. Отладьте приложение. Сохраните его.

Листинг 18.13. Код приложения "Температура воздуха"

```
Private Sub Form_Load()
  hsbТемпература.Max = 40
  hsbТемпература.Min = -40
  hsbТемпература.SmallChange = 5
  hsbТемпература.LargeChange = 5
  FontSize = 12
End Sub
Private Sub hsbТемпература_Change()
  Cls
  t = hsbТемпература.Value
  Select Case t
    Case -40 To -31: Print " t = "; t, " Сильный мороз"
    Case -30 To -11: Print " t = "; t, " Мороз"
    Case -10 To 10: Print " t = "; t, " Холодно"
    Case 11 To 20: Print " t = "; t, " Тепло"
    Case 21 To 30: Print " t = "; t, " Жарко"
    Case 31 To 40: Print " t = "; t, " Сильная жара"
  End Select
End Sub
```

18.15. Фигура *(Shape)*

Элемент управления `Shape` (фигура) используется при необходимости получить фигуру из показанного на рис. 18.21 набора, в котором первая строка содержит варианты фигур, вторая — варианты заполнения фигуры.

- Свойства элемента `Shape`:
- Name (имя) — записывается с префиксом `shp`;
- `Shape` — определяет внешний вид фигуры;
- `BorderWidth` — определяет ширину границы;
- `BorderStyle`, `FillStyle` — определяют тип линии и заполнения;
- `BorderColor`, `FillColor` — определяют цвет границы и заливки фигуры.

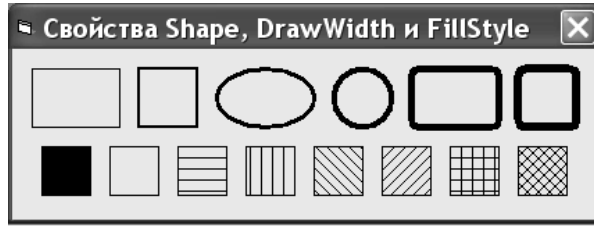


Рис. 18.21. Демонстрация свойств элемента `Shape`

Подробнее с элементом `Shape` познакомимся в следующем разделе.

18.16. Счетчик *(UpDown)*

Управляющий элемент `UpDown` (счетчик) служит для установки различных значений. Он представляет собой две кнопки со стрелками, как в полосах прокрутки, и используется совместно с элементами управления, способными работать с числовой информацией. Если он связывается с элементом `TextBox`, то каждый щелчок на одной из стрелок будет увеличивать, а на другой — уменьшать хранимое в текстовом окне число на заданную величину.

Свойства элемента `UpDown`:

- `Alignment` — определяет положение счетчика относительно связанного с ним элемента;
- `AutoBuddy` — при значении True связанным становится ближайший по `TabIndex` объект;
- `BuddyControl` — имя связанного со счетчиком элемента;
- `Increment` — шаг изменения показаний счетчика;
- `Max`, `Min` — наибольшее и наименьшее допустимые значения счетчика;
- `Orientation` — определяет ориентацию кнопок в счетчике;
- `SuncBuddy` — при значении True требует обновлять значения в связанном элементе;

- Value — показания счетчика;
- Wrap — при значении True обеспечивает цикличность счета (по модулю Max).

События элемента UpDown:

- Change — возникает при изменении показаний счетчика;
- DownClick — возникает при нажатии стрелки, направленной вниз;
- UpClick — возникает при нажатии стрелки, направленной вверх.

Пример 16. Элемент *UpDown* и свойства элемента *Shape*

Создадим EXE-проект с демонстрацией свойств элементов UpDown и Shape. Проект содержит одну форму с именем frmUpDown_СвойстваShape и Caption — Элементы UpDown и Shape, элемент Shape1, четыре элемента UpDown и четыре связанных с ними элемента TextBox. Имена восьми последних элементов оставим по умолчанию.

В окне, вызываемом командой главного меню **Проект | Компоненты**, в списке на вкладке **Управления** установите флажок **Microsoft Windows Common Controls – 2 6.0**. При этом на панель инструментов управления будет выведена группа стандартных элементов, среди которых находится элемент UpDown.

Рассмотрим связывание первой пары: Text1 и UpDown1. Выведите в форму первый TextBox (Text1) и задайте ему небольшие размеры (рис. 18.22).

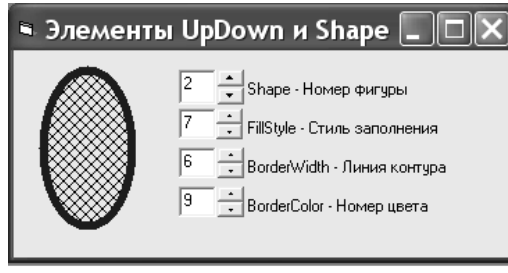


Рис. 18.22. Свойства элемента Shape

Задайте первый элемент UpDown (UpDown1). Свойства TabIndex этих элементов различаются на единицу, что упрощает их связывание. Достаточно выделить UpDown1 и в окне свойств установить значение свойства AutoBuddy равным True. При этом свойство BuddyControl автоматически примет значение Text1. Установите в значение True свойства SyncBuddy и Wrap. Это

обеспечит обновление значений и цикличность счета. Значение свойства `Max` будет установлено при загрузке формы.

Связывание остальных трех пар выполняется по рассмотренным правилам.

В листинге 18.14 приведен код приложения.

Листинг 18.14. Элементы `UpDown` и `Shape`

```
Private Sub Form_Load()
  UpDown1.Max = 5: UpDown2.Max = 7
  UpDown3.Max = 10: UpDown3.Min = 1
  UpDown4.Max = 15
End Sub
Private Sub UpDown1_Change()
  Shape1.Shape = UpDown1.Value
End Sub
Private Sub UpDown2_Change()
  Shape1.FillStyle = UpDown2.Value
End Sub
Private Sub UpDown3_Change()
  Shape1.BorderWidth = UpDown3.Value
End Sub
Private Sub UpDown4_Change()
  Shape1.BorderColor = QBColor(UpDown4.Value)
End Sub
```

18.17. Элемент *ProgressBar*. Индикация хода процесса

В операционных системах и приложениях элемент `ProgressBar` используется для извещения пользователя о том, что процесс продолжается, запущенная задача решается, зависания нет. При этом наглядно отображается информация о том, какая часть процесса уже выполнена и сколько времени осталось ждать его завершения. Элемент может быть использован, например, при выполнении учащимися некоторых тестовых заданий в условиях заранее заданных ограничений по времени.

Элемент `ProgressBar` относится к группе стандартных элементов Windows и вызывается на панель элементов управления так же, как и элемент `UpDown`. Но в окне **Компоненты** требуется установить флажок **Microsoft Windows Common Controls – 6.0**.

Свойства элемента `ProgressBar`:

- `Height Width` — высота и ширина индикатора;
- `Max`, `Min` — максимальное и минимальное значения свойства `Value`;
- `Value` — состояние элемента, определяет размер выполненной части процесса.

Размещение его в форме выполняется обычным способом. Обратите внимание на то, что размер и количество фрагментов, заполняющих окно элемента, зависят от его высоты.

Пример 17. *ProgressBar*. Конец работы

В следующем приложении (рис. 18.23) значения свойств `Max` и `Min` элемента `ProgressBar` оставлены заданными по умолчанию (100 и 1 соответственно). Поэтому при выбранном интервале работы таймера, равном 0,1 секунды, поле элемента `ProgressBar` заполняется за 10 секунд.

Выбирая интервал срабатывания таймера и значения свойств `Max` и `Min` элемента `ProgressBar`, можно задавать требуемое время заполнения окна элемента.

В приложении форме присвоено имя `frmКонецРаботы` и `Caption` — `Конец работы`. Две кнопки имеют соответственно имена `cmdПуск`, `Caption = Пуск` и `cmdВыход`, `Caption = Выход`. Имя таймера оставлено по умолчанию — `Timer1`. Текст выводится в форму методом `Print`.

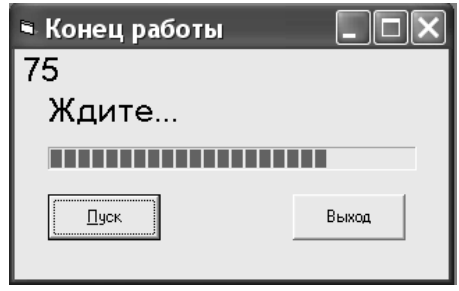


Рис. 18.23. Приложение с элементом `ProgressBar`

Листинг 18.15. Приложение с элементом `ProgressBar`

```
Private Sub cmdПуск_Click()
  Timer1.Interval = 100
  ProgressBar1.Max = 100
End Sub
```

```
Private Sub Timer1_Timer()
  Static p As Integer
  p = p + 1
  ProgressBar1.Value = p
  frmКонецРаботы.FontSize = 18
  Cls
  Print p: Print "    Ждите..."
  If p = 100 Then
    Timer1.Interval = 0
    ProgressBar1.Value = 0
    Cls
    Print p: Print "    Конец работы!!!"
  End If
End Sub
```

18.18. Элемент *Slider*. Бегунок.

Управляющий элемент `Slider` служит для ввода в программу численных значений. Он входит в одну группу с элементом `ProgressBar` и находится в той же строке списка стандартных элементов окна **Компоненты**: **Microsoft Windows Common Controls – 6.0**.

Свойства элемент `Slider`:

- `Value` — задает или возвращает установленное бегунком значение;
- `LargeChange` — задает величину изменения показаний бегунка при щелчках на нем мышью или нажатии клавиш <PageUp> и <PageDown>;
- `SmallChange` — задает величину изменения показаний бегунка при нажатии клавиш <<> и <>>;
- `Max`, `Min` — определяют диапазон значений бегунка;
- `BorderStyle` — определяет наличие вокруг элемента рамки;
- `Orientation` — определяет горизонтальную или вертикальную ориентацию элемента;
- `TickStyle` — определяет внешний вид элемента, расположение меток;
- `TickFrequency` — определяет частоту делений на линейке элемента.

Из событий чаще других используются `Change` и `Click`.

Пример 18. *Slider*. Приложение "Агрегатные состояния воды"

Приложение состоит из формы, двух меток и элемента `Slider`. Размещение элементов в форме показано на рис. 18.24. Форме присвоено имя `frmСостоянияВоды`, и `Caption` — `Агрегатные состояния воды`. Меткам и бегунку оставлены имена по умолчанию: `Label1`, `Label2` и `Slider1`. У меток значения свойства `Alignment` равны 2 (Центровка), а `BorderStyle` — 1. Свойствам бегунка значения присваиваются в процедуре загрузки формы. Код приложения приведен в листинге 18.16.

Рис. 18.24. Приложение с элементом `Slider`

Листинг 18.16. Код приложения "Агрегатные состояния воды"

```
Dim t As Integer
Private Sub Form_Load()
  Slider1.Value = 50
  Slider1.LargeChange = 10
  Slider1.SmallChange = 5
End Sub
Private Sub Slider1_Change()
  t = Slider1.Value
  Label1.Caption = "t = " & t & " град"
  Select Case t
    Case Is < 1
      Label2.Caption = "Вода - лед"
    Case 0 To 99
      Label2.Caption = "Вода - жидкость"
```

```
    Case Is > 99
      Label2.Caption = "Вода - лед"
  End Select
End Sub
```

После отладки приложения проверьте задание значений элемента "бегунок" с помощью мыши, клавиш <PageUp> и <PageDown>, клавиш <<> и <>>.

18.19. Линия *(Line)*

Элемент `Line` (линия) — последний графический элемент панели управляющих элементов. Он позволяет вывести в форму графические элементы `PictureBox` и `Image` линии заданной ширины и заданного типа.

Свойства элемента `Line`:

- `X1`, `Y1` и `X2`, `Y2` — координаты соответственно начала и конца линии;
- `BorderWidth` — ширина линии;
- `BorderStyle` — стиль линии (0 — прозрачная, 1 — сплошная, 2 — штриховая, 3 — пунктирная и т. д.);
- `BorderColor` — цвет линии.

Создайте EXE-проект, разместите в форме элементы `PictureBox`, `Image` и нарисуйте в форме и в этих элементах линии разной ширины и различного типа и цвета.

Глава 19

Создание Windows-приложений

19.1. Разработка интерфейса приложения

Работа по созданию нового приложения начинается с постановки задачи, уточнения функций приложения, учета пожеланий пользователя. Выполняется поиск и изучение прототипов — приложений, близких по требованиям к создаваемому проекту. Чем тщательнее выполняется работа на начальных этапах, тем успешнее, без многочисленных возвратов, доработок, переделок и настроек проходит дальнейшая работа над приложением.

Рекомендуется начинать разработку внешнего вида форм приложения с эскизов на бумаге. Количество и расположение элементов управления играют существенную роль в простоте и удобстве работы с приложением. Перенасыщенность формы элементами создает, особенно у начинающего пользователя, мнение о сложности освоения работы с приложением, рассеивает его внимание и затрудняет работу.

Интерфейс приложения должен быть прост и понятен. Просмотр окна пользователем идет, как правило, сверху вниз и слева направо. Это определяет и расположение элементов на форме. Например, элементы, завершающие работу с формой (кнопки **OK**, **Далее**, **Выход** и т. п.), удобно располагать в правом нижнем углу формы.

Функционально и логически связанные элементы желательно группировать, располагать в наиболее выгодных для совместного использования местах. Нередко удобно визуализировать группы элементов по мере необходимости, используя меню или командные кнопки.

Размеры элементов, их цвет, используемые шрифты и другие характеристики желательно выбирать наиболее привычные, например, используемые в приложениях фирмы Microsoft. Обычно рекомендуется использовать в приложении стандартные шрифты, например, Arial New или Times Roman, не более двух или трех размеров.

Некоторые элементы дизайна приложения должны быть ориентированы на предупреждение и исключение ошибок пользователя. К ним относятся различные уместные по ситуациям всплывающие подсказки, контекстные ме-

ню, другая информация. Важную роль может играть простая справочная система, вызываемая командами меню.

Привлечение пользователя на всех этапах разработки к тестированию приложения — лучший способ проверки простоты и удобства работы с создаваемым приложением.

19.2. Области и время действия переменных

При знакомстве в предыдущей главе с управляющими элементами строились простые приложения. При этом некоторые важные моменты не оговаривались, чтобы не затенять изучение общих принципов построения. Это относится, в частности, к определению области и времени действия переменных.

Области действия переменных определяют области их видимости. Эти области связаны со структурой проектов, которая состоит из процедур, модулей (контейнеров, общих процедур) и всего проекта в целом. Эти три ступени определили деление переменных на следующие типы:

- *Локальные переменные* — это переменные уровня процедуры, в которой они определены. Определяются локальные переменные в начале процедуры при помощи инструкций `Dim` и `Static`. Они распознаются и действуют только в своей процедуре. Локальные переменные с одинаковыми именами из разных процедур и, тем более, из разных модулей — это разные переменные.
- *Контейнерные переменные, переменные уровня модуля* — доступны только в модуле или форме, в которых они определены. Определяются контейнерные переменные в секции `General` кода при помощи инструкций `Dim` или `Private`. Контейнерные переменные с одинаковыми именами, определенные таким образом в разных модулях или формах, — это разные переменные.
- *Глобальные переменные* — доступны во всех процедурах проекта. Определяются в секции `General` кода одного из модулей с помощью инструкции `Public`.
- *Статические переменные.* Кроме области видимости переменных используется понятие времени их жизни. Переменная сохраняет и изменяет свои значения, пока выполняется процедура, в которой она определена. При выходе из процедуры переменные удаляются из памяти. Чтобы сохранять значение переменной в памяти, их объявляют как статические. Переменная становится статической при ее определении с помощью инструкции `Static`. При этом ее значение сохраняется все время работы приложения.

Пример 1. Области и время действия переменных

Создадим приложение, демонстрирующее работу локальных, контейнерных и статических переменных. Приложение состоит из двух форм: `Form1` и `Form2`

(рис. 19.1, 19.2). Первая форма вводится в проект с началом создания EXE-приложения, вторая форма добавляется командой **Проект | Добавить форму**. При этом в проводнике проекта добавляется модуль `Form2` (рис. 19.3).

Каждая форма будет иметь свой код. Напомним, что на этапе конструирования кнопки проводника проекта упрощают переходы от формы к форме и от кода к коду. Поэтому желательно вывести окно **Проводник проекта** на экран.

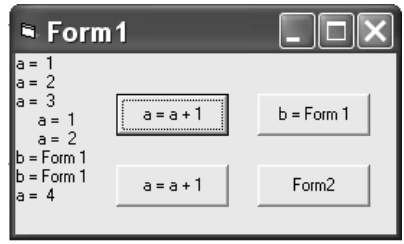


Рис. 19.1. Первая форма к примеру 1

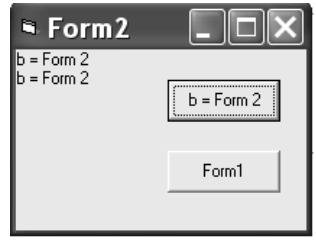


Рис. 19.2. Вторая форма к примеру 1

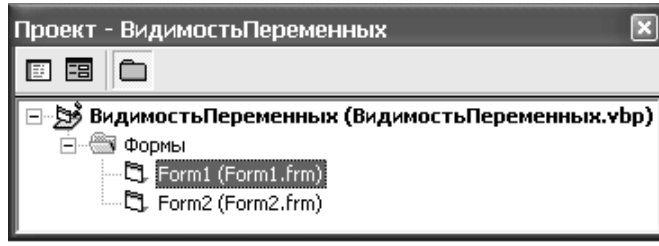


Рис. 19.3. Проводник проекта к примеру 1

Код модуля *Form1*

Код модуля `Form1` состоит из процедур обработки события `Click` для каждой из четырех кнопок формы (листинг 19.1).

Первые две процедуры работают со статическими, локальными переменными с одинаковыми именами a. Каждая из них обрабатывает событие Click своей кнопки с надписью a=a+1. Нажатие на кнопки увеличивает на единицу значения своих переменных. Эти значения печатаются в форме. Различие значений доказывает независимость одноименных локальных переменных.

Третья процедура работает с контейнерной переменной b. Нажатие на третью кнопку присваивает этой переменной значение "b = Form 1" и выводит это значение в форму.

Четвертая кнопка выводит на экран форму Form2. Для этого используется метод Show.

Листинг 19.1. Код модуля Form1 проекта "Видимость переменных"

```
Dim b As String
Private Sub Command1_Click()
  Static a As Integer
  a = a + 1
  Print "a = "; a
End Sub
Private Sub Command2_Click()
  Static a As Integer
  a = a + 1
  Print "     a = "; a
End Sub
Private Sub Command3_Click()
  b = "Form 1"
  Print "b = "; b
End Sub
Private Sub Command4_Click()
  Form2.Show
End Sub
```

Код модуля *Form2*

Код модуля Form2 состоит из двух процедур обработки событий Click кнопок формы (см. листинг 19.2). Первая процедура присваивает контейнерной для модуля Form2 переменной b значение "b = Form 2". После присвоения значение переменной выводится в форму. Различие значений одноименных контейнерных переменных разных форм показывает, что это разные переменные. Вторая процедура выводит на передний план экрана форму Form1.

Листинг 19.2. Код модуля Form2 проекта "Видимость переменных"

```
Dim b As String
Private Sub Command1_Click()
  b = "Form 2"
  Print "b = "; b
End Sub
Private Sub Command2_Click()
  Form1.Show
End Sub
```

Рассмотренные примеры позволяют сделать следующие выводы. Во-первых, как локальные (определяемые в разных процедурах), так и контейнерные (определяемые в разных модулях) переменные проекта могут иметь одинаковые имена. Во-вторых, при желании сохранять значение переменной все время работы приложения (например, показания счетчика), ее следует определить как статическую переменную.

19.3. Создание меню

Меню является одним из важных элементов большинства приложений и служит для быстрого выбора команд, объектов приложений и т. п. Меню может состоять из нескольких уровней. Выбор меню первого уровня открывает меню второго уровня, выбор пункта которого открывает меню третьего уровня и т. д. Меню, как и другие объекты приложений, обладает свойствами.

Основные свойства меню:

- `Name` — начинается с префикса `mnu` и должно быть единственным в проекте;
- `Caption` — это название пункта меню; позволяет постановкой символа `&` использовать быстрые клавиши для выбора пунктов меню;
- `Enabled` — разрешает или запрещает выполнение команды меню;
- `Visible` — определяет видимость или невидимость пункта меню.

Пример 2. Приложение с меню и списком

Проект содержит одну форму (`Name` (имя) — `frmРекиГоры`, `Caption` — `Реки и Горы Мира`). В форме создано двухуровневое меню (`Caption` — `Части Света`) и список (`Name` — `List1`).

Список заполняется в процессе исполнения кода при выборе пунктов меню второго уровня названиями трех самых длинных рек или самых высоких гор выбранной части света (см. рис. 19.4 и листинг 19.3).

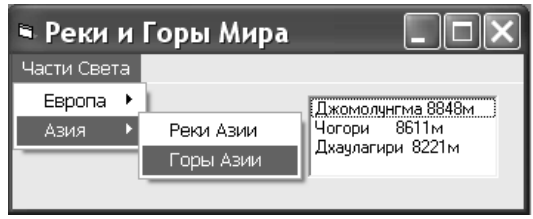


Рис. 19.4. Окно приложения "Реки и Горы Мира"

Для создания меню выполним следующие действия.

1. Нажмем на главной панели инструментов кнопку **Редактор Меню**. Открывается диалоговое окно **Редактор Меню**. На рис. 19.5 это окно показано на завершающем этапе построения меню.

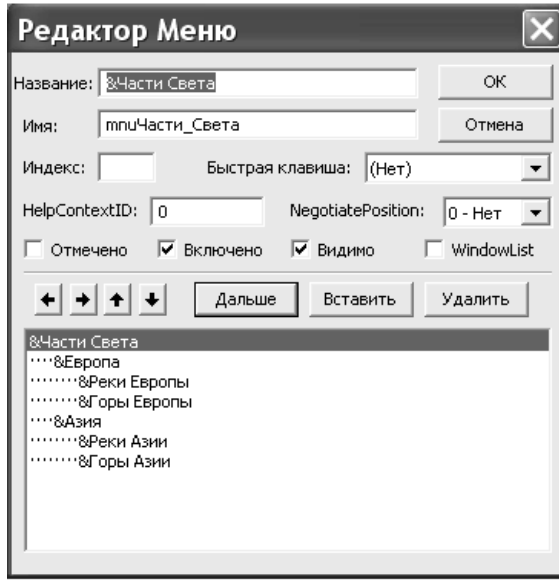


Рис. 19.5. Диалоговое окно **Редактор Меню** с меню приложения "Реки и Горы Мира"

2. В текстовом поле ввода **Название** впишем текст `&Части Света`. Переход к следующему текстовому окну удобно выполнять клавишей <Tab>.
3. В текстовом поле ввода **Имя** впишем текст `mnuЧасти_Света` и нажмем кнопку **Дальше**. С помощью кнопки **Дальше** выполняются переходы от одного пункта меню к другому.

4. Один щелчок в диалоговом окне **Редактор Меню** на стрелках, направленных вправо или влево, смещает вправо или влево следующий пункт меню. Это смещение определяет уровень пункта меню. Последовательно впишем названия и имена всех пунктов меню.
5. Вернемся к каждому пункту меню и выберем символы для определения быстрых клавиш. Это делается постановкой перед выбранными символами знака &. По завершении создания меню нажмем кнопку **OK**.

При работе с приложением быстрые клавиши будут работать в комбинации с управляющей клавишей <Alt>, например: <Alt>+<ч> или <Alt>+<Ч>.

Щелчок на пункте меню в процессе разработки вызывает начальную и конечную строку процедуры для записи инструкций обработки этого события.

Листинг 19.3. Код приложения "Реки и Горы Мира"

```
Private Sub mnuРеки_Европы_Click()
  List1.Clear
  List1.AddItem "Волга  3531км"
  List1.AddItem "Дунай  2857км"
  List1.AddItem "Урал   2428км"
End Sub
Private Sub mnuГоры_Европы_Click()
  List1.Clear
  List1.AddItem "Эльбрус 5642м"
  List1.AddItem "Казбек  5033м"
  List1.AddItem "Монблан 4807м"
End Sub
Private Sub mnuРеки_Азии_Click()
  List1.Clear
  List1.AddItem "Янцзы  6300км"
  List1.AddItem "Хуанхэ 4670км"
  List1.AddItem "Меконг 4500км"
End Sub
Private Sub mnuГоры_Азии_Click()
  List1.Clear
  List1.AddItem "Джомолунгма 8848м"
  List1.AddItem "Чогори      8611м"
  List1.AddItem "Дхаулагири  8221м"
End Sub
```

При работе с приложением обратите внимание на возможность выбора пунктов меню клавишами со стрелками и клавишей <Enter>.

19.4. Создание MDI-приложений

Различают три вида Windows-приложений, разрабатываемых с помощью Visual Basic 6.0.

- Однодокументные приложения — SDI (Single-Document Interface). Создаются на основе, как правило, одной формы, в которую может вызываться для работы только один документ. Это наиболее простые приложения.
- Многодокументные приложения — MDI (Multiple-Document Interface). Строятся на нескольких формах, одна из которых (родительская, главная) служит в качестве окна для размещения остальных форм (дочерних, подчиненных). При этом дочерние окна (документы) могут разворачиваться, сворачиваться, располагаться каскадом только в пределах родительского окна. Примером MDI-приложения может служить текстовый редактор Word.
- Приложения типа проводник (Explorer) можно увидеть, раскрыв окно Проводника системы Windows. Такие приложения создаются на основе элементов управления, не все из которых рассматриваются в учебнике.

Пример 3. MDI-приложение "Круги и Квадраты"

Создадим MDI-приложение с двумя дочерними формами. Формы имеют по одной кнопке. Нажатие кнопки на первой форме выводит разноцветные круги, на второй — жонглирует разноцветными квадратами.

1. Создадим новый проект командами **Файл | Новый Проект | Стандартный EXE | OK**.
2. В проекте вначале имеется только одна форма — `Form1`. Присвойте значения ее свойствам: `Name = frmКруги` и `Caption = Дочерняя форма Круги`.
3. Командами главного меню **Проект | Добавить форму** дополните проект еще одной дочерней формой — `Form2`. Присвойте ее свойствам значения: `Name = frmКвадраты` и `Caption = Дочерняя форма Квадраты`.
4. Командами главного меню **Проект | Добавить MDI-форму** введите в проект родительскую форму. Присвойте ее свойствам значения: `Name = frmКругиКвадраты` и `Caption = Родительская форма`.
5. Присвойте проекту имя и выполните действия, которые позволят при запуске приложения первой выводить на экран родительскую форму. Для этого командой **Проект | Проект1 Свойства** откройте окно **Проект1 –**

Свойства проекта. В текстовое поле ввода **Имя проекта** впишите текст `КругиКвадратыMDI`, а в списке **Объект запуска** выберите строку материнской формы `frmКругиКвадраты`. Закрепите выбор кнопкой **OK**.

6. Последовательно выделяя дочерние формы, присвойте их свойству `MDIChild` значения True. Выделять формы удобно двойным щелчком мышью на их именах в окне **Проводник проекта**. При смене значения свойства `MDIChild` картинки перед именами дочерних форм в Проводнике проекта изменятся на более понятные, указывающие на подчиненность дочерних форм родительской (рис. 19.6).

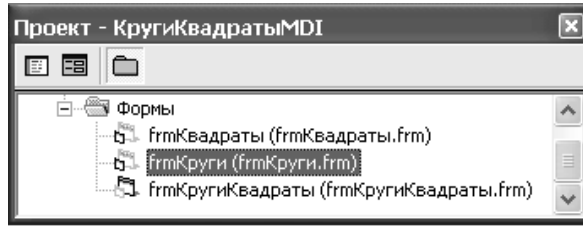


Рис. 19.6. Проводник проекта `КругиКвадратыMDI`

7. Создайте в родительской форме меню для вызова на экран дочерних форм. Для этого выделите родительскую форму. Командой главного меню **Инструменты | Редактор Меню** откройте окно редактора и создайте меню **Выбор фигуры**, второй уровень которого состоит из двух пунктов: **Круги** и **Квадраты**.
8. Поместите на дочерние формы по одной кнопке (рис. 19.7). Кнопкам форм дайте одинаковые имена и названия: `Name =cmdПоказать` и `Caption = Показать`.

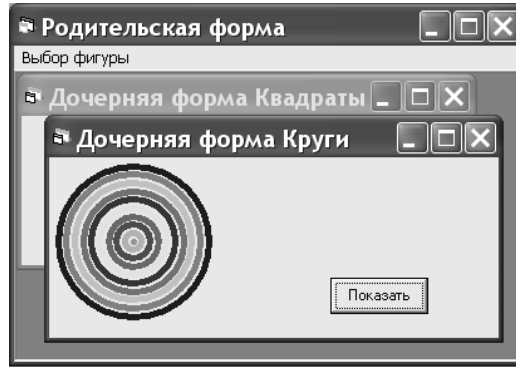


Рис. 19.7. Каскад дочерних форм в материнской форме

9. Проверьте работу меню и запишите коды для всех трех форм. Запись кода начинается с щелчка на пункте меню.

Запись кода

Команды меню родительской формы определяют процедуры вывода в родительскую форму дочерних форм. Щелчки на кнопках **Показать** дочерних форм записывают начальную и конечную строки процедур обработки событий `Click`. Заполните процедуры показанными в листингах 19.4—19.6 инструкциями и проверьте работу MDI-приложения.

Листинг 19.4. Код родительской формы

```
Private Sub mnuКвадрат_Click()
  frmКвадраты.Show
End Sub
Private Sub mnuКруги_Click()
  frmКруги.Show
End Sub
```

Листинг 19.5. Код дочерней формы `Круги`

```
Private Sub cmdПоказать_Click()
  Static r, c
  r = (r + 100) Mod 1000
  c = (c + 1) Mod 16
  DrawWidth = 5
  Circle (1000, 1000), r, QBColor(c)
End Sub
```

Листинг 19.6. Код дочерней формы `Квадраты`

```
Private Sub cmdПоказать_Click()
  Static k, c
  Cls
  k = (k + 1) Mod 2
  c = (c + 1) Mod 16
  If k = 0 Then
    Line (500, 500)-(1500, 1500), QBColor(c), BF
```

```
  Else
    Line (2000, 500)-(3000, 1500), QBColor(c), BF
  End If
End Sub
```

19.5. Работа с мышью

Если основным устройством ввода информации и управления приложениями служит клавиатура, то манипулятор "мышь" является вторым по важности устройством ввода и управления.

Основные события, связанные с манипулятором мышь, такие:

- `Click` — одиночный щелчок мыши;
- `DblClick` — двойной щелчок мыши;
- `MouseDown` — нажатие кнопки мыши;
- `MouseUp` — отпускание кнопки мыши;
- `MouseMove` — перемещение указателя мыши.

Все эти события имеются в списке событий окна редактирования кода. При выделенной форме щелчок на строке выбранного события инициирует запись в код граничных строк процедуры обработки этого события. Заголовки процедур содержат параметры, записанные в скобках после названия события, например:

```
Private Sub Form_MouseDown (Button As Integer, Shift As Integer, X As Single, Y As Single)
```

Рассмотрим эти параметры (в скобках они выделены) на примерах.

