ИНФОРМАТИКА

Учебное пособие для 11 класса
общеобразовательных учреждений
с белорусским и русским языками обучения
с 12-летним сроком обучения
(базовый и повышенный уровни)

Допущено
Министерством образования
Республики Беларусь

2-е издание, дополненное

Минск «Народная асвета» 2008

УДК 004(075.3=161.1)
ББК 32.81я721
И74

Авторы:

А. Е. Пупцев — главы 1, 2, 4, 6, Н. П. Макарова — главы 2, 3,
Г. А. Заборовский — глава 5, А. А. Черняк — глава 5

Рецензенты:

кафедра прикладной математики и информатики
Белорусского государственного педагогического университета им. Максима Танка
(зав. кафедрой, канд. физ.-мат. наук, доцент *А. А. Бейда*),
кафедра экономической информатики Белорусского государственного аграрного
технического университета (зав. кафедрой, канд. пед. наук, доцент *О. Л. Сапун*),
учитель информатики высшей категории СШ № 209 г. Минска *Л. И. Мосткова*

Информатика : учеб.. пособие для 11-го кл. общеобразоват. учреж-
И74 дений с белорус. и рус. яз. обучения с 12-летним сроком обучения (базовый
и повышенный уровни) / А. Е. Пупцев [и др.]. — 2-е изд., дополненное. —
Минск : Нар. асвета, 2008. — 223 с. : ил.
ISBN 978-985-12-1998-4.

УДК 004(075.3=161.1)
ББК 32.81я721

ISBN 978-985-12-1998-4

От авторов

Уважаемый старшеклассник!

В этом году Вы продолжите углублять свои знания в области информационных технологий, начнете осваивать основы программирования.

Данное пособие содержит материал для изучения информатики на базовом и повышенном уровнях. Учебный материал, предназначенный для изучения на повышенном уровне, отмечен звездочкой (*).

Учебное пособие состоит из шести глав.

Изучив главу 1 «Информационные модели, системы и технологии», Вы углубите знания в области построения и использования информационных моделей, расширите представления об информационных системах и технологиях.

В главе 2 «Аппаратное и программное обеспечение компьютера» содержится материал об архитектуре современного компьютера, дается классификация программного обеспечения, рассматриваются возможности различных операционных систем.

В главе 3 «Основы программирования» Вы познакомитесь со средой программирования, с основными алгоритмическими структурами, графическими возможностями языка программирования Паскаль. Освоив азбуку программирования, Вы сможете решать задачи из разных предметных областей и сфер человеческой деятельности, помогая в учебе себе, товарищам, в работе — учителям, родителям.

Глава 4 «Технология обработки информации в системах управления базами данных» изучается на повышенном уровне. Она познакомит Вас с возможностями системы управления базами данных Microsoft Access. В процессе изучения материала главы Вы будете создавать таблицы, формы, отчеты в базе данных. Научитесь использовать фильтры, выполнять сортировку данных и создавать различного вида запросы.

Материал главы 5 «Вычислительные методы и технологии» предназначен для изучения на повышенном уровне. Изучив его, Вы познакомитесь с использованием универсальной математической системы Mathcad для решения уравнений, систем уравнений, неравенств, а также построения графиков функций.

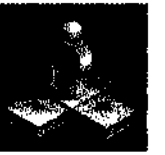
Глава 6 «Коммуникационные технологии» содержит материал о средствах обмена информацией в режиме реального времени в сети Интернет, о способах и скорости передачи информации в сети, о телекоммуникациях в сфере образования, культуры, коммерческой и рекламной деятельности.

Чтобы облегчить усвоение учебного материала, новые понятия в тексте выделены синим цветом. Материал, на который следует обратить внимание, отмечен значком  .

Главы учебного пособия разбиты на параграфы, а параграфы — на пункты. К некоторым пунктам и параграфам предложены контрольные вопросы, для практической работы даны упражнения.

Желаем успехов в изучении информационных технологий и основ программирования.

1 ГЛАВА



ИНФОРМАЦИОННЫЕ МОДЕЛИ, СИСТЕМЫ И ТЕХНОЛОГИИ

§ 1. Информация. Представление и измерение информации

1.1. Понятие информации. Виды и свойства информации

Жизнь человека постоянно связана с информацией, которую он получает из окружающего мира.

Информация является фундаментальным понятием и не имеет строгого определения. Свое видение этого понятия предлагают многие науки: философия, антропология, физика, биология, экономика, информатика и др.

Термин «информация» произошел от латинского слова *informatio*, что означает *разъяснение, осведомление, изложение*.

Информация такое же важное понятие, как и материя. Н. Винер писал: «Информация — это информация, а не материя или энергия».

Вещество и энергия, материя в целом, никуда не исчезают, а только переходят из одной формы в другую. В отличие от них информация может появляться и исчезать.

Физика изучает информацию, присущую процессам, существующим в неорганическом мире. Биология рассматривает информацию о живой природе, а экономическая информация отражает социально-экономические процессы.

Под **информацией** в информатике будем понимать сведения, отражающие свойства и состояния объектов, явлений и процессов в природе, обществе, технических системах, которые уменьшают степень неопределенности и неполноту знаний о них.

На рисунке 1.1 представлены некоторые существующие классификации видов информации по различным признакам.

Эффективность использования информации человеком во многом зависит от ее качества. Качество информации определяется такими ее свойствами, как *достоверность, полнота, актуальность, полезность, понятность* и др.

Достоверность информации определяется ее способностью отображать реальное состояние объектов или процессов.

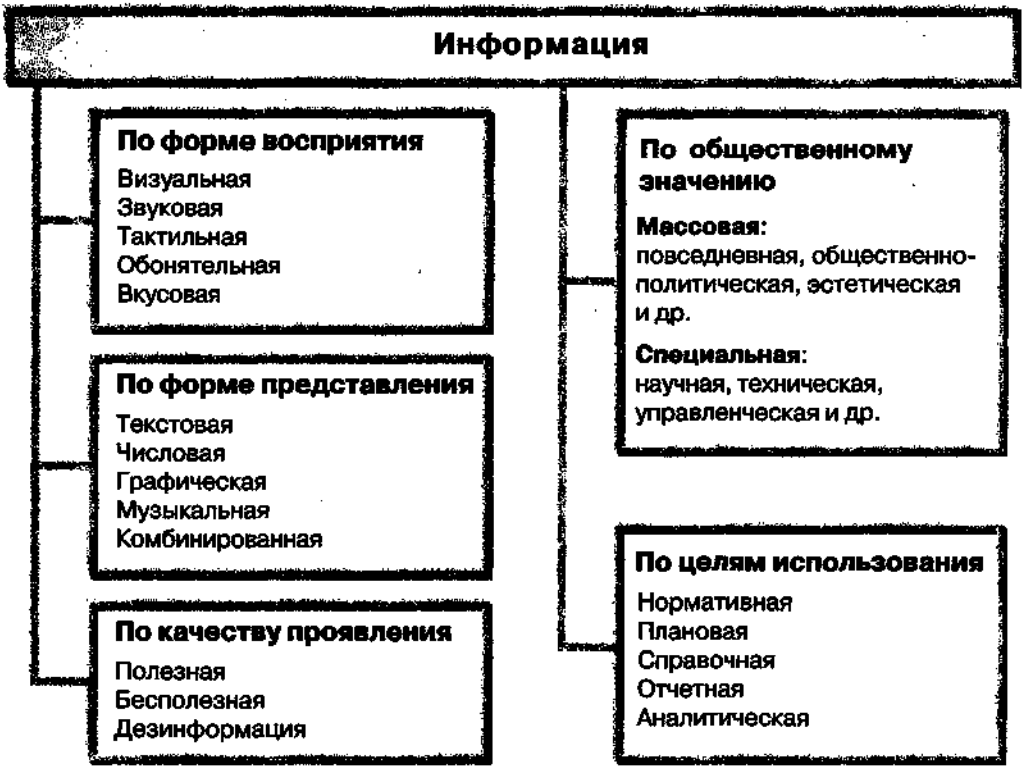


Рис. 1.1

Полнота информации означает, что ее достаточно для принятия получателем правильных решений или формирования объективных выводов.

Актуальность информации определяется степенью ее ценности в данный момент времени.

Информация является полезной, если получатель может на ее основе решать все требуемые задачи.

Понятность характеризует степень восприятия информации получателем.

1.2. Представление информации

Представление информации в живой природе и технике осуществляется в различных формах.

Одной из форм представления информации и обмена ею между людьми являются языки. Языки разделяются на *естественные* и *формальные*.

Люди используют естественные языки, например белорусский, русский, английский и т. д. В основе естественных языков лежит алфавит, который состоит из определенного набора знаков. Таким образом, язык является знаковой системой. В каждом языке существуют правила для работы с ним. В устной речи на естественном языке элементами информации являются слова, в письменной — буквы или иероглифы.

Кроме естественных языков, человек для представления информации и обмена ею использует формальные языки: азбука Морзе, языки программирования, языки представления химических, математических формул, язык музыки (нотная грамота) и др.

Формальные языки — это языки, придуманные и разработанные человеком для определенных целей. В отличие от естественных языков формальные состоят из специальных знаков и записываются с помощью строгих правил синтаксиса и грамматики.

Информация передается от источника к приемнику с помощью сигналов, например электрических, световых, звуковых.

Сигнал (от латинского *signum* — *знак*) — изменяющийся во времени физический процесс, с помощью которого передается сообщение.

! Сообщение — форма представления информации в виде совокупности знаков (символов).

В современном компьютере обрабатывается числовая, текстовая, графическая и звуковая информация. Процессор обрабатывает информацию, представленную в цифровой форме. Задачей устройств ввода—вывода компьютера в зависимости от их назначения является преобразование информации в форму, понятную компьютеру или человеку.

Мы уже знакомились с понятием *кодирования* информации и формами представления числовой, текстовой и графической информации в компьютере. Под кодированием информации понимается процесс ее преобразования в соответствии с определенными правилами. В результате этого процесса изменяется форма представления информации. Процесс, обратный кодированию, называют **декодированием**.

Числовая информация в компьютере представляется в двоичной системе счисления.

Для кодирования текстовой информации в компьютере используются кодовые таблицы, например ASCII или Unicode. Нам уже известно из базового курса информатики, что таблица ASCII поддерживает 8-разрядный двоичный код. Это значит, что каждый символ закодирован последовательностью из 8 нулей и единиц.

Мы знаем, что существует два вида представления изображений: растровое и векторное.

Растровое изображение — это совокупность точек (пикселей). Координаты каждой точки и ее свойства: цвет, яркость — выражаются с помощью двоичного кода. Векторное изображение хранится в виде совокупности числовых значений свойств объектов изображения, которые представлены в двоичном коде.

Звук представляет собой волну, которая характеризуется частотой и амплитудой. Двоичное кодирование звуковой информации в компьютере заключается в представлении звуковой волны в виде последовательности нулей и единиц. Компьютерное устройство, например микрофон, преобразует звуковую волну в электрический сигнал, который затем преобразуется в цифровую форму.

1.3. Измерение информации

Рассматривая измерение информации, мы будем использовать понятие *данные*.

! Данные — это информация, представленная в формализованном виде, пригодном для автоматической обработки техническими устройствами, при участии человека.

Для измерения информации применим *синтаксический* подход.

Определение количества информации при этом подходе предполагает рассмотрение носителей информации, способов представления и кодирования информации, скорости ее передачи и др.

При этом подходе для измерения информации применяется параметр — *объем данных* $V_д$.

Для передачи информации используются сообщения. Объем данных $V_д$ в сообщении, закодированном с помощью знаковой системы, определяется числом знаков (символов), умножаемым на количество информации, которое выделено под один знак.

Минимальное количество информации, для кодирования которой достаточно одного двоичного разряда, называют *битом (bit)*. Слово «бит» сформировано из двух английских слов: *binary — двоичный, digit — знак*. Бит может принимать только два значения: 0 или 1.

Для удобства используется единица измерения количества информации — *байт (byte)*, который состоит из 8 последовательных бит:

$$1 \text{ байт} = 2^3 \text{ бит} = 8 \text{ бит}.$$

При работе с большими объемами информации используются и более крупные единицы измерения объема данных:

1 кбайт (килобайт) = 1024 байт = 2^{10} байт;
1 Мбайт (мегабайт) = 1024 кбайт = 2^{20} байт;
1 Гбайт (гигабайт) = 1024 Мбайт = 2^{30} байт;
1 Тбайт (терабайт) = 1024 Гбайт = 2^{40} байт.

?

1. **Что понимается под информацией в информатике?**
2. **Какие виды информации Вы знаете?**
3. **Назовите основные свойства информации.**
4. **Для чего человек использует естественные и формальные языки? Приведите примеры таких языков.**
5. **Как представляется в компьютере информация различных видов?**
6. **Какие единицы измерения информации при синтаксическом подходе Вы знаете?**
7. **Что такое бит и байт?**

Упражнения

1. Ответьте на вопрос, что больше:

   а) 9 бит или 1 байт; б) 1 кбайт или 1000 байт;
   в) 2^{30} байт или 1 Тбайт; г) 1025 Мбайт или 1 Гбайт.

2. Определите емкость типичных носителей информации компьютера, которые Вы используете в кабинете информатики:

   а) винчестер (жесткий диск); в) CD диск;
   б) DVD диск; г) Flash-память.

§ 2. Информационные модели

Понятия модель и моделирование не являются для нас новыми. Под *моделью* понимается объект, используемый с какой-то целью вместо другого объекта. Процесс *моделирования* позволяет создавать и исследовать разные модели. Модели могут быть разделены на два больших класса: *материальные* и *информационные*.

Информационная модель — это совокупность информации, описывающая наиболее существенные свойства, состояния объекта, процесса, явления, представленная в знаковой или образной форме.

Знаковая информационная модель представляет собой описание объекта, процесса на каком-либо естественном или формальном языке, а образная модель представляется в виде рисунков, фотографий, схем и т. д.

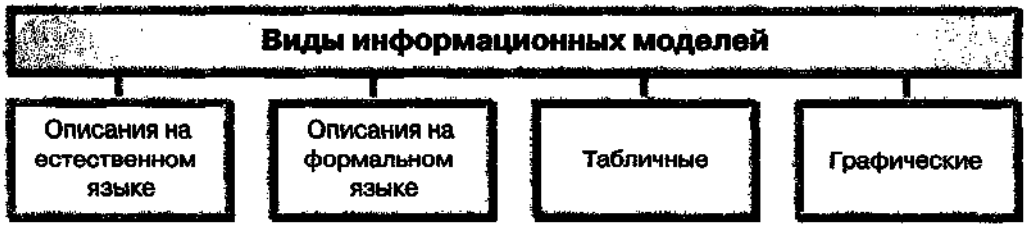


Рис. 1.2

Информационная модель размещается на определенном носителе информации: бумаге, видеопленке, магнитном диске и др.

Виды информационных моделей представлены на рисунке 1.2.

Человек на протяжении всей жизни составляет описания предметов, объектов, ситуаций, происшествий на естественном языке. При составлении модели на этом языке необходимо ясно и понятно формулировать предложения, использовать проверенные факты, нужные понятия и термины.

Примерами словесных моделей является информация в учебниках, произведения художественной литературы, сводки происшествий и др.

Для создания словесных описаний на компьютере мы используем текстовые редакторы, с помощью которых создаются текстовые документы. При разработке словесного описания необходимо иметь объект — текст, среду для набора текста, например текстовый редактор Word, определить параметры оформления текста: тип шрифта, размер, начертание, абзацный отступ и т. д.

Многие модели, представленные на естественном языке, описывают последовательности действий, процессов и представляются в виде алгоритмов.

Информационные модели, содержащие описания на формальном языке, содержат математические и химические формулы, алгоритмы, представленные на языках программирования и т. д. Например, формулы математики описывают соотношения между количественными характеристиками объекта моделирования на математическом языке.

К графическим информационным моделям относятся схемы, карты, чертежи, планы, графики и др.

Широкое распространение получили табличные информационные модели. В табличных моделях информация может размещаться разными способами: «объект — свойство», «объект — объект».

В таблице вида «объект — свойство» в ячейках первого столбца располагаются названия объектов (например, «Дата» в таблице «Погода»), а в остальных столбцах размещаются их свойства (информация о температуре, давлении, влажности).

Погода

Дата	Температура, °С	Давление, мм рт. ст.	Влажность, %
29.06.07	22	745	79
30.06.07	24	747	73
1.07.07	27	748	68
2.07.07	29	744	78
3.07.07	25	745	77

В таблице вида «объект — объект» в первой строке размещаются объекты (например, «Высказывание *A* и *B*» в «Таблице истинности логической операции **И**»), а в последующих строках — значения их свойств («Истина» или «Ложь»).

Таблица истинности логической операции И

Высказывание *A*	Высказывание *B*	Высказывание *A* и *B*
Истина	Истина	Истина
Истина	Ложь	Ложь
Ложь	Истина	Ложь
Ложь	Ложь	Ложь

Для более полного описания свойств или характеристик какого-либо объекта может использоваться несколько информационных моделей разных видов.

Пример. Описание химического вещества «углекислый газ».

Вид модели	Модель
Описание на естественном языке	Бесцветный газ, молекула которого состоит из одного атома углерода и двух атомов кислорода. Относительно хорошо растворим в воде, образует слабую угольную кислоту.
Описание на формальном языке	CO_2
Графическая модель (схема)	

?

1. Что понимается под моделью?
2. Какой процесс называют моделированием?
3. Какая модель называется информационной?
4. Какие виды информационных моделей Вам известны?
5. Приведите примеры информационных моделей, описанных на естественном языке и на формальном языке.
6. Какие модели относятся к графическим информационным моделям? Приведите примеры графических моделей.
7. Приведите примеры табличных информационных моделей.

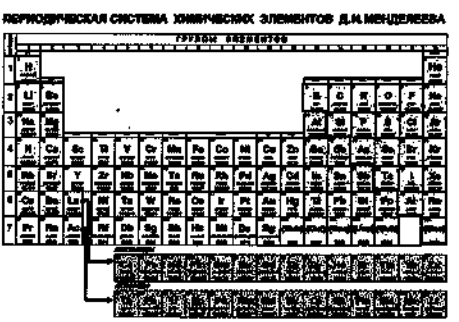
Упражнения

1. Ответьте на вопрос, к какому виду информационных моделей относятся эти модели:

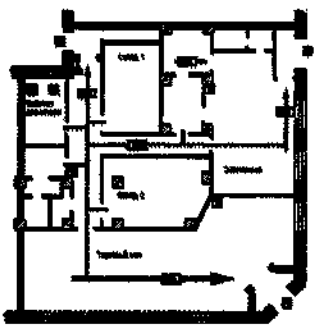
а) $a = \frac{4SR}{bc} = 2R\sin\alpha$;

б) описание автомобильной аварии очевидцем;

в)



г)



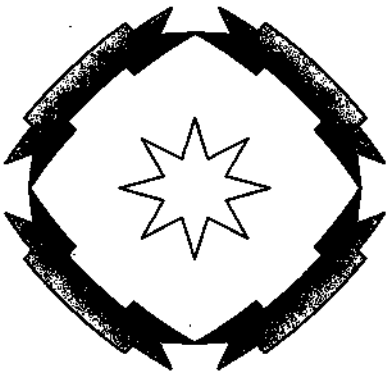
2. В табличном процессоре Excel создайте таблицу.

Название книги	Цена, р.	Количество книг на складе, шт.	Продано, шт.	Сумма, р.
Компьютерный дизайн	39 000	203	15	
Интернет для всех	15 000	502	37	
СУБД Access	25 000	240	32	
Редактор Word	18 000	379	48	
ОС Windows XP	26 000	233	26	
Электронная почта	10 000	387	25	
Итого				

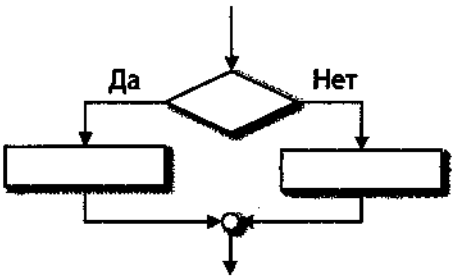
Проведите необходимые вычисления для пустых ячеек. Постройте столбчатую диаграмму, отражающую продажу книг. Модели каких видов Вы получили?

*3. Создайте в текстовом редакторе рисунок *а* и схему *б*. Какого вида информационные модели Вы получили?

а)



б)



§ 3. Информационные системы и технологии

3.1. Системы и информационные системы

Человек соприкасается с различными по своему назначению системами в быту, на производстве, в образовании, медицине и других сферах деятельности. Системы существуют в живой и неживой природе. Они состоят из различных элементов: планет, рек, растений, животных, людей, технических устройств, деталей механизмов, информационных ресурсов, математических уравнений и т. д.

Слово «система» (*systema*) имеет греческое происхождение и означает *целое, состоящее из частей*.

Мы часто встречаем такие слова и словосочетания, как биосистема, экологическая система, государственная система, система обучения, система социального обеспечения, система здравоохранения и т. д. Практически любой работающий механизм или устройство (компьютер, телевизор, автомобиль, банкомат) можно назвать сложной системой.

Разные науки подходят к определению системы по-разному, поэтому строгого определения этого понятия не существует.

Под *системой* будем понимать целостную совокупность элементов, которые находятся в определенных связях или отношениях друг с другом.

В курсе информатики мы уже знакомились с разными системами: операционными, файловыми, системами управления базами данных и т. д.

Добавление к понятию «система» слова «информационная» позволяет определить ее основное функциональное назначение — работа с информацией.

Информационная система (ИС) — совокупность средств и методов сохранения, обработки, поиска и выдачи информации, обслуживаемая и используемая человеком.

Широкое применение ИС на основе вычислительной техники началось во второй половине XX в. Вначале информационные системы обрабатывали расчетные бухгалтерские документы. Затем еще одной их функцией стала быстрая подготовка отчетности на производстве. Последние десятилетия XX в. ознаменовались появлением ИС управленческого назначения, что способствовало проведению эффективного контроля управленческих решений.

В настоящее время информационные системы позволяют принимать стратегические решения при управлении производством, в науке, распределять материальные и информационные ресурсы, обеспечивать конкурентоспособность предприятий различных форм собственности.

Возможности информационно-поисковых (справочных) систем, географических и обучающих информационных систем нам уже известны из базового курса информатики.

Любая современная информационная система состоит из подсистем, включающих *техническое, программное, информационное, организационное* и *правовое обеспечения*.

Техническое обеспечение включает в себя комплекс технических средств, обеспечивающих работу ИС: компьютеры разных моделей, устройства сбора, накопления, обработки, выдачи и передачи информации, различная оргтехника и т. д.

Программное обеспечение ИС представляет собой совокупность компьютерных программ, математических методов обработки информации, предназначенных для реализации необходимых системе целей и задач.

Информационное обеспечение ИС имеет единую систему классификации и кодирования информации, унифицированные системы документов, создаваемых согласно международным и государственным стандартам.

Организационное и **правовое** обеспечения ИС включают в себя определение порядка работы обслуживающего персонала, правовые нормы, регламентирующие порядок получения информации и доступа к ней пользователей.

3.2. Технологии и информационные технологии

Определение информационных технологий тесно связано с понятием «технология».

Термин «технология» происходит от греческого слова *technē* — *наука об умении, мастерстве, искусстве*. Этот термин имеет множество значений.

В узком смысле под технологией понимают процесс функционирования определенных орудий производства, методы, приемы и режимы работы механизмов, станков, различной аппаратуры.

В более широком смысле с помощью технологий описываются многие производственные, экономические, социальные, культурные и другие процессы, происходящие в обществе.

Целью технологии материального производства является выпуск продукции, целью информационной технологии — создание информации, используемой человеком.

Понятие «информационные технологии» впервые было применено в конце 50-х годов XX в. в Англии и США, однако его активное использование началось в 80-е годы XX в., когда под влиянием новых технологий в обществе начали широко использовать термин «информация».

Под **информационной технологией (ИТ)** понимается процесс, состоящий из методов, способов и приемов, позволяющих осуществлять обработку, хранение, передачу, поиск и выдачу информации.

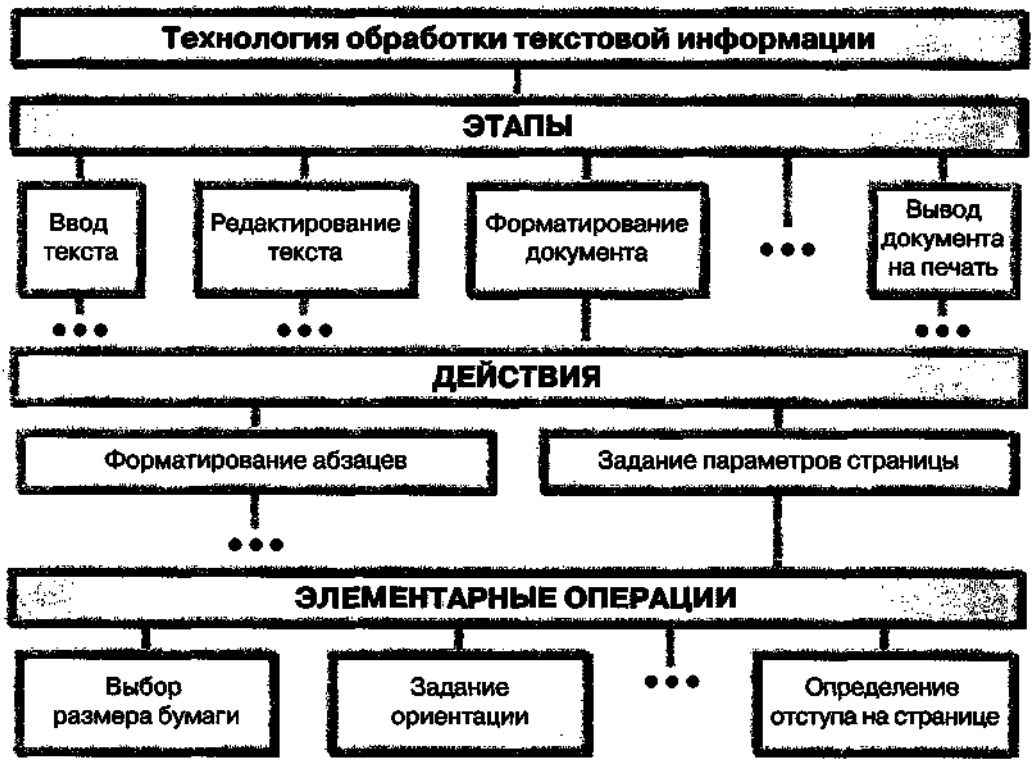


Рис. 1.3

В широком смысле ИТ часто рассматривают как совокупность действий по осуществлению информационных процессов во всех сферах человеческой деятельности: производственной, управленческой, финансовой, научной, социальной, культурной.

Информационную технологию, описывающую процесс, можно представить в виде иерархической структуры, разделив ее на *этапы*, *действия* и *элементарные операции*. Например, технология обработки текстовой информации содержит следующие этапы, действия и элементарные операции (рис. 1.3).

Компьютерные информационные технологии классифицируют по методам, способам и приемам обработки информации, представленной в разной форме. Эта классификация показана на рисунке 1.4.

Рис. 1.4

Классификация технологий является трудной задачей. Например, мультимедийная технология включает в себя технологию обработки текстовой и графической информации, а сетевая технология — технологию создания веб-страниц.

Различие между информационными системами и информационными технологиями является очень важным. Следует четко понимать, что в основе информационной технологии лежит, прежде всего, процесс, выполнение которого построено на основе методов и способов деятельности, а в основе информационной системы лежат программные и технические средства.

Например, независимо от того, какие информационные системы мы используем при работе с базами данных, действия и операции эффективного поиска информации в них, которые отражает технология работы с базами данных, являются одинаковыми.

Когда мы говорим о технологии обработки текстовой информации, мы можем не связывать ее с конкретной компьютерной программой, так как такое понятие этой технологии, как редактирование текста, является общим для любых компьютерных программ, в которых обрабатывается текстовая информация.

?

1. Что понимается под системой?
2. Какую систему называют информационной?
3. Из каких подсистем состоит современная информационная система?
4. Что понимается под технологией?
5. Какую технологию называют информационной?
6. Назовите ИТ, которые Вы знаете.
7. В чем различие между информационными системами и информационными технологиями?

ГЛАВА 2

АППАРАТНОЕ И ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ КОМПЬЮТЕРА

§ 4. Архитектура компьютера

4.1. Общее представление об архитектуре компьютера

Архитектура компьютера определяется конструкцией и структурной организацией его функциональных блоков (компонентов), описанием принципов их работы и взаимодействия на аппаратном и программном уровнях.

Архитектура компьютера часто разделяется на отдельные части: аппаратную, программную, сетевую архитектуру и др. Она обеспечивает пользователю возможность подключения различных устройств и их замены, а также совместимость аппаратного, программного и информационного обеспечения.

Назначение основных устройств современного компьютера, логическая схема взаимосвязи его функциональных блоков нам уже известны из базового курса информатики.

* Классификация архитектуры компьютера может базироваться на характере информационных связей между процессором, памятью и устройствами ввода—вывода. Основываясь на данной классификации, архитектуру всего многообразия персональных компьютеров и электронных вычислительных машин можно представить двумя типами структур:

- структуры использования каналов ввода—вывода (рис. 2.1);
- магистральной структуры (рис. 2.2).

Первый тип структуры предполагает, что между центральным процессором и оперативной памятью существует непосредственная связь. Связь между центральным процессором и устройствами ввода—вывода, а также между памятью и этими устройствами осуществляется с помощью специальных процессоров, которые называют *каналами ввода—вывода*. Это позволяет выполнять одновременно несколько операций ввода—вывода (см. рис. 2.1).

Второй тип структуры предполагает, что взаимодействие центрального процессора, памяти и устройств ввода—вывода выполняется через единое подключение к системной магистрали (см. рис. 2.2). Системная магистраль используется для передачи данных и адресов.

Рассматривая архитектуру компьютеров, следует выделить особенности использования компьютерной памяти. Отметим, что существует несколько подходов к хранению команд программ и данных в памяти компьютера.

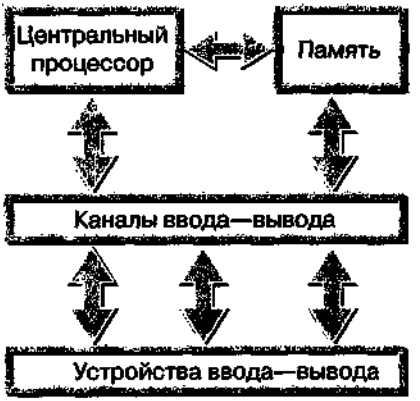


Рис. 2.1

Рис. 2.2

Один подход предложил известный американский ученый Джон фон Нейман (1903—1957). При этом подходе выполняемые команды программ и данные хранятся в одной и той же области памяти. Команды указывают, что необходимо выполнить и адреса данных, которые необходимо использовать. Подход Неймана позволил упростить устройство процессора.

Другой подход предполагает, что данные и программы используют разные области памяти. Это позволяет выполнять несколько параллельных операций: пока одна команда выполняется, вторая выбирается для выполнения. *

В следующих параграфах систематизируем и углубим наши знания о таких основных функциональных компонентах компьютера, как процессоры, виды памяти, типы внешних устройств.

?

1. Что понимается под архитектурой компьютера?

***2. Какими двумя типами архитектуры могут быть представлены персональные компьютеры и ЭВМ?**

4.2. Процессоры и их характеристики

Нам уже известно, что центральным устройством любого компьютера является *процессор*, который располагается на материнской плате, размещенной в системном блоке.

!

Процессор представляет собой устройство, предназначенное для обработки данных, которые находятся в его регистрах, в оперативной памяти, а также данных, размещенных во внешних портах процессора.

Для установки процессора на системной плате предназначен разъем, который называется **сокет**. Сокеты различаются по числу контактов и их расположению.

Существуют различные типы процессоров. Например, корпорация Intel предлагает процессоры Celeron, Pentium, а корпорация AMD — Duron и Athlon и т. д. У каждого типа процессоров есть свои достоинства и недостатки. Обычно выбор процессора основан на соотношении между ценой и производительностью для определенного круга задач.

Наиболее важными характеристиками современных процессоров является *тактовая частота, разрядность, рабочее напряжение, размер кэш-памяти.*

Исполнение каждой команды в процессоре занимает определенное количество тактов. Чем выше **частота тактов**, тем большее число команд может выполнить процессор в единицу времени. Тактовая частота измеряется в герцах (Гц) и более крупных единицах мегагерцах (МГц) и гигагерцах (ГГц). 1 МГц равен одному миллиону тактов в секунду, а 1 ГГц — одному миллиарду тактов в секунду. Таким образом, тактовая частота определяет важную характеристику процессора — его быстродействие.

Разрядность процессора показывает, сколько бит данных он может обработать в своих регистрах за один такт. Разрядность процессоров современных компьютеров 32 бит или 64 бит.

Рабочее напряжение процессора (измеряется в вольтах) обеспечивает материнская плата. С развитием вычислительной техники происходило постоянное снижение рабочего напряжения от 5 В до 3 В, что позволило повысить производительность процессора.

Нам уже известно назначение кэш-памяти для ускорения работы внешних устройств компьютера. Процессор также использует внутри себя **кэш-память**. Это снижает количество обращений процессора к оперативной памяти компьютера. Повышенный объем кэш-памяти имеют высокопроизводительные процессоры. Например, для процессора Pentium применен кэш объемом 16 кбайт.

Разделение микропроцессоров в зависимости от фирмы-производителя используется достаточно часто. В таблице 1 представлены некоторые характеристики процессоров.

Таблица 1

Производитель	Тип микропроцессора	Тактовая частота микропроцессора, ГГц	Кэш, кбайт	Разрядность, бит
Intel	P4-661	3,6	2048	64
Intel	Celeron	3,33	512	64
AMD	Athlon 64	3,5	128 + 512	64
AMD	Semptron	3,4	128 + 256	64

1. Для чего предназначен процессор?
2. Назовите характеристики современных процессоров.
3. Процессоры каких типов производятся в настоящее время?

4.3. Виды памяти компьютера

Одним из важнейших компонентов компьютера является его память. В памяти компьютера хранятся программы и данные. Существуют различные способы классификации памяти. Компьютерная память может быть разделена по скорости доступа к ней (медленная или быстрая), по времени хранения данных (долговременная, кратковременная, постоянная) и др.

Рассмотрим внутреннюю (электронную) и внешнюю память компьютера (рис. 2.3).

Оперативная память используется в основном для размещения выполняемых пользователем программ и данных в течение всего времени работы компьютера. Кроме этого, в оперативную память после включения компьютера записываются некоторые программы операционной системы. Эта часть оперативной памяти во время сеанса работы недоступна пользовательским программам.

Постоянная память содержит данные, используемые при работе с компьютером, которые при его отключении сохраняются. Содержимое ПЗУ записывается при изготовлении компьютера и не допускает корректировки в процессе эксплуатации. Если такая корректировка необходима, используются перепрограммируемые постоянные запоминающие устройства (ППЗУ). Перепрограммирование информации может осуществляться как в самом компьютере в ходе его эксплу-

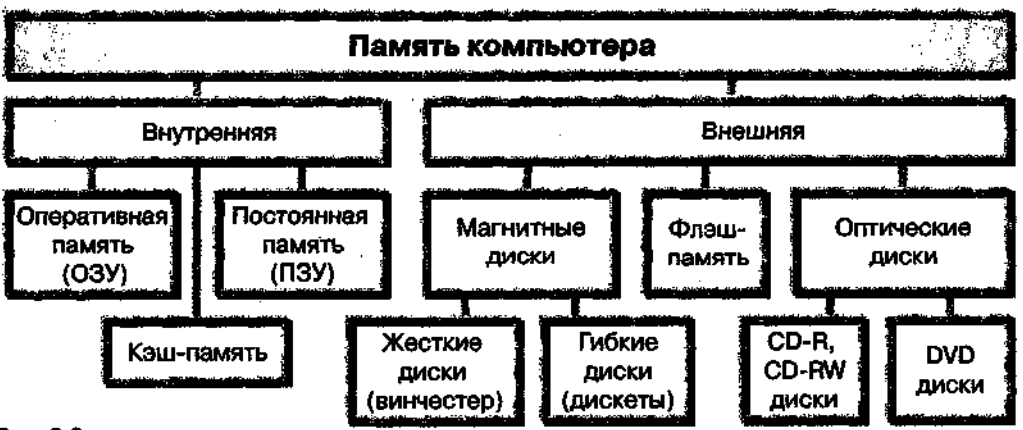


Рис. 2.3

атации (такая технология называется флэш-технологией), так и вне его (с помощью специальных устройств, называемых программаторами).

Для ускорения работы компьютера разработчиками была создана быстрая по времени доступа кэш-память. Идея использования кэш-памяти строится на том, что в ней хранятся наиболее часто используемые данные.

Каким бы ни был объем внутренней памяти компьютера, его всегда не хватает. В связи с этим большие объемы информации хранятся на внешней памяти, которая размещается на *магнитных* и *оптических* дисках (см. рис. 2.3).

К магнитным носителям информации относятся жесткий диск (винчестер) и гибкие диски (дискеты).

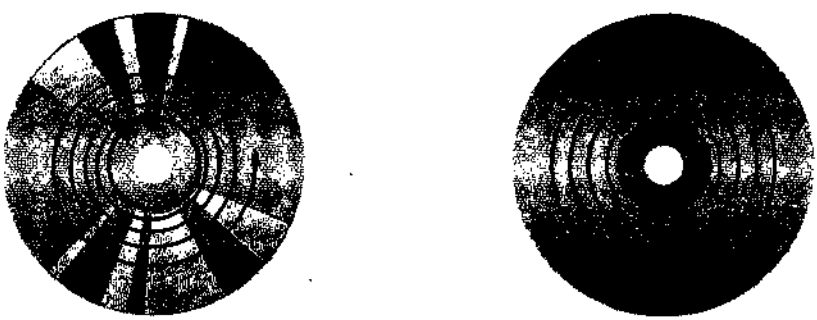
В настоящее время помимо магнитных дисков широко используются оптические (лазерные) диски.

Оптические компакт-диски делятся на CD-ROM (CD-R) (от английского Compact Disk-Read Only Memory — компакт-диск, предназначенный только для чтения) и CD-RW (от английского Compact Disk-ReWritable — перезаписывающийся компакт-диск), а также DVD (от английского Digital Video Disk — цифровой видеодиск). Объем информации, хранящийся на оптических дисках, может в настоящее время достигать 17 Гбайт и более.

В качестве внешней памяти широко используется также *флэш-память*.

* Компакт-диск CD-R представляет собой прозрачный полимерный диск, на одной стороне которого напыляется светоотражающий слой алюминия. Дополнительно этот слой защищается от повреждений слоем прозрачного лака. Компакт-диск CD-RW производится с помощью более сложной технологии. Слой красителя, размещенного на таком диске, может менять свои характеристики при нагреве лазерным лучом.

Дорожки оптических дисков, на которые записываются данные, имеют спиральную форму (рис. 2.4, *а*), дорожки магнитных дисков — кольцевидную (рис. 2.4, *б*).



а *б*

Рис. 2.4

Информация на магнитном диске располагается на дорожках, которые разделены на секторы. В одном секторе дорожки может быть размещено 128 байт, 256 байт, 512 байт или 1024 байт. Для сохранения файла дисковое пространство выделяется *кластерами*. Кластер является минимальной единицей размещения информации на диске и состоит из одного или нескольких смежных секторов дорожки (рис. 2.5). Если для размещения файла смежных кластеров не хватает, то для него выделяются другие несмежные кластеры. В результате возникает фрагментирование, которое снижает скорость считывания файла. *

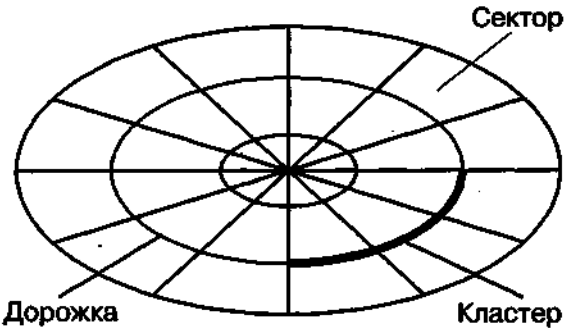


Рис. 2.5

1. Как подразделяется память по способу размещения?
2. Из каких частей состоит внутренняя память компьютера?
3. Для чего используется оперативная и постоянная память компьютера?
4. Какие виды оптических дисков вам известны?
*5. Как располагается информация на магнитных дисках?

4.4. Внешние типовые устройства

Современный компьютер имеет различные внешние типовые (периферийные) устройства. Они могут использоваться для ввода, вывода информации или обеспечивать выполнение сразу двух этих функций. Например, модем, даже если он встроенный, передает информацию от компьютера в сеть и наоборот.

Ввод и вывод информации могут выполняться с помощью специальных *интерактивных досок*, подключенных к компьютеру. **Интерактивная доска** — это устройство, позволяющее лектору или докладчику объединить два различных инструмента: экран для отображения информации и обычную маркерную доску. Такая доска может быть также разделена на две области: одна область служит для отображения информации, а другая содержит специальные элементы для управления этой доской и ввода информации.

Каждое устройство ввода или вывода информации подключается с помощью специального кабеля к контроллеру этого устройства через соответствующий порт (разъем). В некоторых случаях контроллер может быть размещен прямо на устройстве. С назначением и типами портов мы знакомились ранее в курсе информатики.

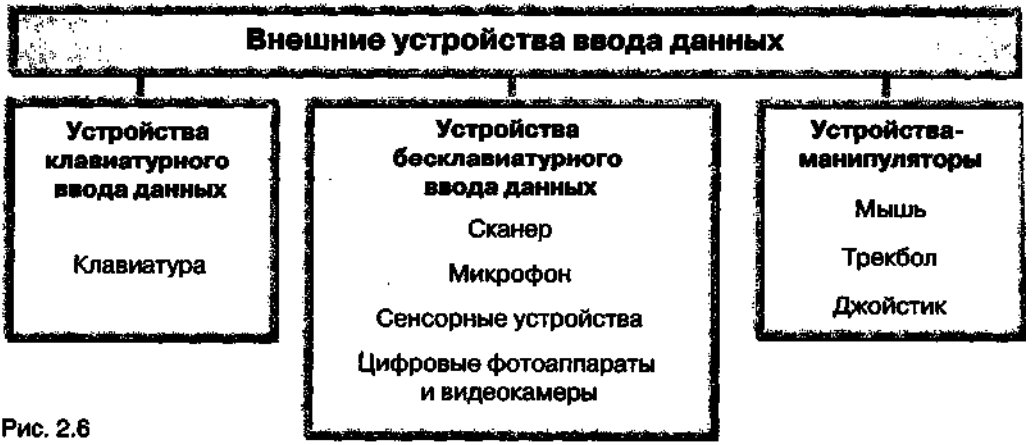


Рис. 2.6

Внешние устройства ввода информации предназначены для передачи информации в компьютер и представляют эту информацию в понятной для компьютера форме.