Пример 4. Мышь и параметр *Button*

Параметр `Button` содержит информацию о номере нажатой кнопки: 1 — левая (`vbLeftButton`), 2 — правая (`vbRightButton`) и 4 — средняя (`vbMiddleButton`).

Приступите к созданию нового проекта. Выберите в списке объектов окна кода объект `Form1`, а в списке событий — событие `MouseDown`.

Следующие два кода (листинги 19.7, 19.8) демонстрируют работу параметра `Button`. Первый код использует оператор `Select Case` и функцию вывода сообщений `MsgBox` (рис. 19.8, листинг 19.7). Окно сообщений `MsgBox` имеет много различных вариантов внешнего вида и часто используется в приложениях для вывода подсказок, предупреждений, другой информации. Подробнее с ним можно познакомиться в справочной системе. Для этого можно выделить в коде приложения слово `MsgBox` и нажать клавишу <F1>.

Листинг 19.7. Код демонстрации нажатия кнопок мыши

```
Private Sub Form_MouseDown(Button As Integer, Shift As Integer,
X As Single, Y As Single)
  Select Case Button
    Case vbLeftButton                    'или Case 1
      MsgBox ("Нажата левая кнопка")
    Case vbRightButton                   'или Case 2
      MsgBox ("Нажата правая кнопка")
  End Select
End Sub
```

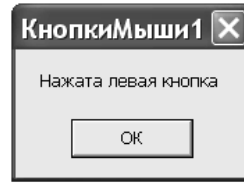


Рис. 19.8. Информационное окно `MsgBox`

В листинге 19.8 приведен код, использующий однострочный оператор `If...Then`, и метод `Print`, служащий для вывода сообщений (рис. 19.9).

Листинг 19.8. Код приложения "Кнопки мыши 2"

```
Private Sub Form_MouseDown(Button As Integer, Shift As Integer,
X As Single, Y As Single)
  If Button = 1 Then Print "1 - нажата левая кнопка"
  If Button = 2 Then Print "2 - нажата правая кнопка"
  If Button = 4 Then Print "4 - нажата средняя кнопка"
End Sub
```

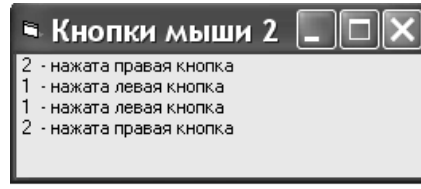


Рис. 19.9. Окно приложения "Кнопки мыши 2"

Пример 5. Мышь и параметр *Shift*

Параметр `Shift` определяет, были ли во время нажатия кнопки мыши нажаты клавиши <Shift>, <Ctrl> и <Alt> или их комбинации. При этом управляющие клавиши выступают в качестве ключа при подаче команд на выполнение ответственных действий. Использование совместно с кнопкой мыши клавиш и их комбинаций значительно увеличивает общее количество возможных команд.

Параметр `Shift` может принимать следующие значения:

- 0 нет нажатых клавиш;
- 1 нажата клавиша <Shift>;
- 2 нажата клавиша <Ctrl>;
- 3 нажаты клавиши <Shift> и <Ctrl>;
- 4 нажата клавиша <Alt>;
- 5 нажаты клавиши <Shift> и <Alt>;
- 6 нажаты клавиши <Ctrl> и <Alt>;
- 7 нажаты клавиши <Shift>, <Ctrl> и <Alt>.

Приступите к созданию нового проекта. Выберите в списке объектов окна кода объект `Form1`, а в списке событий — событие `MouseDown`. Наберите код приложения, показанный в листинге 19.9.

Листинг 19.9. Код демонстрации нажатия клавиш клавиатуры

```
Private Sub Form_Load()
  Show
End Sub
Private Sub Form_MouseDown(Button As Integer, Shift As Integer,
X As Single, Y As Single)
  Select Case Shift
    Case 0: Print Shift; " - нет нажатых клавиш"
    Case 1: Print Shift; " - нажата клавиша Shift"
    Case 2: Print Shift; " - нажата клавиша Ctrl"
    Case 3: Print Shift; " - нажаты клавиши Shift и Ctrl"
    Case 4: Print Shift; " - нажата клавиша Alt"
    Case 5: Print Shift; " - нажаты клавиши Shift и Alt"
    Case 6: Print Shift; " - нажаты клавиши Ctrl и Alt"
    Case 7: Print Shift; " - нажаты клавиши Shift, Ctrl и Alt"
  End Select
End Sub
```

Пример 6. Мышь и координаты ее указателя

Параметры X и Y определяют координаты указателя мыши в форме или элементе управления.

Следующее приложение (его код приведен в листинге 19.10) демонстрирует работу событий MouseDown и MouseUp и параметров X и Y

При нажатии кнопки мыши ее указатель меняется, принимая вид большого крестика. При отпускании кнопки мыши указатель мыши опять принимает форму стрелки; рисуется окружность небольшого диаметра с точкой в центре и печатаются координаты точки отпускания (рис. 19.10).

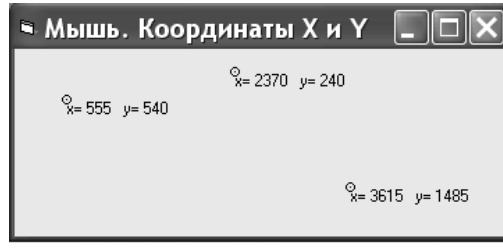


Рис. 19.10. Демонстрация параметров мыши X и Y

Листинг 19.10. Код обработки событий MouseDown и MouseUp

```
Private Sub Form_MouseDown(Button As Integer, Shift As Integer,
_X As Single, Y As Single)
  MousePointer = vbCrosshair
End Sub
Private Sub Form_MouseUp(Button As Integer, Shift As Integer,
_X As Single, Y As Single)
  MousePointer = vbDefault
  Pset(X,Y): Circle (X, Y), 50
  Print "  x="; X; "  y="; Y
End Sub
```

Пример 7. Мышь — примитивный чертежник

В следующем простом приложении мышь с нажатой левой кнопкой чертит синим цветом, а с нажатой правой кнопкой — красным. При этом используется событие MouseMove. Код приложения приведен в листинге 19.11.

Листинг 19.11. Код приложения "Мышь — чертежник"

```
Private Sub Form_MouseMove(Button As Integer, Shift As Integer,
_X As Single, Y As Single)
  DrawWidth = 2
  If Button = 1 Then
    PSet (X, Y), vbBlue
  End If
  If Button = 2 Then
    PSet (X, Y), vbRed
  End If
End Sub
```

19.6. Работа с графикой

Пример 8. Графический редактор

Набор событий мыши позволяет создать простой графический редактор, в котором возможны: выбор цвета и ширины кисти, очистка графического поля и сохранение созданного рисунка.

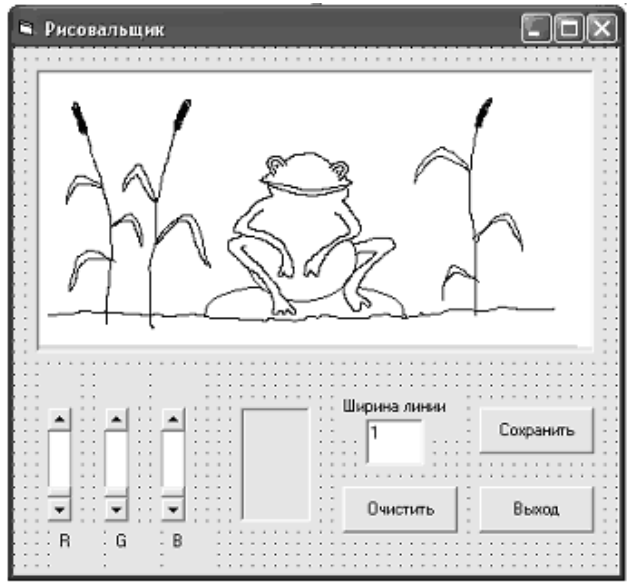


Рис. 19.11. Окно графического редактора

Внешний вид окна редактора показан на рис. 19.11. Для его создания потребовались: одна форма, два графических поля `PictureBox`, три элемента `VScrollBar`, шесть элементов `Label`, одно текстовое поле `TextBox` и три кнопки.

В коде программы (листинг 19.12) для организации процесса рисования используются три рассмотренных выше связанных с мышью события: `MouseDown`, `MouseMove` и `MouseUp`.

Интересными моментами при разработке и записи кода данного проекта можно считать:

- запись в файл полученного в окне редактора рисунка;
- использование при выборе цвета линий массива элементов `VScrollBar`, что существенно сокращает длину кода;
- использование графического метода `Line` с постановкой начальной точки методом `Pset` в целях исключения разрывов в линиях при рисовании.

Листинг 19.12. Код графического редактора

```
Dim R, G, B, s
Dim d As Integer
'Процедура записи рисунка в файл
Private Sub Command1_Click()
  SavePicture Picture1.Image, "C:\Мои документы\Мои рисунки\Рисунки
↳приложений\Рисовальщик.bmp"
End Sub
Private Sub Command2_Click()
  End                              'завершение работы
End Sub
  Private Sub Command3_Click()
  Picture1.Cls                     'удаление рисунка
End Sub
Private Sub Picture1_MouseDown(Button As Integer, Shift As Integer,
↳X As Single, Y As Single)
  Picture1.PSet (X, Y)             'точка для последующей работы метода
Line
  s = 1                            'разрешение рисования
End Sub
Private Sub Picture1_MouseMove(Button As Integer, Shift As Integer,
↳X As Single, Y As Single)
```

```
  d = Val(Text1.Text) 'ширина линии рисования
  Picture1.DrawWidth = d
  If s = 1 Then Picture1.Line -(X, Y), RGB(R, G, B)
                      'рисование
End Sub
Private Sub Picture1_MouseUp(Button As Integer, Shift As Integer, X
↳As Single, Y As Single)
  s = 0               'запрет рисования при отпускании кнопки
End Sub
                      'Ниже записана процедура выбора цвета линий
Private Sub VScroll1_Change(Index As Integer)
  Select Case Index
    Case 0:  Label2.Caption = VScroll1(0).Value: R = VScroll1(0).Value
    Case 1:  Label3.Caption = VScroll1(1).Value: G = VScroll1(1).Value
    Case 2:  Label4.Caption = VScroll1(2).Value: B = VScroll1(2).Value
  End Select
  Picture2.BackColor = RGB(R, G, B)
End Sub
```

Пример 9. Графика и метод Монте-Карло

Создадим приложение для демонстрации использования метода Монте-Карло для определения площади фигуры сложной конфигурации.

Приложение (рис. 19.12 и 19.13) содержит по одному элементу `Picture`, `Label`, `TextBox` и `Timer` и две кнопки: **Пуск** и **Выход**.

Идея применения метода Монте-Карло заключается в следующем. Берут объект с известной площадью (`P`) — например, квадрат или прямоугольник. В этом объекте размещают фигуру, площадь которой (`X`) требуется определить, и набрасывают в объект большое количество (`N`) точек со случайными координатами. При набрасывании подсчитывают количество (`T`) точек, "упавших" в границах фигуры. При хорошем генераторе случайных чисел и достаточно большом числе `N` точки равномерно распределяются по площади прямоугольника и позволяют записать равенство `X / T=P / N`, откуда можно получить сооотношение: `X = P*T / N`.

Для удобства расчетов с помощью свойства `Scale` в окне элемента `Picture` установлена новая система координат: `Scale (x1,y1)-(x2,y2)`, где в скобках записаны координаты верхнего левого и нижнего правого углов графического поля. Выбор значений координат (листинг 19.14) удобен для выполнения вычислений. Рисунок следует нарисовать или выбрать заранее. Как правило, для демонстрации метода в качестве фигуры выбирают круг, опре-

деляют его площадь и, зная радиус круга, определяют число π и сравнивают его с известным значением 3,14.

Рис. 19.12. Окно приложения перед заполнением точками

Рис. 19.13. Окно приложения с результатом работы

В рассматриваемом примере в графическом редакторе Paint была нарисована и сохранена в памяти зеленая елочка. Для записи кода программы нужно

знать номер зеленого цвета, которым изображена елка. Этот номер можно получить в графическом редакторе Paint или в начале работы с рисунком с помощью кода, приведенного в листинге 19.13.

Листинг 19.13. Код определения цвета елочки

```
Private Sub Form_Load()
  Show
       'для визуализации результатов печати
  Picture1.Scale (0, 0)-(100, 100)          'установка системы координат
  Picture1.Picture = LoadPicture("E:\Documents and Settings\Александр\
↳Мои документы\Мои рисунки\Елочка.bmp")
  Picture1.Circle (45, 55), 3       'для проверки попадания на зелень елки
  Print Picture1.Point(45, 55)      'печать номера цвета (32768)
End Sub
```

Напомним, что метод `Point(x, y)` возвращает номер цвета в точке с координатами `x` и `y`. Показанный выше фрагмент кода можно после удаления лишних строк использовать в коде приложения, демонстрирующего применение метода Монте-Карло (листинг 19.14).

Листинг 19.14. Код демонстрации метода Монте-Карло

```
Dim k As Long
Private Sub cmdПуск_Click()
  Randomize Timer
For i = 1 To Val(Text1.Text)      'число циклов = числу точек
  x = Rnd * 100: y = Rnd * 100    'выработка случайных координат
    r = Int(Rnd * 30 + 5): c = Int(Rnd * 16)
    If Picture1.Point(x, y) = 32768 Then  'проверка попадания на елку
      k = k + 1                   'количество точек попало на елку
    Else
      Picture1.PSet (x, y)        'точки вне елки
    End If
  Next
  x = k * 10000 / Val(Text1.Text)   'вычисление площади
  lblИнфо.Caption = "Площадь рисунка елки в квадрате 100x100 равна:"
  Text1.Text = Str(x)               'вывод результата в TextBox
End Sub
```

```
Private Sub Form_Load()
  Picture1.Scale (0, 0)-(100, 100)  'установка системы координат
  Picture1.Picture = LoadPicture("E:\Documents and Settings\Александр\
↳Мои документы\Мои рисунки\Елочка.bmp")
  lblИнфо.Caption = "Сколько точек набросать в квадрат?"
End Sub
Private Sub cmdВыход_Click()
  End
End Sub
```

Пример 10. Новогодняя елка

Используя предыдущий проект, украсьте елку разноцветными шарами. Код нового проекта приведен в листинге 18.15. В нем изменена система координат на более удобную для задания случайного радиуса шаров. Изменены надписи у элемента `lblИнфо`.

На поверхность элемента `Picture1` набрасываются точки, пока число попаданий на елку не сравняется с задаваемым числом украшений. С центром в попавших на елку точках рисуются шарики случайного радиуса и цвета.

Листинг 18.15. Код украшения елки

```
Private Sub Form_Load()
  Picture1.Scale (0, 0)-(1000, 1000)       'смена системы координат
  Picture1.Picture = LoadPicture("E:\Documents and Settings\Александр\
↳Мои документы\Мои рисунки\Елочка.bmp")
  Picture1.FillStyle = 0                   'заливка краской шаров на елке
  lblИнфо.Caption = "Сколько украшений повесить на елку?"
End Sub
Private Sub cmdПуск_Click()
  Randomize Timer
  Do While k <= Val(Text1.Text)     'проверка: все ли шары нарисованы?
    x = Rnd * 1000: y = Rnd * 1000  'две строки выработки случайных
    r = Int(Rnd * 30 + 5): c = Int(Rnd * 16)      'значений параметров
    If Picture1.Point(x, y) = 33768 Then    'проверка попадания точки
      Picture1.FillColor = QBColor(c)       'цвет заливки
      Picture1.Circle (x, y), r, QBColor(c) 'рисование очередного шара
      k = k + 1                             'подсчет количества шаров
    End If
```

```
  Loop
  lblИнфо.Caption = "Елка украшена!"       'сообщение о конце работы
End Sub
Private Sub cmdВыход_Click()
  End
End Sub
```

19.7. Звуковое сопровождение

Музыку в приложениях Windows можно прослушивать в различных форматах, в частности — в формате файлов, имеющих расширения wav и mid.

Подайте команды **Пуск | Найти | Файлы и папки**. В окне **Результаты поиска** в текстовом поле **Искать имена файлов и папок** запишите *.wav и нажмите кнопку **Найти**. В результате поиска на экран будет выведен список файлов.

Щелчок на имени файла выводит на экран диалоговое окно **Проигрыватель Windows Media** с управляющими элементами запуска, останова, регулировки громкости и т. п. Проигрыватель позволяет прослушать выбранный файл.

Выберите, например, файл GLASS. Если щелкнуть на его имени правой кнопкой мыши и в контекстном меню выбрать пункт **Свойства**, то можно просмотреть его свойства и путь к нему. Если файл предполагается использовать, то этот путь необходимо запомнить или записать. Например, для файла GLASS это: C:\WINDOWS\MEDIA\Microsoft Office 2000\GLASS.wav.

Пример 11. Воспроизведение аудиофайлов

Запустите программу Visual Basic 6.0 и приступите к созданию нового проекта, позволяющего воспроизводить аудиофайлы. Задайте свойствам формы значения: `Name = frmЗвук`, `Caption = Воспроизведение звука`.

Откройте в главном меню окно **Проект**. Выберете строку **Компоненты**. В открывшемся списке поставьте галочку в начале строки **Microsoft Multimedia Control 6.0** и нажмите кнопку **OK**. В окне управляющих элементов при этом появится новый элемент — `MMControl1` (рис. 19.14). Обычным способом перенесите его в форму создаваемого приложения и присвойте ему имя `Плеер`, а свойству `DeviceType` присвойте значение `WaweAudio` (в коде).

Для простоты будем запускать `Плеер` при загрузке формы. Щелкнем на форме мышью и запишем код программы, приведенный в листинге 19.16.

Листинг 19.16. Код программы воспроизведения аудиофайлов

```
Private Sub Form_Load()
  Плеер.DeviceType = "WaveAudio"
```

```
  Плеер.FileName = "C:\windows\help\tour\audio\wav\wmpaud6.wav"
  Плеер.Command = "Open"
  Плеер.Command = "Play"
End Sub
Private Sub Form_Terminate()
  Плеер.Command = "Close"
End Sub
```

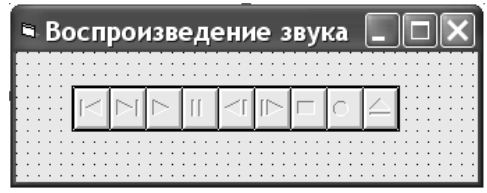


Рис. 19.14. Элемент `MMControl1` в форме

Первая строка процедуры загрузки формы, приведенной в листинге 19.16, задает тип устройства, обеспечивающего проигрывание мелодии из файла с расширением *.wav. Если выбран файл с расширением *.mid, то используется устройство "Sequencer".

Во второй строке указываются имя файла и путь к нему.

Третья строка открывает файл.

Четвертая строка начинает воспроизведение.

Последняя процедура с событием `Terminate` служит для закрытия файла. Чтобы это событие произошло, следует щелкнуть на кнопке **Закрыть формы**.

Пример 12. Озвучивание работы таймера

Нередко при создании проектов озвучивают результаты каких-либо действий пользователя или операторов программы. Например, можно озвучить моменты нажатия кнопок или моменты срабатывания таймера. В этих случаях элемент `MMControl` можно скрыть, присвоив его свойству `Visible` значение False.

Следующее приложение (листинг 19.17) содержит одну форму, таймер и элемент `MMControl` (рис. 19.15). Каждую секунду при срабатывании таймера форма меняет свой цвет. Это сопровождается звуковыми сигналами.

Имена проигрывателя, таймера и командной кнопки для экономии времени оставим заданными по умолчанию. Имя проекта и имя формы изменим соответственно на `Плеер2` и `frmПлеер2` (иначе при желании сохранить проект в

памяти будет накапливаться много файлов проектов и форм под одинаковыми названиями `Проект1` и `Form1`).

Рис. 19.15. Окно приложения "Таймер, Звук и Цвет"

Листинг 19.17. Код приложения "Таймер, Звук и Цвет"

```
Dim n As Integer
Private Sub Command1_Click()
Timer1.Interval = 1000
MMControl1.Visible = False
End Sub
Private Sub Timer1_Timer()
n = (n + 1) Mod 2
If n = 0 Then
frmПлеер2.BackColor = vbRed
MMControl1.FileName = "C:\Program Files\PLUS!\PINBALL\SOUND3"
Else
frmПлеер2.BackColor = vbBlue
MMControl1.FileName = "C:\Program Files\PLUS!\PINBALL\SOUND19"
End If
MMControl1.Command = "Open"
MMControl1.Command = "Sound"
MMControl1.Command = "Close"
End Sub
```

19.8. Процедуры-функции и процедуры

Кроме процедур обработки событий, определяющих событийный принцип работы Windows-приложений, важное место в языке Visual Basic 6.0 занимают общие процедуры. Под процедурами понимается часть программы, реа-

лизующая вспомогательный алгоритм и допускающая многократное обращение к ней из различных частей программы.

Процедуры помогают значительно упростить коды программ, в которых имеются повторяющиеся действия, возможно, с отличающимися параметрами. Это *процедуры-функции* `Function...End Function`, позволяющие возвращать определяемые в них значения функций, и *процедуры* `SubEnd Sub`, способные изменять значения нескольких переменных. Кроме процедур `Function` и `Sub` имеются процедуры `Property`, используемые для присвоения и чтения значений свойств.

Различают процедуры общего назначения на уровне проекта и закрытые процедуры. *Процедуры на уровне проекта* записываются в отдельных модулях, создаваемых командой главного меню **Проект | Добавить модуль**. Они доступны для всех процедур всех модулей проекта. *Закрытые процедуры* записываются в коде модуля (например, модуля формы) и доступны только для процедур своей формы. Процедуры на уровне проекта и закрытые процедуры по области действия можно сравнить с глобальными и контейнерными переменными.

Процедуры-функции *Function...End Function*

Visual Basic имеет множество функций, значения которых вычисляются в стандартных подпрограммах. Это стандартные арифметические, тригонометрические, строковые функции (`Sqr(x)`, `Log(x)`, `Sin(x)`, `Len(a)` и т. п.). Где бы в коде ни была записана функция, достаточно задать значение аргумента, чтобы было вычислено значение функции. Такой подход удобен и для функций, создаваемых в процессе работы с приложениями самим пользователем.

Синтаксис процедур `Function End Function` следующий:

```
[Public | Private] [Static] Function имя [(список)]
   [операторы]
   [Exit Function]
   имя = выражение
   [операторы]
End Function
```

Слово `Public` указывает на то, что процедура доступна всем модулям проекта. Слово `Private` ограничивает доступность процедуры только процедурами своего модуля. Слово `Static` указывает, что переменные являются локальными в функции и сохраняются между ее вызовами. Параметр *имя* объявляет имя функции. Параметр *список* — это список разделенных запятыми аргументов — формальных параметров, которым из основной программы

передаются и присваиваются значения аргументов — фактических параметров. При работе с процедурами должно соблюдаться взаимно однозначное соответствие между аргументами и параметрами по количеству, типам и порядку следования. Запись *имя = выражение* возвращает значение функции, присвоенное ее имени. В теле процедуры-функции можно использовать оператор `Exit Function`.

Пример 13. Процедура-функция — текст без пробелов

Такая строковая функция полезна при желании проверить текст на наличие в нем палиндрома, при автоматическом поиске палиндромов в большом тексте.

Палиндромон ("палиндром", "перевертень") — редкий и очень древний жанр стиха. Он записывается в одну строку и читается одинаково слева направо и справа налево. В старину палиндромонами украшали чашу: как ни верти — читается одно и то же. В наше время в ходу такие палиндромы: "А роза упала на лапу Азора" (Ф. Фет), "Яро закусала ренегата генерала сука Зоря", "Молебен о коне белом", "Осело колесо", "Искать такси", "Несун гнусен" и многие другие. Заметим, что палиндромами могут быть и числа.

Для исключения пробелов можно использовать алгоритм последовательного посимвольного просмотра текста и операцию склеивания (конкатенацию) всех символов текста кроме пробелов.

Для проверки текста на палиндром после удаления всех пробелов можно применить функцию `StrReverse` (изменяет порядок следования символов в строке на обратный) и сравнить тексты с помощью функции `StrComp`.

Visual Basic имеет функции, исключающие только лидирующие и замыкающие пробелы. Решим следующие задачи.

1. Построим функцию, исключающую все, в том числе и внутренние пробелы (листинг 19.18).
2. Создадим проект с общей закрытой процедурой-функцией, записывающей заданный текст без пробелов (листинг 19.19).
3. Создадим проект с общей открытой процедурой-функцией, записывающей текст без пробелов (листинги 19.20, 19.21 и 19.22).

Решение первой задачи потребует использования созданного кода в каждом месте модуля или проекта, где возникает необходимость исключить из строки пробелы.

Решение второй задачи позволяет использовать в модуле один и тот же исключающий пробелы код, к которому обращаются из разных мест модуля.

Решение третьей задачи позволяет использовать один и тот же исключающий пробелы код для всех модулей проекта.

Листинг 19.18. Код исключает из текста все пробелы

```
Private Sub Form_Load()
  Show
  x = InputBox("Введите текст с пробелами")
  y = ""
  For i = 1 To Len(x)
    z = Mid(x, i, 1)
    If z <> " " Then y = y & z
  Next
  Print "y= "; y
End Sub
```

Проверив работу программы, создадим закрытую процедуру-функцию с именем `БезПробелов`. При этом код записывается так, как показано в листинге 19.19.

Листинг 19.19. Код приложения с процедурой-функцией `БезПробелов`

```
Private Sub Form_Load()
  Show
  x = InputBox("Введите текст с пробелами")
  y = БезПробелов(x)
  Print "y= "; y
End Sub
Function БезПробелов(t)
  s = ""
  For i = 1 To Len(t)
    z = Mid(t, i, 1)
    If z <> " " Then s = s & z
  Next
  БезПробелов = s
End Function
```

Из любой процедуры модуля, в котором находится закрытая процедура `БезПробелов`, по запросу вида `<переменная> = БезПробелов (<строка>)`

управление передается процедуре-функции `БезПробелов`. Там из строки удаляются все пробелы, после чего строка возвращается в процедуру запроса и присваивается переменной `y`.

Последний вариант приложения строится на двух формах `Form1` и `Form2` и одном модуле общей открытой процедуры (рис. 19.16, 19.17 и 19.18, листинги 19.20, 19.21 и 19.22). Такая структура проекта позволяет решить поставленную задачу — показать создание и работу общей открытой процедуры. Вторая форма и модуль добавляются в проект командами **Проект | Добавить форму** и **Проект | Добавить модуль**.

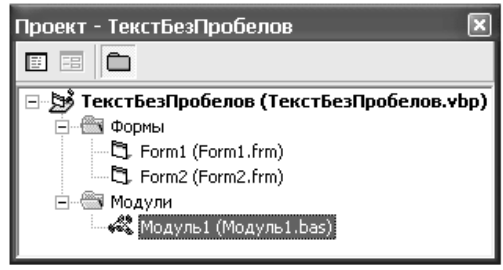


Рис. 19.16. Проводник проекта "ТекстБезПробелов"

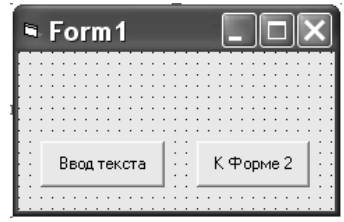


Рис. 19.17. Первая форма проекта

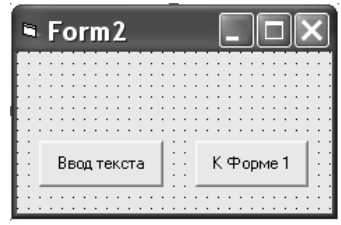


Рис. 19.18. Вторая форма проекта

Листинг 19.20. Код первой формы с обращением к общей процедуре-функции

```
Private Sub Form_Load()
  Show
End Sub
Private Sub Command1_Click()
  Cls
  x = InputBox("Текст с пробелами")
  y = БезПробелов(x)
  Print "x=  "; x
  Print "y=  "; y
End Sub
Private Sub Command2_Click()
  Form2.Visible = True
  Form1.Visible = False
End Sub
```

Листинг 19.21. Код второй формы с обращением к общей процедуре-функции

```
Private Sub Form_Load()
  Show
End Sub
Private Sub Command1_Click()
  Cls
  g = InputBox("Введите текст с пробелами")
  h = БезПробелов(g)
  Print "g=  "; g
  Print "h=  "; h
End Sub
Private Sub Command2_Click()
  Form1.Visible = True
  Form2.Visible = False
End Sub
```

Листинг 19.22. Код общей процедуры-функции

```
Public Function БезПробелов(z)
  w = ""
  For i = 1 To Len(z)
```

```
    v = Mid(z, i, 1)
    If v <> " " Then w = w & v
  Next
  БезПробелов = w
End Function
```

Процедуры *Sub...End Sub*

Синтаксис процедуры `Sub...End Sub` такой:

```
[Private | Public] [Static] Sub имя [(список)]
   [операторы]
   [Exit Sub]
   [операторы]
End Sub
```

Слова `Private`, `Public` и `Static` играют ту же роль, что и в процедуре-функции. Параметр *имя* — глобальное имя процедуры, ограниченное длиной в 40 символов. Параметр *список* — список разделенных запятыми имен переменных, передаваемых процедуре при ее вызове. В операторе предусмотрен альтернативный выход с помощью оператора `Exit Sub`. В отличие от процедуры-функции `Function` имя процедуры `Sub` не может быть использовано в выражениях. Процедуры могут быть рекурсивными, т. е. могут вызывать сами себя.

Вызов процедуры `Sub...End Sub` допускается двумя способами: с оператором `Call` и без него. Оператор `Call` передает управление процедуре. Вызов процедуры имеет два вида синтаксиса.

Синтаксис 1:

```
CALL имя процедуры [(список аргументов)]
```

Синтаксис 2:

```
имя процедуры [список аргументов]
```

Пример 14. Музыка в кнопках

Visual Basic 6.0 позволяет сопровождать работу пользователя с приложениями музыкальными эффектами. Например, можно организовать сопровождение звуками моментов нажатия кнопок.

Создадим простое приложение с одной формой (`frmМузыка_в_кнопках`), тремя кнопками (`cmdКруг`, `cmdКвадрат` и `cmdТреугольник`) и элементом `Microsoft Multimedia Control 6.0` — сокращенно `MMControl`. В программе приложения будем использовать процедуру `Sub...End Sub`.

На рис. 19.19 показан внешний вид формы с расположенными в ней элементами управления.

Рис. 19.19. Элементы приложения "Музыка в кнопках"

Напомним, что элемент `MMControl` обычно отсутствует в окне элементов управления. Поместить его в окне можно командами **Проект | Компоненты** главного меню. В раскрывшемся диалоговом окне **Компоненты** на вкладке **Управления** установите флажок **Microsoft Multimedia Control 6.0**. После нажатия кнопки **OK** значок элемента появится внизу окна элементов управления. После этого элемент переносится на форму обычным способом.

Работа с приложением сводится к нажатию трех кнопок. При нажатии на каждую из кнопок на форме рисуется соответствующая фигура. При этом предыдущие рисунки стираются и звучит музыкальный фрагмент. Во время работы приложения элемент `MMControl1` не виден, т. к. его свойство `Visible` равно False.

В качестве музыкальных файлов выбраны файлы с расширением wav. Это потребовало применить устройство воспроизведения (Device Type) типа `WaveAudio`. Напомним, что для файлов с расширением mid следует использовать устройство типа `Sequencer`.

Выбор файлов удобно выполнять их прослушиванием в режиме поиска: **Пуск | Найти | Файлы и папки**. Например, в поле ввода имени файла в окне поиска файлов и папок записать *.wav и нажать кнопку **Найти**.

В окно **Результаты поиска** выводится список имен файлов и пути к ним. Щелчки на именах файлов запускают их воспроизведение. При этом на экране появляется проигрыватель с элементами управления.

Если не использовать процедуры при написании кода, то большую часть программы будут занимать строки воспроизведения звуков. Строки повторяются "один к одному" в каждой из трех процедур обработки событий. С этим еще можно было бы согласиться, если бы хотя бы имена файлов были различными. Заметим, что с увеличением числа кнопок существенно увеличивается и длина кода.

Для упрощения программы удобно использовать общие процедуры. В этом случае общие части всех трех процедур обработки событий заменяются одной из них, но записанной отдельно, в виде процедуры. Эта общая процедура имеет имя и вызывается по этому имени из процедур обработки событий.

Присвоим общей процедуре имя `music`. На рис. 19.20, *а* показана структура рассмотренной выше программы без использования общей процедуры. На рис. 19.20, *б* приведена структура той же программы с использованием общей процедуры.

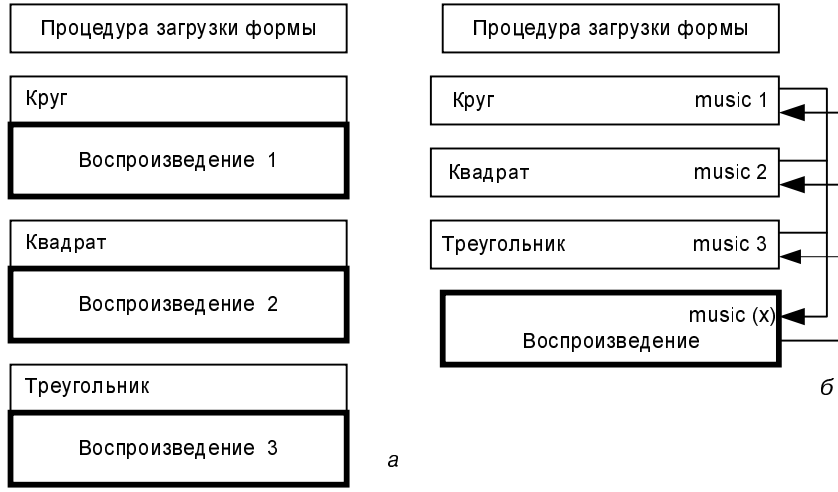


Рис. 19.20. Структура приложения "Музыка в кнопках"

Для воспроизведения разных мелодий дописываем к имени общей процедуры разные значения аргументов: 1, 2 и 3. Эти значения при передаче управления процедуре присваиваются параметру `x`. Использование оператора `Select Case` позволяет по значению параметра `x` выбирать разные файлы для воспроизведения различных мелодий.

Код приложения "Музыка в кнопках" приведен в листинге 19.23.

Листинг 19.23. Код приложения "Музыка в кнопках"

```
Private Sub Form_Load()
Show
End Sub
Private Sub cmdКруг_Click()
Cls
```

```
music 1
Circle (1000, 1000), 500
End Sub
Private Sub cmdКвадрат_Click()
Cls
music 2
Line (2000, 500)-(3000, 1500), , B
End Sub
Private Sub cmdТреугольник_Click()
Cls
Call music(3)
Line (3500, 1500)-(4500, 1500)
Line -(4000, 500)
Line -(3500, 1500)
End Sub
Public Sub music(x)
Select Case x
Case 1: MMControl1.FileName = "c:\Windows\Media\Chord.wav"
Case 2: MMControl1.FileName = "c:\Windows\Media\Chimes.wav"
Case 3: MMControl1.FileName = "c:\Windows\Media\Ding.wav"
End Select
With MMControl1
.Visible = False
.DeviceType = "WaveAudio"
.Command = "Open"
.Command = "Sound"
.Command = "Close"
End With
End Sub
```

19.9. Анимация

Анимация — это движение различных изображений. Простейший вид анимации заключается в перемещении или смене (переключении) изображений. Рисунок выводится на экран, стирается, выводится через короткий промежуток времени на экран в новой точке, стирается и т. д. При этом создается впечатление движения рисунка *(см. "Пример 15. Затмение Луны")*. При смене изображений рисунок стирается, а новый рисунок остает-

ся на том же месте *(см. "Пример 17. Танцор")*. При совмещении перемещения и смены рисунков можно имитировать, например, бег человека.

Элемент Visual Basic `Timer` упрощает задание скорости перемещения и смены изображений.

Пример 15. Затмение Луны

Создайте новый проект (рис. 19.21) "Затмение Луны". Разместите в центре формы элемент `Shape1` (круг). Скопируйте его с целью получить одинаковый по размеру элемент `Shape2`. При копировании откажитесь от создания массива. Поместите `Shape2` слева от `Shape1` на одинаковой высоте. Поместите в форму элемент `Timer1`.

Задайте `Shape1` светло-голубой цвет (свойства `FillColor` — палитра и `FillStyle` — заливка), а форме и `Shape2` задайте темно-синий цвет (`BackColor` — палитра).

Запишите приведенный ниже код (листинг 19.24). При проверке работы приложения подберите наиболее удачный интервал таймера.

Рис. 19.21. Окно приложения "Затмение Луны"

Листинг 19.24. Код приложения "Затмение Луны"

```
Код программы
Private Sub Form_Load()
  Shape2.Left = 0
  Timer1.Interval = 50
End Sub
Private Sub Timer1_Timer()
```

```
  Shape2.Left = Shape2.Left + 10
  'можно Shape2.Move = Shape2.Left + 10
End Sub
```

Пример 16. Светофор

Приложение "Светофор" также может служить простейшим примером анимации с переключением и наглядным пособием по правилам уличного движения (рис. 19.22, листинг 19.25).

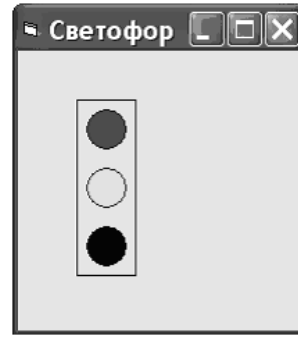


Рис. 19.22. Окно приложения "Светофор"

Листинг 19.25. Код приложения "Светофор"

```
Private Sub Timer1_Timer()
  Static t
  t = (t + 1) Mod 10
  Select Case t
    Case 0
      Shape1.FillColor = vbRed
      Shape2.FillColor = vbBlack
    Case 3
      Shape2.FillColor = vbYellow
    Case 5
      Shape1.FillColor = vbBlack
      Shape2.FillColor = vbBlack
      Shape3.FillColor = vbGreen
```

```
    Case 8
      Shape2.FillColor = vbYellow
      Shape3.FillColor = vbBlack
    End Select
End Sub
```

Пример 17. Танцор

Большие возможности по анимации дают графические элементы Visual Basic.

Создайте стандартный EXE-проект. Разместите на форме четыре элемента `Image`, как показано на рис. 19.23, и таймер. Свойству `Stretch` элементов `Image` присвойте значение True. Таймеру задайте интервал 500. Затем при настройке работы приложения интервал можно будет изменить, а элементы `Image` — совместить по площади.

Код приложения "Танцор" приведен в листинге 19.26.

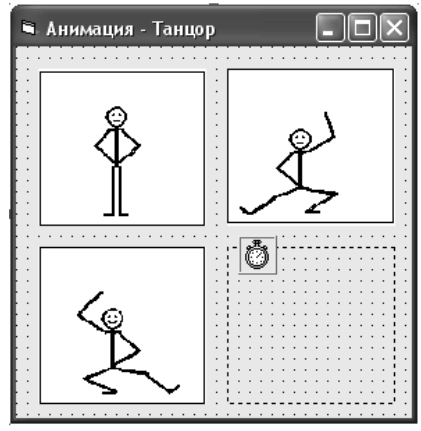


Рис. 19.23. Пример приложения с анимацией (окно приложения "Танцор")

В графическом редакторе Paint нарисуйте человечка и через буфер обмена скопируйте его в Image1 (в левом верхнем углу формы). В редакторе Paint, работая инструментом **Ластик**, стирая и подрисовывая руки и ноги фигурки, получите еще два рисунка. Через буфер обмена скопируйте их в Image2 и Image3.

Запишите приведенный ниже код программы.

Листинг 19.26. Код приложения "Танцор"

```
Private Sub Form_Load()
  Image1.Stretch = True: Image1.Visible = False
  Image2.Stretch = True: Image2.Visible = False
  Image3.Stretch = True: Image3.Visible = False
  Timer1.Interval = 500
End Sub
Private Sub Timer1_Timer()
  Static k
  k = (k + 1) Mod 4
  Select Case k
    Case 0
      Image4.Picture = Image1.Picture
    Case 1
      Image4.Picture = Image2.Picture
    Case 2
      Image4.Picture = Image1.Picture
    Case 3
      Image4.Picture = Image3.Picture
  End Select
End Sub
```

Пример 18. Видеофильм

Использованный в предыдущих примерах, входящий в набор Micrisoft Multimedia 6.0 стандартный элемент `MMControl` можно использовать для просмотра видеофильмов. Для поиска и просмотра видеофайлов с расширением avi можно использовать систему поиска файлов и средства их просмотра системы Windows. В показанном ниже проекте (листинг 19.27) код программы демонстрирует работу файла GLOBE.avi.

Создайте стандартный EXE-проект. Выведите в форму элемент `MMControl` и запишите код программы.

Листинг 19.27. Код приложения "Видеофильм"

```
Private Sub Form_Load()
  Кино.DeviceType = "AVIVideo"
  Кино.FileName = "c:\Мои документы\Анимация\GLOBE.avi"
```

```
  Кино.Command = "Open"
  Кино.Command = "Play"
End Sub
```

Запустите работу приложения и поработайте с пультом управления.

19.10. Работа с текстовыми файлами

Необходимость работы с файлами возникает не только при их открытии, сохранении, переименовании и других подобных действиях. Хранить информацию в виде текстовых файлов последовательного доступа удобно во многих случаях, например, при создании приложений для обучения, для контроля знаний обучаемых, для их тестирования и т. п.

Файлы по виду доступа к их содержимому принято разделять на файлы последовательного доступа, файлы произвольного доступа и двоичные (бинарные) файлы. Рассмотрим работу с текстовыми файлами последовательного доступа.

В текстовом файле последовательного доступа информация сохраняется и считывается построчно. Вся процедура общения с файлом состоит из следующих действий: открытие файла (оператор `Open`), чтение (оператор `Input`) запись (оператор `Output`), добавление данных (оператор `Append`) и закрытие файла (оператор `Close`).

Открытие файла

Оператор `Open` имеет следующий синтаксис:

```
Open <имя файла> For[Input | Output | Append] As <номер канала>
```

Пример:

```
Open  "C:\ ТестДаНет.exe"  For Input  As  #1
```

Чтение файла

После того как файл открыт, его содержимое можно построчно считывать и передавать по каналу связи. Это можно делать с помощью оператора `Line Input #` или в цикле, или при обработке события событийной процедурой приложения, пока не наступит конец файла `EOF` (`End Of File`). В показанном листинге 19.28 примере за одно обращение считываются две строки, содержимое которых присваивается переменным `x` и `y`.

Листинг 19.28. Фрагмент кода чтения из файла

```
If Not (EOF(1)) Then
  Line Input #1, x
```

```
  Line Input #1, y
Else
  Close
End If
```

Запись в файл

Если файл для записи отсутствует, то при записи (`Output`) или добавлении (`Append`) он создается в заданном месте и с заданным именем, после чего в него производится запись. Если файл имеется, то при записи он предварительно очищается. В листинге 19.29 показан простой пример записи в файл.

Листинг 19.29. Фрагмент кода записи в файл

```
Private Sub Command1_Click()
  Open "C:\Мои документы\Файл для записи.txt" For Output As #1
  Print #1, "стол", "стул", "шкаф"
  Close
End Sub
```

Пример 19. Приложение "Тест Да или Нет"

Проекту присвоим имя `ТестДаНет`. Для создания проекта требуются (рис. 19.24):

- одна форма (присвоено имя `frmДаНет`);
- одна метка (имя по умолчанию — `Label1`, свойство `Caption` равно `"Укажите истинны или ложны следующие утверждения:"`);
- одно текстовое поле (`TextBox` присвоено имя — `txtВопрос`);
- две кнопки (присвоены имена `cmdВопрос` и `cmdВыход`);
- два переключателя (присвоены имена `optYes` и `optNo`).

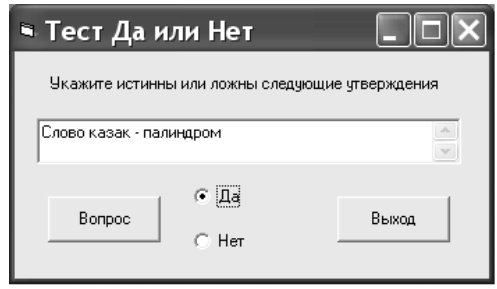


Рис. 19.24. Окно теста "Да или Нет"

Текст вопросов создается с помощью блокнота и записывается в файл C:\Тест Да и Нет.txt (рис. 19.25).

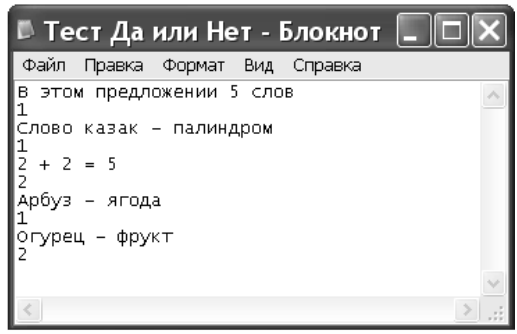


Рис. 19.25. Тест в файле Тест Да и Нет.txt (просмотр в редакторе Блокнот)

Код приложения приведен в листинге 19.30.

Листинг 19.30. Код приложения "Тест Да и Нет"

```
Dim k, kp, n
Private Sub Form_Load()                'с загрузкой формы открывается файл
  Open "C:\Мои документы\ Тест Да и Нет.txt" For Input As #1
End Sub

Private Sub cmdВопрос_Click()
                                       'kp - число правильных ответов
  If n = 1 And optYes.Value = True Then kp = kp + 1
  If n = 2 And optNo.Value = True Then kp = kp + 1
  k = k + 1
                                       'номер вопроса
  optYes.Value = False
  optNo.Value = False
  If Not (EOF(1)) Then                 'если нет конца файла, то
    Line Input #1, a$                  'читается текст очередного вопроса
    Line Input #1, n                   'и номер правильного ответа
    txtВопрос.Text = a$                'вывод вопроса на экран
  Else
   txtВопрос.Text = "Из " + Str(k - 1)
```

```
+ " вопросов правильно отвечено на" + Str(kp)
    cmdВопрос.Enabled = False              'кнопка делается недоступной
  End If
End Sub
Private Sub cmdВыход_Click()
  Close #1
                                           'закрытие файла
  End
End Sub
```

Пример 20. Приложение "Тест с Флажками"

В приложении "Тест с Флажками" используется еще один из множества способов построения тестов. Окно приложения (рис. 19.26) содержит метку, текстовое поле, четыре флажка и две кнопки. Имена элементов и их надписи можно увидеть в коде приложения (листинг 19.31). Вопросы и правильные ответы последовательно считываются из созданного в Блокноте текстового файла Тест с Флажками.txt (рис. 19.27).

Нажатие кнопки **Вопрос** выводит в текстовое поле задание и надписи к флажкам. После этого следует поставить флажки в соответствующие заданию строки. Завершается тест сообщением о числе вопросов и правильных ответов.

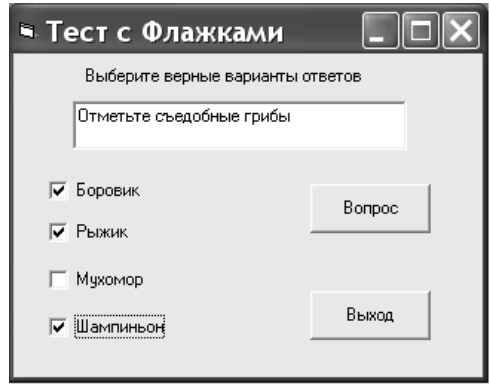


Рис. 19.26. Окно приложения "Тест с Флажками"

При записи в Блокнот теста следует оставить курсор в последней заполненной строке, как показано на рис. 19.27.

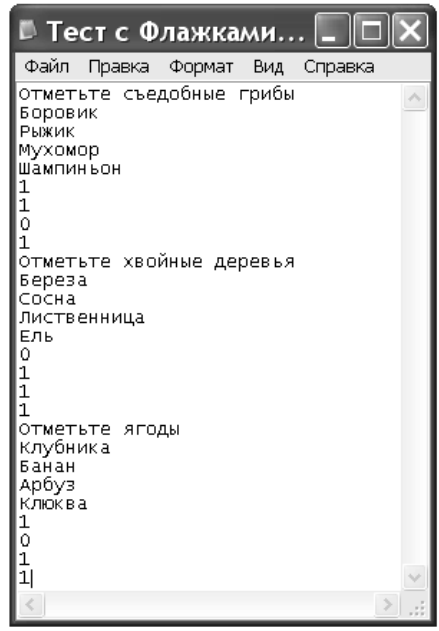


Рис. 19.27. Текст задания и ответов приложения "Тест с Флажками"

Листинг 19.31. Код приложения "Тест с Флажками"

```
Dim kp, k, a1, a2, a3, a4
Private Sub Form_Load()
  Open "C:\Мои документы\Тексты Тестов\Тест с Флажками.txt" For Input As
#1
End Sub
Private Sub cmdВопрос_Click()
  If Check1.Value = a1 And Check2.Value = a2 _
    And Check3.Value = a3 And Check4.Value = a4 Then
    kp = kp + 1      'подсчет числа правильных ответов
  End If
    k = k + 1
    Check1.Value = 0: Check2.Value = 0: Check3.Value = 0: Check4.Value =
0
'ниже, если еще нет конца файла (EOF(1)), то из памяти считывается текст
вопроса, надписи к флажкам и перечень правильных ответов
  If Not (EOF(1)) Then
    Line Input #1, t$: txtВопрос.Text = t$
```

```
    Line Input #1, t1$: Check1.Caption = t1$
    Line Input #1, t2$: Check2.Caption = t2$
    Line Input #1, t3$: Check3.Caption = t3$
    Line Input #1, t4$: Check4.Caption = t4$
    Line Input #1, a1
    Line Input #1, a2
    Line Input #1, a3
    Line Input #1, a4
  Else
    txtВопрос.Text = "Из " + Str(k - 1) + " вопросов правильно отвечено на
" + Str(kp - 1)
    cmdВопрос.Enabled = False   'кнопка недоступна
  End If
End Sub
Private Sub cmdВыход_Click()
  End
End Sub
```

19.11. Приложения-игры

Пример 21. Отгадай число

Закрепить рассмотренный в *гл. 3* оптимальный алгоритм отгадывания числа из заданного диапазона чисел можно с помощью изучения приложения "Отгадай число", код которого приведен в листинге 19.32.

Листинг 19.32. Код приложения "Отгадай число"

```
Dim a As Integer, k As Integer, b As Integer
Private Sub Form_Load()
  Show
  Randomize
  a = Int(Rnd * 100 + 1)
  b = InputBox("Введите число в диапазоне от 1 до 100", , , 3000, 4000)
  Do While b <> a
    If b > a Then
      Print "Введено число "; b; "  Оно больше загаданного компьютером."
    End If
    If b < a Then
      Print "Введено число "; b; "  Оно меньше загаданного компьютером."
```

```
    End If
    k = k + 1
    b = InputBox("Введите число", , , 3000, 4000)
  Loop
  Print "Вы отгадали число. Оно равно: "; b
  Print "Было сделано "; k + 1; "попыток."
End Sub
```

Пример 22. Отгадай слово

Приложение "Отгадай слово" построено на элементах, показанных на рис. 19.28:

- Форма (`Name` — `frmИграСлова`, `Caption` — `Отгадай слово`);
- Два текстовых поля — оставлены имена по умолчанию `Text1` и `Text2`;
- Метка — оставлено имя `Label1`;
- Две кнопки — `cmdПоказать` и `cmdСравнить`.

В этом приложении используется несколько интересных приемов: секретность ввода текста, перемешивание букв в массиве и т. п. Игра заключается в следующем.

1. Первый игрок загадывает слово и записывает его в левое текстовое поле (рис. 19.28). При записи слова буквы заменяются на звездочки.
2. При нажатии на кнопку **Показать** буквы случайным образом перемешиваются и выводятся в левое поле. Надпись меняется на "Запиши слово в правое окно".
3. Второй игрок делает попытку отгадать слово, записывая его в правое поле.
4. При нажатии на кнопку **Сравнить** в левое поле выводится загаданное слово и, в зависимости от результата, в метке записывается слово Да или Нет.

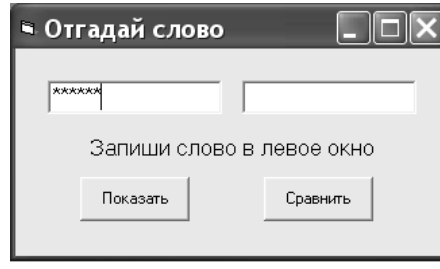


Рис. 19.28. Окно игры "Отгадай слово" перед показом задания

Код приложения приведен в листинге 19.33.

Листинг 19.33. Код приложения "Отгадай слово"

```
Dim a() As String, b As String
Dim c As String, d As String
Private Sub cmdПоказать_Click()
  Randomize             'для случайного перемешивания букв
  b = Text1.Text        'запоминаем слово для вывода
  ReDim a(Len(b))       'определяем размер динамического массива
  For I = 1 To Len(b)   'формируем массив букв
    a(I) = Mid(b, I, 1)
  Next
  For I = 1 To Len(b)   'перемешиваем буквы в массиве
    J = (Int(Rnd * Len(b) + 1))
    c = a(I): a(I) = a(J): a(J) = c
  Next
  For I = 1 To Len(b)   'формируем закодированное слово
    d = d & a(I)
  Next
  Text1.PasswordChar = "" 'отключаем секретность вывода
  Text1.Text = d          'выводим закодированное слово
  Label1.Caption = "Запиши слово в правое окно"
End Sub
Private Sub cmdСравнить_Click()
  Text1.Text = b          'вывод исходного слова
  If Text2.Text = b Then  'проверка отгадки
    Label1.Caption = "Да"
  Else
    Label1.Caption = "Нет"
  End If
End Sub
Private Sub Form_Load()
  Text1.PasswordChar = "*"  'секретность ввода
  Text1.FontSize = 12
  Text2.FontSize = 12
  Label1.FontSize = 12
  Label1.Caption = "Запиши слово в левое окно"
End Sub
```

Пример 23. Тараканьи бега

Приложение "Тараканьи бега", реализующее игру "тараканьи бега", демонстрирует возможности использования для построения простых игр стандартных управляющих элементов. Основой игры служат три горизонтальные полосы прокрутки `HScrollBar` с именами `Hscroll1`, `Hscroll2` и `Hscroll3` (рис. 19.29). К ним добавлены три метки номеров дорожек с именами `Label1`, `Label2` и `Label3`. В нижней части окна расположены три кнопки с именами: `cmdСтарт`, `cmdНовыйЗабег` и `cmdВыход`. Скорость перемещения "бегунов" определяет таймер и величина случайных приращений к значениям свойства `Value` полос прокрутки.

Рис. 19.29. Окно приложения "Тараканьи бега" с результатами забега

Листинг 19.34. Код приложения "Тараканьи бега"

```
Private Sub cmdСтарт_Click()
  Timer1.Interval = 50          'запуск таймера
End Sub
Private Sub Form_Load()
  HScroll1.Max = 100: HScroll2.Max = 100: HScroll3.Max = 100
End Sub
Private Sub Timer1_Timer()
  Static a, b, c
  Randomize Timer
                    'две строки - бег тараканов и визуализация бега
  a = a + Int(Rnd * 2): b = b + Int(Rnd * 2): c = c + Int(Rnd * 2)
  HScroll1.Value = a: HScroll2.Value = b: HScroll3.Value = c
                    'три строки: проверка финиша, остановка таймера,
↳вывод результатов, сброс в исходное состояние
```

```
  If a >= 100 Or b >= 100 Or c >= 100 Then
    Timer1.Interval = 0
    Print "Результат: "; "1 -"; a, "2 -"; b, "3 -"; c
    a = 0: b = 0: c = 0
  End If
End Sub
Private Sub cmdНовыйЗабег_Click()
              'две строки: очистка формы, сброс в исходное состояние
  frmТараканьиБега.Cls
  HScroll1.Value = 0: HScroll2.Value = 0: HScroll3.Value = 0
End Sub
Private Sub cmdВыход_Click()
  End
End Sub
```

Пример 24. Тренинг для игрока

Как известно, успехи на международных чемпионатах по компьютерным играм зависят в том числе и от мастерства владения мышью. Быстроту и точность работы с мышью можно проверить с помощью следующего простого приложения (листинг 19.35).

На развернутой во весь экран форме с заданной частотой в случайных местах рисуются окружности небольшого диаметра. Рядом с окружностями записывается их общее количество. Требуется щелкнуть мышью внутри окружности до появления следующей окружности. При щелчке выводится общее число попаданий. Для запуска приложения служит кнопка **Старт** (имя — `Command1`). Завершается работа нажатием кнопки **Стоп** (имя — `Command2`).

Листинг 19.35. Код приложения "Тренинг для игрока"

```
Dim x1 As Integer, y1 As Integer
Dim k As Integer, s As Integer
Private Sub Command1_Click()
  Form1.Show
  Timer1.Interval = 2000
End Sub
Private Sub Command2_Click()
  Timer1.Interval = 0
End Sub
```

```
Private Sub Form_MouseDown(Button As Integer,
↳Shift As Integer, X As Single, Y As Single)
  Static s
  Circle (X, Y), 10
  r = Sqr((X - x1) ^ 2 + (Y - y1) ^ 2)
  If r <= 50 Then s = s + 1
  Print Tab; s
End Sub
Private Sub Timer1_Timer()
  Static k
  Cls
  x1 = Int(Rnd * 6000 + 200)
  y1 = Int(Rnd * 4000 + 200)
  Circle (x1, y1), 50
  k = k + 1
  Print k
End Sub
```

Приложение можно модернизировать, добавив управление заданием диаметра окружности, частоты ее появления, выводом результатов тренировки.

19.12. Работа с принтером

Команда *Печать* меню *Файл*

Простейший вариант использования принтера в Visual Basic — это распечатать форму и код приложения. Для этого нужно вызвать из памяти приложение и подать команду **Файл | Печать**. На экран по этой команде выводится диалоговое окно с заголовком **Печать — <*Имя приложения*>**. Следует заметить, что связанные с печатью окна и их содержание могут существенно различаться в зависимости от версии операционной системы.

При нажатии в окне **Печать** кнопки **Настройка** раскрывается окно **Настройка печати**, в котором можно установить формат бумаги (по умолчанию A4), режим подачи бумаги (ручная или автоматическая) и ориентацию печати (книжная или альбомная).

В окне **Печать** в группе флажков **Что печатать** предлагаются варианты печати.

- Выбор варианта **Изображение формы** приводит к печати внешнего вида формы в режиме конструирования (без заголовка).

- При выборе варианта **Код** распечатывается код приложения.
- Установка флажка **Форму как текст** приводит к распечатке перечня всех использованных в приложении элементов управления со значениями свойств, измененных в процессе проектирования.
- Допускается выбор одновременно нескольких вариантов.

Форму со всеми размещенными на ней элементами управления можно распечатать также с помощью метода `PrintForm`, записав в код программы инструкцию `ИмяФормы.PrintForm`.

Объект *Printer*

Этот виртуальный объект служит для распечатки текста и графики. Он обладает всеми необходимыми для высокоточной и качественной печати свойствами и методами, но требует больших затрат времени для написания кода. Свойства и методы объекта `Printer` во многом совпадают со свойствами и методами формы. Это свойства выбора шрифта, его типа и размера, все графические методы, свойства `CurrentX` и `CurrentY` задания координат точек вывода и т. д.

В качестве примера в листинге 19.36 приведен характерный фрагмент кода программы, работающей с объектом `Printer`.

Листинг 19.36. Фрагмент кода программы печати ведомости с помощью объекта `Printer`

```
'Код вывода Ведомости (опорного плана участка и таблицы его
характеристик)
Private Sub cmdПечать_Click()
  Printer.FontSize = 10
  Printer.CurrentX = 25
  Printer.CurrentY = 10
  Printer.Print "Область Ленинградская"; Spc(26); "Район Приозерский"
.....
  Printer.Print Spc(18); "Ведомость координат и площадей земельного
  участка"
.....
  Printer.Line (20, 50)-(195, 62 + n * 6), , B
.....
  Printer.CurrentX = 25: Printer.CurrentY = 51
```

```
  Printer.Print "№"; Spc(6); "Внутренние углы"; Spc(6); "Дирекционные
углы"; Spc(4); "Меры"
.....
  Printer.EndDoc
End Sub
```

19.13. Создание EXE-файла и инсталляционного пакета

Пусть все файлы проекта (с расширениями vbp, frm, bas и т. д.) находятся в одной папке с именем этого проекта. Это удобно для работы и исключает путаницу.

Обычно перед созданием приложения проект проверяется, тестируется, дорабатывается для улучшения его внешнего вида, информативности, удобства работы и т. п.

Загрузите проект и проверьте его работу, т. к. созданный EXE-файл уже нельзя будет переделать. При желании устранить ошибки придется поработать с проектом в прежнем виде и создавать новый EXE-файл.

Если проект не требует для своей работы обращений к другим файлам для чтения или записи информации, то создание EXE-файла выполняется без особой подготовки проекта командой **Создать ИмяПроекта.exe** из меню **Файл**.

При подаче этой команды на экран выводится диалоговое окно **Создать проект** с предложением имени создаваемого проекта и места его сохранения. Можно присвоить приложению новое имя или оставить старое имя проекта. Если проект вызывался из своей папки, то записать EXE-файл приложения удобно в эту же папку. Нажатие кнопки **OK** завершает создание EXE-файла приложения.

После создания исполняемого вне среды Visual Basic приложения его можно скопировать в специально созданную для приложений папку Мои приложения, размещенную в папке Мои документы. Кроме того, можно создать ярлык приложения и вывести его на рабочий стол.

Замечание о вспомогательных файлах

Если проект требует для своей работы обращения к вспомогательным файлам, то создание исполняемого файла потребует некоторой подготовки. Дело в том, что адреса вспомогательных файлов в EXE-файле будут фиксированы и их нельзя будет изменить. Поэтому при случайном переносе этих файлов в другие папки или при их удалении приложение окажется неработоспособным.