Для систематизации знаний об устройствах ввода данных представим их классификацию на рисунке 2.6.

Среди устройств ввода данных в компьютер нам уже знакомы клавиатура, микрофон, мышь и сканер. К устройствам бесклавиатурного ввода данных в компьютер относятся также сенсорные устройства. Эти устройства способны распознавать зрительные образы, звуки, прикосновения, уровень температуры и т. д. В настоящее время сенсорные устройства широко используются в робототехнике и при разработке искусственного интеллекта. Наиболее часто встречающимися на практике сенсорными устройствами являются: оптические перья, сенсорные интерактивные экраны, графические планшеты (дигитайзеры), сенсорная клавиатура.

Внешние устройства вывода информации предназначены для передачи информации из компьютера и преобразуют эту информацию в форму, понятную человеку.

В настоящее время существуют мониторы с вакуумным кинескопом (электронно-лучевой трубкой) и жидкокристаллические мониторы. Толщина жидкокристаллического монитора составляет только 15—20 % от толщины традиционного монитора.

Существуют принтеры нескольких типов: матричные, струйные и лазерные. Выбор принтера зависит от объема печати, качества печатаемых документов и др.

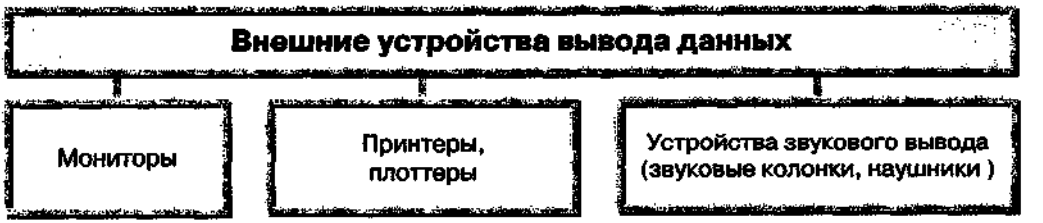


Рис. 2.7

Матричные принтеры работают медленно по сравнению с другими. Струйные принтеры позволяют получать документы хорошего качества, в том числе и цветные. Если количество печатаемых документов очень большое, лучше использовать лазерный принтер.

Для вывода графических документов большого размера (схем, карт) используются плоттеры.

Для систематизации знаний об устройствах вывода данных представим их классификацию на рисунке 2.7.

Обычно периферийные устройства компьютера подключаются к нему с помощью проводов.

Однако в настоящее время развивается технология беспроводного соединения устройств Bluetooth. Она позволяет разным устройствам обмениваться информацией, присоединяться к компьютерным сетям, управлять бытовой техникой.

Технология Bluetooth использует небольшие приемопередатчики малого радиуса действия (до 100 м), встроенные в устройства или подключаемые через свободный порт. В отличие от связи между устройствами, которая базируется на инфракрасном излучении в зоне прямой видимости, связь с помощью технологии Bluetooth может осуществляться между устройствами, разделенными препятствиями.

?

1. Для чего предназначены устройства ввода и вывода информации в компьютер?
2. Перечислите типовые внешние устройства:
   а) ввода информации;
   б) вывода информации;
   в) ввода и вывода информации.

*4.5. Выбор аппаратного обеспечения компьютера

Рассмотрим некоторые основные подходы к выбору аппаратного обеспечения.

Компьютеры могут быть собраны в широко известных фирмах-производителях, например IBM, Hewlett Packart и др. Тогда они имеют специальные товарные знаки. Эти товарные знаки указывают на изготовителя компьютера, являются определенной гарантией оказания разных сервисных услуг пользователю.

Компьютеры, сборка которых производилась на неизвестных фирмах, должны иметь сертификаты. Важными являются сертификаты на совместимость, энергосбережение, внешнее излучение и др.

При выборе конфигурации компьютера пользователь обращает внимание на различные блоки и устройства:

- тип микропроцессора и материнской платы;
- объем оперативной памяти;
- объем внешней памяти и состав устройств, относящихся к ней;
- тип видеомонитора и видеоадаптера;
- тип принтера, клавиатуры, мыши, модема и других дополнительных устройств.

Нам уже известно, что производительность компьютера отражается в характеристиках его процессора (тактовая частота, разрядность, количество регистров, объем кэш-памяти) и зависит от наличия сопроцессора, объема ОЗУ и его быстродействия, пропускной способности системной и локальной шины, быстродействия накопителей на жестких дисках, объема памяти видеоадаптера и т. д.

Компьютерные блоки и устройства постоянно совершенствуются, улучшаются многие их характеристики. При выборе микропроцессора старого типа следует помнить, что его быстродействие уступает более новым современным типам.

Оперативная память компьютера должна быть достаточно большой, например 256 Мбайт, что позволит работать с разнообразным программным обеспечением.

Винчестер в компьютере, конечно, должен иметь гигабайты памяти. Следует обратить внимание на время доступа к информации при работе с ним (8—10 мс).

При выборе видеомонитора необходимо уточнить его цветность, тип, размер экрана, разрешающую способность и др. Стоимость монитора часто составляет 25 % и более от стоимости всего компьютера. Частота кадровой развертки менее 70 Гц приводит к мерцанию экрана и плохо влияет на зрение.

Монитор при работе излучает (ультрафиолетовое и радиоизлучение). Имеет место и электростатическое поле. Поэтому рекомендуется использовать фильтры (сеточные, пленочные или стеклянные). Чем современнее монитор, тем уровень его излучения меньше.

При выборе принтера следует сразу решить, каким он будет: черно-белым или цветным, с узкой или широкой кареткой. Самые лучшие принтеры на данный момент лазерные. Они имеют высокую разрешающую способность и скорость печати. При этом лазерные принтеры — самые дорогие. Печать хорошего качества

могут предложить и более дешевые современные струйные принтеры. Их выбор оправдан, если количество распечатываемых документов невелико, так как высока стоимость расходных материалов.

1. На какие функциональные блоки и устройства следует обратить внимание пользователю при подборе конфигурации компьютера?
2. Какие факторы влияют на производительность компьютера?

*§ 5. Программное обеспечение компьютера

Основу компьютера наряду с его аппаратным обеспечением (от английского *hardware — твердое изделие*) составляет программное обеспечение (от английского *software* — мягкое изделие, что подчеркивает возможность программного обеспечения модифицироваться, изменяться).

Программное обеспечение (ПО) — совокупность программ, используемых в компьютере. Под программным обеспечением также понимают область деятельности по его проектированию и разработке, которая включает в себя технологию проектирования программ, методы тестирования, методы доказательства правильности, методы документирования программ и др.

Являясь неотъемлемой частью компьютерной системы, программное обеспечение определяет сферу применения конкретного компьютера. Используя имеющееся программное обеспечение и устанавливая на компьютере новые программы, можно превратить его в рабочее место бухгалтера или конструктора, директора школы или агронома, решать различные задачи: готовить рефераты, редактировать документы, отправлять письма и др.

Программы выполняют функцию обработки информации. Каждая программа предназначена для решения определенной задачи. При этом она тесно взаимодействует с аппаратными устройствами компьютера: центральным процессором, устройствами ввода—вывода, запоминающим устройством и др.

Программа, так же как и данные, хранится в памяти компьютера в двоичном коде.

В составе программного обеспечения компьютера можно выделить четыре уровня: *базовое*, *системное*, *инструментальное* и *прикладное* ПО (рис. 2.8).

Базовое программное обеспечение образует самый низкий уровень программного обеспечения. Его назначение — обеспечение взаимодействия с базовыми аппаратными средствами, которые хранятся в специальных микросхемах (ПЗУ). В ПЗУ хранятся некоторые константы (например, π), программы для вычисления стандартных функций, программа тестирования компьютера при его включении.

Программы системного уровня выполняют посреднические функции между аппаратурой компьютера, базовым ПО и другими программами. В состав системного ПО входят операционные системы; средства обеспечения пользовательского

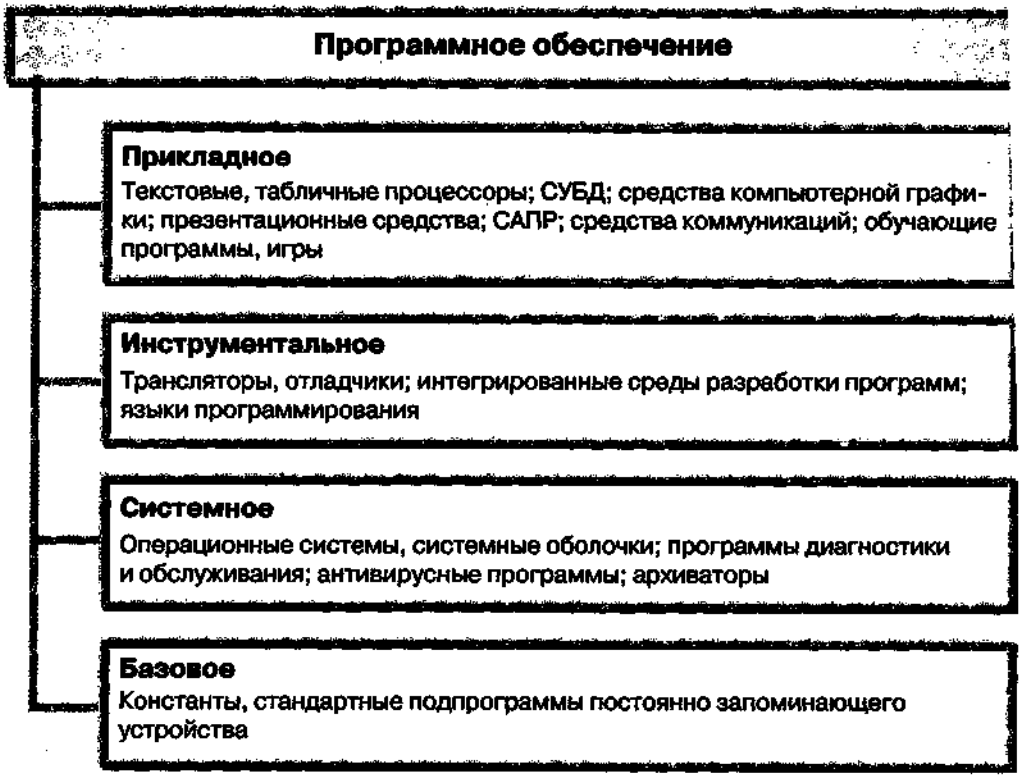


Рис. 2.8

интерфейса; утилиты (от латинского *utilitas — польза*) — средства проверки, наладки и настройки компьютерной системы; драйверы — программы, расширяющие возможности операционной системы по управлению устройствами ввода—вывода, оперативной памятью, позволяющие подключать к компьютеру новые устройства; файловые менеджеры (диспетчеры файлов); архиваторы; программы обеспечения компьютерной безопасности (антивирусное программное обеспечение) и др.

Инструментальное ПО предназначено для разработки другого программного обеспечения (системного и прикладного).

Примеры специализированных инструментальных сред:

- инструментальная среда Cyber Book (система создания гипертекстовых мультимедийных книг) позволяет визуализировать гипертекст, содержащий ссылки на текст, мультимедийные объекты и программы;
- интегрированные среды разработки приложений Borland Pascal 7.0, Visual Basic, Delphi, C++.

Прикладной уровень образуют программы для решения конкретных задач. Сюда относятся текстовые и графические редакторы, табличные и текстовые процессоры, издательские системы, системы управления базами данных, системы автоматизированного проектирования, экспертные и бухгалтерские системы, Web-редакторы, браузеры, системы видеомонтажа, геоинформационные системы и др.

Данная классификация программного обеспечения построена на основе назначения программ каждого уровня. Известна и другая классификация, основанная на длительности нахождения программ в оперативной памяти. В соответствии с ней программы делят на *резидентные* и *нерезидентные*. Резидентные программы находятся в памяти постоянно (например, операционные системы, антивирусные программы), нерезидентные — только в процессе выполнения (например, прикладные программы).

Программное обеспечение классифицируется также по стоимости. Различают программы, распространяемые платно, бесплатно, условно-бесплатно.

Большинство программ распространяется на коммерческой основе. При этом набор дискет или компакт-дисков с записанной программой называется *дистрибутивом*.

Некоторые программы распространяются бесплатно (*freeware*), например через глобальную компьютерную сеть Internet, электронные доски объявлений (BBS) и т. д.

Условно-бесплатные программы (*shareware*) предоставляются, как правило, для использования в определенный период (например, 30 дней) на бесплатной основе. Для систематического использования предлагается заплатить определенную сумму, затем пользователь получает регистрационный ключ, позволяющий использовать программу в полном объеме.

Производители некоторых программ делают их защищенными от копирования или предоставляют право их копирования на ограниченное количество компьютеров.

В нарушение прав производителей программных средств создаются пиратские копии программ, которые обычно распространяются на компакт-дисках без сопроводительной документации. Качество работы таких программ, как правило, низкое. Распространение и использование пиратских копий программ является нарушением закона.

?

1. Какие уровни программного обеспечения Вам известны?
2. К какому уровню программных средств относятся программы, обеспечивающие работу сканера? Принтера?

*§ 6. Операционные системы

Операционная система (ОС) — комплекс программ системного уровня предназначенный для функционирования всех устройств компьютера и поддержки работы его программ. ОС выполняет две основные функции: обеспечение пользователю-программисту удобств при работе с компьютером и повышение эффективности использования компьютера путем рационального управления его ресурсами.

В настоящее время наиболее распространенными являются следующие ОС Windows (различные версии, разработчик — фирма Microsoft, США) Mac OS (разные версии для персональных компьютеров Macintosh, разработчик — фирма Apple, США), Linux (разработчик — Линукс Торвалдс, Финляндия).

Linux — современная Unix-подобная многопользовательская сетевая операционная 32-разрядная система с сетевой оконной графической системой X Window System для персональных компьютеров и рабочих станций. ОС Linux поддерживает протоколы сети Internet, работает с сетями на базе Novell и MS Windows. Все компоненты системы распространяются бесплатно с правом установки для неограниченного числа пользователей. ОС Linux широко распространена на платформах Intel PC, DEC AXP, Power Macintosh и др. Основная часть системы может работать на 8 Мбайт памяти. С утилитами Linux занимает 10—20 Мбайт на жестком диске.

ОС Windows — современная удобная операционная система для старших моделей персональных компьютеров IBM PC. Эта система может использоваться на компьютерах с оперативной памятью более 2 Мбайт и памятью на жестких дисках не менее 80 Мбайт. Перенос информации с одного компьютера на другой возможен при наличии в них совместимых ОС. Компьютеры серии IBM PC и компьютеры Macintosh несовместимы на уровне программного обеспечения.

Основные отличительные особенности современных ОС: дружественный графический (унифицированный пользовательский) интерфейс, где в различные программы заложены одинаковые принципы работы; автоматическое подключение новых устройств и программ (ОС сама выполняет настройку нового оборудования; такой принцип работы получил название «*plug and play*» — «подключи и используй»); многозадачность (единый программный интерфейс, который позволяет выполнять работу одновременно в нескольких программах, окнах, переносить информацию из одной программы в другую; многозадачный режим работы требует от ОС умения разделять память между программами и данными); режим полного соответствия (формирование такого же изображения, как на экране монитора, принцип WYSIWYG — «*What You See Is What You Get*»). Последний режим работы является отличительным признаком современных сред визуального программирования (Delphi, Visual Basic и др.).

ОС хранится на диске (обычно, C:\), который называют системным, а также на системной дискете или компакт-диске. Основные функции ОС связаны с обеспечением различных интерфейсов: пользователя (между пользователем и аппаратно-программными средствами), аппаратно-программного (между аппаратным и программным обеспечением) и программного (между разными видами ПО). В процессе работы ОС осуществляет диалог с пользователем, обеспечивает реакцию компьютера на ошибки и аварийные ситуации, обслуживает работу всей компьютерной системы.

В составе ОС можно выделить программы (модули), предназначенные для: организации и обслуживания файловой системы, работы с устройствами компьютера (драйверы устройств), выполнения вводимых пользователем команд (командный процессор), обеспечения графического интерфейса пользователя (с помощью пиктограмм), обслуживания дисков, работы в сети (сервисные программы, утилиты), получения справочной информации об ОС (справочная система).

Часть ОС компьютера, постоянно находящаяся в оперативной памяти и управляющая всей ОС, образует ядро ОС, без которого невозможна организация работы на компьютере.

Автоматическая загрузка ОС происходит при включении компьютера. Этапы загрузки ОС:

1. При включении компьютера обеспечивается запуск программы BIOS (*Basic Input/Output System* — базовая система ввода—вывода) и тестирование всех его устройств. Программа BIOS хранится в ПЗУ. Результаты работы программы BIOS можно увидеть, если приостановить загрузку ОС, нажав клавишу Pause/Break.

2. Загрузка ОС с системного диска в оперативную память: с помощью программы *Master Boot* (загрузчик ОС), которая находится в специальном загрузочном секторе системного диска. Если системный диск отсутствует, на экран выдается сообщение *Non system disk*, загрузка ОС прекращается, работа пользователя невозможна.

3. Передача управления командному процессору, отображение графического интерфейса. Компьютер готов к работе. Дальнейшая работа пользователя осуществляется под управлением ОС. Для решения конкретной задачи на компьютере требуется выбрать подходящую программу и приступить к работе в ней. Например, для копирования файлов используются средства ОС, предназначенные для работы с файлами (**Проводник**, файловые менеджеры); для редактирования текстового документа — прикладные программы (текстовый редактор).

?

1. **Что такое операционная система?**
2. **Какие основные функции ОС?**
3. **Какие этапы загрузки ОС Вы знаете?**

ГЛАВА 3

ОСНОВЫ ПРОГРАММИРОВАНИЯ

§ 7. Языки программирования

Решение задачи с помощью компьютера можно разбить на ряд этапов, носящих общий характер (хотя возможна некоторая их корректировка):

- построение математической модели изучаемого объекта и осуществление математической постановки задачи;
- выбор метода решения задачи;
- разработка алгоритма решения задачи;
- составление программы решения задачи и ее реализация на компьютере;
- тестирование и отладка программы;
- непосредственное решение задачи на компьютере.

Для описания алгоритмов решения задач широко используются языки программирования, специально созданные для записи программ.

Язык программирования строится на совокупности трех составляющих: алфавита, синтаксиса (жестких правил написания объектов языка) и семантики (правил их использования).

Паскаль — один из популярных языков программирования. Названный в честь французского математика и философа Блеза Паскаля (1623—1662), он был создан как учебный язык программирования в 1971 г. Никлаусом Виртом в Высшей технической школе в Цюрихе. Среди достоинств языка выделяются: простота освоения (незначительный объем базовых понятий; несложный синтаксис); распространенность (выполнение программы практически на всех современных ПК); возможность развития (разработка на его основе современной среды визуального программирования Delphi, азы программирования в которой будут постигаться в 12-м классе на повышенном уровне обучения).

Средствами языка Паскаль можно составлять программы для автоматического решения задач различной тематики и направленности: построение графиков функций, выполнение расчетов, организация экспериментов, ведение диалога, обучение и контроль знаний, программная реализация трансляторов, компьютерная графика и др.

Процесс создания программы в общем случае предполагает:

Упражнения

1. Приостановите загрузку ОС, изучите информацию, которая выводится на экран, и заполните следующую таблицу:

Тип процессора	
Тактовая частота процессора	
Количество жестких дисков	
Объем жестких дисков	
Наличие дисководов гибких дисков, их параметры	
Объем ОЗУ	
Тип монитора	

2. Представьте в виде таблицы способы запуска программы Проводник, указав методы запуска и элементы уравнения:

Метод запуска	**Элемент управления**
1. Через контекстное меню кнопки **Пуск**	Кнопка **Пуск**

- ввод текста программы в компьютер, его редактирование и длительное хранение в виде отдельного файла;
- перевод исходного текста на машинный язык с помощью специальной программы (такие программы называют трансляторами);
- использование стандартных библиотек, реализующих различные стандартные подпрограммы;
- обращение к системе подсказок и помощи.

Для создания программы используются интегрированные среды программирования (ИСП), которые включают в себя сам язык программирования и средства разработки программ. Такие среды позволяют повысить эффективность программирования.

В состав современной ИСП входят текстовый редактор, транслятор, редактор связей, библиотеки подпрограмм, система помощи и подсказок. Некоторые ИСП содержат еще один компонент — отладчик (специальную программу, которая позволяет в пошаговом режиме анализировать работу созданной программы, что облегчает процесс отладки крупных программ).

Популярные среды программирования: Turbo Basic, Turbo Pascal, Turbo C, Borland Pascal, Borland Delphi, Microsoft Visual Basic, Borland C++.