Один из способов несколько обезопасить проект от потери связи с вспомогательными файлами заключается в переносе этих файлов в папку с приложением.

Создание инсталляционного пакета

Простые проекты, не связанные с обращениями к вспомогательным файлам, перенести на другой компьютер просто — для этого достаточно обычной дискеты. При этом необходимо перенести все файлы всех форм и модулей проекта. Это удобно сделать, собрав их в отдельную папку.

Дело существенно усложняется, если возникает потребность перенести сложное по структуре и взаимосвязям файлов приложение. В этом случае приходится создавать *Инсталляционный пакет* и инсталлировать программу приложения на этом компьютере.

После создания исполняемого файла приложения командами главного меню **Добавления | Менеджер дополнений** откройте диалоговое окно, в котором выделите строку **Package and Deployment Wizard** (Мастер упаковки и развертывания) и нажмите кнопку **OK**. На экран выводится первое окно мастера. Мастер также можно найти и запустить, используя известные способы поиска файлов в системе Windows. Он находится в папке PDWizard, которая в свою очередь находится в папке VB98 (файл PDCMDLN.EXE).

В первом открывшемся окне из трех вариантов (**Package** — упаковка, **Deploy** — Распаковка, и **Manage Scripts** — Менеджер сценария) выберем первый. Всего мастер выведет на экран около двадцати различных окон. И почти в каждом случае следует соглашаться с предлагаемым мастером вариантом, нажимая на кнопку **Да** или **Next**.

Так можно поступать в окнах **Package Type**, **Package Script**, **Package Folder**. Если в окне **Package Folder** согласиться с предложенным вариантом, то мастер запишет все файлы создаваемого инсталляционного пакета в папку Package и поместит ее в папку с файлами проекта. Нажав на кнопку **New Folder**, можно выбрать и другой вариант размещения пакета.

В окне **Included Files** предлагается выполнить важную операцию включения в инсталляционный пакет всех файлов, к которым в процессе работы обращается ваше приложение и без которых оно может оказаться неработоспособным.

В окне **Cab Options** мастер интересуется, как будет переноситься пакет на другие компьютеры: одним большим файлом (**Single cabs**) или несколькими небольшими (**Multiple cabs**), например, размером с дискету.

В окне **Installation Title** следует утвердить или задать новое имя приложения и так далее до финиша и сообщения мастера о завершении работы.

Инсталляция

После завершения работы по созданию инсталляционного пакета можно установить приложение на своем компьютере. В созданной компьютером папке Package содержатся папка Support и несколько файлов, среди которых находятся сжатый файл приложения и файл установки Setup.exe. Файлы папки Support не входят в инсталляционный пакет и не подлежат переносу на другой компьютер. Остальные файлы необходимо переносить.

Для инсталляции приложения на своем компьютере следует завершить работу со всеми ранее загруженными программами и запустить программу Setup.exe установки приложения. После запуска программы установки последовательно появляются три диалоговых окна, ответив на вопросы которых, получим возможность запускать приложение, например, из стартового меню.

Глава 20

Информация и системы

20.1. Информационные управляющие системы

Информатика тесно связана с кибернетикой. Кибернетика — это наука о процессах управления в машинах, живых организмах, обществе. Она также занимается изучением общих законов получения, хранения, передачи и переработки информации в управляющих системах.

В кибернетике рассматриваются более сложные *информационные управляющие системы*, включающие управляющий объект, канал связи и управляемый объект (рис. 20.1). При этом управляющий и управляемый объекты могут представляться в виде *"черного ящика"*, что предполагает изучение работы системы независимо от внутреннего содержания и структуры этих объектов. Они могут быть любой природы: механическими, электрическими, биологическими, общественными структурами и т. д.

Кроме понятия черного ящика в кибернетике используют важное понятие *обратной связи*. Функционирование управляющей системы рассматривается с разомкнутой и замкнутой обратной связью, которая может быть положительной или отрицательной.

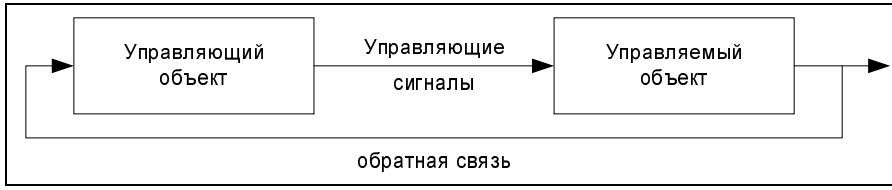


Рис. 20.1. Информационная управляющая система

Положительная обратная связь усиливает результаты работы управляемого объекта. Как правило, положительная обратная связь приводит к усилению управляющих сигналов, что ведет к перевозбуждению и неустойчивой рабо-

те системы. *Отрицательная обратная связь* ослабляет управляющие воздействия на управляемый объект.

Для стабильной, устойчивой, без излишних колебаний работы системы необходимо, чтобы управляющие сигналы в каждый момент времени имели соответствующие величину и знак. Это обеспечивается работой *устройства сравнения*. Оно находится на входе управляющего объекта, сравнивает передаваемые по обратной связи реальные результаты работы управляемого объекта с требуемыми результатами и вырабатывает *сигнал ошибки* соответствующего знака и величины. Это корректирует в нужном направлении управляющие сигналы и работу управляемого объекта.

Пусть управляющий объект — шофер автомашины, управляемый объект — автомашина. Управляющие сигналы — это воздействия шофера на рулевое управление. Обратная связь — зрительная. Качественная работа такой системы может определяться плавным, без колебаний вписыванием автомобиля в виражи дороги. Без обратной связи такая система не работоспособна.

Примером управляющей системы может служить система "учитель-ученик". Управляющие сигналы в такой системе — это информация, передаваемая на учебных занятиях. Качество функционирования системы оценивается уровнем знаний и умений ученика, которые существенно зависят от работы обратной связи — различных видов контроля усвоения учебного материала.

20.2. Алфавитный подход

Было показано, что состояние выключателя можно закодировать одним битом информации ($2 = 2^1$, $\log_2 2 = 1$), четыре стороны света кодируются двоичным словом длиной два бита ($4 = 2^2$, $\log_2 4 = 2$), восемь углов куба — трехбитовым словом ($8 = 2^3$, $\log_2 8 = 3$), и т. д.

Рассмотренные примеры показывают, что для кодирования в двоичном алфавите N вариантов (объектов, событий, сообщений, состояний и т. п.) требуется $I = \log_2 N$ бит информации. Можно сделать вывод: чем больше требуется закодировать объектов, тем длиннее требуется двоичное слово. Его длина определяется по формуле Хартли, названной по имени предложившего ее американского инженера:

$$I = \log_2 N.$$

Эта формула определяет количество информации (I), которое имеем в случае получения одного из N *равновероятных* сообщений.

Замечание

Напомним, что логарифмом называют показатель степени (I), в которую нужно возвести основание логарифма (2), чтобы получить заданное число (N). При $I = \log_2 N$ получаем $N = 2^I$.

Для определения величины I как функции от N можно воспользоваться таблицей значений функции I (табл. 20.1).

Таблица 20.1. *Значения функции I*

N **(сообщений)**	***I*** **(бит)**	***N*** **(сообщений)**	***I*** **(бит)**
2	1	64	6
4	2	128	7
8	3	256	8
16	4	512	9
32	5	1024	10

Пример 1

Требуется отгадать задуманное число из восьми чисел (от 1 до 8), задавая вопрос: "В первой половине находится задуманное число?" и получая в ответ "Да" или "Нет".

Задав первый вопрос и услышав первый ответ, получаем один бит информации. При этом неопределенность уменьшается в два раза (из 8 чисел остается только 4 числа). После второго вопроса получаем еще один бит информации, уменьшающий неопределенность еще в два раза (из 4 чисел осталось только 2 числа). Получая еще один бит информации, после третьего вопроса отгадываем число.

Рассмотренное правило отгадывания числа можно считать оптимальным. Для отгадывания потребовалось получить три ответа, три бита информации. Столько же бит требуется для оптимального кодирования каждого из восьми чисел.

На рис. 20.2 показана последовательность шагов отгадывания числа 6. На вопросы: "В какой половине находится число?" дается ответ в виде жирной линии. При решении задачи использован оптимальный по числу попыток прием деления чисел пополам. Метод половинного деления находит широкое применение в математике, информатике, других науках.

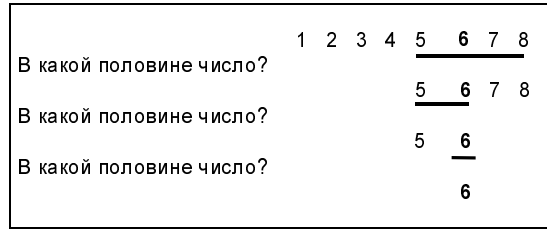


Рис. 20.2. Пример процедуры отгадывания числа

20.3. Вероятностный подход

Напомним, что вероятность — это числовая характеристика возможности появления какого-либо случайного события. При подбрасывании монеты появление орла или решки — события равновероятные. Вероятность появления каждой из этих сторон монеты равна 1/2 При подбрасывании игральной кости вероятность выпадения каждого конкретного числа равна 1/6 (у кубика 6 граней).

Для пояснения *вероятностного подхода* к определению количества информации формулу Хартли запишем в следующем виде:

$$I = \log_2 N = \log_2(1/p) = -\log_2 p,$$

где N — число равновероятных событий; p — вероятность появления события. Так как вероятности появления событий равны, то можно записать:

$$p = 1/N,$$

откуда:

$$N = 1/p.$$

Последнее преобразование основано на свойстве логарифмов: $\log_2 1/p = -\log_2 p$.

Проверим равенство для простейшего случая двух равновероятных событий:

$$I = -\log_2 1/2 = -(-1)=1 \text{ бит информации.}$$

Для рассмотренного выше примера с отгадыванием числа (пример 1) вероятность с первой попытки отгадать одно число из восьми равна 1/8. Подставляя $p = 1/8$ в формулу, получаем:

$$I = -\log_2 1/8 = -(-3) = 3 \text{ бита.}$$

Пример 2

При отгадывании одного задуманного числа из восьми будем пользоваться той же методикой, но делить числа на две неравные части.

Разделим все восемь чисел на две группы — 5 чисел и 3 числа.

Вероятности нахождения задуманного числа в группах различны. Чем больше чисел в группе, тем больше эта вероятность. Для первой группы она равна $p_1 = 5/8$. Для второй — $p_2 = 3/8$.

Можно ожидать, что и количество информации, содержащейся в ответе на вопрос "В какой группе находится загаданное число?" будет различным для первой и второй группы. Можно также предположить, что, чем меньше вероятность нахождения числа в группе, тем большее количество информации содержится в сообщении, что число находится в этой группе.

Вычислим количества информации. Сообщение о нахождении числа в первой группе содержит $I_1 = -\log_2 5/8 = 0{,}678$ бита. Сообщение о нахождении числа во второй группе содержит $I_2 = -\log_2 3/8 = 1{,}415$ бита. Предположения подтвердились.

Пусть получен первый ответ — число находится во второй группе (в ней три числа). Разделим ее на две группы — 1 число и 2 числа. Вычислим количество информации, содержащейся в ожидаемых сообщениях о нахождении числа в группе. $I_3 = -\log_2 1/3 = 1{,}585$ бита $I_4 = -\log_2 2/3 = 0{,}585$ бита.

Если на второй вопрос получен ответ о нахождении числа в первой группе (в ней одно число), то число отгадано за две попытки. При этом получено $I = I_2 + I_3 = 1{,}415 + 1{,}585 = 3$ бита информации.

Если же на второй вопрос получен ответ о нахождении числа во второй группе (в ней два числа), то требуется задавать еще один вопрос и получать ответ, который будет содержать $I_5 = -\log_2 1/2 = 1$ бит информации. В этом случае общее количество информации, которое будет получено при отгадывании числа, также будет равно: $I = I_2 + I_4 + I_5 = 1{,}415 + 0{,}585 + 1 = 3$ бита. Это объясняется тем, что события, состоящие в отгадывании с первой попытки любого из восьми чисел, имеют одинаковую вероятность, равную 1/8.

Использование калькулятора

Калькулятор имеет кнопки для работы с натуральными логарифмами (основание $e = 2{,}73$) и десятичными логарифмами (основание 10). Вычисления количества информации требуют работы с двоичными логарифмами (основание 2). Из элементарной математики известна формула перехода от натуральных логарифмов к двоичным: $\log_2 p = \ln p / \ln 2$. Воспользуемся ей.

В примере 2 утверждалось, что $I_1 = -\log_2 5/8 = 0{,}678$ бита. Проверим вычислениями на калькуляторе. Для этого требуется в соответствии с формулой перехода получить результат вычисления по формуле: $I = \ln 5/8 / \ln 2$.

Будем действовать так. Сначала получим значение натурального логарифма числа 2 (кнопка **ln**) и запишем его в память (кнопка **MS**). Затем вычислим частное от деления 5 на 8 (кнопка /) и получим значение натурального логарифма полученного частного (снова кнопка **ln**). Результат получим после деления логарифма частного (числителя формулы) на число из памяти — знаменатель (кнопка **MR**). По такой схеме можно легко и быстро вычислять значения количества информации.

Последовательность действий при работе с калькулятором показана в табл. 20.2 Напомним, что приложение "Калькулятор" запускается командой **Пуск | Программы | Стандартные | Калькулятор**. Для вычислений логарифмов калькулятор следует переключить в меню **Вид** на вариант **Инженерный**.

Таблица 20.2. *Работа с калькулятором*

Шаг	Кнопка	Результат
1	**2**	2
2	**ln**	0,693147
3	**MS** (в память)	0,693147

Таблица 20.2 *(окончание)*

Шаг	Кнопка	Результат
4	**5**	5
5	/	5
6	**8**	8
7	**=**	0,625
8	**ln**	–0,470003
9	/	–0,470003
10	**MR** (из памяти)	0,693147
11	**=**	–0,678071

Пример 3

В колоде 36 карт. Из них 12 карт с "портретами" валетов, дам и королей. Какое количество информации содержит сообщение о том, что из колоды была взята карта с портретом (I_1), туз (I_2), любая карта от шестерки до десятки (I_3), туз пик (I_4)?

Все четыре события имеют разную вероятность выполнения. Эти вероятности соответственно равны: $p_1 = 12/36 = 1/3$, $p_2 = 4/36 = 1/9$, $p_3 = 20/36 = 5/9$, $p_4 = 1/36$. Для вычисления количеств информации используем формулу $I = \log_2(1/p) = -\log_2 p$. Вычисления дают следующие результаты: $I_1 = -\log_2 12/36 = 1{,}585$; $I_2 = -\log_2 4/36 = 3{,}17$; $I_3 = -\log_2 20/36 = 0{,}847$; $I_4 = -\log_2 1/36 = 5{,}174$. Последний результат показывает, что для кодирования всех 36 карт требуется 5,174 бита или шестибитовое двоичное слово ($2^5 = 32$ — мало, т. к. $32 < 36$, $2^6 = 64$, $64 > 36$).

Пример 4

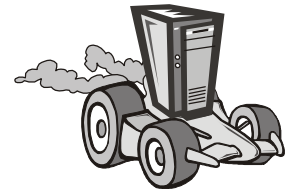
В составе поезда 16 вагонов. Среди них есть вагоны купейные и плацкартные. Сообщение о том, что ваш знакомый приезжает в купейном вагоне, несет 2 бита информации. Определить, сколько в поезде купейных вагонов.

Обозначим x — искомое число купейных вагонов. Вероятность того, что знакомый приезжает в купейном вагоне, равна $p = x/16$. Для решения используем формулу $I = \log_2(1/p)$. Для нашего примера имеем: $I = \log_2 16/x$. Напомним, что I — это показатель степени, в которую нужно возвести основание логарифма 2, чтобы получить число $1/p$. Величина I по условию задачи равна 2. Подставим значения: $2^2 = 16/x$, откуда получаем $x = 16 / 4 = 4$, т. е. в поезде 4 купейных вагона.

Задания для самостоятельной работы

1. Сколько бит информации нужно получить, чтобы отгадать одно задуманное число: из 32 чисел, из 64 чисел?
2. В лукошке находятся 16 фруктов: 8 яблок, 6 груш и 2 лимона. Какое количество информации содержится в сообщениях о том, что из лукошка случайным образом были последовательно взяты с возвратом: яблоко (I_1), груша (I_2), лимон (I_3).
3. В корзине с грибами 8 белых грибов, 8 подосиновиков и 16 подберезовиков. Сколько информации содержат события, состоящие в том, что из корзины случайным образом последовательно без возврата взяли: один белый гриб (I_1), один подберезовик (I_2)?
4. Из кошелька с восемью монетами наугад взяли монету 1 рубль, и это событие содержит 2 бита информации. Сколько в кошельке монет достоинством в один рубль?
5. В доме 64 квартиры. Среди них имеются трехкомнатные, двухкомнатные и однокомнатные квартиры. Сообщение о том, что ваш знакомый живет в двухкомнатной квартире, содержит 4 бита информации. Сколько в доме двухкомнатных квартир?

Глава 21



Арифметические основы работы компьютеров

21.1. Перевод чисел

В *гл. 4* были определены основные понятия, связанные с системами счисления, и действия с представленными в них числами. Для простоты изложения все действия были ограничены целыми числами. Настоящая глава является продолжением *гл. 4*. В ней рассматривается перевод смешанных дробей и связанные с машинной арифметикой способы и формы представления чисел.

Формула разложения числа по степеням основания

Пусть в десятичной системе задано некоторое число $A_{(10)} = 247{,}35$. Каждая позиция, занимаемая цифрами, называется *разрядом числа*. Разряды имеют названия и номера: разряд единиц, разряд десятков, разряд десятых, разряд сотых и т. д. Названия разрядов определяют их *вес*: единицы, десятки, десятые, сотые, тысячные.

Заданное число, не изменяя его количества, можно записать следующими способами:

$A_{(10)} = 247{,}35;$

$A_{(10)} = 200 + 40 + 7 + 3/10 + 5/100;$

$A_{(10)} = 2 \cdot 100 + 4 \cdot 10 + 7 + 3 \cdot 1/10 + 5 \cdot 1/100;$

$A_{(10)} = 2 \cdot 10^2 + 4 \cdot 10^1 + 7 \cdot 10^0 + 3 \cdot 10^{-1} + 5 \cdot 10^{-2}.$

Последнюю запись называют разложением числа по степеням основания.

Запишем десятичное число, имеющее по три разряда в целой и дробной части, и формулу его разложения в общем виде:

$A_{(10)} = a_2a_1a_0,\ a_{-1}a_{-2}a_{-3} = a_2 \cdot 10^2 + a_1 \cdot 10^1 + a_0 \cdot 10^0 + a_{-1} \cdot 10^{-1} + a_{-2} \cdot 10^{-2} + a_{-3} \cdot 10^{-3}.$

Для числа в позиционной системе с основанием p, имеющего n разрядов в целой и m разрядов в дробной части, несколько упрощенная (без p^0) формула записывается так:

$$A_{(p)} = a_{n-1}a_{n-2}...a_2a_1a_0,\ a_{-1}a_{-2}a_{-3}...a_{-m+1}a_{-m} =$$
$$= a_{n-1} \cdot p^{n-1} + a_{n-2} \cdot p^{n-2} + ... + a_2 \cdot p^2 + a_1 \cdot p + a_0 + a_{-1} \cdot p^{-1} + a_{-2} \cdot p^{-2} + ...$$
$$... + a_{-m} \cdot p^{-m}.$$

В этой записи каждая степень основания p^i определяет вес своего (i-го) разряда. Влево и вправо от запятой он соответственно увеличивается и уменьшается в число раз, равное основанию системы p. Произведение цифры разряда числа на вес своего разряда определяет *количественный эквивалент цифры* $a_i \cdot p^i$. Как видно из формулы, сумма всех количественных эквивалентов числа определяет его величину.

Перевод с использованием формулы разложения

Как было показано в *гл. 4*, наиболее простой способ перевода заключается в суммировании количественных эквивалентов цифр заданного числа. Действия при переводе выполняются в новой системе, поэтому способ удобно использовать для перевода чисел в десятичную систему. В основе способа лежит использование веса разрядов числа — значений степеней основания чисел:

- для $p = 10$ — $...1000, 100, 10, 1, \frac{1}{10}, \frac{1}{100}, \frac{1}{1000}, \frac{1}{10000}, ...$
- для $p = 2$ — $...1024, 512, 256, 128, 64, 32, 16, 8, 4, 2, 1, \frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \frac{1}{16}, \frac{1}{32}, ...$
- для $p = 8$ — $...4096, 512, 64, 8, 1, \frac{1}{8}, \frac{1}{64}, \frac{1}{512}, \frac{1}{4096}, ...$
- для $p = 16$ — $...65536, 4096, 256, 16, 1, \frac{1}{16}, \frac{1}{256}, \frac{1}{4096}, \frac{1}{65536}, ...$

Пример 1

Дано : $A_{(2)} = 10111{,}1101$. *Найти* : $A_{(10)}$.

Решение :

Запишем формулу разложения двоичного числа:

$$A_{(2)} = a_4 \cdot 16 + a_3 \cdot 8 + a_2 \cdot 4 + a_1 \cdot 2 + a_0 + a_{-1} \cdot \frac{1}{2} + a_{-2} \cdot \frac{1}{4} + a_{-3} \cdot \frac{1}{8} + a_{-4} \cdot \frac{1}{16}.$$

Подставляем в формулу значения двоичных разрядов и выполняем действия:

$$A_{(10)} = 1 \cdot 16 + 0 \cdot 8 + 1 \cdot 4 + 1 \cdot 2 + 1 + 1 \cdot \frac{1}{2} + 1 \cdot \frac{1}{4} + 0 \cdot \frac{1}{8} + 1 \cdot \frac{1}{16}.$$

Записи при переводе можно значительно уменьшить, если непосредственно суммировать степени основания, соответствующие единичным значениям разрядов заданного числа (сколько единиц в двоичной записи числа — столько слагаемых):

$$A_{(10)} = 16 + 4 + 2 + 1 + \frac{1}{2} + \frac{1}{4} + \frac{1}{16} = 23\frac{13}{16}.$$

$$\textit{Ответ}: A_{(10)} = 23\frac{13}{16}.$$

Пример 2

$\textit{Дано}: B_{(8)} = 217{,}42.$ $\quad \textit{Найти}: B_{(10)}.$

Решение:

Формула разложения числа в восьмеричной системе следующая:

$$B_{(8)} = b_2 \cdot 8^2 + b_1 \cdot 8 + b_0 + b_{-1} \cdot 8^{-1} + b_{-2} \cdot 8^{-2}.$$

Подставляем в формулу значения разрядов заданного восьмеричного числа:

$$B_{(10)} = 2 \cdot 64 + 1 \cdot 8 + 7 + 4 \cdot \frac{1}{8} + 2 \cdot \frac{1}{64} = 143\frac{34}{64}.$$

$$\textit{Ответ}: B_{(10)} = 143\frac{34}{64}.$$

Пример 3

$\textit{Дано}: C_{(16)} = 2B{,}3E.$ $\quad \textit{Найти}: C_{(10)}.$

Решение:

Формула разложения числа в шестнадцатеричной системе:

$$C_{(16)} = c_1 c_0, c_{-1} c_{-2} = c_1 \cdot 16 + c_0 + c_{-1} \cdot \frac{1}{16} + c_{-2} \cdot \frac{1}{256}.$$

Подставляем значения разрядов заданного числа:

$$C_{(10)} = 2 \cdot 16 + 11 + 3 \cdot \frac{1}{16} + 14 \cdot \frac{1}{256} = 43\frac{62}{265}.$$

$$\textit{Ответ}: C_{(10)} = 43\frac{62}{256}.$$

Программа, код которой приведен в листинге 21.1, реализует алгоритм перевода, основанный на сложении количественных эквивалентов разрядов заданного числа. Программа легко трансформируется для случая перевода восьмеричных чисел в десятичную систему.

Листинг 21.1. Перевод целого $A_{(2)}$ в $A_{(10)}$.

```
Dim a As Integer, b As Integer, k As Integer
Private Sub Form_Load()
  Show
  a = InputBox("Введите двоичное целое")  'ввод числа
  Do
    b = b + (a Mod 10) * 2 ^ k  'формирование результата
    k = k + 1                   'формирование показателя степени
    a = a \ 10                  'отбрасывание учтенных разрядов
  Loop While a > 0              'проверка окончания перевода
  Print "a(10)="; b             'вывод результата
End Sub
```

Такое приложение требует запуска (загрузки формы) для перевода каждого числа. При желании организовать последовательный перевод неограниченного количества чисел можно строки кода, начиная со второй и заканчивая выводом результата, сделать телом цикла еще одной пары операторов `DoLoop`. В этом случае для исключения накапливания результата после ввода числа следует обнулить значения b и k (`b=0`, `k=0`).

Код приложения, приведенного в листинге 21.2, использует для перевода чисел строковые функции. Запуск процедуры перевода чисел выполняется кнопкой **Перевод**. Имеется кнопка завершения работы **Выход**.

Листинг 21.2. Перевод чисел с использованием строковых функций

```
Dim a As String
Dim c As Integer
Private Sub cmdПеревод_Click()
  a = InputBox("Введите целое двоичное число", "Перевод чисел")
  c = 0
  For i = 1 To Len(a)
    c = 2 * c + Val(Mid(a, i, 1))
  Next
```

```
  Print "a(2)="; a, "a(10)="; c
End Sub
Private Sub cmdВыход_Click()
  End
End Sub
```

При переводе чисел с использованием формулы разложения числа все действия выполняются в новой системе счисления. В компьютерах этот метод находит применение при переводе чисел из десятичной системы в "привычную" для компьютера двоичную систему.

Перевод целых чисел делением на основание новой системы

Алгоритм перевода чисел заключается в следующем. Сначала исходное число, затем получающиеся частные делим на основание новой системы. Действия выполняем в старой системе. Записываем *последнее частное* и *остатки* в обратном порядке их получения. Полученное число является записью заданного числа в новой системе.

Пример 4

Дано : $A_{(10)} = 23.$ *Найти* : $A_{(2)}.$

Решение :

$\frac{23}{2} = 11,$ *Остаток* 1; $\frac{11}{2} = 5,$ *Остаток* 1;

$\frac{5}{2} = 2,$ *Остаток* 1; $\frac{2}{2} = 1,$ *Остаток* 0.

Ответ : $A_{(2)} = 10111.$

Проверку правильности полученного результата удобно выполнить, используя формулу разложения числа по степеням основания. Так как действия выполняются в старой системе, рассматриваемое правило удобно использовать при переводе чисел из десятичной системы.

Пример 5

Дано : $A_{(10)} = 92.$ *Найти* : $A_{(8)}.$

Решение :

$\frac{92}{8} = 11,$ *Остаток* 4; $\frac{11}{8} = 1,$ *Остаток* 3.

Ответ : $A_{(8)} = 134.$

Пример 6

$Дано: A_{(10)} = 78. \quad Найти: A_{(16)}.$

$Решение:$

$\frac{78}{16} = 4, \quad Остаток\ 14.$

$Ответ: A_{(16)} = 4E.$

Код программы, реализующей алгоритм перевода числа из десятичной в двоичную систему, приведен в листинге 21.3.

Листинг 21.3. Перевод чисел 10 — 2

```
Private Sub Form_Load()
  Show
  a = InputBox("Введи десятичное число")
  b = a   'сохранение a для печати
  k = 0   'номера двоичных разрядов
  Do
    r = a Mod 2 'определение остатков
    a = a \ 2   'определение частных
    c = r * 10 ^ k + c 'формир. результатата
    k = k + 1
  Loop While a > 0
  Print "A(10) = "; b, "A(2) = "; c
End Sub
```

Перевод правильных дробей умножением на основание

Алгоритм перевода правильных дробей из одной системы счисления в другую следующий. Последовательно умножаем сначала исходное число, затем дробные части получаемых произведений на основание новой системы. При этом в целую часть будут выходить цифры записи числа в новой системе. Действия выполняются в старой системе.

Пример 7

Дано : $A_{(10)} = 0{,}375$. *Найти* : $A_{(2)}$.

Решение :

$$
\begin{array}{r}
0,\,375 \\
\times \quad\quad 2 \\
\hline
0\;\;750 \\
\times \quad\quad 2 \\
\hline
1\;\;500 \\
\times \quad\quad 2 \\
\hline
1\;\;000
\end{array}
$$

Ответ : $A_{(2)} = 0{,}011$.

В примере 7 в целые части произведений вошли цифры 011. Можно предположить, что результат равен $A_{(2)} = 0{,}011$. Проверим полученный результат обратным переводом:

$$0{,}011_{(2)} = \frac{1}{2} + \frac{1}{8} = \frac{3}{8}_{(10)} = 0{,}375_{(10)}.$$

В рассмотренном примере перевод заканчивается на третьем шаге, т. к. дробная часть становится равной нулю. Это бывает далеко не всегда.

Поставим следующие вопросы. Когда нужно заканчивать умножение на основание новой системы? Сколько нужно получать значащих разрядов результата перевода?

Можно принять естественное правило: точность записи числа в новой системе должна быть не ниже точности числа в старой системе. Например, если число $C_{(10)} = 0{,}6$, т. е. задано с точностью до 1/10, то запись этого числа в двоичной системе должна иметь четыре разряда дробной части. Это обеспечит точность до 1/16 (три разряда мало, т. к. третий разряд имеет вес $1/8 < 1/10$). Если же исходное десятичное число задано, например, с точностью до 1/100, то принятое правило требует получения в двоичной системе семи разрядов после запятой ($2^{-7} = 1/128$), в восьмеричной системе — трех разрядов ($8^{-3} = 1/512$), в шестнадцатеричной системе — двух разрядов ($16^{-2} = 1/256$). При вычислениях следует использовать правила округления результатов. Другим ограничением может служить заранее заданное количество разрядов для записи числа в новой системе.

Пример 8

Дано : $B_{(10)} = 0{,}67$. *Найти* : $B_{(2)}$.

Решение :

0,67	1,34	0,68	1,36	0,72	1,44	1,88	1,76
× 2	× 2	× 2	× 2	× 2	× 2	× 2	× 2
1,34	0,68	1,36	0,72	1,44	0,88	1,76	1,52

$B_{(2)} = 0{,}10101011$.

Ответ : $B_{(2)} = 0{,}101011$.

В примере 8, записав последовательно целые части произведений (они подчеркнуты), получаем результат: $B_{(2)} = 0{,}10101011$. Вычислено восемь двоичных разрядов. Так как исходное число задано с точностью 1/100, а $2^{-8} = 1/256$, то получен один лишний разряд, который используем для округления результата.

Выполним проверку. Переведем полученный результат в десятичную систему:

$$B_{(10)} = \frac{1}{2} + \frac{1}{8} + \frac{1}{32} + \frac{1}{64} = \frac{43}{64} = 0{,}672 \approx 0{,}67.$$

Лишние 0,002 получены в результате округления в большую сторону.

В заключение в примере 9 рассмотрим схему перевода правильной дроби. Будем использовать рассмотренное выше правило умножения дробных частей получающихся произведений на основание новой системы. Получающиеся в каждом шаге цифры записи числа в двоичной системе в примере 9 подчеркнуты.

Пример 9

Дано : $C_{(10)} = \frac{9}{11}$. *Найти* : $C_{(2)}$.

Решение :

$$\underline{0},\frac{9}{11}\cdot 2 \to \frac{18}{11} \to \underline{1}\,\frac{7}{11}\cdot 2 \to \frac{14}{11} \to \underline{1}\,\frac{3}{11}\cdot 2 \to \frac{6}{11} \to \underline{0}\,\frac{6}{11}\cdot 2 \to \frac{12}{11} \to$$

$$\underline{1}\,\frac{1}{11}\cdot 2 \to \ldots$$

Ответ : $C_{(2)} = 0{,}1101$.

Выписав подчеркнутые цифры, получаем результат $C_{(2)} = 0{,}1101$ с недостатком. Это число $C_{(10)} = 13/16$. Перевод исходного числа (9/11) и результата (13/16) в десятичную дробь дает результаты: 0,818 и 0,813.

Смешанная дробь переводится в новую систему счисления по частям: целая часть — методом деления, дробная часть — методом умножения на основание новой системы.

Поразрядные способы перевода

Перевод чисел существенно упрощается, если основания старой (p) и новой (q) систем связаны соотношением $p = q^k$ или $p^k = q$, где k — целое число. Это, например, системы восьмеричная и двоичная ($k = 3$), шестнадцатеричная и двоичная ($k = 4$), девятеричная и троичная ($k = 3$) и т. д.

Рассмотрим на примерах переводы чисел из восьмеричной и шестнадцатеричной систем в двоичную и обратно.

Пример 10

Дано : $A_{(8)} = 132{,}52$. *Найти* : $A_{(2)}$.

Решение :

Для получения результата нужно каждую восьмеричную цифру заданного числа записать тремя двоичными разрядами — триадой ($k = 3$):

$A_{(8)} = \quad 1 \quad 3 \quad 2, \quad 5 \quad 2;$

$A_{(2)} = 001 \quad 011 \quad 010, \quad 101 \quad 010;$

Ответ : $A_{(2)} = 1011010{,}10101.$

Пример 11

Дано : $B_{(16)} = 20E, B8;$ *Найти* : $B_{(2)}$.

Решение :

В этом примере каждая шестнадцатеричная цифра записывается четырьмя двоичными разрядами — тетрадой ($k = 4$):

$B_{(16)} = \quad 2 \quad 0 \quad E, \quad B \quad 8;$

$B_{(2)} = 0010 \quad 0000 \quad 1110, \quad 1011 \quad 1000;$

Ответ : $B_{(2)} = 1000001110{,}10111.$

Пример 12

Дано : $C_{(2)} = 11001111{,}00011$. *Найти* : $C_{(8)}, C_{(16)}$.

Решение :

Для перевода нужно разбить заданное двоичное число влево и вправо от запятой на триады (тетрады), при необходимости дополняя их нулями. Затем

каждую триаду (тетраду) записать цифрами восьмеричной (шестнадцатеричной) системы:

$C_{(2)} = 11001111{,}00011;$ $C_{(2)} = 11001111{,}00011;$

$C_{(2)} = 011\ \ 001\ \ 111,\ 000\ \ 110;$ $C_{(2)} = 1100\ \ 1111,\ 0001\ \ 1000;$

$C_{(8)} = \ \ 3\ \ \ 1\ \ \ 7,\ \ 0\ \ \ 6;$ $C_{(16)} = \ B\ \ \ F,\ \ 1\ \ \ 8;$

$Ответ: C_{(8)} = 317{,}06.$ $Ответ: C_{(16)} = BF{,}18.$

Поразрядные способы перевода чисел можно использовать для сокращения действий при переводе числа, например, из десятичной системы в двоичную. Для этого целое число делением (дробное — умножением) сначала переводят в восьмеричную систему, а затем из восьмеричной системы поразрядно в двоичную систему.

Если в качестве промежуточной системы использовать двоичную, то существенно упрощается перевод из восьмеричной системы в шестнадцатеричную, и обратно. Это показано в следующем примере.

Пример 13

$Дано: A_{(8)} = 275{,}034.$ $Найти: A_{(16)}.$

Решение:

$A_{(8)} = \ \ 2\ \ \ \ 7\ \ \ \ 5,\ \ \ 0\ \ \ \ 3\ \ \ \ 4;$

$A_{(2)} = \ 010\ \ \ 111\ \ \ 101,\ \ \ 000\ \ \ 011\ \ \ 100;$

$A_{(2)} = 1011\ \ \ 1101,\ \ 0000\ \ \ 1110;$

$A_{(16)} = \ \ B\ \ \ \ D,\ \ \ 0\ \ \ \ E;$

$Ответ: A_{(16)} = BD{,}0E.$

Быстрый способ перевода, использующий устный счет

Записав единицу, приписываем к ней справа нули (1, 10, 100, 1000, 10000, ...) и переводим в десятичную систему. Получаем числа 1, 2, 4, 8, 16, ... С приписыванием справа нуля двоичное число увеличивается вдвое. Если же приписать единицу, то число увеличится вдвое плюс единица.

Пример 14

$Дано: A_{(2)} = 1010011{,}100101.$ $Найти: A_{(10)}.$

Решение:

Последовательно открывая разряды целой части числа, получаем:

1, 2, 4 + 1 = 5, 10, 20, 40 + 1 = 41, 82 + 1 = 83.

С дробной частью поступаем так же:

1, 2, 4, 8 + 1 = 9, 18, 36 + 1 = 37.

Шестой разряд после запятой имеет вес $2^6 = 64$. Поэтому дробная часть равна $\frac{37}{64}$.

Ответ : $A(10) = 83\frac{37}{64}$.

21.2. Сравнение систем счисления

На первый взгляд вне конкуренции очень хорошо знакомая нам десятичная система. Однако она используется только при вводе информации в компьютер и выводе из него. Это обусловлено следующими причинами.

При хранении и передаче информации каждую цифру необходимо представлять некоторой физической величиной, например, амплитудой напряжения, тока, направлением намагниченности магнитного материала и т. п. В условиях помех, чем больше число градаций этих физических величин (для десятичной системы 10), тем больше вероятность переходов с одной градации к другой и вероятность появления ошибок. Это приводит к уменьшению надежности хранения и передачи информации. Вероятность появления таких ошибок минимальна при использовании двоичной системы.

При кодировании информации в двоичной системе наиболее просто технологически реализуются электронные схемы, выполняющие операции над числами (транзистор открыт или закрыт, импульс тока есть или нет, участок поверхности магнитного диска намагничен или размагничен). К тому же и действия над двоичными числами, как было показано, выполняются весьма просто.

К недостаткам двоичной системы следует отнести необходимость и трудоемкость перевода чисел из десятичной системы при вводе информации в компьютер и в десятичную систему при выводе результатов. Отметим также, что двоичная система счисления самая неэкономная по записи чисел. Она требует больше разрядов, чем запись того же числа в других системах.

Задания для самостоятельной работы

Перевести числа из одной системы счисления в другую:

1. Дано: $A_{(2)} = 101011{,}000111$. Найти: $A_{(8)}$, $A_{(10)}$, $A_{(16)}$.
2. Дано: $B_{(8)} = 150{,}74$. Найти: $B_{(2)}$, $B_{(10)}$, $B_{(16)}$.
3. Дано: $C_{(10)} = 87{,}29$. Найти: $C_{(2)}$, $C_{(8)}$, $C_{(16)}$.

4. Дано: $D_{(16)} = A0,F8$. Найти: $D_{(2)}$, $D_{(8)}$, $D_{(10)}$.
5. Дано: $E_{(10)} = 7/11$. Найти: $E_{(10)}$.
6. Дано: $F_{(5)} = 304,12$. Найти: $F_{(10)}$.
7. Дано: $G_{(10)} = 47,8$. Найти: $G_{(3)}$.
8. Дано: $H_{(\text{р.с.с.})} = \text{MDCCCXII}$. Найти: $H_{(10)}$.
9. В задании 8 под буквами (р.с.с.) понимается "римская система счисления".

Выполнить действия:

1. Дано: $A_{(2)} = 10101$, $B_{(2)} = 111$. Найти: $C_{(2)} = A_{(2)} + B_{(2)}$.
2. Дано: $D_{(2)} = 1101$, $E_{(2)} = 1011$. Найти: $F_{(2)} = D_{(2)} \cdot E_{(2)}$.

21.3. Способы представления чисел в компьютере

Желание обойтись в процессоре одним только сумматором привело к замене вычитания чисел сложением специальных кодов чисел. Поясним это простейшим примером действий с числами в десятичной системе.

Пусть требуется выполнить действие 74 − 48. Код положительного числа равен самому числу. Код отрицательного числа формируется как дополнение этого числа до 100: 100 − 48 = 52. В компьютере это выполняется очень просто, без применения вычитания.

При сложении кодов чисел получаем результат 74 + 52 = 126. При этом возникает перенос (одна сотня) из старшего разряда чисел (переполнение). Этот перенос отбрасывается, устраняется схемным путем. Результат вычитания равен 26.

В качестве кодов используют прямой, обратный и дополнительный коды. Машинный код числа состоит из знакового разряда и цифровых разрядов. Принято кодировать знак плюс цифрой 0, знак минус — цифрой 1. Для того чтобы отличить запись машинных кодов от других записей чисел, в ней используют квадратные скобки. Знаковый разряд от цифровых разрядов будем отделять точкой.

Примем для простоты примеров разрядность чисел равной пяти разрядам: один разряд для записи знака и четыре разряда для записи цифровых разрядов числа. Рассмотрим примеры записи чисел в виде машинных кодов. В качестве чисел будем брать числа $1 > A \geq 1/16$, т. е. правильные дроби.

Прямой код

Прямой код формируется посредством записи в знаковый разряд знака — числа, в цифровые разряды — значений цифровых разрядов числа.

Пример 15

$$Дано: A_{(10)} = \frac{12}{16},\ B_{(10)} = -\frac{5}{16}.$$

Найти: запись A и B в прямом коде.

Решение: $A_{(2)} = +0{,}1100; \quad B_{(2)} = -0{,}0101;$

Ответ: $[A]_{пк} = 0.1100;\ [B]_{пк} = 1.0101.$

Обратный код

Обратный код положительных чисел равен их прямому коду. Обратный код отрицательных чисел формируется из прямого и прямой код из обратного инверсией (изменением на обратное, противоположное значение) цифровых разрядов числа.

Пример 16

$$Дано: A_{(10)} = \frac{5}{16},\ B_{(10)} = -\frac{13}{16}.$$

Найти: запись A и B в прямом и обратном коде.

Решение: $A_{(2)} = +0{,}0101;\ B_{(2)} = -0{,}1101;$

Ответ: $[A]_{пк} = 0.0101;\ [B]_{пк} = 1.1101; \quad [A]_{ок} = 0.0101;\ [B]_{ок} = 1.0010.$

Дополнительный код

Дополнительный код положительных чисел равен их прямому коду. Дополнительный код отрицательных чисел формируется как дополнение заданного числа до единицы. Дополнительный код удобно получать из обратного кода прибавлением единицы к младшему цифровому разряду.

Пример 17

$$Дано: A_{(10)} = \frac{5}{16}, B_{(10)} = -\frac{13}{16}.$$

Найти: $[A]_{пк},\ [A]_{ок}, [A]_{дк}, [B]_{пк}, [B]_{ок}, [B]_{дк}.$

Решение: $A_{(2)} = +0{,}0101; \quad B_{(2)} = -0{,}1101;$

Ответ: $[A]_{пк} = 0{,}0101;\ [A]_{ок} = 0{,}0101;\ [A]_{дк} = 0{,}0101;$

$[B]_{пк} = 1.1101;\ [B]_{ок} = 1.0010;\ [B]_{дк} = 1.0011.$

Выполнение арифметических операций

При выполнении операций со знаковыми разрядами кодов чисел действуют как с цифровыми. При действиях в обратном коде перенос из знакового разряда циклически переносится к младшему цифровому разряду и суммируется с ним, а в дополнительном коде перенос из знакового разряда отбрасывается и не учитывается.

Результаты выполнения действий в примерах будем переводить в прямой код. При положительном результате $[A]_{ок} = [A]_{дк} = [A]_{пк}$. Прямой код отрицательных чисел из обратного кода получаем инверсией цифровых разрядов. Прямой код отрицательных чисел из дополнительного получаем инверсией значений цифровых разрядов и прибавлением единицы младшего разряда.

Пример 18

$$Дано: A_{(10)} = \frac{5}{16}, B_{(10)} = -\frac{12}{16}.$$

Найти: $C = A + B$, действия выполнить в кодах.

Решение:

$A_{(2)} = 0{,}0101; \quad B_{(2)} = -0{,}1100;$

$[A]_{пк} = 0.0101;$	$[B]_{пк} = 1.1100$	
	$+ \quad [A]_{ок} = 0.0101$	$+ \quad [A]_{дк} = 0.0101$
	$\underline{[B]_{ок} = 1.0011}$	$\underline{[B]_{дк} = 1.0100}$
Ответы:	$[C]_{ок} = 1.1000$	$[C]_{дк} = 1.1001$
	$[C]_{пк} = 1.0111$	$[C]_{пк} = 1.0111$
	$C_{(2)} = -0{,}0111$	$C_{(2)} = -0{,}0111$
	$C_{(10)} = -\frac{7}{16};$	$C_{(10)} = -\frac{7}{16}.$

Пример 19

$$Дано: A_{(10)} = \frac{11}{16}, B_{(10)} = \frac{3}{16}.$$

Найти: $C = A - B$, действия выполнить в обратном и дополнительном кодах.

Решение :

$[A]_{пк} = 0.1011;$ $\quad [A]_{ок} = 0.1011$ $\quad [A]_{дк} = 0.1011$

$[-B]_{пк} = 1.0011;$ $\quad +[-B]_{ок} = 1.1100$ $\quad +[-B]_{дк} = 1.1101$

10.0111 $\quad 10.1000$

$+ \quad 1$

Ответы : $\quad [C]_{ок} = 0.1000$ $\quad [C]_{дк} = 0.1000$

$C_{(10)} = \frac{8}{16};$ $\quad C_{(10)} = \frac{8}{16.}$

Для получения ответа в обратном коде перенос из старшего разряда суммируется с младшим разрядом результата. В дополнительном коде такой перенос отбрасывается.

Приведенные примеры показывают, что действия над числами в кодах приводят к получению правильных результатов, т. е. сумма кодов дает код суммы.

Переполнение и машинные нули

Одна из важнейших характеристик компьютера — разрядность обрабатываемых им двоичных чисел. Количество разрядов, отводимых в компьютере для хранения, передачи и обработки двоичных чисел называют разрядной сеткой компьютера. Первые процессоры на интегральных схемах были четырехразрядными. В настоящее время широко распространены 16-разрядные и 32-разрядные процессоры. Появились 64-разрядные процессоры. Чем больше разрядность, тем больше диапазон обрабатываемых чисел, тем выше точность вычислений.

Если результат выполнения операции выходит за диапазон представления чисел и не может быть записан в заданном числе разрядов, то такую ситуацию называют переполнением разрядной сетки. С переполнением (Overflow) мы нередко имеем дело при решении на компьютере задач вычислительного характера.

Если же результат настолько мал, что не может быть представлен в заданной разрядной сетке, то компьютером в дальнейшем он принимается равным нулю. Ситуацию, когда число не равно нулю, но принимается компьютером за ноль, называют появлением машинного нуля.

Рассмотрим примеры.

Пример 20

Дано : $A_{(10)} = \frac{11}{16};\ B_{(10)} = \frac{9}{16}.$

Найти : $C = A + B.$

Действия выполнить в обратном коде.

Решение : $A_{(2)} = 0{,}1011; [A]_{пк} = 0.1011; B_{(2)} = 0{,}1001; [B]_{пк} = 0.1001.$

а) $[A]_{ок} = 0.1011$
$+ [B]_{ок} = 0.1001$
$[C]_{ок} = 1.0100$

б) $[A]_{мок} = 00.1011$
$+ [B]_{мок} = 00.1001$
$[C]_{мок} = 01.0100.$

Оценим результат решения примера 20, *а*. Суммировали два положительных числа, а получили отрицательный результат. Перенос из старшего цифрового разряда занял знаковый разряд и исказил знак результата. Произошло переполнение разрядной сетки, сумма заданных чисел больше единицы.

Для выявления подобных переполнений применяют модифицированные коды, отличающиеся тем, что они имеют по два знаковых разряда, как показано в примере 20, *б*. Положительные числа в модифицированных кодах имеют в знаковых разрядах два нуля (00.), отрицательные — две единицы (11.). Признаком переполнения служит различие значений знаковых разрядов (01. или 10.). При этом первый знак указывает на знак результата.

Пример 21

Дано : $A_{(10)} = \frac{2}{16}$, $B_{(10)} = \frac{5}{16}$. *Найти* : $C = A \times B$.

Решение :

При умножении чисел знак результата определяется по знакам сомножителей логическим путем по известным правилам. Цифровые разряды произведения определяются перемножением цифровых разрядов сомножителей.

$A_{(2)} = 0{,}0010;$
$B_{(2)} = 0{,}0101;$

$0{,}0010$
$\times 0{,}0101$
0010
$+ 0010$
$0{,}00001010$

Ответ : $C_{(2)} = 0{,}0000.$

В результате умножения получили результат, значащие цифры которого не вписываются в разрядную сетку.

Произведение равно $0{,}00001010_{(2)} = 10/256_{(10)}$. В разрядную сетку записывается $C_{(2)} = 0{,}0000$, т. е. машинный ноль.

Для уменьшения возможности появления машинных нулей нужно или увеличивать число разрядов в разрядной сетке компьютера, или вводить некоторые масштабные множители. Например, результат в примере 21 можно

записать так: $C_{(2)} = 0{,}1010 \times 2^{-4}$. Использование масштабных коэффициентов реализовано в нормальной форме представления чисел.

21.4. Формы представления чисел в компьютере

Различают естественную, с фиксированной точкой (запятой) и нормальную (экспоненциальную), с плавающей точкой (запятой) формы представления чисел.

При *естественной* форме представления чисел точка (запятая), отделяющая разряды целой части числа от дробной части, фиксируется в определенном месте разрядной сетки, например, перед старшим цифровым разрядом, непосредственно после знакового разряда. В этом случае процессор оперирует с правильными дробями, что исключает переполнение при умножении и облегчает подбор масштабных коэффициентов.

Шестнадцатиразрядная сетка позволяет получить 216 различных чисел без знака. Знак занимает один разряд разрядной сетки. При выполнении действий над числами возможно получение результата в виде +0 (0.0000) и −0 (1.0000). Последнее представление нуля в компьютере изменяется на +0. Двойное кодирование нуля уменьшает общее количество представляемых чисел на одно число.

Нормальную форму также называют *экспоненциальной*. Нормальная форма позволяет при одинаковой разрядности получать существенно больший диапазон представления чисел. Это приводит к уменьшению вероятности переполнения разрядной сетки и появления машинных нулей. Кроме того, нормальная форма просто решает вопрос изменения значений сопровождающих числа масштабных коэффициентов.

До недавнего времени процессоры обрабатывали числа в форме с плавающей точкой по специальным микропрограммам. Затем к обычным процессорам в помощь стали добавлять так называемые математические процессоры, работающие с числами с плавающей точкой в 5—15 раз быстрее обычных процессоров. Современные процессоры обрабатывают числа как в естественной, так и в нормальной форме.

В нормальной, экспоненциальной форме числа представляются в виде $A = Ma \times 2^{Pa}$, где Ma — мантисса числа, Pa — порядок.

Каждое число записывается четырьмя элементами: "<знак мантиссы, он же знак числа> + <мантисса> + <знак порядка> + <порядок>". В современных компьютерах примерное соотношение числа разрядов между этими элементами следующее: 1 + 23 + 1 + 7 = 32 — для вещественных чисел одинарной точности и 1 + 52 + 1 + 10 = 64 — для чисел двойной точности.

Сравним диапазоны представления чисел для естественной и нормальной форм. Для наглядности возьмем десятиразрядные числа. В естественной форме один разряд занят знаком числа и девять разрядов — цифровые. Диапазон представления чисел при этом: от $-(2^9 - 1)$ до $+(2^9 - 1)$, т. е. от -511 до $+511$.

В нормальной форме отведем 6 разрядов под мантиссу со знаком и 4 разряда под порядок со знаком. В этом случае максимальное значение мантиссы равно $2^5 - 1 = 31$, а максимальный порядок равен $2^3 - 1 = 7$, поэтому диапазон представления чисел: от -31×2^7 до $+31 \times 2^7$, т. е. от -31×128 до $+31 \times 128$.

Сравнение показывает, что при одинаковой разрядности диапазон чисел в экспоненциальной форме значительно больше, чем в естественной форме. Заметим, что с увеличением разрядности это различие в диапазонах быстро возрастает.

Можно ошибочно предположить, что с увеличением диапазона увеличивается и количество представляемых различных чисел. Но при заданной разрядности количество различных комбинаций нулей и единиц увеличить невозможно. Количество чисел при представлении в нормальной форме меньше, чем при представлении их в естественной форме. Сказывается двойственное представление нуля (+0 и −0), но теперь уже в значениях порядка.

Если в естественной форме шаг изменения чисел постоянный и равен единице младшего разряда чисел, то экспоненциальная форма приводит к неравномерному шагу, существенно увеличивающемуся к краям диапазона.

Действия над числами в нормальной форме

Как правило, все операнды поступают из памяти для выполнения операций и результаты операций отправляются для хранения в память в нормализованном виде, т. е. старший значащий разряд мантиссы записан непосредственно после знакового разряда числа. Нормализованное число представляется с максимально возможной точностью.

Пример 22

Дано: $A_{(2)} = 11{,}1$; $B_{(2)} = 0{,}01$; $C_{(2)} = -0{,}1011$.

Найти: запись A, B и C в экспоненциальной форме, в нормализованном виде, в прямом коде; разрядность: 6 разрядов для мантиссы со знаком, 4 разряда для порядка.

Решение :

$A_{(2)} = 11{,}1 = +0{,}111 \times 10^{+010}$; $B_{(2)} = 0{,}01 = +0{,}10 \times 10^{-001}$; $C_{(2)} = -0{,}1011 \times 10^{+000}$.

Ответ :

$[A]_{пк} = 0.11100.0.010$; $[B]_{пк} = 0.10000.1.001$; $[C]_{пк} = 1.10110.0.000$.

При выполнении операций сложения и вычитания чисел с плавающей точкой необходимо предварительно выровнять порядки. При этом сдвигается вправо число, имеющее меньший порядок, на количество разрядов, равное разности порядков. Рассмотрим примеры.

Пример 23. Сложение

Дано : $A_{(10)} = 2\frac{1}{4}$; $A_{(2)} = 10{,}01$; $B_{(10)} = -\frac{3}{8}$; $B_{(2)} = -0{,}011$.

Найти : $C = A + B$.

Сложение выполнить в нормальной форме, в обратном коде.

Решение : $A_{(2)} = 10.01$, $[A]_{пк} = [A]_{ок} = 0.10010.0.010$;

$B_{(2)} = -0.011$, $[B] = 1.11000.1.001$, $[B] = 1.00111.1.110$.

Определяем разность порядков:

$$[Pa]_{ок} = 0.010, \quad [Pb]_{ок} = 1.110, \quad [-Pb]_{ок} = 0.001.$$

$$\begin{array}{r} [Pa]_{ок} = 0.010 \\ + \quad [-Pb]_{ок} = 0.001 \\ \hline [R]_{ок} = 0.011 \end{array}$$

Положительный результат определения разности порядков R указывает на то, что меньший порядок у второго слагаемого, у числа B. Величина разности показывает, на сколько разрядов необходимо сдвинуть вправо мантиссу числа B для выравнивая порядков.

Сдвинутая на три разряда вправо мантисса равна $[Мв]_{ок} = 1.11100$. При сдвиге учитываем, что в обратном коде отрицательного числа значения разрядов инвертированы, поэтому при сдвиге освобождающиеся разряды заполняются единицами (в прямом коде — нулями).

Порядки выровнены, поэтому можно выполнить сложение мантисс и приписать им больший, теперь общий для обоих слагаемых порядок. Мантиссы складываем в модифицированном коде (два знаковых разряда числа), что позволяет выявлять переполнение разрядной сетки.

$$
\begin{array}{rl}
[Ma]_{мок} & = 00.10010 \\
+\,[Mb]_{мок} & = 11.11100 \\
\hline
 & 100.01110 \\
 & + \qquad 1 \\
\hline
[Mc]_{мок} & = \quad 00.01111
\end{array}
$$

Результат сложения мантисс положителен и ненормализован. Переполнение отсутствует. Для нормализации нужно сдвинуть мантиссу результата на один разряд влево. При этом, чтобы не исказить величину числа в целом, необходимо уменьшить на единицу порядок.

Получили результат: $[C]_{мок} = 00.01111.0.010$. После нормализации имеем: $[C]_{мок} = 0.11110.0.001$ (увеличиваем мантиссу — уменьшаем порядок).

Ответ: $[C]_{ок} = [C]_{пк} = 0.1111.0.001$; $C_{(2)} = 1.111$; $C_{(10)} = 15/16 \times 21 = 1\ 7/8$.

Умножение и деление

В нормальной форме числа A и B можно записать в следующем виде: $A = Ma \times 2^{Pa}$ и $B = Mв \times 2^{Pв}$. При умножении и делении чисел A и B справедливы следующие формулы:

$$A \times B = Ma \times 2^{Pa} \times Mв \times 2^{Pв} = Ma \times Mв \times 2^{Pa + Pв};$$

$$A/B = Ma \times 2^{Pa} \,/\, Mв \times 2^{Pв} = Ma/Mв \times 2^{Pa - Pв}.$$

Для выполнения умножения (деления) чисел в экспоненциальной форме (форме с плавающей точкой) нужно перемножить (разделить) мантиссы и приписать результату порядок, равный сумме (разности) порядков исходных чисел.

21.5. Примеры использования других систем

Теоретически было доказано, что самой экономичной по затратам оборудования для построения компьютеров является троичная система счисления. В то время, когда вопросы экономии оборудования стояли очень остро (компьютеры первого и второго поколения), была построена единственная в своем роде вычислительная машина "Сетунь", в которой использовалась троичная система.

Быстродействие компьютера зависит от скорости работы в процессоре сумматора чисел. В худшем случае, при операции сложения 11...111 + 00...001 перенос должен пройти через все разряды сумматора. Чем больше разряд-

ность сумматора, тем больше затрачивается времени на выполнение операции. Имеются системы, в которых переносы из разряда в разряд отсутствуют. К ним относится "Система счисления в Остаточных Классах" (СОК). Ее еще называют системой вычетов, китайской системой (вычетами занимались математики конфуцианской школы), чешской системой (чешский профессор Свобода первый рассматривал возможность использования СОК для построения компьютеров).

СОК — непозиционная система счисления. Основаниями системы служат взаимно простые числа, например: $p_1 = 3$, $p_2 = 5$, $p_3 = 7$. Диапазон представления чисел определяется произведением оснований: $D = p_1 \cdot p_2 \cdot p_3 = 3 \cdot 5 \cdot 7 = 105$, т. е. при этих основаниях можно однозначно представлять числа от 0 до 104. Любое число в этом диапазоне записывается остатками от его деления на выбранные основания: $A(_{COK}) = (a_1,a_2,a_3)$, где $a_1 = A \bmod p_1$, $a_2 = A \bmod p_2$, $a_3 = A \bmod p_3$. Например, число $A_{(10)} = 13$ запишется в СОК с выбранными основаниями так: $A_{(COK)} = (1, 3, 6)$.

Действия сложение, вычитание и умножение выполняются в СОК поразрядно, по модулю оснований. Это позволяет выбирать результаты из таблиц, получать схемным путем. Рассмотрим пример сложения и умножения чисел.

Сложение и умножение

Пример 24

Дано: $A_{(10)} = 12$, $B_{(10)} = 8$. *Основания СОК*: 3, 5 и 7.

Найти: $C = A + B$, $D = A \cdot B$.

Решение:

$$\begin{array}{rr} A_{(COK)} = (0, 2, 5) & A_{(COK)} = (0, 2, 5) \\ + B_{(COK)} = (2, 3, 1) & \times B_{(COK)} = (2, 3, 1) \\ \hline C_{(COK)} = (2, 0, 6) & D_{(COK)} = (0, 1, 5) \end{array}$$

Правильность результата можно проверить переводом чисел $12 + 8 = 20$ и $12 \cdot 8 = 96$ в СОК.

Ответ: $C_{(COK)} = (2, 0, 6)$, $D_{(COK)} = (0, 1, 5)$.

К сожалению, деление в системе остатков затруднено, что и определило малое распространение ее при построении компьютеров. Тем не менее на ее основе была построена быстродействующая специализированная ЭВМ, успешно решавшая задачи идентификации объектов при обзоре космического пространства.

Задания для самостоятельной работы

1. Заданные числа перевести в двоичную систему и записать в прямом, обратном и дополнительном коде (один разряд знаковый и четыре цифровых). Выполнить сложение и вычитание чисел в обратном и дополнительном кодах. Проверить правильность полученных результатов.

   а) $A_{(10)} = 11/16$, $B_{(10)} = -4/16$; б) $C_{(10)} = -6/16$, $D_{(10)} = -7/16$;

   в) $E_{(10)} = -9/16$, $F_{(10)} = 5/16$; г) $G_{(10)} = 13/16$, $H_{(10)} = 2/16$.

2. Заданные числа перевести в двоичную систему и записать в прямом коде в нормальной форме в нормализованном виде. Для записи мантиссы со знаком отвести 6 разрядов, для записи порядка со знаком — 4 разряда. Выполнить сложение и вычитание чисел в модифицированных обратном и дополнительном кодах. Результат нормализовать и проверить на правильность получения.

   а) $A_{(10)} = 7{,}5$; $B_{(10)} = 2{,}5$; б) $C_{(10)} = 3{,}25$; $D_{(10)} = -9{,}5$;

   в) $E_{(10)} = 5{,}25$; $F_{(10)} = -6{,}25$; г) $G_{(10)} = 6{,}5$; $H_{(10)} = 8{,}25$.

Глава 22

Логические основы работы компьютеров

22.1. Общие сведения о двоичных алгебрах

В широком смысле под *алгеброй* понимают раздел математики, изучающий общие свойства операций над элементами множества произвольной природы.

Чтобы *задать алгебру*, нужно задать некоторое множество элементов (в частности, переменных и констант) и определяемое для них некоторое множество операций. Кроме того, требуется задать алфавит алгебры, правила записи и преобразования формул.

В школьной алгебре в качестве множества констант и переменных выступает множество вещественных чисел, а множество операций включает операции: сложение, вычитание, умножение, деление, возведение в степень и извлечение корня. Множество входящих в алгебру операций называют *базисом*. Результаты операций определяют функции: сложение — сумму, вычитание — разность, умножение — произведение и т. д.

Заметим, что набор операций школьной алгебры избыточен. Например, возведение в степень можно заменить умножением, умножение можно заменить сложением. Вычитание также можно заменить сложением, как это делается в компьютерах, имеющих только схему сумматора. Уменьшение набора операций привело бы к громоздким, трудно обозримым формулам. Увеличение набора добавлением других операций также нежелательно.