Познакомимся с работой в интегрированной среде программирования Borland Pascal (BP), позволяющей создавать программы для ОС Windows. В среде BP процесс разработки программы содержит следующие этапы:

- создание программного файла с расширением .pas;
- обработка программы (компиляция и редактирование связей), приводящая к формированию выполняемого exe-файла;
- выполнение программы (обработка данных).

7.1. Главное окно интегрированной среды программирования Borland Pascal

Для вызова интегрированной среды Borland Pascal (BP) необходимо запустить на выполнение файл bp.exe (он находится в подпапке BIN папки BP).

На рисунке 3.1 показан вид главного окна после загрузки программы.

Верхняя строка экрана — строка заголовка, ниже — строка основного меню и окно редактора. На основном экране могут появляться другие окна, например окно помощи, компиляции, результатов работы программы и т. д.

Рис. 3.1

С помощью мыши или клавиши F10 (и курсорных клавиш) можно открыть тот или иной пункт меню.

Пункт меню	Назначение
File (Alt+F)	Выполнение операций с файлами текстов программ
Edit (Alt+E)	Выполнение операций редактирования текста программы
Search (Alt+S)	Поиск фрагментов текста
Run (Alt+R)	Запуск на выполнение программы, находящейся в окне редактора
Compile (Alt+C)	Выполнение команд, связанных с компиляцией и компоновкой программы
Debug (Alt+D)	Выполнение команд, связанных с отладкой программы
Options (Alt+O)	Выполнение команд, связанных с управлением параметрами компиляции и среды программирования
Window (Alt+W)	Выполнение команд управления окнами
Help (Alt+H)	Выполнение команд получения справочной информации об языковых средствах языка Паскаль и возможностях ИСП

Основные команды, содержащиеся в меню, приведены в Приложении 1.

Окно редактора предназначено для создания и редактирования текстов программ. Одновременно могут быть открыты несколько окон (файлов). Для перехода от одного окна к другому используют клавишу F6.

Нижняя строка главного окна — информационная; здесь размещаются подсказки по назначению функциональных клавиш, которые упрощают работу в интегрированной среде.

Последовательность действий в среде программирования BP при работе с новой или набранной ранее программой может быть такой:

- запуск интегрированной среды программирования;
- закрытие окна программы (Window — Close, или Alt+F3);
- установка текущего каталога (File — Change Dir);
- создание окна для ввода текста новой программы (File — New). Вновь создаваемый файл всегда имеет имя NONAME.PAS (оно отображается на верхней рамке окна редактора);
- присвоение имени файлу, в котором будет храниться текст программы (File — Save As);
- открытие (загрузка в окно редактора) программного файла (File — Load, или F3);
- сохранение текста программы с прежним именем (File — Save, или F2);
- компиляция и запуск программы на выполнение (Run — Run, или Ctrl+F9);
- завершение работы программы при ее «зацикливании» (Ctrl+Break);
- просмотр окна вывода результатов выполнения программы (Debug — User screen, или Alt+F5);
- завершение работы с ИСП (File — Exit, или Alt+X).

7.2. Редактирование текста программы

Выполнению программы предшествует подготовительная работа компьютера, направленная на выявление синтаксических и логических ошибок.

Если при наборе текста программы допущены синтаксические ошибки, их можно устранить путем редактирования отдельных символов или части строки. Для этого следует поместить курсор в нужную позицию конкретной строки и внести исправления.

При наборе и редактировании текста программы можно использовать такие возможности текстового редактора, как копирование, перенос и удаление выделенных фрагментов (так же, как это делается в текстовом редакторе).

Копировать фрагменты текста можно из одного окна (файла) в другое.

7.3. Обращение к справочной системе Help

Установим курсор на интересующее нас зарезервированное слово (например, в программе PRIVET выберем слово Writeln) и с помощью комбинации клавиш Ctrl+F1 вызовем справку по выбранному слову.

```
Program PRIVET;
Begin
   Writeln('Здравствуйте!');
   Writeln('Начинаем изучать язык');
   Writeln('программирования Паскаль');
End.
```

При нажатии клавиши F1 можно вызвать справочную систему Help среды программирования и получить нужную информацию о режимах работы в ней (например, о правилах редактирования программы). Если нажать клавишу F1 после появления сообщения об ошибке, то будет получена расшифровка сообщения с указанием возможных причин возникновения ошибки (правда, на английском языке).

Справочная информация представляет собой текстовый фрагмент и может быть скопирована и помещена в окно редактора в место, определенное расположением курсора. Таким образом, можно заимствовать из справочной системы примеры программ и обучаться по ним самостоятельно.

7.4. Компиляция и выполнение программы

Команда Compile из пункта меню Compile предназначена для преобразования программы из файла типа .pas, отображаемого в активном окне редактора, в EXE-файл с выполняемой программой (файл с расширением EXE).

Команда Run (меню Run) позволяет осуществить одновременно несколько операций: компиляцию, компоновку (редактирование связей) и выполнение программы. Если программа уже была откомпилирована, команда Run осуществляет лишь запуск программы на выполнение.

7.5. Отладка программы

При написании программы могут быть допущены ошибки, которые относятся к одному из типов:

1. **Ошибки компиляции** — они связаны с нарушением синтаксиса языка Паскаль (неправильное использование разделителей, неверное описание идентификаторов, использование неописанных переменных, неверный тип данных и т. д.). При наличии ошибки такого рода во время трансляции программы происходит позиционирование курсора на месте ошибки и на экране появляется сообщение об ошибке:

Error номер (код) ошибки: Смысл (характер) ошибки.

Например, сообщение **Error 2: Identifier expected.** означает, что на месте позиционирования курсора должен находиться идентификатор («требуется идентификатор»).

2. **Ошибки выполнения** — появляются при нарушении семантических правил языка Паскаль (попытка деления на нуль, извлечения квадратного корня из отрицательного числа и др.). При наличии ошибки такого рода программа завершает свою работу и выводит сообщение об ошибке. Например, сообщение **`Error 200: Division by zero.`** означает, что предпринята попытка деления на нуль.

Сообщения об ошибках во время компиляции и выполнения программы приведены в Приложении 2.

3. **Логические ошибки** — они связаны с неверной постановкой задачи или ошибкой в алгоритме на уровне использования алгоритмических конструкций. Эти ошибки наиболее сложно обнаружить, однако они могут быть выявлены на этапе отладки программы.

Процесс поиска и исправления ошибок программы называется **отладкой**, процесс проверки правильности ее функционирования во всем диапазоне допустимых значений исходных данных — **тестированием** программы.

Итак, программа разработана, введена с помощью текстового редактора среды программирования в компьютер, сохранена на диске, откомпилирована и выполнен ее запуск. Вы ввели некоторые данные и получили результат. Однако не спешите кричать «Эврика!». Это можно сделать, лишь когда Вы уверены, что результат правильный. Для этого нужно протестировать программу, придавая исходным данным значения, близкие к реальным, и сравнивая полученные результаты с ожидаемыми (полученными при устных вычислениях, на электронном калькуляторе или из надежных источников). Каждый тест представляет собой совокупность исходных данных, для которых известен результат.

Познакомимся с технологией отладки программ на примере программы VITEBSK:

```
Program VITEBSK;
Begin
  Writeln('Витебский чугунолитейно-машиностроительный за-
вод');
  Writeln('существовал в г. Витебске с 1877 г. по 1940 г.');
  Writeln('  Затем на базе его был создан завод Вистан.');
  Writeln('Предшественник Вистана работал ',1940-1877,'
года');
  Readln;
End.
```

Предположим, что при ее наборе допущены ошибки:

```
Progra VITEBSK  {опущены буква m, символ ; и Begin}
Write('Витебский чугунолитейно-машиностроительный завод);
                {Опущено "ln" и нет символа '}
Witeln('существовал в г. Витебске с 1877 г. по 1940 г.')
                {Ошибка в операторе Writeln и нет ;}
Writeln('  Затем на базе его был создан завод Вистан.');
Writeln('Предшественник Вистана работал, 1840-1877, года');
                {Плохой список вывода}
Redln;          {Пропущена буква a}
End             {Отсутствует точка}
```

При запуске программы на выполнение будут последовательно появляться сообщения об ошибках.

Обдумаем причины возникновения ошибок, внесем в текст программы соответствующие правки и будем запускать программу до тех пор, пока не получим ожидаемый результат.

Ошибка		Действия пользователя
`Error 36: BEGIN expected.`	Требуется Begin	Написать Begin перед первым оператором `Write`
`Error 36: BEGIN expected.`	Требуется Begin	Правильно написать ключевое слово `Program`
`Error 85: ";" expected.`	Требуется ";"	Дописать «;» в конец заголовка программы
`Error 8: String constant exceeds line.`	Строковая константа превышает размеры строки	Закрыть список вывода символом «апостроф» (')
`Error 3: Unknown identifier.`	Неизвестный идентификатор	Исправить служебное слово `Writeln` под курсором
`Error 85: ";" expected.`	Требуется ";"	Дописать «;» в конец строки выше курсора
`Error 3: Unknown identifier.`	Неизвестный идентификатор	Исправить служебное слово `Readln` под курсором
`Error 10: Unexpected end of file.`	Неожиданный конец файла	Поставить точку в конце программы

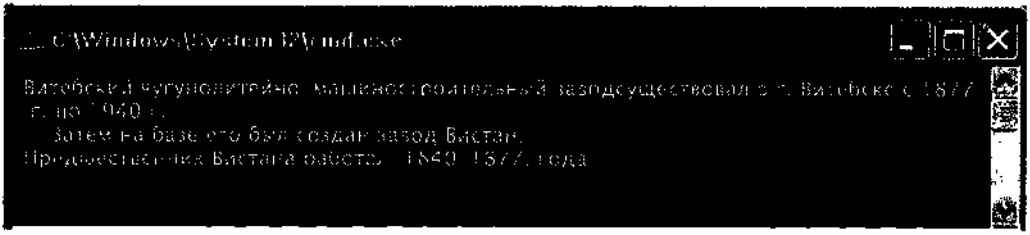


Рис. 3.2

После очередного запуска программы получим результат, представленный на рисунке 3.2.

Сравнивая результаты работы программы с ожидаемыми, замечаем, что после вывода текста «Витебский чугунолитейно-машиностроительный завод» курсор остается в этой же строке и не переходит на новую строку. Изменим оператор `Write('Витебский ...');` на `Writeln('Витебский ...');`. Вновь запустим программу на выполнение и проследим за дальнейшим выводом: остается неясным, сколько же лет работал предшественник Вистана (результат вычитания чисел не выводится на экран). Причиной этого является неверное оформление списка вывода в последнем операторе вывода: вычисляемое выражение не должно содержаться в апострофах. Внесем исправления в программу и вновь запустим ее на выполнение; результат работы программы: `Предшественник Вистана работал -37 года` (отрицательное число). Проверим, как записаны уменьшаемое и вычитаемое, и находим ошибку в записи числа 1940. Исправляем оператор вывода. Повторный запуск программы выдает уже правильные результаты (рис. 3.3). Эврика!

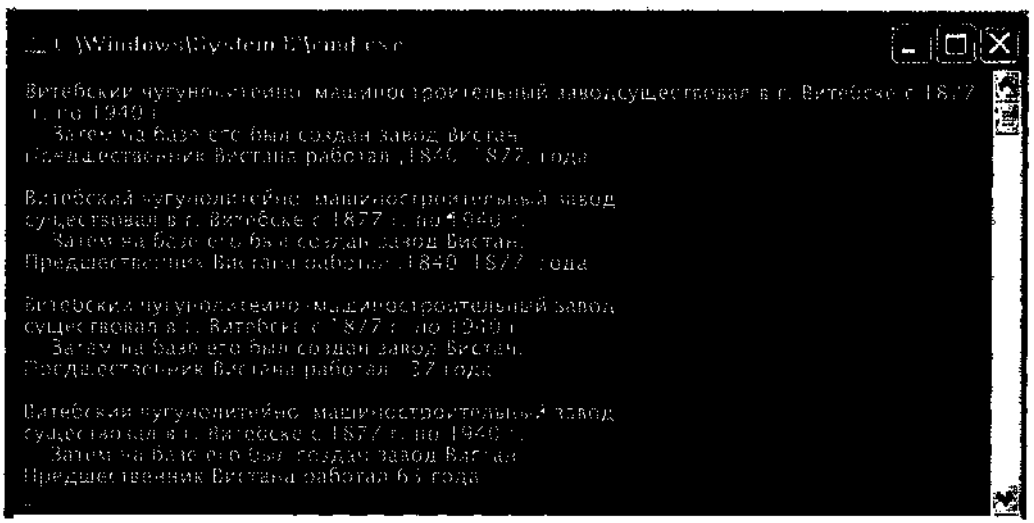


Рис. 3.3

1. Что такое язык программирования? Среда программирования?
2. Почему среду программирования Borland Pascal называют интегрированной? Из каких компонентов она образуется?
3. Какое расширение имеет файл в языке Паскаль?
4. Каковы функции текстового редактора среды программирования?
5. Как запустить программу на выполнение?
6. Что такое отладка программы?

§ 8. Основные объекты языка программирования Паскаль

8.1. Алфавит языка

В основе языка программирования Паскаль лежит определенная совокупность символов, образующих алфавит языка:

буквы латинские (строчные и прописные);

цифры (арабские от 0 до 9);

символы операций (арифметических + - * / и логических = > <);

разделители и прочие символы (. , ; ( ) { } [ ] ' _: пробел и др.).

Из символов алфавита формируются:

слова (константы, переменные) — последовательность символов, образованная по определенным правилам;

выражения — группы слов, объединенные с помощью символов алфавита и имеющие определенный смысл;

операторы — предложения для описания некоторых операций в процессе обработки данных;

программы — записи алгоритмов на языке программирования.

8.2. Величины

Конкретная величина в информатике определяется именем и совокупностью ее допустимых значений. В алгоритмах и программах к величинам относятся **константы, переменные, обращения к функциям, выражения**. Значения, которые может принимать величина, принадлежат к некоторому вполне определенному множеству (целые числа, дробные числа, логические константы или др.). Множество значений величины определяет ее тип. При представлении в памяти компьютера всякая величина имеет вполне определенное (текущее) значение.

Идентификаторы — зарезервированные (служебные) слова, имена пользователя (имена констант, переменных, программ, процедур, функций).

Зарезервированные слова составляют основу языка программирования Паскаль и имеют строго определенное значение. Сюда относятся, например, идентификаторы: `program`, `const`, `var`, `if`, `then`, `else`, `begin`, `end`, `and`, `case`, `not`, `of`, `while`, `uses`, `for`, `do`, `downto`, `div`, `mod`, `or`, `string`, `array`, `procedure`, `function` и др. Эти служебные слова нельзя использовать в качестве имен переменных, которые задает пользователь (программист).

Стандартные идентификаторы (`True`, `False`, `Integer`, `Real`, `Byte`, `Boolean`, `Char`, `Sin`, `Cos`, `Round`, `Length`, `Read`, `Write`, `Delete` и др.), смысл и способ использования которых определены правилами языка программирования, можно, но не рекомендуется использовать в иных целях.

Идентификатор пользователя записывается в виде последовательности латинских букв, цифр и символа подчеркивания, начиная с буквы или символа подчеркивания. Примеры идентификаторов: `ALFA`, `as_`, `mmm`, `PriMeR`, `Y70`, `_5`. Записи `2_6`, `X+Y`, `МИNOGA` не являются идентификаторами, так как первая начинается с цифры, а остальные содержат недопустимые символы («+» и русскую букву «И»).

В записи идентификаторов прописные и строчные буквы не различаются, например `opera`, `Opera`, `OPERA`, `OPeRa` обозначают один и тот же идентификатор.

При выборе идентификатора обычно стремятся отразить в нем его назначение. Это облегчает понимание программы и является признаком хорошего стиля программирования. Так, для обозначения скорости самолета удобно использовать, например, идентификаторы `V`, `V0`, `Speed`, `Flugzeug`, `avion`; для обозначения номера дня недели — `d`, `day`, `Tag`, `jour`; месяца — `m`, `month`, `Monat`, `mois`; года — `g`, `year`, `Jahr`, `annee` или другие.

Если в математике выражение ab *понимают как произведение переменных* a *и* b, *где опущен знак умножения, то в программировании* `ab` — *идентификатор (имя некоторой переменной).*

Упражнения

1. Подберите подходящие идентификаторы для обозначения:
   а) имени, фамилии, возраста ученика;
   б) стоимости линейки, тетради и карандаша;
   в) объема и длительности звучания музыкальной композиции;
   г) углов четырехугольника;
   д) размера начисленной заработной платы;
   *е) радиуса, длины окружности, площади круга;
   *ж) синуса и косинуса угла γ.

2. Укажите слова, которые не могут являться идентификаторами пользователя (объясните почему):

а) ABCdEFgH	б) Intal	в) ome5ga	г) 4TOM
д) _5(as)	е) Lπ	ж) Read Me	з) Begin
и) КЛАСС	к) X_1	л) ЯNDEX	м) a-1

*3. Предположим, что идентификатор должен содержать ровно 3 символа. Сколько разных идентификаторов можно получить, если использовать:

а) две буквы X и Y б) буквы X, Y и цифру 5
в) букву X и цифры 1 и 2

8.2.1. Константы

К стандартным типам в языке Паскаль относятся величины: *числа, логические величины, строки.*

Познакомимся с числовыми величинами.

Константы — это величины, которые определены при написании программы и не изменяют своего значения в процессе ее выполнения.

Константы бывают **неименованные** и **именованные**. Именованная константа представляет собой идентификатор, который сопоставлен с конкретным значением, например g=9.8, x=-2.3, k=25. Тип константы может быть определен по ее внешнему виду, например 25; 423; 0; -26; 1000; +17 — целые константы; 9.8; -2.3; 1.5; -15.28; 0.0; -0.6E05; .25; 4E2; 2.5E+4; -21E-7 — вещественные константы.

Для записи целых констант используют цифры, которым может предшествовать знак «+» или «-». В вещественных константах (они соответствуют действительным числам в математике) присутствует точка, которая разделяет целую и дробную части числа, или буква E. Использование E приводит к представлению числа в показательной форме записи: запись mEp соответствует числу $m \cdot 10^p$; например,

$4E2 \rightarrow 4 \cdot 10^2 = 400$ $\quad -2.35E+4 \rightarrow -2{,}35 \cdot 10^4 = -23500$

$1E6 \rightarrow 1 \cdot 10^6 = 1000000$ $\quad -26.3E-3 \rightarrow -26{,}3 \cdot 10^{-3} = -0.0263$

Упражнения

1. По внешнему виду константы определите ее тип:

а) 32320	б) -4.0	в) 17	г) 25E3
д) +724003	е) -15.723E-3	ж) -324E+35	з) 45.263
и) +5	к) +7777.3E-01	л) -17E-10	

*2. Какие записи являются числовыми константами в языке Паскаль?

а) +1	б) 56,5	в) 1.234,56	г) 45.56-2
д) 56E+9	е) 45A3	ж) 0.(3)	з) .5
и) 4+2	к) -11111	л) 34*2	м) X=2
н) -E3	о) E15	п) 1/12	р) +7.7
с) 0.333...	т) VIII	у) sqrt(2)	ф) pi

8.2.2. Переменные

Переменные — это величины, которым присваиваются значения в процессе выполнения программы. Они могут изменять свои значения.

Переменная задается тремя составляющими: именем, значением, типом. Тип переменной выбирается на основании требуемого диапазона и точности представления данных.

Любая константа, переменная, значение функции и выражение в языке Паскаль имеют определенный тип. Тип определяет следующие характеристики объекта: множество допустимых значений, допустимый набор операций, формат внутреннего представления и объем занимаемой памяти. Принадлежность того или иного объекта к определенному типу осуществляется с помощью специальных команд описания (декларации, объявления). Каждому типу соответствует уникальное ключевое слово:

Тип	Ключевое слово	Объем памяти (байт)	Диапазон значений
Целый	Byte	1	0 .. 255
	Word	2	0 .. 65535
	Integer	2	-32768 .. 32767
	LongInt	4	-2147483648 .. 2147483647
Вещественный	Real	6	$2{,}9 \cdot 10^{-39}$..$1{,}7 \cdot 10^{38}$

Каждая переменная программы должна быть непременно объявлена до момента своего использования в программе!

8.3. Арифметические операции

Вы уже знаете, что тип величины определяет допустимый для нее набор операций. Числовым величинам присущи следующие операции:

+ сложение	- вычитание
* умножение	/ деление
Div целочисленное деление	Mod остаток от деления

Операции умножения и деления имеют более высокий приоритет, чем сложение и вычитание, и выполняются раньше.

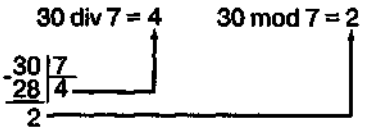
Результат выполнения операции зависит как от самой операции, так и от типа используемых в ней величин (операндов):

Операция	Действие	Типы операндов	Тип результата
+	Сложение	Целый или вещественный	Целый или вещественный
-	Вычитание	Целый или вещественный	Целый или вещественный
*	Умножение	Целый или вещественный	Целый или вещественный
/	Деление	Целый или вещественный	Вещественный
Div	Целочисленное деление	Целый	Целый
Mod	Остаток от целочисленного деления	Целый	Целый

Для операций +, -, * тип результата может оказаться целым (если оба операнда целочисленные) или вещественным (если хотя бы один операнд вещественный).