Особое место среди алгебр занимают двоичные алгебры. Из множества двоичных алгебр познакомимся только с двумя: *алгеброй логики* и *булевой алгеброй*. Это алгебры двоичных переменных, констант и функций, принимающих только два значения: *истина* и *ложь* в алгебре логики и соответственно *единица* и *ноль* в булевой алгебре. Начнем с булевой алгебры, имеющей более простой базис, состоящий всего из трех операций.

Важность знакомства с двоичными алгебрами заключается в следующем. Во-первых, они являются математической основой построения всех логических схем компьютеров, обрабатывающих информацию в двоичной системе счисления. Во-вторых, они служат математической основой решения сложных логических задач.

22.2. Двоичные переменные и функции

Двоичными называют переменные, способные принимать только два значения: 0 и 1. Двоичные функции — это функции двоичных переменных. Они также принимают только два значения — 0 и 1. Двоичные переменные и функции называют булевыми переменными и булевыми функциями (БФ).

Область определения ***n*** аргументов БФ составляют:

при $n = 1$ — значения 0 и 1 этого аргумента ($2^n = 2$);

при $n = 2$ — наборы 00, 01, 10 и 11 — 4 набора значений аргументов ($2^n = 4$);

при $n = 3$ — наборы 000, 001, 010, 011, 100, 101, 110 и 111 — 8 наборов ($2^n = 8$) и т. д.

Можно сделать вывод, что область определения n аргументов БФ состоит из 2^n наборов. Область значений БФ — это множество всего из двух значений 0 и 1.

Конечность области определения и области значений БФ позволяют задавать эти функции не только формулами, но и таблично (табл. 22.1). В виде таблицы БФ задают значениями 0 или 1 на каждом из наборов значений ее аргументов. Такие таблицы называют *таблицами истинности* (в алгебре логики в таких таблицах вместо 0 пишут Л — ложь, вместо 1 пишут И — истина).

Таблица 22.1. *Таблица истинности*

a	x_1x_2	$f_m(x_1, x_2)$	$f_{13}(x_1, x_2)$
0	0 0	$f(0, 0)$	1
1	0 1	$f(0, 1)$	1
2	1 0	$f(1, 0)$	0
3	1 1	$f(1, 1)$	1

Таблица 22.1 представляет собой пример таблицы истинности функций $f_m(x_1, x_2)$ и $f_{13}(x_1, x_2)$ двух аргументов (x_1 и x_2).

Величина ***a*** обозначает номер набора значений аргументов, ***m*** — номер функции.

Функция $f_m(x_1, x_2)$ — это задание в общем виде любой из 16-ти функций двух аргументов. $f(0, 0)$ — значение функции при подстановке в ее формулу значений $x_1 = 0$ и $x_2 = 0$, т. е. на нулевом наборе аргументов. $f(0, 1)$ — значение функции на первом наборе аргументов и т. д.

Значение булевой функции на всех наборах своих аргументов образуют двоичное число, называемое *кодом функции*. Перевод кода функции в десятичную систему дает *номер функции*. В табл. 22.1 записана функция $f_{13}(x_1, x_2)$, с кодом 1101 и номером 13.

С ростом числа аргументов n число различных функций резко возрастает. Если число наборов определяется формулой $K_н = 2^n$, то число функций вычисляется по формуле $K_ф = 2^{K_н}$. При $n = 1$ $K_н = 2$, $K_ф = 4$; при $n = 2$ $K_н = 4$, $K_ф = 16$; при $n = 4$ $K_н = 16$, $K_ф = 65\,536$ и т. д. При $n = 10$ $K_ф$ сравнимо с числом элементарных частиц во Вселенной.

Все функции изучить невозможно. Поэтому, как и в обычной школьной алгебре, выбирают некоторое количество функций малого числа (один, два) аргументов и применяют их в качестве операций для построения формул функций любой сложности.

Рассмотрим булевы функции одного и двух аргументов и подробно остановимся на функциях, входящих в базис булевой алгебры, базис алгебры логики, базисы других двоичных алгебр.

22.3. Булевы функции одного аргумента

Все четыре функции на двух наборах одного аргумента показаны в табл. 22.2.

Таблица 22.2. *Функции одного аргумента*

x	$f_0(x)$	$f_1(x)$	$f_2(x)$	$f_3(x)$
0	0	0	1	1
1	0	1	0	1

Функция $f_0(x)$ называется *константа ноль*. Она независимо от значений аргумента равна нулю.

Функция $f_3(x)$ называется *константа единица*. Она независимо от значений аргумента равна единице.

Функция $f_1(x)$ — это *переменная x*. Она повторяет значения переменной.

Функция $f_2(x)$ называется *инверсией* (*отрицание x*). Она входит в базис булевой алгебры и алгебры логики. Рассмотрим ее подробнее.

Инверсия — функция одного аргумента. Логическая операция над аргументом — *отрицание*. Часто отождествляют функцию с операцией и говорят: "функция отрицание" или "операция инверсия". Однако при строгом формальном подходе отождествлять результат с действием нежелательно.

Знак операции — черта над аргументом, например: $\bar{x}$. Такая запись читается: "не x" или "отрицание x". В языках программирования также широко используются логические операции, реализующие булевы функции. Функция инверсия аргумента x записывается так: `NOT x`.

Функцию инверсия в схемах компьютера реализует логический элемент *инвертор* (элемент НЕ). Схема инвертора показана на рис. 22.1. Работает инвертор так: если на входе 0, то на выходе 1, если на входе 1, то на выходе 0.

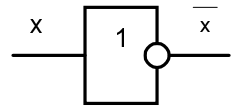


Рис. 22.1. Условное обозначение инвертора

22.4. Булевы функции двух аргументов

Двум аргументам соответствуют четыре набора их значений, что приводит к шестнадцати различным кодам функций ($n = 2$, $K_н = 2^n = 4$, $K_ф = 2^{K_н} = 16$). Все 16 функций двух аргументов записаны в табл. 22.3.

Таблица 22.3. *Булевы функции двух аргументов*

a	x_1x_2	f_0	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}	f_{11}	f_{12}	f_{13}	f_{14}	f_{15}
0	00	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
1	01	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
2	10	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
3	11	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

Среди функций двух аргументов имеются уже знакомые нам по функциям одного аргумента:

f_0 и f_{15} — соответственно функция константа 0 и ее инверсия — константа 1;

f_3 и f_{12} — переменная x_1 и ее инверсия $\bar{x}_1$;

f_5 и f_{10} — переменная x_2 и ее инверсия $\bar{x}_2$.

Функции f_0 и f_{15} фиктивно зависят от обоих аргументов. Изменения значений аргументов не влияют на значение этих функции. Функции f_3, f_5, f_{10} и f_{12} фиктивно зависят от одного аргумента и существенно зависят от другого аргумента.

Из оставшихся десяти функций также можно выделить пять пар — пять функций и их инверсий: f_1 и f_{14}, f_2 и f_{13}, f_4 и f_{11}, f_6 и f_9, f_7 и f_8. Рассмотрим их.

Конъюнкция

Булева функция $f_1(x_1,x_2)$ с кодом 0001 называется конъюнкцией. *Конъюнкция* — это такая булева функция, которая равна единице тогда и только тогда, когда все аргументы функции равны единице. Другое определение — это такая функция, которая равна нулю, если хотя бы один аргумент функции равен нулю.

Таблица истинности функции конъюнкция — см. табл. 22.4.

Функцию конъюнкция получаем как результат операции *логическое умножение*. Знак операции: &. В формулах, как и в обычной алгебре, знак чаще всего опускается: $f(x_1,x_2) = x_1 \& x_2 = x_1 \cdot x_2 = x_1 x_2$. Читается формула так: "x_1 и x_2".

Запись в языках программирования: `x1 AND x2`.

Таблица 22.4. *Таблица истинности функции конъюнкция*

a	$x_1\ x_2$	$f_1(x_1 x_2)$
0	0 0	0
1	0 1	0
2	1 0	0
3	1 1	1

Функцию конъюнкция реализует логический элемент *конъюнктор* (элемент И). Условное обозначение конъюнктора в логических схемах показано на рис. 22.2.

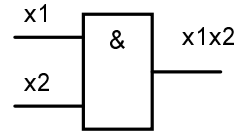


Рис. 22.2. Обозначение конъюнктора

Используя принцип суперпозиции (подстановку функции в качестве аргументов в другую функцию), функцию конъюнкция можно обобщить на n аргументов: $f(x_1, x_2, ..., x_n) = x_1x_2 ...x_n$.

В качестве содержательного примера реализации функции конъюнкция рассмотрим схему голосования "только все!" На рис. 22.3 показана цепь с N кнопками, позволяющими включать индикаторную лампочку. На электрических выключателях принято отмечать: 0 — выключено и 1 — включено. Лампочка засветится только в том случае, если будут замкнуты все ключи, т. е. на все N входов будут "поданы" единицы. Такая схема реализует функцию конъюнкция.

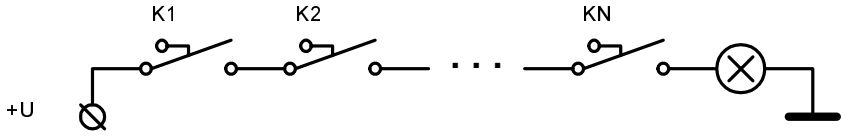


Рис. 22.3. Реализация конъюнкции в схеме голосования "только все!"

Дизъюнкция

Булева функция $f_7(x_1,x_2)$ с кодом функции 0111 называется дизъюнкцией. *Дизъюнкция* — это такая двоичная функция, которая равна нулю тогда и только тогда, когда все аргументы функции равны нулю. Другое определение: дизъюнкция — это такая функция, которая равна единице, если хотя бы один аргумент равен единице.

Таблица истинности функции дизъюнкция — см. табл. 22.5.

Функции дизъюнкции соответствует операция *логическое сложение*. Знак операции: $\vee$. Пример записи формулы функции дизъюнкция:

$$f(x_1, x_2) = x_1 \vee x_2.$$

Читается формула так: "x_1 или x_2".

Запись на языках программирования: `x1 OR x2`.

Таблица 22.5. *Таблица истинности функции дизъюнкция*

a	$x_1 x_2$	$f_1(x_1 x_2)$
0	0 0	0
1	0 1	1
2	1 0	1
3	1 1	1

Функцию дизъюнкция реализует логический элемент *дизъюнктор* (элемент ИЛИ). Условное обозначение дизъюнктора показано на рис. 22.4.

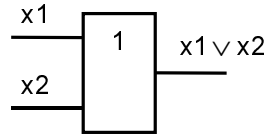


Рис. 22.4. Обозначение дизъюнктора

Функция дизъюнкция обобщается на n аргументов:

$$f(x_1, x_2, ..., x_n) = x_1 \vee x_2 \vee ... \vee x_n.$$

В качестве примера реализации функции дизъюнкция рассмотрим схему голосования "хотя бы один!" На рис. 22.5 показана цепь с N кнопками, позволяющими включать индикаторную лампочку. Лампочка засветится в том случае, если будет замкнут хотя бы один ключ, т. е. схема реализует функцию дизъюнкция.

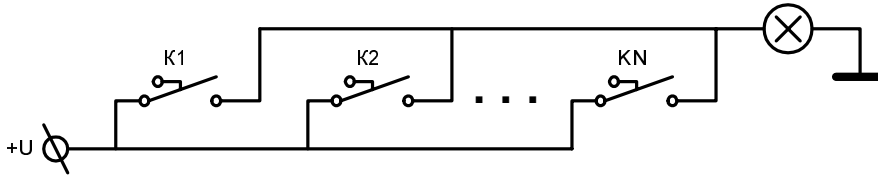


Рис. 22.5. Реализация дизъюнкции в схеме голосования "хотя бы один!"

Инверсия конъюнкции. Функция Шеффера

В таблице истинности функций двух аргументов (табл. 22.3) *функция инверсия конъюнкции* — это функция $f_{14}(x_1, x_2)$. Она имеет код функции 1110.

Запись формулы функции следующая:

$$f(x_1, x_2) = \overline{x_1 \,\&\, x_2} = \overline{x_1 \cdot x_2} = \overline{x_1 x_2}\,.$$

Инверсия конъюнкции, как видно из ее формулы, образована из двух более простых функций: конъюнкции и инверсии. Эта функция замечательна тем, что она в единственном числе образует базис алгебры — алгебры Шеффера, позволяющей записать в виде формулы любую двоичную логическую функцию.

В алгебре Шеффера единственная функция — *функция Шеффера*. Знак операции: | — штрих Шеффера. В алгебре Шеффера функция инверсия конъюнкции записывается так:

$$f(x_1, x_2) = x_1 \mid x_2.$$

Логический элемент, реализующий функцию Шеффера, называется элементом Шеффера или элементом И-НЕ. Условное обозначение элемента И-НЕ показано на рис. 22.6. Функция Шеффера также обобщается на n аргументов.

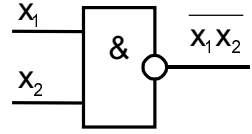


Рис. 22.6. Обозначение элемента И-НЕ

Инверсия дизъюнкции. Функция Пирса

В таблице истинности функций двух аргументов (см. табл. 22.3) *функция инверсия дизъюнкции* — функция под номером 8. Она имеет код функции 1000. Запись формулы функции:

$$f_8(x_1, x_2) = \overline{x_1 \vee x_2}\ .$$

Функцию инверсия дизъюнкции еще называют функцией Пирса. Она, как и функция Шеффера, универсальна в том смысле, что в единственном числе образует алгебру Пирса и позволяет записать формулой любую двоичную функцию. Знак операции: ↓ — стрелка Пирса. Запись функции Пирса:

$$f(x_1, x_2) = x_1 \downarrow x_2.$$

Логический элемент, реализующий функцию Пирса, называют элемент Пирса или элемент ИЛИ-НЕ. Условное обозначение элемента приведено на рис. 22.7.

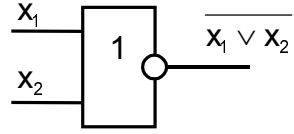


Рис. 22.7. Обозначение элемента ИЛИ-НЕ

Импликация

Функция импликация это логическая функция двух двоичных логических переменных x_1 и x_2. Различают импликацию от x_1 к x_2 (функция f_{13} с кодом

1101) и импликацию от x_2 к x_1 (функция f_{11} с кодом 1011). Обе эти функции приведены в табл. 22.6. Операцию, соответствующую функции импликация, также называют импликацией и обозначают стрелкой от одной логической переменной к другой.

Таблица 22.6. *Функции импликация*

x_1 x_2	$x_1 \to x_2$	$x_2 \to x_1$
0 0	1	1
0 1	1	0
1 0	0	1
1 1	1	1

Операция импликация входит в базис алгебры логики, где оперирует с истинными и ложными высказываниями. Об этом подробнее будет рассказано в *гл. 23*.

Запись формул функций импликация в булевой алгебре и в алгебре логики следующая:

$$f_{13}(x_1, x_2) = \bar{x}_1 \vee x_2 = x_1 \to x_2;$$

$$f_{11}(x_1, x_2) = x_1 \vee \bar{x}_2 = x_2 \to x_1.$$

Логическая операция `IMP` (импликация) используется в языках программирования. Запись следующая: `x1 IMP x2`.

Неравнозначность

Логическую *функцию неравнозначность* двух аргументов еще называют *исключающим ИЛИ* ("или один, или другой, но не оба"), а также *суммой по модулю 2*. Код функции — 0110, номер — 6. Функция принимает единичное значение только на тех наборах значений двух аргументов, на которых эти значения различны (01 и 10). Функция неравнозначность — многоместная, ей соответствует логическая операция сумма по модулю 2. Операция входит в базис алгебры Жегалкина и имеет знак операции: $\oplus$. Иногда ее включают в базис алгебры логики. Запись формулы функции неравнозначность следующая:

в булевой алгебре:

$$f_6(x_1, x_2) = \bar{x}_1 x_2 \vee x_1 \bar{x}_2$$

в алгебре Жегалкина:

$$f_6(x_1,x_2)=x_1 \oplus x_2$$

в языках программирования:

```
x1 XOR x2
```

Равнозначность

Функция равнозначность является обратной функцией к функции неравнозначности. Функция равнозначность (код функции 1001, номер функции 9) принимает значение 1 только на наборах с одинаковыми значениями аргументов (00 и 11). В качестве операции двойной импликации включается в базис алгебры логики (знак операции: $\leftrightarrow$). Функцию равнозначность также называют *эквивалентностью*. Запись формулы функции

в булевой алгебре:

$$f_9(x_1, x_2) = \bar{x}_1\bar{x}_2 \vee x_1x_2 ,$$

в алгебре логики:

$$f_9(x_1, x_2) = x_1 \leftrightarrow x_2,$$

в языках программирования:

```
x1 EQV x2.
```

Функции запрета

Функции запрета — это функции f_2 и f_4 (см. табл. 22.3). Они являются обратными функциями к функциям импликации. Функции запрета широко использовались при проектировании и описании работы логических элементов ЭВМ 1-го и 2-го поколений.

22.5. Алгебры. Сравнение по набору операций

Школьная алгебра и двоичные алгебры значительно отличаются друг от друга по множеству переменных и констант и по набору операций, производимых с ними. В табл. 22.7 показаны эти отличия.

Таблица 22.7. *Школьная и двоичные алгебры*

Алгебра	Множество переменных и констант	Базис алгебры (набор операций)
Школьная	вещественные числа	+, –, *, /, $\sqrt{\ }$, степень
Булева	0 и 1 (наборы)	&, $\vee$, $\neg$

Таблица 22.7 *(окончание)*

Алгебра	**Множество переменных и констант**	**Базис алгебры (набор операций)**
Шеффера	0 и 1 (наборы)	$\mid$ — штрих Шеффера
Пирса	– " –	$\downarrow$ — стрелка Пирса
Жегалкина	– " –	&, $\oplus$, 1
Алгебра логики	– " –	&, $\vee$, $\neg$, $\rightarrow$, $\leftrightarrow$

Из перечисленных в табл. 22.7 двоичных алгебр широко используются на практике только две — алгебра Буля (булева) и алгебра логики. Алгебры Шеффера и Пирса активно изучались на предмет возможности построения компьютера из однотипных элементов. К тому же, ввиду технологических особенностей, наиболее просто реализуются именно логические элементы Шеффера (И-НЕ) и Пирса (ИЛИ-НЕ).

22.6. Булева алгебра. Основные законы и тождества

Многие законы и правила преобразования формул булевой алгебры совпадают с законами в школьной алгебре. Но среди них есть тождества, присущие только булевой алгебре. Перечислим все законы булевой алгебры. Это тождества, используемые для преобразования формул логических функций.

1. $XY = YX$ — переместительные законы
   $X \vee Y = Y \vee X$ — (коммутативность)
2. $X(YZ) = (XY)Z$ — сочетательные законы
   $X \vee (Y \vee Z) = (X \vee Y) \vee Z$ — (ассоциативность)
3. $X(Y \vee Z) = XY \vee XZ$ — распределительные законы
   $X \vee YZ = (X \vee Y)(X \vee Z)$ — (дистрибутивность)
4. $XX = X$ — законы тавтологии
   $X \vee X = X$ — (идемпотентность)
5. $X \cdot 1 = X$ — законы выполнения операций
   $X \vee 0 = X$ — над константами
   $X \cdot 0 = 0$
   $X \vee 1 = 1$

6. $X\overline{X} = 0$	законы инверсии для конъюнкции
$X \vee \overline{X} = 1$	и дизъюнкции
7. $\overline{X \cdot Y} = \overline{X} \vee \overline{Y}$	законы де Моргана
$\overline{X \vee Y} = \overline{X} \cdot \overline{Y}$	
8. $XY \vee X\overline{Y} = X$	законы склеивания
$(X \vee Y)(X \vee \overline{Y}) = X$	
9. $X \vee XY = X$	законы поглощения
$X(X \vee Y) = X$	
10. $\overline{\overline{X}} = X$	закон снятия двойной инверсии

Обратим внимание на то, что большинство законов записано в две строки. При этом вторая строка образуется из первой заменой конъюнкции на дизъюнкцию, дизъюнкции на конъюнкцию и заменой констант 1 на 0 и 0 на 1. В этом выражается двойственный характер функций конъюнкция и дизъюнкция. Это замечательное свойство двойственности позволяет, например, сделать следующее. Если в логической схеме заменить все конъюнкторы на дизъюнкторы, дизъюнкторы на конъюнкторы, единичные сигналы на нулевые и нулевые на единичные, то схема будет реализовывать ту же исходную функцию.

Докажем справедливость некоторых тождеств. Первый способ доказательства заключается в приведении путем преобразования одной части равенства к другой части. Если это удается сделать, то тождество можно считать доказанным. Второй способ доказательства состоит в определении кодов функций левой и правой части равенства. Если коды равны, то и функции равны, что также доказывает справедливость тождества.

❒ Первый способ доказательства применим ко второму распределительному закону (3).

$X \vee YZ = (X \vee Y)\ (X \vee Z)$.

Правую часть равенства преобразуем к виду левой части. Справа будем записывать используемые при преобразовании законы и правила:

$(X \vee Y)\ (X \vee Z) = XX \vee XY \vee XZ \vee YZ =$	'раскрываем скобки, $X \cdot X = X$
$X \vee XY \vee XZ \vee YZ =$	'выносим X за скобки
$X(1 \vee Y \vee Z) \vee YZ =$	'используем $X \vee 1 = 1$ и $X \cdot 1 = X$
$X \vee YZ$	'результат преобразований.

Справедливость тождества доказана.

- Второй способ доказательства применим к законам де Моргана (7).

  Словесно их можно сформулировать так: 1) инверсия конъюнкции равна дизъюнкции инверсий и 2) инверсия дизъюнкции равна конъюнкции инверсий. Докажем первую часть правила: $\overline{X \cdot Y} = \overline{X} \vee \overline{Y}$. Для доказательства используем табл. 22.8, строки которой соответствуют наборам значений аргументов X и Y.

Таблица 22.8. *Доказательство второго закона де Моргана*

XY	$\overline{XY}$	$\overline{X} \vee \overline{Y}$
0 0	$\overline{0 \cdot 0} = \overline{0} = 1$	$\overline{0} \vee \overline{0} = 1 \vee 1 = 1$
0 1	$\overline{0 \cdot 1} = \overline{0} = 1$	$\overline{0} \vee \overline{1} = 1 \vee 0 = 1$
1 0	$\overline{1 \cdot 0} = \overline{0} = 1$	$\overline{1} \vee \overline{0} = 0 \vee 1 = 1$
1 1	$\overline{1 \cdot 1} = \overline{1} = 0$	$\overline{1} \vee \overline{1} = 0 \vee 0 = 0$

Подставляя в формулы левой и правой части равенства значения аргументов, вычисляем значения функций. В итоге убеждаемся, что коды обеих функций совпали. Тождество доказано.

Выполненный пример показывает возможность перехода от записи функции формулой к записи ее в таблице истинности. Для этого достаточно, подставляя в формулу функции все наборы значений аргументов, определить значение функции на этих наборах.

Правила преобразования формул

Кроме перечисленных выше законов для преобразования и упрощения формул булевых функций используются тождества, получившие название правил или операций. Рассмотрим их.

Правило отрицания

$$\bar{f}(x, y, ..., z)_{\&,\vee} = f(\bar{x}, \bar{y}, ..., \bar{z})_{\vee,\&}.$$

Правило отрицания утверждает, что для получения отрицания некоторого выражения достаточно заменить в нем знаки дизъюнкции знаками конъюнкции, знаки конъюнкции знаками дизъюнкции, а все аргументы — их отрицаниями. Если в выражении имеются константы 0 и 1, то их также нужно заменить противоположными значениями.

Примеры:

$$\overline{x_1 \vee \bar{x}_2(x_3 \vee \overline{\bar{x}_4 \cdot x_5})} = \bar{x}_1(x_2 \vee \bar{x}_3(\overline{x_4 \vee \bar{x}_5}));$$

$$\overline{x_1 \vee \bar{x}_2 x_3 \vee 0} = \bar{x}_1 \cdot (x_2 \vee \bar{x}_3) \cdot 1.$$

Правило свертки

Правило свертки позволяет упростить формулу, исключив из нее инверсию одной из переменных:

$$x \vee \bar{x}y = x \vee y;$$

$$x(\bar{x} \vee y) = xy.$$

Правило обобщенного склеивания

Правило свертки — это теорема русского математика П. С. Порецкого. В силу простоты оно не требует словесного описания. Правило задается тождествами:

$$xy \vee \bar{y}z \vee xz = xy \vee \bar{y}z;$$

$$(x \vee y)(\bar{y} \vee z)(x \vee z) = (x \vee y)(\bar{y} \vee z).$$

Справедливость тождеств можно доказать рассмотренными выше способами.

22.7. Канонические формы булевых функций

Одна и та же функция может быть задана множеством различных формул. Поэтому функции трудно сравнивать. Канонические формы функций — это запись функций по единым правилам. Такие формы еще называют совершенными. Различают дизъюнктивную и конъюнктивную совершенные нормальные формы. Рассмотрим только дизъюнктивную форму.

Совершенная дизъюнктивная нормальная форма

Дизъюнктивной нормальной формой (ДНФ) булевой функции называют дизъюнкцию конъюнкций (логическую сумму логических произведений). Формула функции в ДНФ не имеет скобок и общих для нескольких аргументов отрицаний.

Пример ДНФ-функции:

$$F(X, Y, Z) = X \vee \bar{Y}Z \vee \bar{X}Z \vee X\bar{Y}Z .$$

Конституентой единицы называют полную (все аргументы) конъюнкцию отрицаемых или неотрицаемых аргументов. Обозначают ее — $K(a)$, где a — номер того единственного набора, на котором $K(a) = 1$.

Пример: $K(5) = X\overline{Y}Z$. $K(5)$ равна 1 только на 5-м наборе значений аргументов (101).

Совершенной дизъюнктивной нормальной формой (СДНФ) функции называют дизъюнкцию конституент единицы, равных единице на тех же наборах, что и функция.

Переход от таблицы истинности функции к СДНФ

Таблица 22.9. *Примеры функций трех аргументов*

a	*XYZ*	*F(X, Y, Z)*	*W(X, Y, Z)*
0	0 0 0	0	0
1	0 0 1	1	1
2	0 1 0	1	0
3	0 1 1	0	1
4	1 0 0	1	0
5	1 0 1	0	1
6	1 1 0	0	1
7	1 1 1	1	0

Пусть заданы две булевы функции 3-х аргументов с помощью таблицы истинности (табл. 22.9). Функцию $F(X, Y, Z)$ запишем в СДНФ. Функцию $W(X, Y, Z)$ используем для следующих построений.

По определению СДНФ выпишем из таблицы конституенты единицы для наборов, на которых функция F равна 1:

$$F = K(1, 2, 4, 7) = K(1) \vee K(2) \vee K(4) \vee K(7) =$$
$$= XYZ \vee XYZ \vee XYZ \vee XYZ =$$
$$\quad 001 \quad 010 \quad 100 \quad 111$$
$$\overline{X} \cdot \overline{Y} \cdot Z \vee \overline{X} \cdot Y \cdot \overline{Z} \vee X \cdot \overline{Y} \cdot \overline{Z} \vee \overline{X} \cdot \overline{Y} \cdot \overline{Z}.$$

Правило перехода от таблицы к формуле функции в СДНФ состоит в следующем.

Для перехода нужно записать дизъюнкцию полных конъюнкций по числу единиц в коде функции (в примере — 4 единицы), подписать под ними наборы, на которых функция равна единице, и поставить отрицания аргументов, соответствующих нулям в этих наборах.

Переход от схемы к формуле функции

Переход от логической схемы (ЛС), построенной из логических элементов (ЛЭ), к формуле булевой функции требует знания булевых функций, реализуемых ЛЭ. Начиная от входов ЛС, используя принцип суперпозиции (подстановки функций в качестве аргументов в другие функции), получаем на выходе функцию, реализуемую всей ЛС. Рассмотрим этот переход на примере ЛС, показанной на рис. 22.8.

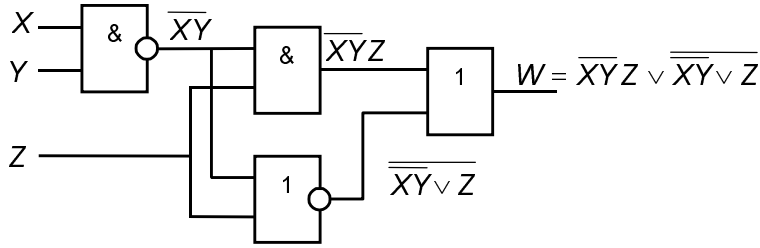


Рис. 22.8. Переход от логической схемы к формуле функции

Переход от формулы к СДНФ и таблице истинности

Такой переход можно сделать, выполнив следующие действия.

1. При наличии общих для нескольких переменных отрицаний, используя правило де Моргана, опускаем их на переменные. При наличии двойных отрицаний убираем их:

$$W = \overline{XY}Z \vee \overline{\overline{XY} \vee Z} = (\overline{X} \vee \overline{Y})Z \vee \overline{\overline{XY}}\,\overline{Z} = (\overline{X} \vee \overline{Y})Z \vee XY\overline{Z}.$$

2. Раскрываем все скобки и добавляем недостающие переменные умножением членов полученной ДНФ на скобки, равные единице:

$$W = \overline{X}Z \vee \overline{Y}Z \vee XY\overline{Z} = \overline{X}(Y \vee \overline{Y})Z \vee (X \vee \overline{X})\overline{Y}Z \vee XY\overline{Z}.$$

3. Раскрываем скобки и оставляем в формуле только по одной из повторяющихся конституент (используем правило $X \vee X = X$). В результате получим СДНФ функции:

$$W = \overline{X}YZ \vee \underline{\overline{X} \cdot \overline{Y}Z} \vee X\overline{Y}Z \vee \underline{\overline{X} \cdot \overline{Y}Z} \vee XY\overline{Z} = \overline{X}YZ \vee \overline{X} \cdot \overline{Y}Z \vee X\overline{Y}Z \vee XY\overline{Z}.$$

4. Осталось определить номера наборов, на которых полученные конституенты единицы равны единице. Это соответственно наборы: 011, 001, 101 и 110 (наборы 3, 1, 5 и 6). Записываем в таблице истинности на этих наборах значение функции 1, на остальных наборах 0. Функция $W(X, Y, Z)$ записана в табл. 22.9.

Переход от алгоритма работы к логической схеме

Такой переход выполняется в последовательности: задача — алгоритм — таблица истинности — формулы функций — логическая схема

Пример

В *гл. 5* была построена логическая схема, суммирующая два одноразрядных двоичных числа *A* и *B* и вырабатывающая их сумму *S* и перенос *P*. Усложним задачу. Построим логическую схему одноразрядного двоичного сумматора на три входа, вырабатывающего значение суммы *S* и переноса *Q* в следующий разряд сумматора.

Обозначим входные сигналы: *A* и *B* — значения одноименных разрядов двух двоичных чисел, *P* — перенос из предыдущего разряда.