Операции div и mod используются как операции целочисленного деления для получения соответственно целой части и остатка от деления. Так, результатом выполнения операции a div b будет [a/b], a mod b — остаток от деления a на b.

Пример 1. Найти результат выполнения операции целочисленного деления и остаток от деления числа a на число b при $a = 30$, $b = 7$.



Пример 2. Определить тип результата выполнения арифметических операций: $4{,}5 + 1{,}5 = 6{,}0$; $\frac{5}{2} = 2{,}5$; $17 - 14 = 3$; $5 \cdot 2 = 10$; $7 + 12 = 19$; $3{,}0 - 7 = -4{,}0$.

Для выполнения задания используем таблицу, предложенную в данном пункте.

`4.5+1.5=6.0;` `3.0-7=-4.0;` `5/2=2.5` (вещественный тип).
`5*2=10;` `17-14=3;` `7+12=19` (целый тип результата).

8.4. Стандартные подпрограммы

В состав библиотеки, которая поставляется вместе со средой программирования, входят стандартные (встроенные) подпрограммы (процедуры и функции). Из набора стандартных подпрограмм для обработки информации одного назначения составляются модули. Каждый модуль имеет свое имя (`System`, `Crt`, `Graph`, `Printer` и др.). Доступ к процедурам и функциям модуля осуществляется при его подключении (см. п. 12.1).

Подпрограммы модуля `System` используются по умолчанию, и поэтому для их применения не требуется дополнительное подключение. Использование подпрограмм модуля позволяет избежать ненужного программирования многих стандартных функций.

Описание и назначение некоторых числовых функций, которыми мы будем пользоваться при программировании, приводятся в таблице:

Подпрограмма (функция)	Назначение	Тип аргумента x (формального параметра)	Тип результата
`Abs(x)`	$\|x\|$	Целый, вещественный	Целый, вещественный
`Arctan(x)`	arctg x	Целый, вещественный	Вещественный
`Sin(x)`	$\sin x$	Целый, вещественный	Вещественный

Продолжение

Подпрограмма (функция)	Назначение	Тип аргумента x (формального параметра)	Тип результата
`Cos(x)`	$\cos x$	Целый, вещественный	Вещественный
`Exp(x)`	e^x	Целый, вещественный	Вещественный
`Ln(x)`	$\ln x$, $x > 0$	Целый, вещественный	Вещественный
`Sqr(x)`	x^2	Целый, вещественный	Целый, вещественный
`Sqrt(x)`	$\sqrt{x}, x \geqslant 0$	Целый, вещественный	Вещественный
`Int(x)`	$[x]$ целая часть числа	Целый, вещественный	Вещественный
`Frac(x)`	$\{x\}$ дробная часть числа	Целый, вещественный	Вещественный
`Trunc(x)`	$[x]$ целая часть числа	Целый, вещественный	Целый
`Round(x)`	Округление до ближайшего целого	Целый, вещественный	Целый

Для обращения к конкретной стандартной функции следует записать в выражении ее имя и конкретный фактический параметр, который должен соответствовать требованиям этой подпрограммы. Так, для вычисления $\sin\alpha$ можно написать `sin(alfa)`, для вычисления $\sqrt{5}$ — `sqrt(5)`.

Примеры выполнения функций `Trunc` и `Round`:

```
trunc(pi)=3,   trunc(-3.9)=-3,   round(pi)=3,   round(-3.9)=-4.
```

1. Во время работы с редактором текста в интегрированной среде программирования можно ознакомиться с требованиями к параметрам функций, если установить курсор клавиатуры на имени функции и нажать комбинацию клавиш CTRL+F1.
2. В тригонометрических функциях аргумент задается в радианах.

1. Что такое константа? Переменная? Идентификатор?
2. Как задается имя переменной?
3. Как записываются целые константы? Вещественные константы?
5. Какие арифметические операции определены в языке Паскаль?
6. Какие стандартные числовые функции имеются в языке Паскаль?

Упражнения

1. Определите результат выполнения операции и его тип:

а) `13 div 2`	б) `13 mod 2`	в) `18 div 23`	г) `18 mod 23`
д) `2*3.1`	е) `18/2`	ж) `5+7`	з) `15.3-7.3`

2. Определите результаты выполнения функции при $x = 2$:

а) `sin(x-2)`	б) `sqrt(2*x)`	в) `sqr(1-x)`
г) `trunc(x)`	д) `cos(x/2-1)`	е) `abs(x-10)`
ж) `int(x/1.5)`	з) `round(x+1.2)`	и) `abs(x-4)`
к) `arctan(1-x)`	л) `exp(x-2)`	м) `int(x-0.3)`
н) `frac(x/3)`	о) `trunc(x+10.9)`	п) `trunc(x-10.1)`

3.5. Арифметические выражения

Арифметические выражения представляют собой аналог алгебраических выражений в математике. Они состоят из операндов (переменных, констант, функций), символов операций и круглых скобок (если это требуется). Операции выполняются над операндами.

1. Символы операций ни в коем случае нельзя опускать, например произведение ab в языке Паскаль записывается в виде `a*b`.
2. В языке Паскаль отсутствует операция возведения в степень. Для вычисления x^n используют определение степени, умножая x на себя n раз, или свойства логарифмической и экспоненциальной функций, записывая x^n в виде арифметического выражения: EXP(N*LN(X)), что соответствует записи $e^{n \ln x}$.

Примеры арифметических выражений:

а) `103.8-X`	б) `4/7`	в) `a+5.6`
г) `5*(X+Y)`	д) `x/sin(x)`	е) `sqrt(a*b)`

При вычислении выражений операции выполняются в таком порядке:

1) вычисление значений стандартных функций; 2) умножение и деление; 3) сложение и вычитание. Например, (для каждого выражения указывается порядок выполнения операций):

①	①②④③⑤	② ① ⑤③④	③② ①
`x+y`	`a*x*x+b*x+c`	`2*(5-x)+3/7*k`	`1+4/(a*b)`

Операции в выражении при одинаковом приоритете выполняются в порядке записи слева направо. Круглые скобки позволяют изменить порядок вычисления выражения. Сравним с примером выражения без скобок:

①④⑤②③

```
2*5-x+3/7*k
```

В состав арифметических выражений могут входить также стандартные функции языка Паскаль. Их аргументами могут быть другие выражения. Например, `sqrt(4*x+2.5); sin((x+alpha)/pi)+sqrt(x*x+1.5); a*cos(pi-alfa/4); sin(sin(x))`.

Тип значения арифметического выражения зависит от типа используемых в нем величин, операций, функций.

Пример. Определить тип арифметических выражений `A+X`, `B+(C+D)/P`, `A*B`, `X-P`, `A/X` согласно описанию:

```
Const X=10;
      P=3.5;
Var A: Integer;
    B,C,D: Real;
```

Результат: тип выражения `A+X` — целый (так как A и X целые), а тип выражений `B+(C+D)/P`, `A*B`, `X-P`, `A/X` — вещественный.

Упражнения

1. Запишите математические выражения средствами языка Паскаль:

а) $2x^3$ б) $\frac{a+b}{2a+b}(a+c)$ в) $\sin\alpha\cos\beta-\cos\alpha\sin\beta$

г) πR^2 д) $\frac{1}{2}ab\sin\alpha$ е) $\sin x+\cos x-\frac{x}{x+1}$

ж) $\sqrt{b^2-4ac}$ з) $\frac{2}{abc}$ и) $\sqrt{x^2+y^2}-1{,}5(x-3)$

к) $\left[\frac{x}{y}\right]-\frac{x}{y}$ л) $\gamma\frac{m_1m_2}{r^2}$ м) $\sqrt{a^2+b^2-2ab\cos\lambda}$

***2.** Пусть `X`, `Y` — вещественные, `K`, `P` — целочисленные переменные. Определите тип результата операции:

а) `1-K`	б) `X+2`	в) `Y-0.5`	г) `K/P`	д) `P+1/3`
е) `Y*K`	ж) `2*K`	з) `X+P`	и) `P+K`	к) `10/K`

*3. Удалите лишние скобки:

а) ((a/b)*c)*d б) a+(b*c)-3
в) (a*(b/c*(x*y)/z)) г) (((2/x)-4)+5)

*4. Запишите арифметические выражения языка Паскаль, значениями которых являются:

а) площадь квадрата, периметр которого равен P;
б) площадь круга, если длина ограничивающей его окружности равна L;
в) длина стороны равностороннего треугольника, площадь которого равна S.

§ 9. Структура программы

Программа на языке Паскаль состоит из двух разделов: *раздела описаний* и *раздела операторов*.

Раздел описаний — это декларативная (описательная) часть программы; включает имя программы, описание констант, переменных и др.

Раздел операторов — выполняемая часть программы, которая всегда начинается словом `Begin` и заканчивается словом `End`.

```
PROGRAM …;
   CONST …;
   VAR …;
BEGIN
   … – тело программы
END.
```

1. В действительности, тело программы может состоять из большего количества разделов описаний. С некоторыми из них мы познакомимся позже.
2. В программе может отсутствовать любая часть, кроме раздела операторов.

Заголовок программы: **PROGRAM название** ;
PROGRAM — ключевое слово заголовка программы;
название — идентификатор пользователя;
CONST — ключевое слово описания констант;
VAR — ключевое слово описания переменных, используемых в программе;
BEGIN — начало исполняемых операторов;
END. — конец программы (обратите внимание на синтаксис: операторы разделяются символом «;», программа заканчивается символом «.»).

Примеры описаний	Результаты описаний
константа:	
`CONST V=10;A=4.6E-4;` `Y=17.3; K=-5;`	V, K — константы целого типа; A, Y — константы вещественного типа
переменных:	
`VAR XX,V0:Real;` `K,N,Pr:Integer;`	XX, V0 — переменные вещественного типа; K, N, Pr — переменные целого типа

`CONST Time=31.5;G=1.9;P=15;`
`VAR Year:Word;`
`Money:LongInt;`
`S1,S2:Real;`
	Объявление констант Time и G со значениями 31,5 и 1,9 вещественного типа, P со значением 15 — целого типа.
	Переменные Year, Money — целого типа;
	S1, S2 — вещественного типа

Всякий объявленный в программе идентификатор должен быть уникальным. Так, например, недопустимым является объявление:

```
Var X: Real; X: Integer;
```

Итак, описание констант осуществляется в разделе описания констант CONST, а описание переменных — в разделе описания переменных VAR:

Описание констант	Описание переменных

`CONST имя1 = значение1 ;`
`имя2 = значение2 ;`
`...`
	`VAR имя1 : тип1 ;`
	`имя2 : тип2 ;`
	`...`

имя1, имя2, ... — имена констант, по которым будет осуществляться обращение к ним в программе	имя1, имя2, ... — имена переменных, по которым будет осуществляться обращение к ним в программе; тип1, тип2, ... — их типы. Несколько величин одного типа перечисляются через запятую: `Var a,b,c:Real;`

В тексте программы могут присутствовать *комментарии*, которые важны для документирования (пояснения) программы.

Комментарий — произвольный текст, ограниченный с обеих сторон фигурными скобками или круглыми скобками и символом *, например

{Это комментарий} и (* Это комментарий *).

При выполнении программы комментарии не учитываются.

Комментарий может размещаться в любом месте программы.

Пример программы с комментариями:

```
{Программа сравнения физических характеристик Солнца и
Земли}
Program Astronomy;
                          {Объявление констант}
CONST MS=2E30;            {Солнце: масса}
      PS=1.416;           {плотность}
      MZ=5.976E24;        {Земля: масса}
      PZ=5.518;           {плотность}
                          {Объявление переменных}
  VAR
     dsZ,Om,Op:Real;      {dsZ — длительность суток на Земле}
                          {Om — отношение масс,
                          Op — отношение плотностей}
  …
Begin
  …
End.
```

Операторы выполняемой части программы рекомендуется записывать в отдельных строчках с отступами, как, например, в программе VITEBSK (п. 7.5). Это способствует наглядности программы и упрощает понимание отдельных элементов оператора.

?

1. Какова общая структура программы на языке Паскаль?
2. Какие служебные (зарезервированные) слова Вы знаете?
3. Что является заголовком, символом начала и конца программы?
4. Каково назначение разделов Var, Const?

Упражнение

Определите типы описанных констант и переменных, предложите возможные значения переменных:

```
Const Year=2000; ST1=2000.0; ST2=2E3;
      ST3=0.2E4; Time=2.3; M=-15;
      ABBA=4; Bel=+100;
Var   Telephone, Money, Number: Integer;
      Price, Secret: Real; DIA: LongInt;
      Zap: Byte; Mass, Mini: Real; Mas, INTER: Word;
```

§ 10. Ввод — вывод данных

10.1. Вывод

Приведенные в § 7 программы PRIVET и VITEBSK предусматривают вывод на экран монитора некоторого текста. Для этой цели используются две стандартные процедуры: `Writeln` и `Write`.

`Writeln` `(Список вывода);` — стандартная процедура, которая используется для вывода данных на экран. Значения выводятся в порядке их записи в списке вывода (рис. 3.4). Затем курсор переходит в начало следующей строки.

`Write` `(Список вывода);` — стандартная процедура для вывода данных на экран без перевода курсора на новую строку.

Целые числа выводятся в привычной (естественной) форме, а вещественные с помощью мантиссы и порядка (в показательной форме с нормализованной мантиссой): *mEp*, где m — мантисса, p — порядок, $1 \leqslant m < 10$ (одна позиция в целой части и десять позиций в дробной части), p выводится со знаком «+» или «-» и далее в две позиции.

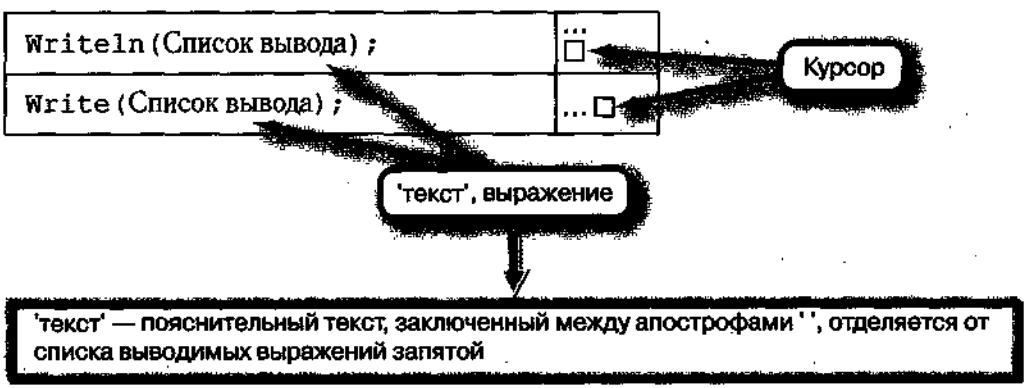


Рис. 3.4

Примеры вывода с указанием местонахождения курсора после выполнения операторов вывода даны в таблице:

Оператор	Результат выполнения
`Writeln('Ответ: ', sqrt(5));`	Ответ: 2.2360679775E+00 □
`Writeln('Введите число');`	Введите число □

Продолжение

Оператор	Результат выполнения
`Write ('X=');`	X=□
`Write(X, ',',Y);` при Y=-3, X=5 (X, Y описаны как Integer)	5,-3□
`Write(2+3);`	5□

`Writeln('Дано: a=', a);`
`Writeln('b=', b);`
`Writeln('a + b = ', a+b);`
при a=-34.6, b=126.93 (a, b описаны как Real)
	Дано: a = -3.4600000000E+01
	b = 1.2693000000E+02
	a + b = 9.2330000000E+01
	□

`Writeln(0.2-0.01);`	1.9000000000E-01 □
`Writeln;`	Переход на новую строку

10.2. Форматный вывод

Процедура вывода позволяет задать требуемый вид записи данных из списка вывода. Такой вывод называют **форматным**. При этом значения переменных выводятся в определенное количество позиций.

1) *Вывод целого числа.* Число после символа «:» (это число называют «формат») определяет количество символов, выделяемых на вывод значения выражения на экран. Если выводимое число содержит меньше символов, чем указано в формате, число сдвигается вправо, а левые позиции заполняются пробелами. Если количество символов выводимого числа больше, чем допускает формат, число выводится с минимально возможным количеством символов.

Например, при описании `Var K: Integer;` и K = 153:

Оператор	Вывод	Оператор	Вывод
`Writeln(K);`	153 □	`Write(K:5);`	_ _153□
`Write(K);`	153□	`Write(K:2);`	153□

Символ «_» в начале строки в данном случае используется для изображения пробела, а символ «□» указывает место расположения курсора после вывода информации.

2) *Вывод вещественного числа.* Формат имеет вид c : d (рис. 3.5), где c — общее количество позиций, d — количество позиций (знаков) после запятой:

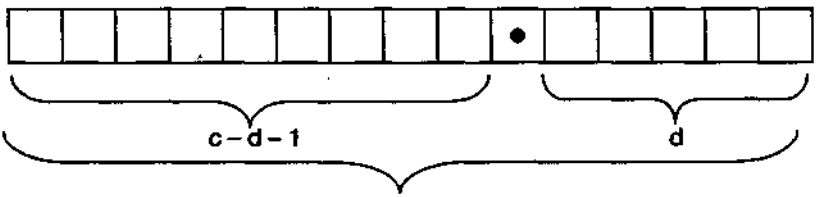


Рис. 3.5

Если d содержит меньше позиций, чем требуется для вывода числа, происходит его округление до требуемого числа знаков; при $c \leqslant d$ выдерживаются требования формата для дробной части выводимого числа, а целая часть выводится в минимально возможное количество знаков.

Пример форматного вывода вещественного числа:

Описание	Значение	Оператор вывода	Вывод
Var X:Real;	X = -29.5684	Write(X);	-2.9568400000E+01□
		Write(X:10:5);	_-29.56840□
		Write(X:10:2);	_ _ _ _-29.57□
		Write(X:4:0);	_-30□
		Write(X:0:3);	-29.568□
		Write(X:0:0);	-30□
		Write(X:7:1);	_ _-29.6□

Примеры вывода:

1. При описании

```
Var I,N:Integer; K:LongInt;
```

и значениях переменных

$$I = 123,\ N = 12345,\ K = 1234567$$

результаты вывода имеют вид:

Оператор	Вывод
`Writeln(I:5,N:7,',',K);`	_ _123 _ _12345,1234567 □
`Writeln(I:10);`	_ _ _ _ _ _ _123 □
`Writeln(I);`	123 □

2. При описании `Var X,Y: Real;` и X = 123.456, Y = -65.4321 результаты вывода имеют вид:

Операторы	Вывод

`Writeln(X:10:3,Y:15:7);`
`Writeln(X:10:4,Y:20:6);`
`Writeln(X);`
	_ _ _123.456 _ _ _ _-65.4321000
	_ _123.4560 _ _ _ _ _ _ _ _ _ _-65.432100
	1.2345600000E+02 □

3. При описании `Var X,B,C:Integer;A:Real;` и X=7, A=3.2, B=5, C=25 результаты вывода имеют вид:

Операторы	Вывод

`Write(X);`
`Writeln('_ _ _ _ _ответ a=');`
`Writeln('_ _ _ _ _ответ a=', X);`
`Write(A:0:1,',',B,'+A=',(sqrt(C)+A):0:1);`
	7_ _ _ _ _ ответ a=
	_ _ _ _ _ответ a=7
	3.2,5+A=8.2 □

4. В соответствии с оператором `Write(2*3);` на экран монитора будет выведено:

6□

Упражнение

Определите, что будет выведено на экран после выполнения следующих операторов (если b=12.4; C=-1.5):

а) `Write('b=',b,',C=');` б) `Writeln(b+C,'=',(b+C):7:2);`

в) `Writeln(C:0:5, ', ',b);` г) `Write(b,C);`

10.3. Ввод

Для ввода значений переменных с клавиатуры используют две стандартные процедуры ввода: Read и Readln.

Read (Список ввода); — стандартная процедура, которая после последнего введенного значения оставляет курсор в строке ввода.

Readln (Список ввода); — стандартная процедура, которая после завершения ввода значения переменной переведет курсор в начало следующей строки.

При наличии в списке ввода нескольких идентификаторов соответствующие значения после запуска программы следует вводить через пробел(ы) или нажимая после каждого значения клавишу Enter. При этом соответствующей переменной будет присвоено очередное по порядку значение.

При выполнении оператора ввода программа приостанавливает работу, пока пользователем не будут введены необходимые значения переменных.

Более комфортным для пользователя является сопровождение ввода данных приглашением к вводу, например:

```
Write('n1='); Readln(n1);
Write('n2='); Readln(n2);
```

Результат выполнения операторов при n1=10, n2=17:

```
n1=10
n2=17
□
```

Пример. Составить программу, которая запросит два целых числа и одно вещественное.

```
...
Var n1,n2:Integer;
    X:Real;
    ...
Begin
    Readln(n1,n2);
    Write('X=');
    Readln(X);
   ...
End.
```

После запуска программы на выполнение она приостанавливает работу в ожидании ввода двух целых чисел. После ввода с клавиатуры через пробел чисел 10 и 17 и нажатия клавиши Enter произойдет присваивание переменным $n1$, $n2$ значений 10 и 17 соответственно. Затем на экран выводится сообщение X= и программа вновь приостанавливает работу в ожидании ввода уже вещественного числа. В нашем примере это -4.7. После его ввода значение передается в область оперативной памяти переменной X.

Результат работы фрагмента программы:

```
10 17
X=-4.7
```

Ввод данных завершается нажатием клавиши Enter.

Упражнение

Найдите ошибки при использовании процедур ввода:

а) `Read(i, x, y: 5:1);`
б) `Readln(x+y, i);`
в) `Read(100, x, sqrt(y));`
г) `Readln(i);`

§ 11. Оператор присваивания

Оператор присваивания позволяет назначить переменной определенное значение, помещая это значение в соответствующую ячейку памяти (старое значение при этом стирается).

Вид (формат) оператора присваивания:

Переменная := выражение;,

где := — символ присваивания.

Оператор задает значение переменной, имя которой записано в левой части оператора присваивания.

Порядок выполнения оператора присваивания:

1) вычисляется значение выражения в правой части оператора; при этом тип результата выражения должен совпадать с типом переменной в левой части оператора или являться подмножеством типа переменной (говорят, что типы переменной и выражения должны быть **совместимыми**);

2) результат выражения присваивается переменной в левой части оператора (т. е. копируется в область памяти, которая выделена этой переменной).

Схематически порядок выполнения оператора присваивания можно изобразить так:

Идентификатор ◄———— выражение ; .

Примеры операторов присваивания даны в таблице:

Оператор	Результат выполнения оператора
`X:=9;`	Переменной *X* присваивается значение 9
`Y:=(a+b)*x/(c-d);`	После вычисления значения выражения $\frac{(a+b)x}{c-d}$ его результат присваивается переменной *Y*
`A:=A+1;`	Переменная *A* увеличит свое значение на 1. Например, при $A=5$ результат выполнения оператора: A = 6

Важно, что числовые значения идентификаторов *a*, *b*, *c*, *d*, *x* должны быть определены до их первого использования в программе в составе других операторов: они могут задаваться с помощью операторов ввода, или как константы в разделе констант, или с помощью отдельных операторов присваивания.

Пример 1. Составить программу, которая по заданным четвертным отметкам ученика по некоторому предмету определит среднюю годовую отметку.

```
Program God_1;      {отметка за год}
  Const a=7; b=8;  {a, b — отметки за 1-ю и 2-ю четверти}
        c=9; d=9;  {c, d — отметки за 3-ю и 4-ю четверти}
  Var y: Real;
Begin
  Write ('Средняя годовая отметка:');
  y:=(a+b+c+d)/4;
  Writeln(y:0:1);
End.
```

Здесь все значения задаются в разделе описания констант `Const`. Для вычисления значения *y* при других значениях *a*, *b*, *c*, *d* следует внести соответствующие изменения в раздел описания констант и вновь запустить программу на выполнение.

```
Program God_2 ;
  Var a,b,c,d: Integer;
      y: Real;
Begin
  a:=7; b:=8;
```

```
   c:=9; d:=9;
   Write('Средняя годовая отметка: ');
   y:=(a+b+c+d)/4;
   Writeln(y:0:1);
End.
```

Здесь значения переменных *a*, *b*, *c*, *d* задаются в программе с помощью операторов присваивания. Для вычисления значения *y* при других значениях *a*, *b*, *c*, *d* следует изменить операторы присваивания и вновь запустить программу на выполнение.

```
Program God_3;
   Var a,b,c,d: Integer;
       y: Real;
Begin
   Writeln ('Введите отметки:');
   Write ('за 1-ю четверть=');
   Readln(a);
   Write ('за 2-ю четверть=');
   Readln(b);
   Write ('за 3-ю четверть=');
   Readln(c);
   Write ('за 4-ю четверть=');
   Readln(d);
   y:=(a+b+c+d)/4;
   Writeln('Средняя годовая отметка=',y:0:1);
End.
```

Здесь значения всех используемых переменных вводятся по одному числу с помощью оператора ввода Readln. Для вычисления значения *y* при других значениях *a*, *b*, *c*, *d* следует вновь запустить программу на выполнение и ввести новые числовые значения.

```
Program God_4;
   Var a,b,c,d: Integer;
       y: Real;
Begin
   Writeln('Введите отметки');
   Readln(a,b,c,d);
   y:=(a+b+c+d)/4;
   Writeln('Средняя годовая отметка=', y:0:1);
End.
```

Здесь значения всех используемых переменных вводятся с помощью одного оператора ввода `Readln`. Для вычисления значения y при других значениях a, b, c, d следует вновь запустить программу на выполнение и ввести новые числовые значения.

Присваивание переменной целого типа выражения вещественного типа запрещается!

* Если в арифметическом выражении содержатся повторяющиеся вычисления, целесообразно ввести дополнительную переменную для хранения результата повторяющегося фрагмента и использовать ее в программе. Например, в формуле $y = \frac{x^2 + 1{,}4\sin x^2}{\sqrt{1+x^2}} - \frac{5\sqrt{1+x^2}}{x^2}$ вычисление x^2 встречается пять раз и два раза $\sqrt{1+x^2}$. В этом случае можно ввести три оператора присваивания:

```
K:=x*x;
S:=sqrt(1+K);
y:=(K+1.4*sin(K))/S-5*S/K; *
```

Пример 2. Составить программу «Калькулятор», которая должна запрашивать у пользователя два целых числа и затем вычислять их сумму, разность, произведение и частное.

Дано: a, b — заданные числа.

Найти: S — их сумма,
R — разность,
P — произведение,
C — частное.

Связь:
$$\begin{cases} S = a + b, \\ R = a - b, \\ P = a \cdot b, \\ C = \frac{a}{b}. \end{cases}$$

При: a, b — целые числа.

Так как a и b — целые числа, то при описании присвоим им тип `Integer`. Сумма, разность и произведение целых чисел также являются целыми числами. Поэтому переменные S, R и P опишем тоже как `Integer`.

Частное — действительное число (это видно, например, из записи 2:4=0,5). С учетом требований к типам переменных, участвующих в операции, переменную C опишем как `Real`.

Получим программу:

```
Program Calculi;
   Var  a,b,S,R,P: Integer;
        C: Real;
Begin
   Writeln('Введите два целых числа');
   Readln(a,b);
   S:=a+b; R:=a-b;
   P:=a*b; C:=a/b;
   Writeln(a,'+',b,'=',S);
   Writeln(a,'-',b,'=',R);
   Writeln(a,'*',b,'=',P);
   Writeln(a,'/',b,'=',C:0:2);
End.
```

Для проверки правильности составления программы выполним ее многократно для различных пар целых чисел. Когда при очередном запуске программы введем 6 0 Enter, на экране появится сообщение (рис. 3.6).

Возникшая ошибка свидетельствует о невозможности выполнения операции деления на нуль. В данной программе нужно предупредить пользователя, напри-

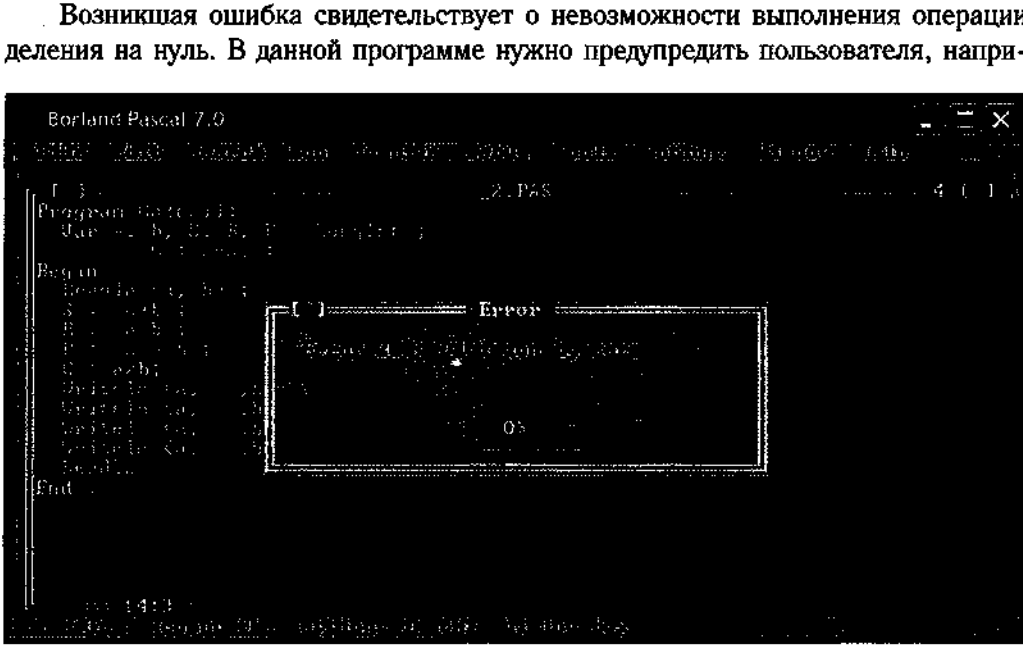


Рис. 3.6

мер с помощью вывода на экран соответствующего сообщения, о недопустимости ввода нуля в качестве второго числа:

```
Writeln('Введите два целых числа');
Writeln('Пожалуйста, не вводите 0 для второго числа!');
Readln(a,b);
```

*__Пример 3.__ Составить программу решения задачи: «Определить, на какой высоте и на каком расстоянии от места броска окажется через время t (с) мяч, брошенный под углом α к горизонту с начальной скоростью v_0 $\left(\frac{\text{м}}{\text{с}}\right)$».

Дано: v_0 $\left(\frac{\text{м}}{\text{с}}\right)$ — начальная скорость;

t (с) — время полета;

α (в градусах) — угол броска.

Найти: x (м) — расстояние от места броска;

y (м) — высота полета.

Связь:
$$\begin{cases} x = (v_0 \cos\alpha)t, \\ y = (v_0 \sin\alpha)t - \dfrac{gt^2}{2}. \end{cases} \qquad (1)$$

При:
$$\begin{cases} 0 < \alpha \leqslant 90, \\ 0 < v_0 \leqslant 20, \\ t > 0. \end{cases} \qquad (2)$$

Метод: Выполнить расчет по формулам (1) при ограничениях (2). При этом следует выполнить перевод значения α из градусной меры в радианную: $\alpha = \frac{\alpha \cdot \pi}{180}$. При $y \leqslant 0$ будем считать, что мяч лежит на земле.

Положим, что данные корректны и проверка ограничений (2) не требуется.

Ниже даны графическое представление алгоритма решения задачи (рис. 3.7) и программа на языке Паскаль.

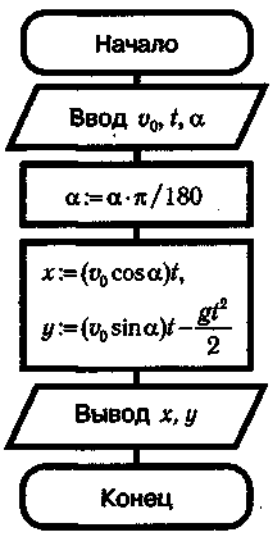


Рис. 3.7

```
Program BROSOK;
  Const g=9.8;
  Var V0,t,alfa,x,y: Real;
Begin
  Write('введите скорость ');
```

```
  Readln(V0);
  Write('введите время ');
  Readln(t);
  Write('введите угол ');
  Readln(alfa);
  alfa:=alfa*pi/180;
  x:=V0*cos(alfa)*t;
  y:=V0*sin(alfa)*t-g*t*t/2;
  Writeln('x=',x:0:1);
  Writeln('y=',y:0:1);
End. *
```

Программист может поручить компьютеру формирование значений переменных в виде случайных чисел. Для этой цели используются стандартные функции: **Random** — для получения случайного действительного числа на промежутке (0; 1) и **Random(n)** — для получения случайного целого числа на отрезке [0; $n-1$]. Данным функциям в программе предшествует процедура **Randomize**, которая позволяет при повторном запуске программы получить новые случайные числа.

Пример 4. Составить программу вычисления среднего возраста двух случайных прохожих.

Программа будет иметь вид:

```
Program SR_LET;
  Var a,b: Integer; C: Real;
Begin
  Randomize;
       {Пусть заданные числа принадлежат отрезку [1; 100]}
  a:=Random(100)+1;
  b:=Random(100)+1;
  C:=(a+b)/2;
  Writeln('Средний возраст=',C:0:1);
End.
```

Пример 5. Составить программу замены значений переменных a и b (пусть дано: $a=3$, $b=5$; требуется получить: $a=5$, $b=3$).

Запись команд присваивания `a:=b; b:=a;` приведет к получению неверного результата: $a=b=5$ (убедитесь в этом!).

На практике для решения задач обмена, как правило, используют еще один объект. Например, для обмена содержимого стакана и чашки, куда налита питьевая вода (газированная и негазированная соответственно), требуется третий предмет, например кружка.

Поступим так: заменим значение переменной a на значение переменной b и наоборот. Для этого запомним в некоторой новой переменной, например c, значение переменной a (`c:=a;`); затем переменной a присвоим значение переменной b (`a:=b;`); переменной b присвоим значение переменной c (`b:=c;`), таким образом, значение переменной b станет равным a. Задача решена: `c:=a; a:=b; b:=c;` (переменные a, b, c должны быть одного типа).

?

1. **Для чего используется оператор присваивания?**
2. **Какой формат имеет оператор присваивания?**
3. **Каков порядок выполнения оператора присваивания?**
4. ***Можно ли получить решение примера 5 без использования дополнительной переменной?**

Упражнения

1. Напишите программу вычисления значения выражения с использованием наименьшего количества операторов:

   а) $x_{1,2} = \dfrac{-b \pm \sqrt{b^2 - 4ac}}{2a}$ б) $d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$

   в) $z = \dfrac{x - y}{5 + |xy|}$ *г) $y = 5x^4 + 7x^3 - 2x^2 + 9x - 4$

2. Составьте программы решения задач:

   а) Андрей положил в сберегательный банк определенную денежную сумму под простые проценты. Определите, какова будет сумма вклада через один год.

   б) Семья Николая планирует провести выходные на озере Нарочь. Рассчитайте стоимость проезда всей семьи в автомобильном транспорте, если известна стоимость билета в междугороднем автобусе. Стоимость билета в экспрессе на 50 % дороже, чем в междугороднем автобусе, а в маршрутном такси — на 60 % дороже, чем в экспрессе. Требуется приобрести два полных билета и два детских со скидкой 50 %.

   *в) Нужно купить обои для оклеивания двух стен, размером $a \times b$ м каждая. Известна стоимость одного рулона обоев длиной 12 м и шириной 1 м. Определите стоимость обоев.

§ 12. Текстовый и графический режимы

12.1. Текстовый режим

При выполнении программы вывод информации на экран может осуществляться в одном из двух режимов: **текстовом или графическом**. После загрузки среды программирования Borland Pascal экран находится в текстовом режиме.

Текстовый экран представляет собой совокупность строк, которые разбиваются на позиции. В каждой позиции можно разместить один знак, координаты которого (горизонтальная и вертикальная) задаются двумя целыми числами: номером позиции в строке и номером строки, например точка экрана (40,12) находится в 40-й позиции 12-й строки (рис. 3.8).

Чаще всего на экране в текстовом режиме размещается 25 строк по 80 знаков в каждой (возможны и другие текстовые режимы, например 50 строк по 80 знаков).

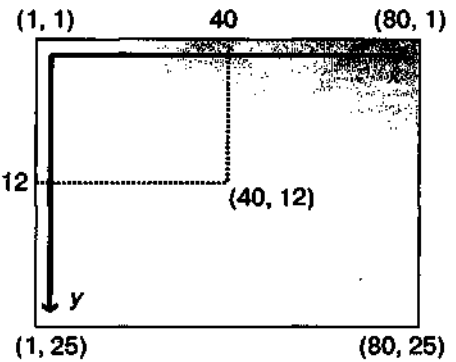


Рис. 3.8

Для управления режимами экрана, цветовой гаммой, звуком, кодами клавиатуры и др. используются подпрограммы модуля CRT (или OPCRT — проверьте имя Вашей библиотеки!). Для подключения модуля укажем его имя в разделе описания модулей, который размещается в описательной части сразу после заголовка программы: **USES CRT;** . После этого пользователю доступны следующие процедуры и функции:

Подпрограмма	Назначение	Пример
`ClrScr;`	Очистка экрана (заполнение цветом фона) и размещение курсора в левом верхнем углу	`ClrScr;`
`Delay(x);`	Приостановка выполнения программы на *x* миллисекунд (практически значение *x* зависит от тактовой частоты процессора)	`Delay(1000);`

`GotoXY(X,Y);`	Установка курсора в позицию экрана с координатами (*X*, *Y*)	Вывод текста, начиная с позиции (30,12):
		`GotoXY(30,12);`
		`Writeln('Здравствуйте!');`

Продолжение

Подпрограмма	Назначение	Пример

`Sound(S);`	Включение внутреннего динамика и генерация звука с частотой тона *S* Гц: восьми нотам до, ре, ми, ... первой октавы соответствуют частоты: 262, 294, 330, 349, 392, 440, 494, 524. При переходе к соседней октаве частоты изменяются в два раза	Звучание звука с задержкой:
		{включение звукового сигнала частотой 1200 Гц}
		`Sound(1200);`
		{задержка звучания на время 1000 мс}
		`Delay(1000);`

`NoSound;`	Выключение динамика	`NoSound;`

`TextColor(C);`	Установка цвета символов C (0 — черный, 1 — синий, 2 — зеленый, 3 — голубой, 4 — красный, 5 — фиолетовый, 6 — коричневый, 7 — светло-серый, 8 — темно-серый, 9 — ярко-синий, 10 — ярко-зеленый, 11 — ярко-голубой, 12 — розовый, 13 — малиновый, 14 — желтый, 15 — белый, 128 — мерцание символа)	Вывод цветного текста:
		`TextColor(2);`
		`GotoXY(30,5);`
		`Writeln('Здравствуйте!');`
		`TextColor(1);`
		`GotoXY(27,6);`
		`Writeln('Начинаем изучать');`
		`TextColor(14);`
		`GotoXY(34,7);`
		`Writeln('язык');`
		`GotoXY(27,8);`
		`Writeln('программирования');`
		`TextColor(4);`
		`GotoXY(33,9);`
		`Writeln('Паскаль');`

`TextBack-Ground(C);`	Установка цвета фона *C* (8 неярких цветов с номерами от 0 до 7)	`TextBackGround(4);`

Пример. Написать программу исполнения музыкальной фразы «Жили у бабуси» (фа-ми-ре-до-соль-соль). Звучание должно сопровождаться выводом текста (как в «Караоке»).

Программа будет такой:

```
Program Kleine_Musik;
   Uses Crt;
Begin
```

```
  ClrScr; Randomize;
  Sound(349); Delay(1000); Sound(330); Delay(1000);
  GotoXY(30,5); TextColor(2); Write('Жили');
  Sound(294); Delay(1000); Sound(262); Delay(1000);
  GotoXY(30,6);
  TextColor(4+128);  {Мерцание букв красного цвета}
  Write('у бабуси'); Sound(392); Delay(2000);
  Sound(392); Delay (2000);
  NoSound; Writeln ('Выход - клавиша Enter');
  Readln;
End.
```

Упражнения

1. Напишите программу для получения собственной музыкальной композиции.
2. Напишите программу вывода на экран монитора красочно оформленного текста с музыкальным сопровождением (например, компьютерной открытки-поздравления «Лучшему другу в день рождения»).

12.2. Графический режим

Всякое графическое изображение представляет собой совокупность точек — пикселей. Каждая из них задается своими координатами и цветом.

Количество отображаемых на экране пикселей называют **разрешающей способностью** (разрешением) графического экрана. Разрешение экрана может быть различным в зависимости от типа графического **адаптера** (печатной платы, на которой размещаются электронные компоненты видеодисплея) и графического **драйвера** (специальной программы, которая осуществляет управление графическим адаптером).

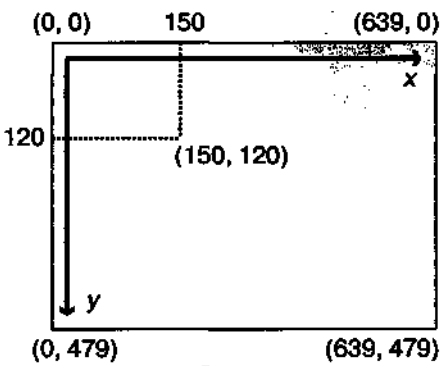


Рис. 3.9

Рассмотрим программирование графики в стандартном видеорежиме среды программирования Borland Pascal, который поддерживается современными компьютерами: 640 × 480 × 16 (640 точек по горизонтали, 480 точек по вертикали, 16 цветов).

Графический экран как совокупность точек (пикселей) координатной плоскости изображен на рисунке 3.9. Каждая точка (позиция) экрана задается номером

пикселя по горизонтали и вертикали. Например, пиксель (150, 120) находится в 150-м столбце и 120-й строке.

Для управления графическим экраном предназначен модуль GRAPH. Сам модуль представляет собой отдельный файл Graph.tpu. При программировании он должен быть доступным для компилятора. Также доступным (для запуска программ) должен быть драйвер EGAVGA.bgi. Для упрощения программирования скопируем файл Graph.tpu и драйвер EGAVGA.bgi в рабочую папку. Файл Graph.tpu хранится в папке UNITS, а файл EGAVGA.bgi — в папке BGI, вложенной в папку BP.