Обратим внимание на важный момент построения логической схемы. В качестве булевых переменных выступают цифровые разряды двоичного числа. Пример показывает применение булевой алгебры для построения всех логических схем компьютера, преобразующего информацию, представленную в двоичной системе счисления.

Зная двоичную арифметику и используя модель работы такого сумматора (рис. 22.9, *а*), заполняем таблицу истинности (табл. 22.10). Условное обозначение сумматора показано на рис. 22.9, *б*.

Таблица 22.10. *Функции суммы и переноса*

a	*ABP*	*S*	*Q*
0	000	0	0
1	001	1	0
2	010	1	0
3	011	0	1
4	100	1	0
5	101	0	1
6	110	0	1
7	111	1	1

Из таблицы истинности по рассмотренным в *разд. "Переход от формулы к СДНФ и таблице истинности"* правилам выписываем формулы функций *S* и *Q* в виде дизъюнкции конституент единицы. Как видно из таблицы, каждая

формула будет содержать по четыре дизъюнктивных члена — по числу единиц в кодах функций:

$$S = \overline{A}\,\overline{B}P \vee \overline{A}B\overline{P} \vee A\overline{B}\,\overline{P} \vee ABP;\quad Q = \overline{A}BP \vee A\overline{B}P \vee AB\overline{P} \vee ABP.$$

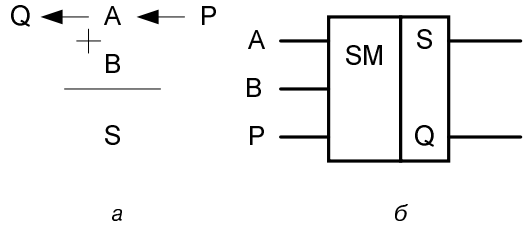


Рис. 22.9. Модель сложения (*а*) и обозначение сумматора (*б*)

Вторую формулу можно упростить. Упрощение формул уменьшает потребное для их реализации количество логических элементов. В упрощенном виде формула переноса запишется так:

$$Q = AB \vee AP \vee BP.$$

Убедиться в справедливости полученной формулы можно обратным переходом к СДНФ функции Q.

Решение задач минимизации сложности логических формул было актуально при построении схем компьютеров первого и второго поколения, когда стоимость каждого логического элемента была высока.

Постройте самостоятельно логические схемы, реализующие полученные формулы работы одноразрядного сумматора на три входа.

22.8. Применение логических операций при программировании

Результаты операций отношения (знаки операций: =, <>, >, <, >=, <=) принимают значение истина (True), если условие выполняется, и ложь (False), если условие не выполняется. Отношения можно интерпретировать как простые высказывания, которые могут быть истинными или ложными.

С помощью логических операций `And`, `Or`, `Not`, `Xor`, `Eqv` и `Imp` из простых отношений можно строить сложные, составные конструкции и использовать их в качестве сложных условий, например, в операторах `If`.

Программа, приведенная в листинге 22.1, печатает результаты операций отношения величин чисел *a*, *b* и *c*.

Листинг 22.1. Печать результатов отношений величин

```
Private Sub Form_Load()
Show
  a = 5: b = 3: c = 1
  f1 = a > b
       'f1=True
  f2 = a < b
       'f2=False
  f3 = a < b Or b > c                              'f3=True
  f4 = a > b And Not (b > c)                       'f4=False
  Print "f1="; f1, "f2="; f2, "f3="; f3, "f4="; f4
End Sub
```

В качестве компонентов отношений могут выступать арифметические и строковые выражения. В выражениях соблюдается следующая очередность выполнения операций: арифметические и строковые, отношения, логические. При этом очередность выполнения логических операций такая: `NOT`, `AND`, `OR`, `XOR` и `EQV`, `IMP`.

Рассмотрим пример использования сложного логического условия. Пусть требуется табулировать функцию, заданную графиком, покзанным на рис. 22.10. Это можно сделать с помощью программы, приведенной в листинге 22.2.

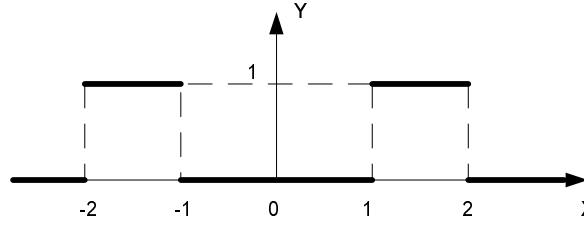


Рис. 22.10. График функции

Листинг 22.2. Табуляция функции, заданной графиком, показанным на рис. 22.10

```
Private Sub Form_Load()
  Show
  x = InputBox("Введите x")
```

```
  x = Val(x)
  If x < -2 Or x > -1 And x < 1 Or x > 2 Then y = 0 Else y = 1
'Эту же функцию можно задать так::
'If x >= -2 And x <= -1 Or x >= 1 And x <= 2 Then y = 1 Else y = 0
  Print "При x="; x, "y="; y
End Sub
```

22.9. Моделирование логических функций

Некоторые из рассмотренных выше задач были связаны с преобразованиями формул булевых выражений, вычислением их значений на различных наборах значений аргументов. К таким задачам можно отнести доказательства справедливости основных законов и тождеств булевой алгебры, переход от произвольной формулы функции к таблице ее истинности, другие задачи. Заметим, что подобные тождественные преобразования даже несложных функций вызывают затруднения, требуют внимательности, решения их связаны с возможными ошибками. В таких случаях следует привлекать компьютер.

Программа, приведенная в листинге 22.3, строит и заполняет таблицу истинности. Она последовательно вырабатывает все наборы значений аргументов и вычисляет значения функции на этих наборах. Программа легко дополняется для генерации наборов 4-х, 5-и и более аргументов. В рассмотренном примере программа работает с формулой функции из *разд. "Переход от формулы к СДНФ и таблице истинности" данной главы*:

$$W = \overline{XY}Z \vee \overline{\overline{XY} \vee Z}.$$

Листинг 22.3. Переход от формулы к таблице истинности

```
Private Sub Form_Load()
  Show
  Print " x"; "  y"; "  z", " w"
  Print
  For x = 0 To 1
    For y = 0 To 1
      For z = 0 To 1
        w = Not (x And y) And z Or Not (Not (x And y) Or z)
        Print x; y; z, w
  Next z, y, x
End Sub
```

В результате работы программы на экран монитора будет выведена таблица истинности функции $W(x, y, z)$, аналогичная табл. 22.9.

При моделировании логических функций возможно получение в коде функций значений −1 вместо 1 и −2 вместо 0. Это легко изменить перед печатью результатов.

Задания для самостоятельной работы

1. В таблице истинности заданы четыре булевы функции трех переменных: f_1, f_2, f_3 и f_4 (табл. 22.11). Для каждой из этих функций выполнить следующее:

Таблица 22.11

α	$x_1\ x_2\ x_3$	f_1	f_2	f_3	f_4
0	0 0 0	1	0	1	1
1	0 0 1	0	1	0	1
2	0 1 0	0	0	0	0
3	0 1 1	1	1	0	0
4	1 0 0	1	1	1	1
5	1 0 1	0	0	1	1
6	1 1 0	0	1	1	0
7	1 1 1	1	0	1	1

   - перейти от таблицы истинности к записи функции в СДНФ (совершенной дизъюнктивной нормальной форме);
   - используя законы и правила преобразования формул булевых функций, в частности, операции склеивания и поглощения, упростить запись функции (уменьшить в формуле число вхождений переменных и их отрицаний);
   - реализовать полученную запись функции логической схемой на элементах И, ИЛИ и НЕ.
2. Для заданной на рис. 22.11 логической схемы, реализующей функции трех аргументов, выполнить следующие действия:
   - записать формулу реализуемой функции;
   - перейти от произвольной формы записи формулы функции к записи в СДНФ;
   - перейти от СДНФ функции к записи ее в таблице истинности;

- подать на входы схемы набор значений двоичных переменных, например, 101, и проследить значения сигналов на выходах логических элементов и на выходах логической схемы; сравнить полученные значения со значениями функции на этом наборе в таблице истинности.

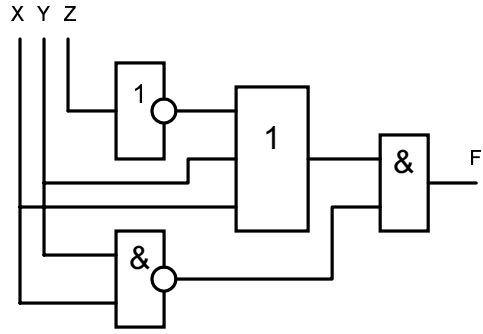


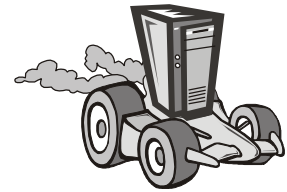
Рис. 22.11. Логическая схемы к заданию 2 самостоятельной работы

3. Как было показано, любую булеву функцию можно записать в СДНФ. Это говорит о функциональной полноте булевого базиса (конъюнкция, дизъюнкция и инверсия). Чтобы убедиться в функциональной полноте базисов алгебр Шеффера, Пирса и Жегалкина, достаточно с помощью функций, входящих в их базисы, реализовать функции булева базиса. Например, инверсия в базисах алгебр Шеффера, Пирса и Жегалкина реализуется так:

$$\bar{x} = x \mid x = x \downarrow x = 1 \oplus x.$$

Реализуйте в этих базисах функции конъюнкцию и дизъюнкцию.

Глава 23



Алгебра логики и логические задачи

23.1. Общие сведения об алгебре логики

Понятия высказываний, истинных и ложных, и примеры их были даны в *гл. 5.* Рассмотрим более подробно вопросы, связанные с логикой высказываний.

Логикой называют науку о способах доказательств, о способах рассуждений, которые от истинных суждений — посылок — приводят к истинным суждениям — следствиям. *Алгеброй логики* называют алгебру, применяемую в разделе логики *Исчисление высказываний.* В этом разделе из простых высказываний с помощью логических операций конъюнкция, дизъюнкция, инверсия, импликация и эквивалентность строятся сложные высказывания. Здесь же решаются вопросы истинности этих сложных высказываний путем анализа на истинность входящих в них простых высказываний.

Основоположником формальной логики считают древнегреческого ученого, философа Аристотеля (384—322 год до н. э.). Если в "Диалогах" Платона (427—347 гг. до н. э.) логика просматривалась в содержательных рассуждениях Сократа (ок. 469—399 гг. до н. э.), то Аристотель вместо конкретных высказываний вводит переменные, отделяет логические правила от содержания, делает первый шаг к математически строгому, формализованному подходу в изучении логики.

Уже у Аристотеля возникла идея составлять более сложные высказывания из простых высказываний. Дальнейшее развитие эта идея получила в трудах Лейбница — немецкого математика, физика, философа (1646—1716 г.). Он работал над приданием аристотелевой логике алгебраической формы. Но лишь в середине XIX века эта идея в работах английского математика и логика Джорджа Буля (1815—1864 г.) воплотилась в законченную форму. Он построил алгебру на такой системе аксиом, которая описывает свойства высказываний, и назвал свою алгебру алгеброй логики.

23.2. Логические операции над высказываниями

Базис современной алгебры логики составляют операции конъюнкция, дизъюнкция, инверсия, импликация и эквивалентность. Запись этих операций на разговорном языке, в алгебре логики, в булевой алгебре и на языке программирования Бейсик показана в табл. 23.1.

Таблица 23.1. *Запись логических операций*

Функции (операции)	**Разговорный язык**	**Алгебра логики**	**Булева алгебра**	**Бейсик**
конъюнкция	*A* и *B*	AB, $A \wedge B$	$A\,B$	`A AND B`
дизъюнкция	*A* или *B*	$A \vee B$	$A \vee B$	`A OR B`
инверсия	не *A*	$\overline{A}$ или ~A	$\overline{A}$	`NOT A`
импликация	если *A*, то *B*	$A \rightarrow B$	$\overline{A} \vee B$	`A IMP B`
эквивалентность (равнозначность)	*A* тогда и только тогда, когда *B*	$A \leftrightarrow B$	$AV \vee \overline{AB}$	`A EQV B`

В некоторых случаях в базис алгебры логики добавляют функцию неравнозначность (сумма по модулю 2, инверсия эквивалентности — `A XOR B`). В следующей таблице истинности (табл. 23.2) собраны коды функций алгебры логики.

Таблица 23.2. *Таблица истинности*

X Y	**не *X***	***XY***	***X∨Y***	***X→Y***	***Y→X***	***X↔Y***
Л Л	И	Л	Л	И	И	И
Л И	И	Л	И	И	Л	Л
И Л	Л	Л	И	Л	И	Л
И И	Л	И	И	И	И	И

Отметим, что при преобразованиях формул логических функций и особенно при их моделировании на компьютере, как правило, пользуются более удобным двоичным алфавитом, т. е. вместо Л используют 0 (ноль), вместо И — 1 (единица).

Рассмотрим функции алгебры логики применительно к конкретным высказываниям. Обозначим высказывания: "Дождь идет" — буквой *D*, "На небе ту-

чи" — буквой *T*. Можно записать: *D* = "Дождь идет" и *T* = "На небе тучи". Включим в рассмотрение и функцию неравнозначность (сумму по модулю 2).

- Конъюнкция (логическое умножение). Сложное высказывание "*D* **и** *T*" понимается в том смысле, что "Дождь идет **и** на небе тучи". Обозначим это высказывание буквой *A*. Используя знаки операций булевой алгебры, можно записать: $A = DT$. На языке программирования это записывается так: `A = D AND T`.
- Дизъюнкция (логическое сложение). Сложное высказывание "*D* **или** *T*" читается так: "Дождь идет **или** на небе тучи". Обозначим его буквой B. Можно записать: $B = D \vee T$ и `B = D OR T`.
- Инверсия (отрицание). Одноместную операцию отрицание можно применить как к простому, так и сложному высказыванию. Высказывание "**не** *D*" читается "**Не верно, что** дождь идет" или "Дождь **не** идет". Это записывается так: $\overline{D}$ и `NOT D`.
- Если $A = D \cdot T$, то под высказыванием "**не** *A*" понимается следующее: "**Не верно, что** дождь идет и на небе тучи". Запись в булевой алгебре: $\overline{A} = \overline{DT}$. На языке Бейсик: `NOT A` или `NOT (D AND T)` или `NOT D OR NOT` (к последнему выражению приводит применение закона де Моргана (инверсия конъюнкции равна дизъюнкции инверсий)).
- Импликация. Операция импликация эквивалентна выражению на разговорном языке "**Если** *D*, **то** *T*", где *D* и *T* — некоторые высказывания. Допустим, что *D* и *T* — это те же приведенные выше высказывания: *D* = = "Идет дождь" и *T* = "На небе тучи". Рассмотрим функцию $D \rightarrow T$.

  Напомним, что если *D* = И, то дождь идет, если же *D* = Л, то дождя нет. Аналогично читается высказывание о тучах на небе. В пятом столбце таблицы истинности (см. табл. 23.2) записано значение функции импликация на всех наборах значений аргументов. Построчно эту функцию в отношении высказываний *D* и T можно интерпретировать так:

  - 1-я строка. Л→Л = И. "Если дождь не идет, то на небе нет тучи" — высказывание истинно;
  - 2-я строка. Л→И = И. "Если дождь не идет, то на небе тучи" — высказывание истинно;
  - 3-я строка. И→Л = Л. "Если дождь идет, то на небе нет тучи" — высказывание ложно;
  - 4-я строка. И→И = И. "Если дождь идет, то на небе тучи" — высказывание истинно.

  Некоторые из строк, например, вторая строка, могут вызвать вопросы. Но следует учитывать, что в алгебрах царствует формальный подход. А в алгебре логики, предназначенной для формализации богатого различны-

ми нюансами разговорного языка, отразить операциями все смысловые тонкости речи практически невозможно. Расширение базиса алгебры (набора операций) усложняет ее, что нежелательно. Но и принятый базис позволяет находить решения сложных логических задач, которые с помощью содержательных рассуждений, без использования аппарата алгебры логики решать трудно.

- Неравнозначность (сумма по модулю 2) — в отношении дождя и тучи может быть записана формулой так: $D \cdot \overline{T} \vee \overline{D} \cdot T$, т. е. "Дождь идет и на небе нет туч или дождя нет и на небе тучи" — или одно, или другое, но не оба. Неравнозначность — инверсия эквивалентности. На языке программирования неравнозначность записывается так: `C = D XOR T`.
- Эквивалентность (равнозначность) — последняя функция базиса алгебры логики. В принятых выше обозначениях (D — "Идет дождь", T — "На небе тучи") эта функция записывается в булевой алгебре, алгебре логики и на языке программирования следующим образом, соответственно: $C = D \cdot T \vee \overline{D} \cdot \overline{T}$, $C = D \leftrightarrow T$ и `C = D EQV T`, т. е. сложное высказывание С истинно, если простые высказывания D и T оба истинны или оба ложны, что означает: "идет дождь **и** на небе тучи **или не** идет дождь **и нет** на небе тучи".

23.3. Формализация высказываний

Естественный язык не поддается полной формализации ввиду неоднозначности слов и выражений и множества трудноуловимых оттенков, передающих эмоциональную сторону высказываний. Поэтому операции в алгебре логики носят собирательный характер в том смысле, что каждая операция формализует некоторое количество тождественных или близких по смыслу выражений разговорной речи. Рассмотрим примеры.

- Конъюнкция: "A и B", "как A, так и B", "не только A, но и B", "A вместе с B", "A, несмотря на B", "A, в то время как B" и т. п.
- Дизъюнкция: "A или B", "A, или B, или оба" и т. п.

  Если встречается связка "либо", например "A либо B", то это означает: "или A, или В, но не оба" или "только A или только B", что формализуется записью $A \cdot \overline{B} \vee \overline{A} \cdot B$, т. е. операцией `XOR`, которая равна инверсии операции `EQV`.
- Инверсия: "не A", "неверно, что A", "A не имеет места".
- Импликация.

  Формулой $A \rightarrow B$ записываются следующие выражения: "если A, то B", "B, если A", "когда А, тогда B", "A только, если B", "A достаточно для B", "для A необходимо B", "A только тогда, когда B" и т. п.

Формулой $B \to A$ записываются следующие высказывания: "если B, то A", "A, если B", "A тогда, когда B", "для A достаточно B" и т. п.

К импликации приводят выражения, содержащие слова "необходимо" и "достаточно". Например, обозначим буквой A высказывание "Пошел дождь", буквой B — "На небе тучи". Рассмотрим высказывание "Чтобы пошел дождь, необходимо наличие на небе туч". Левая часть этого высказывания — посылка, правая — заключение. Формализованно оно записывается так: $A \to B$.

Высказывание "Чтобы пошел дождь, достаточно на небе туч" записывается так: $B \to A$. Можно сделать вывод, что необходимые условия в формуле записываются в качестве заключения справа, а достаточные условия — в качестве посылки слева.

При решении задач полезно помнить запись импликации в булевой алгебре: $A \to B = \overline{A \cdot \overline{B}} = \overline{A} \vee B$, $B \to A = \overline{\overline{A} \cdot B} = A \vee \overline{B}$.

Некоторые выражения разговорного языка по форме напоминают импликацию, а по содержанию требуют запись в виде конъюнкции. Например: "Если Петр любитель ходить по гостям, то Павел домосед"; "Если в планиметрии изучают плоские фигуры, то в стереометрии изучают трехмерные геометрические тела".

- Неравнозначность: "A или B, но не оба", "A либо B", "либо А, либо B", либо не А, либо не B".
- Эквивалентность (равнозначность): "A эквивалентно B", "A тогда и только тогда, когда B", "A необходимо и достаточно для B". Напомним, что в булевой алгебре, алгебре логики и на языке программирования эквивалентность (равнозначность) записывается, соответственно, так:

  $C = \overline{A} \cdot \overline{B} \vee A \cdot B$, $C = A \leftrightarrow B$, `C = A EQV B`.

  В алгебре логики справедливы все законы и правила преобразования формул булевой алгебры. Включение в базис алгебры логики функций импликация и эквивалентность добавляет несколько полезных тождеств, например:

  $A \to B = \overline{B} \to \overline{A}$, $(A \leftrightarrow B) = (A \to B)(B \to A) = A \cdot B \vee \overline{A} \cdot \overline{B}$.

Пример 1

Рассмотрим примеры на запись формулами сложных высказываний. Введем и обозначим несколько простых высказываний:

Z = "Яблоко зеленое", K = "Яблоко красное", V = "Яблоко вкусное",

S = "Яблоко сладкое", G = "Яблоко кислое", W = "Яблоко крупное",

L = "Яблоко мелкое", T = "Яблоко твердое", M = "Яблоко мягкое".

Составим из этих высказываний несколько сложных высказываний:

X_1 = "Яблоко красное и вкусное";

X_2 = "Яблоко зеленое или мягкое";

X_3 = "Яблоко не только крупное, но и сладкое";

X_4 = "Яблоко красное, хотя и кислое, но вкусное";

X_5 = "Яблоко сладкое, но невкусное";

X_6 = "Яблоко зеленое или красное, но твердое";

X_7 = "Не верно, что если яблоко крупное, то оно сладкое";

X_8 = "Если яблоко мелкое или зеленое, то оно твердое";

X_9 = "Яблоко крупное или вкусное, если оно красное и мягкое";

X_{10} = "Если яблоко мелкое, то оно должно быть твердым, и все это только в случае, если яблоко зеленое";

X_{11} = "Чтобы яблоко было сладким, необходимо, чтобы оно было красным";

X_{12} = "Чтобы яблоко было сладким, достаточно, чтобы оно было красным";

X_{13} = "Чтобы яблоко было сладким, необходимо и достаточно, чтобы оно было красным";

X_{14} = "Яблоко кислое тогда и только тогда, когда оно мелкое и зеленое".

Запишем эти сложные высказывания формулами алгебры логики и на языке программирования:

$X_1 = KV =$ `K AND V`	$X_2 = Z \vee M =$ `Z OR M`
$X_3 = WS =$ `W AND S`	$X_4 = KGV =$ `K AND G AND V`
$X_5 = S\overline{V} =$ `S AND NOT V`	$X_6 = (Z \vee M)\ T$ `(Z OR K) AND T`
$X_7 = \overline{W \to S} =$ `NOT(W IMP S)`	$X_8 = (M \vee Z) \to T =$ `(M OR Z) IMP T`
$X_9 = KM \to (W \vee V) =$ `K AND M IMP W OR V`	$X_{10} = Z \to (L \to T) =$ `Z IMP (L IMP T)`
$X_{11} = S \to K =$ `S IMP K`	$X_{12} = K \to S =$ `K IMP S`
$X_{13} = S \leftrightarrow K =$ `S EQV K`	$X_{14} = G \leftrightarrow (LZ) =$ `G EQV(L AND Z)`

Напомним очередность выполнения операций: `NOT`, `AND`, `OR`, `XOR` и `EQV`, `IMP`. Скобки вводятся в формулы с целью изменить эту очередность, так как сначала выполняются действия в скобках.

Для проверки правильности записей иногда полезно рассмотреть таблицу истинности записываемых функций. Например, могут вызвать сомнения функции X_{11} и X_{12}. Представим их таблицей истинности (табл. 23.3).

Таблица 23.3. *Таблица истинности функций X_{11} и X_{12}*

S K	*S→K*	*K→S*
0 0	1	1
0 1	1	0
1 0	0	1
1 1	1	1

Рассмотрим функцию $X_{11} = S \rightarrow K$. Первая строка говорит о том, что несладкое яблоко может не быть красным. Это высказывание истинно. Вторая строка указывает на то, что несладкое яблоко может быть красным. Это высказывание также истинно. Четвертая строка также соответствует истинному высказыванию: сладкое яблоко должно быть также и красным. И только третья строка говорит, что не может быть, чтобы сладкое яблоко не было красным, т. е. **необходимо**, чтобы оно было красным. Высказывание "Яблоко сладкое, но не красное" — ложное и противоречит смыслу исходной формулы: "Если яблоко сладкое, то оно красное".

Аналогичным образом рассматривается функция $X_{12} = K \rightarrow S$. Она принимает значение 0 (ложь) только на наборе $SK = 01$, который соответствует высказыванию о том, что красное яблоко оказалось не сладким. Чтобы яблоко было сладким, **достаточно**, чтобы оно было красным. Но, как видно из таблицы, сладкое яблоко может и не быть красным. Видим, что требование достаточности менее жесткое, чем требование необходимости.

23.4. Решение логических задач

Структура логических задач может быть различной. Также различными бывают и подходы к их решению. Рассмотрим некоторые из приемов, которые приходится использовать для решения задачи.

При решении логических задач наиболее интересен этап формализации высказываний. На этом этапе желательно по возможности не вводить лишние переменные, которые будут увеличивать длину наборов аргументов и усложнять формулы функций. Например, каждую пару альтернативных высказываний: "тепло" и "холодно", "большой остров" и "маленький остров", "острый угол" и "тупой угол" и т. п. можно закодировать одной переменной и ее инверсией. Но это можно сделать только в том случае, если по условию задачи

нет третего варианта выбора, например, такого, как: "остров средней величины" (для второй пары) или "прохладно" (для первой).

Пусть в процессе решения задачи задано или получено несколько простых или сложных высказываний f_1, f_2, ..., f_n, истинность которых известна. В этом случае обычно достаточно взять конъюнкцию этих высказываний $F = f_1 f_2 \ldots f_n$ и определить набор простых высказываний, на котором эта конъюнкция принимает значение истина. Этот набор и будет решением задачи. Возможно, что таких наборов будет несколько, что укажет на наличие нескольких решений. Возможен также вариант отсутствия решения.

Пример 2

Школьник попросил троих друзей отгадать, какое он задумал число из конкретного набора чисел: положительное, отрицательное, четное, нечетное, целое или дробное. Первый друг сказал, что если это число четное, то оно положительное. Второй предположил, что задуманное число четное или целое и положительное. Третий был уверен, что если это число положительное, то оно нечетное. Оказалось, что все три друга правы. Их высказывания истинны. Какое было задумано число?

Формализация и решение задачи

Введем обозначения:

A = "Число положительное", $\overline{A}$ = "Число отрицательное",

B = "Число четное", $\overline{B}$ = "Число нечетное",

C = "Число целое", $\overline{C}$ = "Число дробное".

Используя введенные обозначения, запишем высказывания всех трех отгадчиков:

$$F_1 = B \to A,$$
$$F_2 = B \vee C \cdot A,$$
$$F_3 = A \to \overline{B}.$$

Итоговая функция равна логическому произведению функций F_1, F_2 и F_3:

$$F = F1 \cdot F2 \cdot F3 =$$
$$= (B \to A) \cdot (B \vee C \cdot A) \cdot (A \to \overline{B}) =$$
$$= (\overline{B} \vee A) \cdot (B \vee C \cdot A) \cdot (\overline{A} \vee \overline{B}) =$$
$$= \overline{B} \cdot B \cdot \overline{A} \vee A \cdot \overline{B} \cdot C \cdot \overline{A} \vee A \cdot C \cdot \overline{A} \vee \overline{B} \cdot B \cdot \overline{B} \vee A \cdot \overline{B} \cdot C \cdot \overline{B} \vee A \cdot \overline{B} \cdot C =$$
$$= A \cdot \overline{B} \cdot C.$$

Получили выражение, которое принимает значение Истина только на одном наборе 101, т. е. $A = 1$, $B = 0$ и $C = 1$, а это означает, что было задумано положительное нечетное целое число.

Может быть задано или получено несколько высказываний, о которых сказано, что из них только некоторое количество высказываний истинно. В этом случае организуется перебор вариантов. Например, пусть заданы высказывания a и b, о которых известно, что из них только одно истинно. В этом случае записывают следующую формулу: $F = a\overline{b} \vee \overline{a}b$.

Если заданы высказывания a, b, c, d, из которых только одно истинно, то это соответствует следующей формуле:

$$F = a\overline{b}\,\overline{c}\,\overline{d} \vee \overline{a}b\overline{c}\,\overline{d} \vee \overline{a}\,\overline{b}cd \vee \overline{a}\,\overline{b}\,\overline{c}d\,.$$

Если сказано, что из этих высказываний истинны лишь какие-то два, то имеем следующую формулу:

$$F = ab\overline{c}\,\overline{d} \vee a\overline{b}c\overline{d} \vee a\overline{b}\,\overline{c}d \vee \overline{a}bc\overline{d} \vee \overline{a}b\overline{c}d \vee \overline{a}\,\overline{b}cd$$

и т. д.

Пример 3

Друзья Андрея (A), Владимира (B) и Сергея (C) обсуждали их шансы на победу в шахматном турнире. Первый сказал, что победит A или C, второй заявил, что ни A, ни B победы не видать. Третий был уверен, что победит A или B. В итоге оказалось, что угадал один из них. Кто из трех шахматистов победил?

Формализация и решение задачи

Обозначим высказывания:

A = "Победил Андрей",

B = "Победил Владимир",

C = "Победил Сергей".

Если $A = 1$, то высказывание истинно, если $A = 0$, то ложно. Аналогично для других участников.

Запишем высказывания болельщиков в виде формул:

1-й болельщик: $F_1 = A \vee C$;

2-й болельщик: $F_2 = \overline{A} \cdot \overline{B}$;

3-й болельщик: $F_3 = A \vee B$.

К этим трем предположениям болельщиков необходимо добавить функцию, которая введет естественное ограничение победы только одного из шахматистов:

$$F_4 = A \cdot \overline{B} \cdot \overline{C} \vee \overline{A} \cdot B \cdot \overline{C} \vee \overline{A} \cdot \overline{B} \cdot C .$$

Формула показывает, что функция F_4 может быть равна 1 только на одном из трех наборов значений аргументов A, B и C: 100, 010 и 001, т. е. при победе только одного из шахматистов.

Аналогичная ситуация сложилась и для болельщиков. По условию задачи прав оказался только один из них. Поэтому справедливо следующее выражение:

$$F_1\overline{F}_2\overline{F}_3 \vee \overline{F}_1F_2\overline{F}_3 \vee \overline{F}_1\overline{F}_2F_3,$$

которое, как и значение функции F_4, может быть истинным только на трех наборах аргументов F_1, F_2 и F_3: 100, 010 и 001, т. е., как и требуется, может быть истинным высказывание только одного из болельщиков.

Запишем итоговое выражение для функции, которое приведет к решению задачи. Оно получается как результат логического произведения сформированных выше условий:

$$F = (F_1 \cdot \overline{F}_2 \cdot \overline{F}_3 \vee \overline{F}_1 \cdot F_2 \cdot \overline{F}_3 \vee \overline{F}_1 \cdot \overline{F}_2 \cdot F_3) \cdot F_4 .$$

Формулы инверсных значений функций $F1$, $F2$ и $F3$ следующие:

$$\overline{F}_1 = \overline{A \vee C} = \overline{A} \cdot \overline{C},$$

$$\overline{F}_2 = \overline{\overline{A} \cdot \overline{B}} = A \vee B ,$$

$$\overline{F}_3 = \overline{A \vee B} = \overline{A} \cdot \overline{B} .$$

Они получены опусканием общих для выражения инверсий на переменные по закону де Моргана.

Подставим в формулу для функции F формулы прямых и инверсных значений функций F_1, F_2 и F_3 и упростим полученное выражение, пользуясь правилами тождественных преобразований булевых функций:

$$F = ((A \vee C) \cdot (A \vee B) \cdot \overline{A} \cdot \overline{B} \vee \overline{A} \cdot \overline{C} \cdot (\overline{A} \cdot \overline{B}) \cdot (\overline{A} \cdot \overline{B}) \vee$$
$$\vee \overline{A} \cdot \overline{C} \cdot (A \vee B) \cdot (A \vee B)) \cdot F_4.$$

Второй и третий члены дизъюнкции имеют одинаковые сомножители. По правилу $X \cdot X = X$ опускаем по одному из равных сомножителей и раскрываем скобки:

$$F = (A \cdot A \cdot \overline{A} \cdot \overline{B} \vee A \cdot B \cdot \overline{A} \cdot \overline{B} \vee C \cdot A \cdot \overline{A} \cdot \overline{B} \vee C \cdot B \cdot \overline{A} \cdot \overline{B} \vee$$
$$\overline{A} \cdot \overline{C} \cdot \overline{A} \cdot \overline{B} \vee \overline{A} \cdot \overline{C} \cdot A \vee \overline{A} \cdot \overline{C} \cdot B) \cdot F_4.$$

Используем тождества

$$X \cdot \overline{X} = 0,\ X \cdot 0 = 0 \text{ и } X \vee 0 = X.$$

В соответствии с этими тождествами первые четыре и шестой члены дизъюнкции равны нулю. Раскроем формулу для фунции F_4. Упорядочим переменные в конъюнкциях. В результате получаем:

$$F = (\overline{A} \cdot \overline{B} \cdot \overline{C} \vee \overline{A} \cdot B \cdot \overline{C}) \cdot (A \cdot \overline{B} \cdot \overline{C} \vee \overline{A} \cdot B \cdot \overline{C} \vee \overline{A} \cdot \overline{B} \cdot C).$$

Раскрываем скобки, опускаем равные нулю члены дизъюнкции и получаем ответ:

$$F = \overline{A} \cdot B \cdot \overline{C}.$$

Полученное логическое выражение истинно только на одном наборе значений составляющих его простых выражений: A — ложно, B — истинно и C — ложно, т. е. победил Владимир.

Программное решение

Упрощение сложных логических формул требует внимательности, связано с затратами времени. Для упрощения сложных формул следует привлекать компьютер. В листинге 23.1 приведена программа решения задачи о шахматистах.

Листинг 23.1. Код решения примера 3 о шахматистах

```
Private Sub Form_Load()
  Show
  Print " A"; "  B"; "  C", " F": Print
  For A = 0 To 1
    For B = 0 To 1
      For C = 0 To 1
        F1 = A Or C: F2 = Not A And Not B: F3 = A Or B
        F4 = A And Not B And Not C Or Not A And B And Not C Or
↳ Not A And Not B And C
        F = (F1 And Not F2 And Not F3 Or Not F1 And F2 And Not F3 Or
↳ Not F1 And Not F2 And F3) And F4
    Print A; B; C, F
  Next C, B, A
End Sub
```

В результате работы программы будет отпечатана таблица истинности функции F (рис. 23.1). Для этого программа последовательно формирует наборы

значений аргументов *A*, *B* и *C* (000, 001, 010, ..., 111), подставляет их в функцию *F* и вычисляет ее значение на этих наборах. В таблице истинности только одна единица — против набора 010. Это и есть решение задачи — победил Владимир.

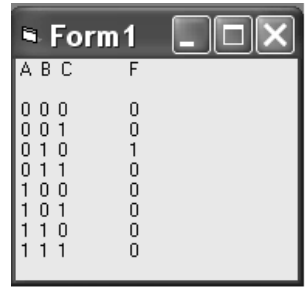


Рис. 23.1. Решение примера о шахматистах

Напомним, что некоторые задачи могут иметь неоднозначное решение. В этом случае итоговая функция может быть истинна не на одном наборе значений аргументов, т. е. код функции содержит несколько единиц. Возможен вариант, когда задача не имеет решения — в коде функции нет единиц.

Пример 4

На вопрос, какая будет завтра погода, синоптик ответил:

1. Если будет мороз, то снег выпадет только при пасмурной погоде.
2. Если не будет мороза, но пойдет снег, то погода будет пасмурной.
3. Не будет ни снега, ни дождя, если погода будет пасмурной.
4. Неверно, что если не будет мороза, то для выпадения снега или дождя достаточно наличия пасмурного неба.

Какую погоду предсказал синоптик?

Формализация и решение задачи

Обозначим высказывания:

M = "мороз",

C = "снег",

P = "пасмурно",

D = "дождь".

Запишем формулы:

$F_1 = M \to (C \to P)$; `F1=m IMP (p IMP c);`

$F_2 = \overline{M} \cdot C \to P$; `F2=(NOT m AND c) IMP p;`

$F_3 = P \to \overline{C} \cdot \overline{D}$; `F3=p IMP (NOT c AND NOT d);`

$F_4 = \overline{\overline{M} \to (P \to C \vee D)}$; `F4=NOT (NOT m IMP (p IMP c OR d).`

Итоговая функция:

$$F = F_1 \cdot F_2 \cdot F_3 \cdot F_4.$$

После подстановки переменных и упрощения формул получаем:

$$F = \overline{M} \cdot P \cdot \overline{C} \cdot \overline{D},$$

что означает "ни мороза, ни снега, ни дождя, но пасмурно" ($F = 1$ на наборе 0100).

Пример 5

В соревнованиях по гимнастике участвуют Алла, Валя, Сима и Даша. Болельщики высказали предположения о возможных победителях:

- 1-е место займет Сима (x_1), Валя будет второй (x_2);
- Второй будет Сима (x_3), Даша займет 3-е место (x_4);
- у Аллы 2-е место (x_5), у Даши 4-е (x_6).

По окончании соревнований оказалось, что в каждом из предположений только одно высказывание истинно, другое — ложно. Как распределились места, если все девушки заняли разные места?

Каждое из трех предположений содержит по два высказывания, которым уже в постановке задачи присвоены идентификаторы x_1, x_2, x_3, x_4, x_5 и x_6.

Программное решение

Программное решение задачи приведено в листинге 23.2.

Листинг 23.2. Соревнования по гимнастике

```
Private Sub Form_Load()
  Show
  For X1 = 0 To 1: For X2 = 0 To 1: For x3 = 0 To 1
  For x4 = 0 To 1: For x5 = 0 To 1: For x6 = 0 To 1
    f1 = X1 And Not X2 Or Not X1 And X2
    f2 = x3 And Not x4 Or Not x3 And x4
```

```
    f3 = x5 And Not x6 Or Not x5 And x6
    f4 = Not (X1 And x3)
    f5 = Not (x4 And x6)
    f6 = X2 And Not x3 And Not x5 Or Not X2 And x3 And Not x5 Or
    ⇗ Not X2 And Not x3 And x5
    f = f1 And f2 And f3 And f4 And f5 And f6
    If f = 1 Then Print X1; X2; x3; x4; x5; x6, f
  Next x6, x5, x4, x3, X2, X1
End Sub
```

Функции f_1, f_2 и f_3 определяют условие истинности только одного из двух высказываний в каждом предположении.

Функции f_4, f_5 и f_6 вносят естественные ограничения: каждое место может занять только одна гимнастка и каждая гимнастка может занять только одно место. Заметим, что при отказе от этих ограничений возможно получение других ответов.

Распечатка итоговой таблицы истинности не умещается полностью на экране монитора ввиду большого числа логических переменных. В таких случаях достаточно выводить только строки таблицы, в которых итоговая логическая функция принимает единичные значения. Напомним, что в коде функций могут выводиться −1 вместо 1 и −2 вместо 0.

В результате работы программы получен следующий результат:

100110 1,

т. е. Сима заняла 1-е место, Даша — 3-е место, у Аллы — 2-е место, а Вале досталось 4-е место.

Задания для самостоятельной работы

1. Задано несколько простых высказываний о погоде. Из них составлены сложные высказывания. Записать эти высказывания формулами алгебры логики и на языке программирования.

   M = "На улице мороз", O = "На улице оттепель", N = "*В*етер северный",

   S = "*В*етер южный", D = "Идет дождь", G = "На дорогах гололедица",

   C = "Идет снег", T = "Температура плюсовая", I = "На деревьях иней",

   U = "На улице туман", P = "Небо пасмурное".

   X_1 = "На улице мороз, идет снег, но гололедицы нет";

X_2 = "На улице оттепель и на деревьях иней или на улице туман";

X_3 = "Если северный ветер или идет снег, то на улице мороз";

X_4 = "На дорогах нет гололедицы, на деревьях нет инея и на улице нет тумана, если дует северный ветер при морозе";

X_5 = "На улице оттепель или на деревьях иней, если температура плюсовая";

X_6 = "Для того чтобы шел дождь или снег, необходимо наличие пасмурного неба";

X_7 = "Для появления на деревьях инея достаточно пасмурного неба и оттепели";

X_8 = "Для гололедицы на дорогах необходимо и достаточно наличие плюсовой температуры при северном ветре и тумане";

X_9 = "Чтобы не было ни снега, ни дождя, необходимо, чтобы небо не было пасмурным";

X_{10} = "На улице туман или на деревьях иней может быть тогда и только тогда, когда на улице оттепель";

X_{11} = "При южном ветре на улице оттепель только тогда, когда пасмурное небо и плюсовая температура";

X_{12} = "На деревьях иней, на улице туман и на дорогах гололедица тогда, когда дует южный ветер и на улице оттепель".

2. Задано несколько простых высказываний о цветах. Из них составлены сложные высказывания, которые записаны формулами алгебры логики и на языке программирования. Запишите эти высказывания на естественном, разговорном языке.

   Простые высказывания:

   A = "Цветок ароматный", K = "Цветок крупный", M = "Цветок мелкий",

   R = "Цветок красный", B = "Цветок белый", G = "Цветок желтый",

   S = "Цветок астра", Z = "Цветок роза", T = "Цветок тюльпан",

   P = "Цветок полевой", D = "Цветок домашний", V = "Цветок садовый".

   Сложные высказывания:

   $Y_1 = K\,A\,(P \vee V) =$ `K AND A AND (P OR V)`,

   $Y_2 = (K \vee M)\,B\,\overline{D} =$ `(K OR M) AND B AND NOT D`,

   $Y_3 = V \to (A \vee K) =$ `V IMP (A OR K)`,

   $Y_4 = A\,K \leftrightarrow (D \vee V) =$ `(A AND K) IMP (D OR V)`,

   $Y_5 = T\,P \to (R \vee G) =$ `(T AND P) IMP (R OR G)`,

$Y_6 = (\overline{S}Z \to (K \vee M))A =$ `(NOT S AND Z IMP (K OR M)) AND A`,

$Y_7 = \overline{(P \to A)}\,\overline{(V \to M)} =$ `NOT (P IMP A) AND NOT (V IMP A)`,

$Y_8 = \overline{(KB \vee GA) \to (S \vee Z)} =$ `NOT (K AND B OR G AND A IMP S OR Z)`.

3. Решите следующую задачу (автор — В. Н. Касаткин). Алеша, Боря и Гриша нашли в земле сосуд. Рассматривая удивительную находку, каждый высказал по два предположения:
   - Алеша: "Это сосуд греческий и изготовлен в V в.";
   - Борис: "Это сосуд финикийский и изготовлен в III в.";
   - Гриша: "Это сосуд не греческий и изготовлен в IV в.".

   Учитель истории сказал ребятам, что каждый из них прав только в одном из двух предположений.

   Где и в каком веке изготовлен сосуд?
4. Решите следующую задачу. Почетный кубок оспаривали четыре команды: "Авангард", "Богатырь", "Горизонт" и "Победа". Болельщики Сеня, Боря и Дима обсуждали их шансы на победу: Сеня считал, что победу одержит либо "Авангард", либо "Богатырь"; Борис был уверен, что "Авангарду" кубка не видать; Дима был уверен, что ни "Богатырь", ни "Победа" трофея не выиграют. Правым оказался один из них. У кого кубок?
5. Решите следующую задачу (автор — В.В. Мадер). Андрей, Ваня и Саша собрались в поход. Учитель, хорошо знавший этих ребят, высказал следующие предположения:
   - Андрей пойдет в поход только тогда, когда пойдут Ваня и Саша;
   - Андрей и Саша друзья, а это значит, что они пойдут в поход вместе или же оба останутся дома;
   - чтобы Саша пошел в поход, необходимо, чтобы пошел Ваня.

   Когда ребята пошли в поход, оказалось, что учитель немного ошибся: из трех его утверждений истинными оказались только два. Кто из названных ребят пошел в поход?
6. Решите такую задачу. Показания свидетелей правонарушения значительно различались. Первый свидетель сказал, что преступник был брюнет с усами. Второй заявил, что это был блондин без усов. Третий свидетель подтвердил, что преступник был блондином, но без портфеля. Четвертый был уверен, что преступник был шатеном с портфелем.

   В действительности оказалось, что каждый из свидетелей ошибся в одном из своих показаний. Каковы приметы правонарушителя?

В заключение опишем постановку двух старинных шуточных логических задач для устного решения.

1. Три мудреца заночевали под пальмой у костра. Пока они спали, какой-то шутник испачкал им лбы сажей. Проснувшись и увидев испачканные лица, все трое стали смеяться. Но смеялись они не долго, т. к. каждый понял, что его лицо тоже испачкано. На то они и мудрецы. Как они рассуждали?
2. Имеется два белых и три черных колпака. Три случайно выбранных из пяти колпаков надели на трех сидящих друг за другом мудрецов. Первый мудрец видит колпаки второго и третьего и говорит: "Я не знаю, какой на мне колпак". Второй мудрец видит перед собой колпак только третьего мудреца и, услышав слова первого мудреца, говорит: "Я тоже не знаю, какой на мне колпак". Услышав ответы первого и второго, третий мудрец говорит: "А я знаю, какой на мне колпак!" Каковы рассуждения всех трех мудрецов, и какой колпак на третьем мудреце?

ПРИЛОЖЕНИЯ

Приложение 1

Действия с папками, файлами, ярлыками

Над элементами файловой системы компьютера (папки, файлы, ярлыки) можно выполнять множество различных действий (создание, перемещение, удаление и т. п.). Для удобства работы пользователя многие меню и команды дублируются. Поэтому каждое из действий можно производить различными способами. Выбор наиболее привлекательного способа остается за пользователем. Но можно посоветовать использование контекстного меню, которое вызывается правой кнопкой мыши.

Заметим, что в разных версиях операционной системы бывают небольшие различия в элементах окон, пунктах меню и т. п. Это затрудняет задачу дать общие советы по работе с элементами файловой системы.

Создание объектов

Создание папки

Для того чтобы создать папку:

1. Выбираем в каталоге папку, в которой требуется создать новую папку.
2. На свободном месте правого окна Проводника щелкаем правой кнопкой мыши. На экран выводится контекстное меню.
3. В контекстном меню выбираем пункты **Создать | Папку**. В окне Проводника появляется значок папки с именем **Новая папка**.
4. Присваиваем созданной папке содержательное имя.
5. Утверждаем создание папки и ее имя щелчком мыши на свободном месте окна или нажатием клавиши <Enter>.

Создание документов

Для того чтобы создать новый документ:

1. На свободном месте правого окна Проводника щелкаем правой кнопкой мыши. На экран выводится контекстное меню.
2. В контекстном меню выбираем пункт **Создать**. Открывается окно со списком типов документов.
3. Выбираем нужный тип. В окне Проводника появляется файл, значок и имя которого указывают на его тип.
4. Присваиваем файлу содержательное имя. Создан пустой файл документа.
5. Утверждаем создание документа щелчком мыши на свободном месте окна или нажатием клавиши <Enter>.
6. Двойной щелчок на имени файла открывает приложение, способное работать с этим файлом.

Создание ярлыка

Ярлыки служат для быстрого вызова программ или документов. Обычно их размещают на рабочем столе. От значков ярлыки отличаются наличием в левом нижнем углу маленькой стрелки. Двойной щелчок на ярлыке запускает программу или открывает документ.

Для того чтобы создать ярлык:

1. Раскрываем каталог так, чтобы нужная папка (файл) находилась в правом окне Проводника.
2. Щелчком правой кнопкой мыши на имени выделяем папку (файл), для которой требуется создать ярлык.
3. В раскрывшемся контекстном меню выбираем пункт **Создать ярлык**. Список в окне проводника добавляется значком и именем **Ярлык для ...**, где вместо многоточия записано имя папки (файла).
4. Утверждаем создание ярлыка и его имя щелчком мыши на свободном месте окна или нажатием клавиши <Enter>.

Выделение группы объектов

При копировании, перемещении, удалении объектов их желательно в целях экономии времени объединять в группы и выполнять эти операции над всей группой. При выделении группы объектов действуют следующие правила:

- если выделяемые объекты расположены в списке подряд, то выделение выполняется при нажатой клавише <Shift>. Сначала выделяется первый в списке объект, затем — последний (или наоборот);

- если выделяемые объекты расположены в разных местах списка, то они добавляются в группу по одному при нажатой клавише <Ctrl>.

Переименование объектов

Для того чтобы переименовать объект:

1. Делаем щелчок на объекте правой кнопкой мыши. Объект выделяется.
2. В раскрывшемся контекстном меню выбираем пункт **Переименовать**. Вид выделения изменяется.
3. Вписываем в прямоугольник имени новое имя объекта.
4. Утверждаем переименование щелчком мыши на свободном месте окна или нажатием клавиши <Enter>.

Копирование объектов

Копирование с помощью контекстного меню

Чтобы скопировать объект или группу объектов с помощью контекстного меню:

1. Выделяем объект или группу объектов копирования щелчком левой кнопки мыши.
2. Щелчком правой кнопкой на выделенных объектах раскрываем контекстное меню, в котором выбираем пункт **Копировать**.
3. Щелчком правой кнопки мыши выделяем объект, в который требуется поместить копию. В появившемся контекстном меню выбираем пункт **Вставить**.

Копирование с помощью панели инструментов Проводника

Чтобы скопировать объект или группу объектов с помощью панели инструментов Проводника:

1. Выделяем объект или группу объектов копирования щелчком левой кнопки мыши.
2. На панели инструментов Проводника нажимаем кнопку **Копировать в**. По этой команде открывается окно **Обзор папок**, в котором требуется найти папку, куда будет помещена копия, или создать ее (имеется кнопка **Создать папку**).
3. После того как папка найдена или создана, нажимаем кнопку **ОК**, инициируя процесс копирования.

Копирование перетаскиванием мышью при нажатой правой кнопке

Чтобы скопировать объект или группу объектов перетаскиванием мышью при нажатой правой кнопке:

1. Выделяем объект или группу объектов копирования щелчком левой кнопки мыши.
2. Нажимаем на выделенном объекте правую кнопку мыши и, не отпуская ее, перетаскиваем контур выделения к выбранной папке копирования. Папка выделяется.
3. При отпускании кнопки мыши рядом с папкой выводится контекстное меню с пунктами **Копировать**, **Переместить**, **Создать ярлыки**, **Отменить**. Выбираем пункт **Копировать**.

Копирование на дискету

Чтобы скопировать объект или группу объектов на дискету:

1. Выделяем объект или группу объектов копирования щелчком левой кнопки мыши.
2. Щелчком на выделенных объектах правой кнопкой мыши открываем окно контекстного меню.
3. В контекстном меню выбираем пункт **Отправить**.
4. В открывшемся списке выбираем пункт **Диск 3,5 (A)**.

Перемещение объектов

Перемещение папок, файлов

Для перемещения папок или файлов:

1. Выделяем объект или группу объектов, которые требуется переместить.
2. На панели инструментов Проводника нажимаем кнопку **Переместить в**. По этой команде открывается окно **Обзор папок**, в котором требуется найти папку, куда будет помещен объект (объекты), или создать ее (имеется кнопка **Создать папку**).
3. После того как папка найдена или создана и ее имя появилось в текстовом окне **Папка**, нажимаем кнопку **OK**, что завершает перемещение.

Перемещение ярлыка на рабочий стол

Для перемещения ярлыка на рабочий стол:

1. Щелчком правой кнопкой мыши на имени выделяем ярлык. Раскрывается контекстное меню.
2. В контекстном меню выбираем пункт **Отправить**. Раскрывается окно адресатов.
3. Выбираем пункт **Рабочий стол (Создать ярлык)**.
4. Кнопкой на панели задач **Свернуть все окна** освобождаем рабочий стол и убеждаемся, что созданный ярлык находится на столе.

Удаление объектов

При освоении работы с объектами файловой системы приходится создавать лишние (учебные) папки, копировать и переименовывать файлы и т. п. Желательно после практической работы весь "мусор" удалять. К операции удаления следует относиться внимательно. Можно случайно удалить файлы, которые могут быть востребованы. Много неприятностей может вызвать, например, переименование, перемещение и удаление файлов операционной системы. Поэтому все эксперименты желательно проводить в пределах папки Мои документы.

При удалении объекты помещаются в Корзину, где они хранятся до окончательного удаления.

Удаление с правой кнопкой мыши

Чтобы удалить объект с помощью правой кнопки мыши:

1. Делаем щелчок на объекте правой кнопкой мыши.
2. В раскрывшемся контекстном меню выбираем пункт **Удалить**.
3. В появившемся окне **Подтверждение удаления файла** нажимаем кнопку **Да** или отменяем удаление кнопкой **Нет**.

Удаление ярлыка с рабочего стола

Чтобы удалить ярлык:

1. Выделяем ярлык правой кнопкой мыши и в контекстном меню выбираем пункт **Удалить**.
2. В появившемся окне **Подтверждение удаления файла** нажимаем кнопку **Да** или отказываемся от удаления нажатием кнопки **Нет**.

Приложение 2

Функция форматирования *Format*

Функция форматирования `Format` возвращает результат преобразования заданного выражения согласно содержавшимся в выражении формата инструкциям форматирования.

Синтаксис функции:

`Format` (*`выражение`* [, *`формат`* [, первый день недели [, первая неделя года]]]),

где

- параметр *`выражение`* — любое допустимое выражение;
- параметр *`формат`* — допустимое наименование формата;
- последние два параметра используются при задании форматов даты.

Числовые форматы

Для задания параметра *`формат`* используются именованные и пользовательские числовые форматы.

- Именованные числовые форматы:
  - General Number — число без разделителей тысяч;
  - Currency — денежный формат, две цифры справа от десятичной точки;
  - Fixed — только две цифры справа от десятичной точки;
  - Standard — с разделителем тысяч и две цифры справа от десятичной точки;
  - Percent — отображает число в виде процентов;
  - Scientific — отображает число в формате с плавающей десятичной точкой;
  - Yes/No — выводит **Нет**, если число равно нулю, и **Да** в противном случае;
  - True/False — выводит **Ложь**, если число равно нулю, и **Истина** в противном случае;
  - On/Off — выводит **Выкл**, если число равно нулю, и **Вкл** в противном случае.

В листинге П2.1 приведен код с использованием именованных числовых форматов.

Листинг П2.1. Примеры именованных числовых форматов

```
Private Sub Form_Load()
  Show
  a = 5736.194
  Print Format(a, "General Number")      'результат: 5736,194
  Print Format(a, "Currency")            'результат: 5 736,19р
  Print Format(a, "Fixed")               'результат: 5736,19
  Print Format(a, "Standard")            'результат: 5 736,19
  Print Format(a, "Percent")             'результат: 573619,40%
  Print Format(a, "Scientific")          'результат: 5?74E+03
  Print Format(a, "Yes/No")              'результат: Да
  Print Format(a, "True/False")          'результат: Истина
  Print Format(a, "On/Off")              'результат: Вкл
End Sub
```

- В пользовательских числовых форматах используются символы:
  - `0` (ноль) — резервирует место для цифр числа; при отсутствии цифры отображается ноль;
  - `#` — резервирует место для цифр числа; при отсутствии цифры ничего не выводится;
  - `%` — резервирует процентное отображение числа;
  - `$` — выводит перед числом знак `$`.

В листинге П2.2 приведен код с использованием пользовательских числовых форматов.

Листинг П2.2. Примеры пользовательских числовых форматов

```
Private Sub Form_Load()
  Show
  a = 375.68
  Print Format(a, "####.###")            'результат: 375,68
  Print Format(a, "0000.000")            'результат: 0375,680
  Print Format(a, "##0.000")             'результат: 375,680
  Print Format(a, "###.#")               'результат: 375,7
  Print Format(45 + 25.7, "##")          'результат: 71
  Print Format(a, "$###.000")            'результат: $375,680
  Print Format(5 - 5, "Yes/No")          'результат: Нет
```

```
    Print Format(5 - 5, "True/False")      'результат: Ложь
    Print Format(5 + 5, "On/Off")          'результат: Вкл
  End Sub
```

Форматы даты и времени

Для задания формата даты и времени используются именованные форматы:

- General Date — выдает дату и время;
- Long Date — выдает дату в полном формате;
- Medium Date — выдает дату в кратком формате;
- Short Date — выдает дату в полном формате;
- Long Time — выдает время в полном формате;
- Medium Time — выдает часы и минуты;
- Short Time — выдает часы и минуты.

В листинге П2.3 приведен код с использованием пользовательских числовых форматов.

Листинг П2.3. Примеры форматирования выражений даты и времени

```
Private Sub Form_Load()
  Show
  Print Format(Now, "General Date")        'результат: 01.10.2003 12:02:25
  Print Format(Now, "Long Date")           'результат: 1 Октябрь 2003 г.
  Print Format(Now, "Medium Date")         'результат: 01-окт-03
  Print Format(Now, "Short Date")          'результат: 01.10.2003
  Print Format(Now, "Long Time")           'результат: 12:02:25
  Print Format(Now, "Medium Time")         'результат: 12:02
  Print Format(Now, "Short Time")          'результат: 12:02
  Print Format(Now)                        'результат: 01.10.2003 12:02:25
  Print Format(Now, "m / d /yy")           'результат: 10 . 1 . 03
  Print Format(Now, "dddd, mmmm dd, yyyy") 'результат: среда,Октябрь
                                           '01,2003
  Print Format(Now, "d-mmm")               'результат: 1-окт
  Print Format(Now, "mmmm-yy")             'результат: Октябрь-03
  Print Format(Now, "hh:mm AM/PM")         'результат: 12:02 PM
  Print Format(Now, "h:mm:ss a/p")         'результат: 12:02:25 p
  Print Format(Now, "d mmm h:mm")          'результат: 1 окт 12:02
  Print Format(Now, "ddddd ttttt")         'результат: 01.10.2003 12:02:25
End Sub
```

Список литературы

1. Алексеев П. А. Информатика 2001. — М.: Издательство СОЛОН-Р, 2001. — 364 с.
2. Ананьев А. И., Федоров А. Ф. Самоучитель Visual Basic 6.0. — СПб.: БХВ-Петербург, 2002. — 624 с.
3. Бирюков Б. В., Тростников В. Н. Жар холодных чисел и пафос бесстрастной логики. Формализация мышления от античных времен до эпохи кибернетики. — М.: "Знание", 1997. — 192 с.
4. Браун С. Visual Basic 6: учебный курс. — СПб.: Питер, 2001. — 576 с.
5. Гарнаев А. Ю. Visual Basic 6.0: разработка приложений. СПб.: БХВ-Петербург, 2001. — 448 с.
6. Гейн А. Г., Сенокосов А. И. Информатика: Учебник для 7—9 классов общеобразовательной школы. М.: Просвещение, 1997.
7. Дженнингс Р. Windows 95 в подлиннике: Пер. с англ. — СПб.: BHV-Петербург, 1995. — 480 с.
8. Есипов А. С. и др. Основы построения вычислительных комплексов средств автоматизации ПВО: Учебник. — М. Изд. МО СССР, 1984. — 592 с.
9. Есипов А. С. Информатика: Учебник по базовому курсу общеобразовательных учебных заведений. Изд. 3-е, перераб. и доп. — СПб.: Наука и Техника, 2003 г. — 400 с.
10. Есипов А. С., Паньгина Н. Н., Громада М. И. Информатика. Сборник задач и решений для общеобразовательных учебных заведений. — СПб.: Наука и Техника, 2001. — 368 с.
11. Зельднер Г. А. Программируем на языке Quick BASIC 4.5. — М.: ABF, 1996. — 432 с.
12. Карпов Б. Visual Basic 6: специальный справочник — СПб.: "Издательство Питер", 2000. — 416 с.
13. Касаткин В. Н. Информация, алгоритмы, ЭВМ: Пособие для учителя. — М.: Просвещение, 1991. — 192 с.
14. Левин А. Самоучитель работы на компьютере. Начинаем с Windows. — М.: Издательский дом "КноРус", 2001. — 688 с.

15. Мадер В. В. Школьнику об алгебре логики: Кн. для внеклас. чтения учащихся 10—11 кл. сред. шк. — М.: Просвещение, 1993. — 128 с.

16. Мельникова О. И., Бонюшкина А. Ю. Начала программирования на языке QBasic: Учебное пособие. — М.: Издательство ЭКОМ, 1997. — 304 с.

17. Мюллер А., Пейтон К. Office 2000 — "Издательство БИНОМ", 1999. — 480 с.

18. Основы информатики и вычислительной техники. В 2-х ч. Под ред. А. П. Ершова и В. М. Монахова. — М.: Просвещение, 1986.

19. Очков В. Ф., Рахаев М. А. Этюды на языках QBasic, Quick Basic, Basic Compiler. — М.: Финансы и статистика, 1995. — 368 с.

20. Паньгина Н. Н. Занятия по Visual Basic//Компьютерные инструменты в образовании, №№ 1—6, 2001, № 1, 2002.

21. Паньгина Н. Н. Изучение VBA в школе//Компьютерные инструменты в образовании, № 2, 2003.

22. Программы для общеобразовательных учреждений: Информатика. 2—11 классы. — М.: БИНОМ. Лаборатория знаний, 2003. — 205 с.

23. Семакин И. Г. и др. Информатика. Базовый курс для 7—9 классов. 2-е изд. — М.: Лаборатория Базовых Знаний, 2003.

24. Титаренко Г. Visual Basic 6.0. — Киев: Издательская группа BHV, 2001. — 448 с. (Серия "Коллекция BHV").

25. Формальная логика. Учебник/Коллектив авторов. Изд-во Ленинградского ун-та, 1977. — 357 с.

26. Шауцукова Л. З. Информатика: Учебник для 7—11 классов общеобразоват. учеб. заведений. В 2-х кн. Кн. 1: Теория (с задачами и решениями). — 2-е изд. перераб. и доп. — Нальчик: Эль-Фа, 1997. — 288 с.

27. Шафрин Ю. А. Основы компьютерной технологии: Учебное пособие для 7—11 классов по курсу "Информатика и вычислительная техника". — М.: АБФ, 1997. — 656 с.

28. Штайнер Г. Microsoft Office 200/. Справочник. — М.: Лаборатория Базовых Знаний, 2000. — 576 с.

29. Microsoft Office 2000. Шаг за шагом: Практич. пособие: Пер. с англ. — М.: "Издательство ЭКОМ", 2000. — 792 с.

30. Visual Basic 6.0: Пер. с англ. — СПб.: БХВ-Петербург, 2000. — 992 с.

Предметный указатель

Г

Д

Ж

З

И

К

Л

М

Н

О

П

Р

С

Т

У

Ф

Ц

Ч

Ш

Э

Я