Для работы необходимо выполнить следующие действия:

1. Подключить модуль GRAPH — `USES GRAPH;`

2. Выполнить инициализацию графического режима (переход в графический режим):

а) в разделе описания переменных описать две целочисленные переменные для указания драйвера и режима работы графического адаптера, например gd и gr соответственно: `Var gd,gr: Integer;`

б) в разделе операторов установить графический режим с помощью процедуры INITGRAPH:

```
...
Begin
   gd:=DETECT;
   INITGRAPH(gd,gr,'');
```

3. Выполнить графические построения.

4. В конце программы выполнить возврат в текстовый режим с помощью процедуры CLOSEGRAPH:

```
...
   Readln; {задержка результата на экране монитора}
   CloseGraph;
End.
```

Общий вид процедуры INITGRAPH: **`INITGRAPH (драйвер, режим, путь);`**. Здесь «путь» — полный путь к папке, в которой хранится нужный графический драйвер. Так как требуемый графический драйвер мы скопировали в текущую рабочую папку, путь будем представлять как «пустую строку». Для задания типа драйвера используем режим автоопределения DETECT.

* При инициализации графического режима можно написать конкретный путь к файлам драйверов, например InitGraph (драйвер, режим,

'C:\BP\BGI');. В этом случае предварительное копирование файлов Graph.tpu и EgaVga.bgi в рабочую папку не требуется.
Можно также соответствующим образом настроить среду программирования BP: для этого в режиме среды Options — Directories — Unit directories указывается каталог, в котором размещаются требуемые файлы; например C:\BP\UNITS; \BP\BGI. *

§ 13. Создание изображений

13.1. Задание цвета

Прежде чем приступить к рисованию, художник готовит чистый холст и краски. При рисовании на языке Паскаль холст (фон рисунка) можно очистить с помощью процедуры **ClearDevice;**. При этом холст заполняется фоновым цветом (по умолчанию черным).

Для выбора цвета фона, на котором будет выполняться рисунок, используется процедура SetBkColor:

SetBkColor (номер цвета); .

Эта процедура изменяет цвет фона, не изменяя изображений (в отличие от процедуры ClearDevice, которая стирает все изображения).

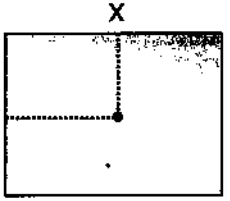
Для выбора цвета изображения используется процедура SetColor:

SetColor (номер цвета); .

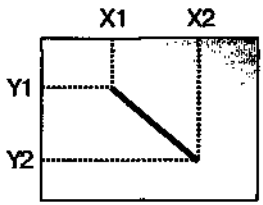
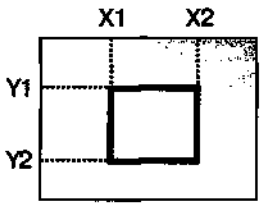
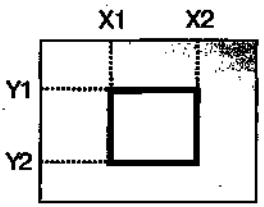
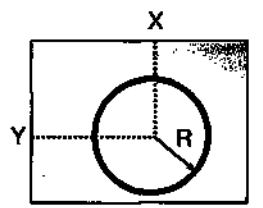
1. При инициализации графического режима экран автоматически очищается (заполняется фоновым цветом).
2. В программе вначале задается цвет изображения, и лишь затем используются средства для получения этого изображения!

Цвет выбирается из таблицы цветов (см. п. 12.1), например процедурой SetColor(4);.

13.2. Построение графических примитивов

Объект	Изображение	Запись на языке Паскаль
Точка заданного цвета		PutPixel(X,Y,цвет);

Продолжение

Объект	Изображение	Запись на языке Паскаль
Отрезок прямой с заданными координатами его начала и конца		`Line(X1,Y1,X2,Y2);` или `Line(X2,Y2,X1,Y1);`
Прямоугольник с заданными координатами его диагональных точек		`Rectangle (X1,Y1,X2, Y2);`
Закрашенный прямоугольник с заданными координатами его диагональных точек		`Bar(X1,Y1,X2,Y2);`
Окружность заданного радиуса с заданными координатами центра		`Circle(X,Y,R);`

Пример 1. С помощью рисунка 3.10 составить программу получения на экране монитора изображения прямоугольного треугольника и точки желтого цвета на зеленом фоне.

Программа будет такой:

```
Program Triangle;
  Uses Graph;
  Var a,b: Integer;
```

```
Begin
   a:=Detect;
   InitGraph(a,b,'');
   SetBkColor(2);
   SetColor(14);
   Line(240,70,240,270);
   Line(240,270,400,270);
   Line(240,70,400,270);
   PutPixel(400,70,14);
   Readln; CloseGraph;
End.
```

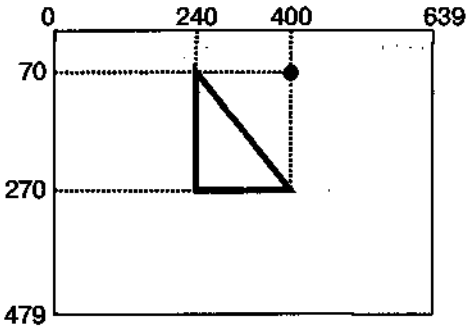


Рис. 3.10

Пример 2. С помощью рисунка 3.11 составить программу получения на экране монитора изображения снежной бабы (белого цвета на синем фоне).

Программа будет такой:

```
Program BABA;
   Uses Graph;
   Var a,b: Integer;
Begin
   a:=Detect; InitGraph(a,b,'');
   SetBkColor(1);
   SetColor(15);
   Circle(320,60,30);
   Circle(320,140,50);
   Circle(320,260,70);
   Readln; CloseGraph;
End.
```

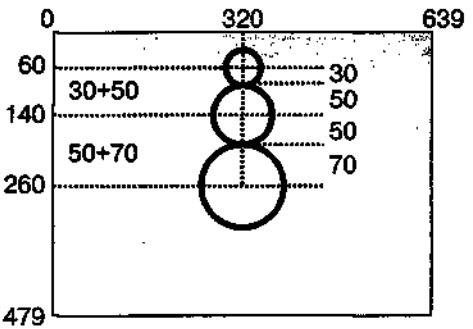


Рис. 3.11

Пример 3. С помощью рисунка 3.12 составить программу получения на экране монитора изображения головы робота.

Программа будет такой:

```
Program ROBOT;
   Uses Graph;
   Var a,b: Integer;
Begin
   a:=Detect; InitGraph(a,b,'');
   SetBkColor(1);
```

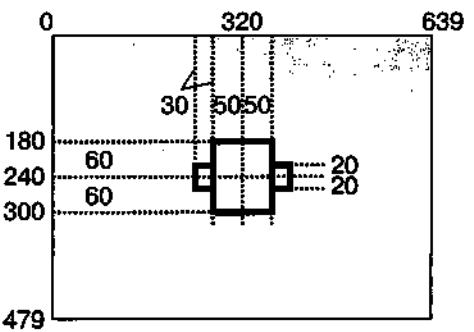


Рис. 3.12

```
   SetColor(15); {голова}
   Rectangle(270,180,370,300); {уши}
   Rectangle(240,220,270,260);
   Rectangle(370,220,400,260);
   Readln;
   CloseGraph;
End.
```

*1. При построении линейных изображений, состоящих из последовательно соединенных отрезков, удобно использовать процедуры:

`MoveTo(X,Y);`	Перемещение текущего указателя в точку с координатами (X,Y)
`MoveRel(dX,dY);`	Перемещение указателя от точки (X,Y) до точки (X+dX, Y+dY)
`LineTo(X,Y);`	Построение отрезка от текущей точки в точку (X,Y)
`LineRel(dX,dY);`	Построение отрезка от текущей точки (X,Y) до точки (X+dX,Y+dY)

Положение текущей точки зависит от графического оператора, который был выполнен последним. Например, после оператора `Line(0, 0, 70, 50);` текущей становится точка (70, 50); после оператора `PutPixel(30, 90);` — точка (30, 90).

Пример 4. С помощью рисунка 3.13 составить программу получения изображения прямоугольной трапеции.

Программа будет такой:

```
Program TRAPECIA_1;
   Uses Graph;
   Var a,b: Integer;
Begin
   a:=Detect; InitGraph(a,b,'');
   MoveTo(240,60);
   LineTo(240,200); {вниз}
   LineTo(450,200); {вправо}
   LineTo(350,60); {влево вверх}
   LineTo(240,60); {влево}
   Readln; CloseGraph;
End.
```

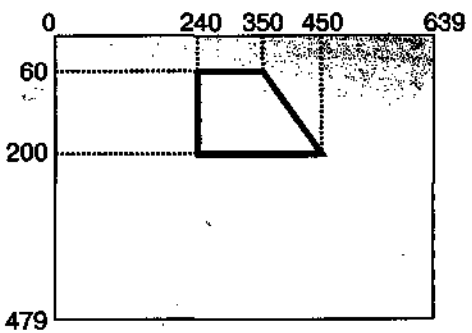


Рис. 3.13

Второй вариант решения примера 4 (с использованием процедуры LineRel):

```
Program TRAPECIA_2;
   Uses Graph; Var a,b: Integer;
Begin
   a:=Detect;InitGraph(a,b,'');
   MoveTo(240,60); LineRel(0,140);
   LineRel(210,0); LineRel(-100,-140);LineRel(-110,0);
   Readln; CloseGraph;
End.
```

2. Язык Паскаль позволяет изображать линии разнообразных стилей (по ширине и начертанию — внешнему виду). Установка стиля производится с помощью процедуры **SetLineStyle (вид линии, шаблон, толщина линии);** . Будем считать шаблон равным нулю. Остальные значения могут быть такими: *вид линии*: сплошная линия (0), точечная (1), штрихпунктирная (2), пунктирная (3); *толщина линии*: нормальной толщины (1), тройной толщины (3). Например, `SetLineStyle(0,0,3);` {Задание линии большой толщины} `SetLineStyle(3,0,1);` {Задание пунктирной линии нормальной толщины}.
3. Возможно совместное подключение модулей Crt и Graph и использование в одной программе подпрограмм обоих модулей. Например, для замедления процесса рисования можно использовать процедуру Delay:

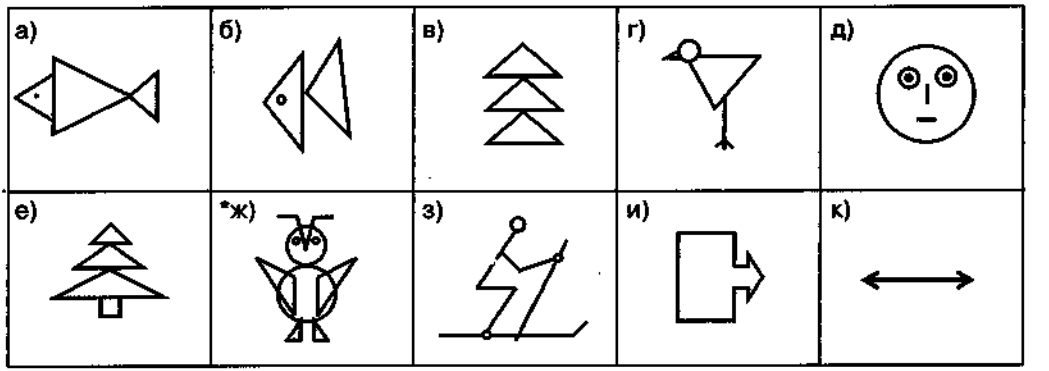
```
   Program CRT_GRAPH;
    Uses Crt,Graph;
    Var a,b: Integer;
Begin …
 LineTo(240,200);Delay(200);
 LineTo(450,200);Delay(300);
 …
 Readln; CloseGraph;
End.*
```

Упражнения

1. Испытайте приведенные в примерах 1—4 программы, дополните рисунки необходимыми деталями:

   а) у снежной бабы (см. рис. 3.11) изобразите глаза-пуговки, нос-морковь;
   б) дорисуйте роботу (см. рис. 3.12) глаза прямоугольной формы, нос, рот, антенны; вместо процедуры Rectangle используйте процедуру Bar.

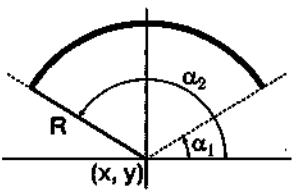
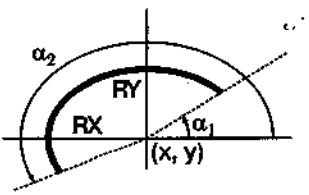
2. Напишите программы получения изображений:



При выполнении изображений используйте различные стили: в прямоугольнике изобразите, например, диагонали пунктирной линией.

*3. Получите с помощью счетчика случайных чисел координаты трех точек графического экрана — вершин треугольника — и изобразите его.

*13.3. Построение сложных графических объектов

Объект	Изображение	Запись на языке Паскаль
Дуга окружности текущего цвета		`Arc(X,Y,α1,α2,R);` (X,Y) — координаты центра дуги, R — радиус дуги окружности
Дуга эллипса текущего цвета		`Ellipse(X,Y,α1,α2,RX,RY);` (X,Y) — координаты центра эллипса, RX, RY — длины горизонтальной и вертикальной полуосей (большого и малого радиусов)

Здесь α_1, α_2 — начальный и конечный углы, которые образуют концы дуги с горизонтальной осью (отсчитываются против хода часовой стрелки).

Пример 1. Составить программу получения изображения эллипса с центром в точке (320, 240) и длинами полуосей 190 и 80 (рис. 3.14).

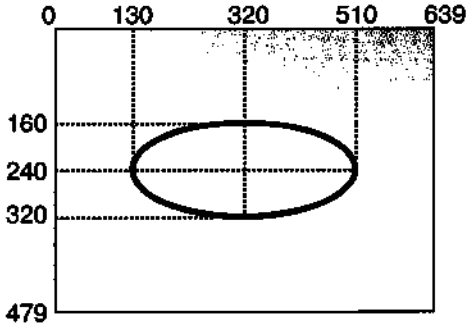


Рис. 3.14

Программа будет такой:

```
Program ELIPS;
   Uses Graph;
   Var a,b: Integer;
Begin
   a:=Detect;
   InitGraph(a,b,'');
   Ellipse(320,240,0,360,190,80);
   Readln;
   CloseGraph;
End.
```

Пример 2. Составить программу получения изображения «левой» полуокружности с центром в точке (250, 200) и радиусом 100 (рис. 3.15).

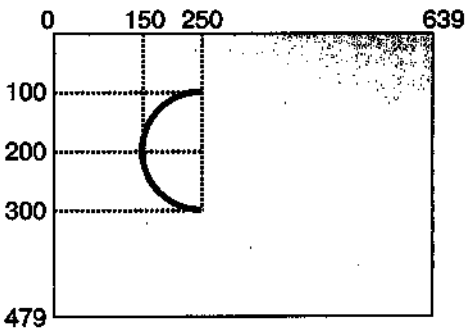


Рис. 3.15

Программа будет такой:

```
Program POLUOKR;
   Uses Graph;
   Var a,b: Integer;
Begin
   a:=Detect;
   InitGraph(a,b,'');
   Arc(250,200,90,270,100);
   Readln; CloseGraph;
End.
```

13.4. Заполнение областей изображения

Вы уже знаете, каким образом в языке Паскаль изображается закрашенный прямоугольник. Имеются другие средства для закрашивания («заливки») замкнутых областей.

Будем понимать под **заливкой** заполнение некоторым узором (цветом или штриховкой различного вида) области изображения, ограниченной **контуром** (непрерывной замкнутой линией).

Пусть на экране монитора получено изображение некоторого замкнутого контура с использованием цвета *C* (*C* — цвет контура области) и задана некоторая точка экрана (X, Y). Если данная точка принадлежит области, ограниченной контуром, то эта область может быть заполнена текущим цветом с помощью процедуры **FloodFill(X,Y,C);**. Если точка находится вне области, происходит заполнение внешней к контуру области.

Для выбора стиля (шаблона и цвета) заполнения области используется процедура SetFillStyle:

SetFillStyle(шаблон, цвет заполнения);.

Значения шаблона задаются равными: 0 (сплошной цвет фона), 1 (сплошной текущий цвет), 2—6 (штриховка различной толщины и наклона), 7—9 (заполнение клеткой различной толщины и наклона), 10—11 (заполнение точками, густо или редко расположенными).

При использовании процедуры SetFillStyle цвет заполнения может быть не равен цвету контура.

Пример 1. С помощью рисунка 3.16 написать программу заполнения прямоугольного треугольника белым цветом (с контуром желтого цвета на зеленом фоне).

Программа будет такой:

```
Program Triangle_1;
   Uses Graph;
   Var a,b: Integer;
Begin
   a:=Detect; InitGraph(a,b,'');
   SetBkColor(2);SetColor(14);
   Line(240,70,240,270);
   Line(240,270,400,270);
   Line(240,70,400,270);
   FloodFill(320,200,14);
   Readln; CloseGraph;
End.
```

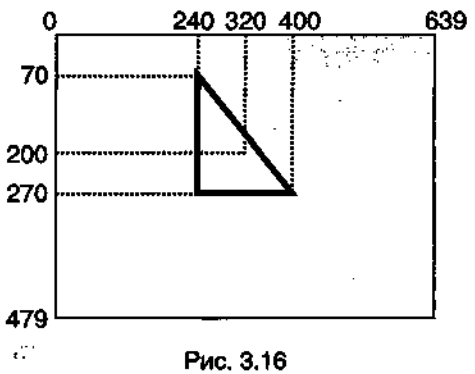


Рис. 3.16

Пример 2. С помощью рисунка 3.16 написать программу заполнения прямоугольного треугольника красным цветом (с контуром желтого цвета на зеленом фоне).

Скорректируем программу Triangle_1:

```
...
{корректировка программы Triangle_1}
```

```
   Line(240,70,400,270);
   SetFillStyle(1,4);
   {сплошной красный цвет заливки}
   FloodFill(320,260,14);
   ...
```

***Пример 3.** С помощью рисунка 3.17 написать программу заполнения прямоугольной трапеции штриховкой светло-серого цвета нормальной толщины.

Программа будет такой:

```
Program TRAPECIA_3;
   Uses Graph;
   Var a,b: Integer;
Begin
   a:=Detect;
   InitGraph(a,b,'');
   SetColor(1); {синий контур}
   MoveTo(240,60);
   LineTo(240,200);
   LineTo(450,200);
   LineTo(350,60);
   LineTo(240,60);
   SetFillStyle(3,7);
   FloodFill(350,175,1);
   Readln;
   CloseGraph;
End.
```

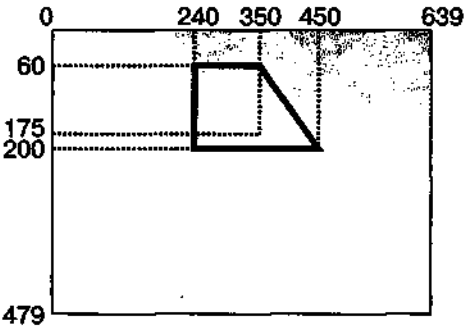


Рис. 3.17

Пример 4. С помощью рисунка 3.18 напишите программу заполнения окружности желтым цветом, которая после нажатия на клавишу Enter изменяет цвет заливки на зеленый.

Программа будет такой:

```
Program CIRC;
   Uses Graph;
   Var a,b: Integer;
Begin
   a:=Detect;
   InitGraph(a,b,'');
   SetColor(14);{желтый контур}
   Circle(320,175,100);
```

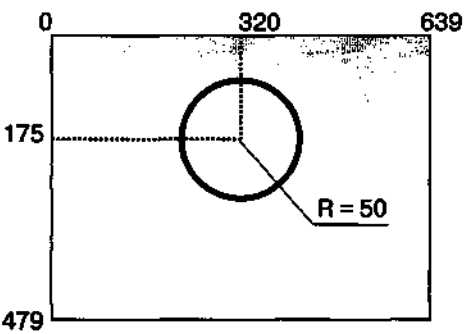
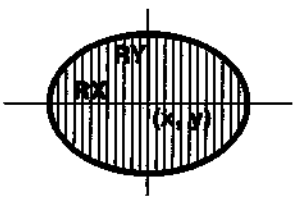
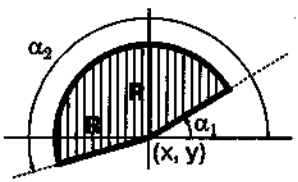
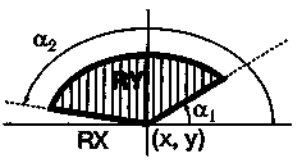
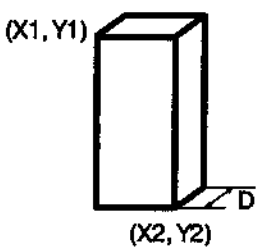


Рис. 3.18

```
  SetFillStyle(1,14);        {желтая заливка}
  FloodFill(320,175,14);
  Readln;
  ClearDevice;               {очистка экрана}
  SetColor(14);              {желтый контур}
  Circle(320,175,50);
  SetFillStyle(1,2);         {зеленая заливка}
  FloodFill(320,175,14);
  Readln; CloseGraph;
End.
```

В языке Паскаль существуют специальные средства для заполнения эллипса, секторов круга и эллипса, параллелепипеда.

Объект	Изображение	Запись на языке Паскаль
Эллипс, закрашенный текущим узором и цветом заполнения		`FillEllipse(X,Y,RX,RY);` где (X,Y) — координаты центра эллипса, RX, RY — длины горизонтальной и вертикальной полуосей
Сектор круга, заполненный текущим узором		`PieSlice(X,Y,α1,α2,R);` где (X,Y) — координаты центра, α_1, α_2 — начальный и конечный углы сектора, R — его радиус
Сектор эллипса, закрашенный цветом по текущему шаблону		`Sector(X,Y,α1,α2,RX,RY);` где (X,Y) — координаты центра эллипса, RX, RY — длины горизонтальной и вертикальной полуосей, α_1, α_2 — начальный и конечный углы сектора
Параллелепипед, передняя грань которого может быть закрашена по текущему шаблону		`Bar3D(X1,Y1,X2,Y2,D,Top);` где (X1, Y1), (X2, Y2) — координаты диагональных точек передней грани (прямоугольника), D — ширина боковой грани, Top — признак отображения верхней грани (TRUE — отображать, FALSE — не отображать)

Пример 5. Написать программу получения изображения геометрических объектов на экране монитора (рис. 3.19).

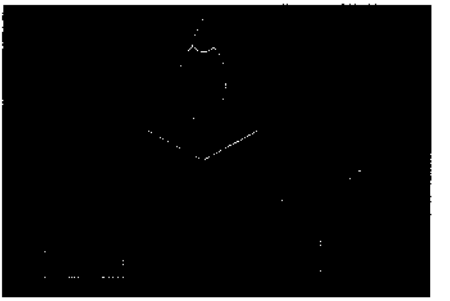


Рис. 3.19

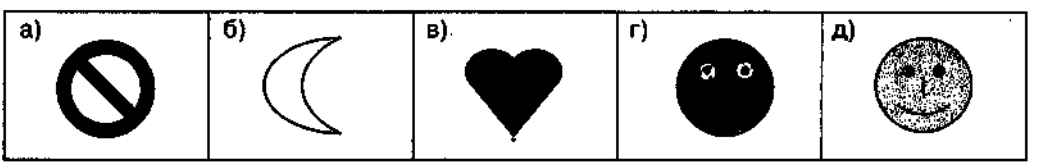
Программа будет такой:

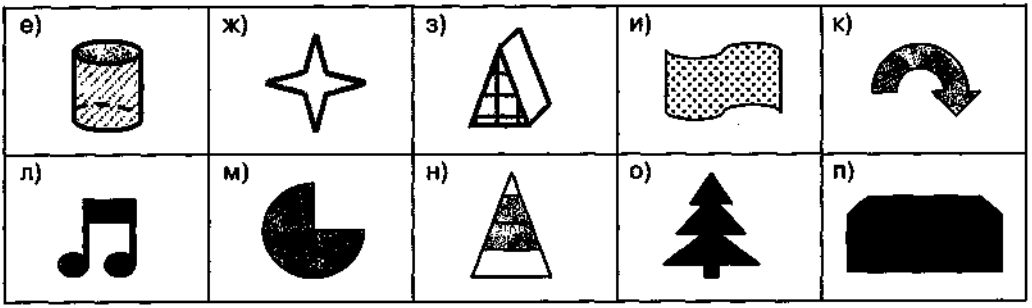
```
Program PRIM;
   Uses Graph;
   Var a,b: Integer;
Begin
   a:=Detect;
   InitGraph(a,b,'');
   SetBkColor(1);                      {синий цвет фона}
   SetColor(14);                       {желтый контур}
   SetFillStyle(1,12);                 {розовая заливка}
   FillEllipse(300,200,30,50);         {эллипс}
   SetFillStyle(1,10);                 {ярко-зеленый}
   PieSlice(300,140,315,225,20);       {сектор}
   SetFillStyle(1,8);                  {темно-серый}
   Sector(300,300,45,135,100,50);      {сектор эллипса}
   Line(50,450,600,450);
   SetFillStyle(1,2);                  {зеленый закрашенный}
   Bar3D(200,400,100,450,20,True);     {параллелепипед}
   SetFillStyle(1,4);                  {красный закрашенный}
   Bar3D(400,450,450,350,50,True);     {параллелепипед}
   Readln; CloseGraph;
End.*
```

1. **В каких режимах может выполняться вывод информации на экран монитора?**
2. **Какова разрешающая способность экрана при выводе графической информации на Вашем компьютере?**
3. **Каково назначение процедур** `InitGraph` **и** `CloseGraph`**?**

Упражнения

1. Напишите программу получения изображения (на выбор):





*2. Случайным образом разместите на экране треугольник и изобразите его медианы.

§ 14. Работа с текстом

14.1. Отображение текста

В графическом режиме на экран можно выводить только текстовую информацию. Для этих целей используются стандартные подпрограммы:

`OutText('текст');` — текст выводится на экран монитора, начиная с текущей позиции указателя (графического курсора).

`OutTextXY(X,Y,'текст');` — текст выводится на экран монитора, начиная с позиции (X, Y).

Пример. Дополнить программу построения треугольника Triangle (п.13.2) выводом текста «Прямоугольный треугольник» (коричневым цветом) ниже изображения, начиная с позиции (220, 330).

Дополнение программы:

```
SetColor(6);
OutTextXY(220,330,'Прямоугольный треугольник');.
```

*14.2. Установка шрифта и стиля

Для изменения внешнего вида выводимого текста (ориентации, направления вывода, размера символов) используется процедура `SetTextStyle`:

`SetTextStyle (шрифт, ориентация, размер);.`

Для параметра «шрифт» определены значения: 0 (обычный матричный шрифт), 1 (полужирный шрифт), 2 (тонкий шрифт), 3 (рубленый шрифт), 4 (готический шрифт).

Значения параметра «ориентация»: 0 (обычное горизонтальное отображение текста), 1 (каждый символ повернут на 90° в направлении против хода часовой

стрелки, при этом текст отображается снизу вверх); 2 (отображение текста в горизонтальном направлении слева направо).

Значения параметра «размер»: от 0 до 10.

Использование процедуры `SetTextStyle` требует наличия в текущем каталоге файлов шрифтов *.chr.

Пример

```
SetTextStyle(0,0,5);
OutText('Горизонтальный вывод большими буквами');
SetTextStyle(0,1,1);
OutText ('Вертикальный вывод');
```

Упражнение

Напишите программы вывода изображений:

а) плаката «Программирование — вторая грамотность»;

*б) текста программы вычисления суммы двух чисел, значения которых вводятся с клавиатуры;

*в) расписания уроков на понедельник.

§ 15. Алгоритмическая структура ВЕТВЛЕНИЕ

В языке Паскаль выделяются два класса операторов: *простые* и *структурные*. **Простой** оператор не содержит в своей записи других операторов, а его выполнение не связано с анализом некоторых условий (программа выполняется последовательно: от первого оператора ко второму, от второго — к третьему и т. д.). **Структурный** оператор содержит в своем составе другие операторы, порядок выполнения которых определяется некоторыми условиями. К первому классу относятся оператор присваивания и операторы обращения к процедурам, ко второму — условный оператор (оператор ветвления), операторы выбора, цикла (`If`, `Case`, `While`, `For`). Несколько операторов с помощью операторных скобок `Begin` и `End` объединяются в составной оператор.

С помощью простых операторов можно реализовать линейные алгоритмы, с помощью условного оператора и оператора выбора — разветвляющиеся алгоритмы, с помощью операторов цикла — циклические алгоритмы.

Параграфы 15, 16 посвящены изучению структурных операторов языка программирования Паскаль.

15.1. Логические величины и выражения

Довольно часто на поставленный вопрос мы получаем ответ «да» или «нет». Величины, которые могут принимать одно из двух значений «истина» (TRUE) или «ложь» (FALSE), называют **логическими**. Им соответствует логический тип данных, который в языке Паскаль описывается как `Boolean`. Значение логического типа занимает в памяти 1 байт: FALSE — 0, TRUE — 1.

Допустимые операции для логического типа данных: операции сравнения (отношения) и логические операции.

Арифметические выражения можно сравнивать между собой с помощью операций сравнения. Эти операции известны нам из курса математики. В языке Паскаль им соответствуют **операции сравнения**: = (равно), <> (не равно), > (больше), >= (больше или равно, т. е., не меньше), < (меньше), <= (не больше).

С помощью операций сравнения строятся **выражения сравнения**, например, `x=5; a>b; a*x+3<=0; 7-x<>5*c; a>=b*c; omega < pi/2; 4=3`. Их можно присваивать логическим переменным или использовать в условных операторах.

Например, если имеется описание `Var Q,W,E: Boolean;R: Real;`, то возможны присваивания: `R:=3; Q:=R<7; W:=True; E:=False;`. В результате выполнения операторов переменные *Q*, *W*, *E* получат соответственно значения True (так как выражение 3 < 7 истинно), True, False.

Пример 1. Составить программу проверки утверждения: верно ли, что $A > B$ (числовые значения величин *A* и *B* вводятся с клавиатуры).

Программа на языке Паскаль может быть такой:

```
Program TRU_FA1;
  Var  A,B: Real;
       Z: Boolean;
Begin
  Write ('Введите число A=');
  Readln(A);
  Write ('Введите число B=');
  Readln(B);
  Z:=A>B;
  Writeln('A>B? ', Z);
End.
```

После запуска программы введем числа 5 (оно будет передано переменной *A*) и 3 (передается переменной *B*). Результат работы программы:

`A>B? True` (действительно, 5 > 3 — истинное неравенство).

Если ввести числа 0 и 4,8, результат будет таким:

`A>B? False`

Пример 2. Составить программу проверки: правильно ли пользователь даст ответ на вопрос: «Когда был основан город Минск?»

Программа на языке Паскаль может быть такой:

```
Program TRU_FA2;
   Var  G: Integer;
        Z: Boolean;
Begin
   Writeln('Когда был основан город Минск?');
   Readln(G);
   Z:=G=1067;
   Writeln(Z);
End.
```

Выражение называется **логическим**, если результатом его вычисления является логическое значение: TRUE или FALSE.

Логические выражения строятся с помощью допустимых операндов и логических операций AND (И), OR (ИЛИ), NOT (НЕ). Примеры логических выражений: `(a>b) and (a>c); (X<=1) or (Y>5) and (a>2); (4*X-1>0) and (X+2<3); not (a=b)`. Операции сравнения имеют низший приоритет по сравнению с другими операциями, поэтому отношения в логических выражениях заключают в скобки.

Ниже записаны логические операции от высшего приоритета к низшему (операции высшего приоритета выполняются в первую очередь):

Операция	Действие
NOT	Отрицание, или логическое НЕ
AND	Логическое умножение, или логическое И
OR	Логическое сложение, или логическое ИЛИ

Результаты выполнения логических операций (T — истина, F — ложь) приведены в следующей таблице:

X	Y	Логическая операция		
		NOT X	X AND Y	X OR Y
T	T	F	T	T
T	F	F	F	T
F	T	T	F	T
F	F	T	F	F

Пример 3. Определить порядок вычисления значения логического выражения `(4*X-1>0) and (X+2<3)`.

Сначала вычисляется значение выражения `4*X-1`, затем оно сравнивается с числом 0 и определяется истинность условия `4*X-1>0`; аналогично для логического выражения `X+2<3`; затем вычисляется значение всего выражения в зависимости от значений каждого выражения:

X	4*X-1>0	X+2<3	(4*X-1>0) and (x+2<3)
5	19>0 → True	7<3 → False	True and False → False
0.5	1>0 → True	2.5<3 → True	True and True → True

Пример 4. Написать программу, проверяющую, верно ли, что человек, возраст которого x, является школьником.

Задача сводится к проверке условия $6 \leqslant x \leqslant 19$. Программа на языке Паскаль может быть записана следующим образом:

```
Program LOGIST;
   Var  x: Byte;
        L,P,Z: Boolean;
Begin
   Write('Возраст=');
   Readln(x);
   L:=x>=6;
   P:=x<=19;
   Z:=L and P;
   Writeln('Это школьник? ', Z);
End.
```

```
Z:=(x>=6) and (x<=19);
```

Результаты работы программы после двух запусков:

```
Возраст=34
Это школьник? False
Возраст=15
Это школьник? True
```

Упражнения

1. Запишите выражение средствами языка Паскаль:

а) $a < 1$ б) $(x-a)^2 + (y-b)^2 < r^2$ в) $|x-a| < \varepsilon$

г) $0 < a < 1$ д) $0{,}1 \leqslant X - a \leqslant 0{,}9$

2. Напишите программу, проверяющую истинность утверждения:

а) введенный с клавиатуры год относится к XXI в.;

б) введенное с клавиатуры число соответствует номеру одного из зимних месяцев;

*в) введенное число является решением неравенства $x^2 - 4 > 0$;

г) Федор может купить на заданную денежную сумму нужное количество тетрадей (стоимость одной тетради, их количество и имеющаяся сумма вводятся с клавиатуры).

15.2. Условный оператор

При решении задач часто возникает необходимость в зависимости от выполнения или невыполнения некоторого условия выполнять то либо другое действие (последовательность действий). Такая форма организации действий в алгоритмах называется **ветвлением**. Ветвление в Паскале реализуется с помощью условного оператора `IF`, который позволяет выбрать для последующего выполнения один из вариантов вычислений либо не выбрать ни одного.

В языке Паскаль существуют две формы записи команды ветвления: сокращенная `IF P THEN S;` (рис. 3.20, *а*) и полная `IF P THEN S1 ELSE S2;` (рис. 3.20, *б*).

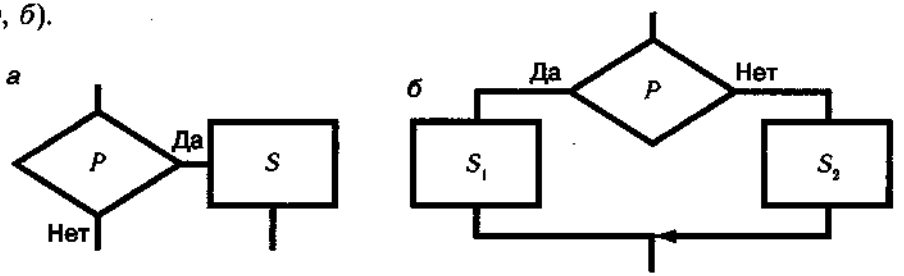


Рис. 3.20

Здесь P — условие разветвления (логическое выражение или выражение отношения), S, S_1, S_2 — выполняемые операторы.

*Условие P называют **простым**, если оно содержит единственное выражение отношения, и **составным**, если оно состоит из нескольких выражений отношения, соединенных логическими операциями. В составных условиях выражения отношения заключаются в круглые скобки.

Примеры простых условий: `a>b, 5+x=3, sqrt(x)<=25.`

Примеры составных условий:

`(x+y=5) and (2*x*sin(x)<5.5)` → $\begin{cases} x + y = 5, \\ 2x\sin x < 5{,}5 \end{cases}$

`(x<y) or not (y<z)` → $\begin{cases} x < y, \\ y \geqslant z \end{cases}$ *

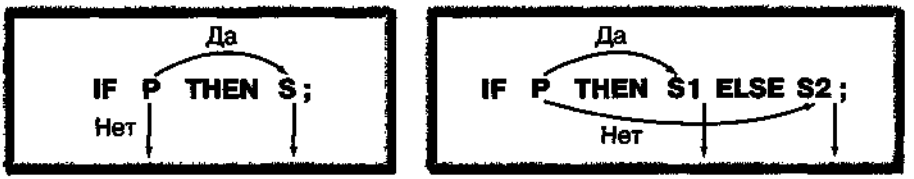


Рис. 3.21

Порядок выполнения операторов приводится на рисунке 3.21. Так, в сокращенной конструкции, если условие P истинно, выполняется оператор S, а затем — оператор, следующий за оператором `IF`.

Если по одной из ветвей после служебных слов `THEN` и (или) `ELSE` требуется выполнить более одного оператора, их ограничивают операторными скобками `BEGIN ... END` и рассматривают как составной оператор.

Например, в программе, определяющей наибольшее число из двух заданных чисел:

$$x = \begin{cases} a, & \text{если } a > b, \\ b, & \text{если } a \leqslant b, \end{cases}$$

условный оператор будет выглядеть так:

```
If a>b Then x:=a Else x:=b;
```

Если требуется определить наибольшее число x из двух заданных чисел a и b и вывести его номер k:

$$x = \begin{cases} a, & \text{если } a > b, \\ b, & \text{если } a \leqslant b; \end{cases}$$

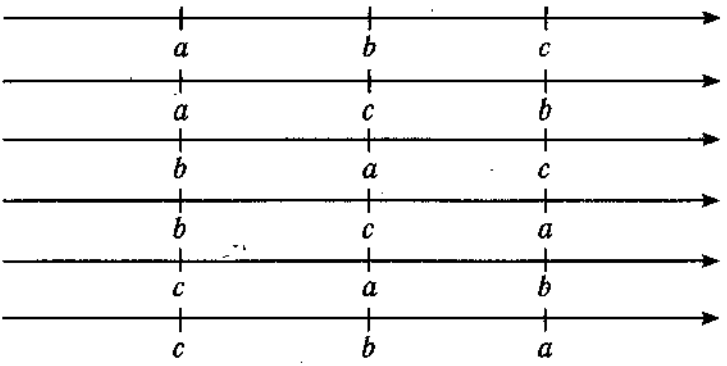
$$k = \begin{cases} 1, & \text{если } a > b, \\ 2, & \text{если } a \leqslant b, \end{cases}$$

условный оператор в программе на языке Паскаль будет таким:

```
If a>b Then
   Begin x:=a; k:=1;
   End
      Else
   Begin x:=b; k:=2;
   End;
```

Пример 1. Записать условный оператор для программы, позволяющей определить наибольшее число среди трех заданных чисел.

Ниже показаны возможные взаимные расположения на числовой прямой точек, соответствующих числам a, b, c.



Условный оператор может выглядеть следующим образом:

```
If a>b  Then
   IF a>c  Then x:=a Else x:=c;
If a<b Then
   IF b>c  Then x:=b Else x:=c;
```

* Использование логической операции and может сделать наш оператор более компактным:

```
If(a>b) and (a>c) Then x:=a;... *
```

Пример 2. Написать программу, моделирующую работу пожарного датчика в помещении: программа должна выводить сообщение «Пожароопасная ситуация», если температура в комнате превысит 60 °С; если температура не более 60 °С, датчик «молчит».

На языке Паскаль программа решения задачи может выглядеть так:

```
Program OGON;
   Var T: Real;
Begin
   Write('T=');Readln(T);
   If T>60 then Writeln ('Пожароопасная ситуация');
End.
```

Пример 3. Написать программу, которая позволит определить, является ли человек, возраст которого x, школьником.

Задача сводится к проверке условия: $6 \leqslant x \leqslant 19$ (сравните с решением примера 4 п. 15.1).

```
Program School_1;
   Var X: Byte;
```

```
Begin
   Write('Возраст=');
   Readln(X);
   If X>=6 Then
      If X<=19 Then
         Writeln ('школьник');
End.
```

* Если использовать логическую операцию and, программа будет выглядеть так:

```
Program School_2;
   Var X: Byte;
Begin
   Write ('Возраст='); Readln(X);
   If (X>=6) and (X<=19) Then Writeln ('школьник');
End.
```

Пример 4. Составить программу, которая определит, на какой высоте и на каком расстоянии от места броска окажется через время t с мяч, брошенный под углом α к горизонту с начальной скоростью v_0 м/с (с учетом проверки корректности исходных данных) (см. пример 3 § 11).

Получим программу:

```
Program BROSOK;
   Const g=9.8;
   Var V0,t,alfa,x,y: Real;
Begin
   Write('введите скорость '); Readln(V0);
   Write('введите время '); Readln(t);
   Write('введите угол '); Readln(alfa);
   alfa:=alfa*pi/180;
   If (alfa>0) and (alfa<=pi/2) and (V0>0) and (V0<=20) and
   (t>0) Then
      Begin
         x:=V0*cos(alfa)*t;
         y:=V0*sin(alfa)*t-g*t*t/2;
         Writeln('x=',x);
         Writeln('y=',y);
      End;
End.
```

В программе используется неполная команда ветвления: если требуемые условия не выполняются, программа не совершает никаких действий. Можно предусмотреть вывод сообщения о некорректных данных:

```
If (alfa>0) and (…) … then
   Begin … End
      Else Writeln('Некорректные данные');
```

Пример 5. Написать программу для решения неравенства $ax > b$ (a, b — действительные числа).

Возможны следующие варианты ответа:

1) $x > b/a$ (при $a > 0$);

2) $x < b/a$ (при $a < 0$);

3) нет решений (при $a = 0$, $b \geqslant 0$);

4) x — любое число (при $a = 0$, $b < 0$).

Графическое представление алгоритма решения задачи показано в виде блок-схемы на рисунке 3.22.

Программа на языке Паскаль может быть такой:

```
Program Neravenstvo;
  Var a,b: Real;
Begin
  Write('a=');Readln(a);
  Write('b=');Readln(b);
  If a>0 Then
     Writeln('x>',b/a)
       Else If a<0 Then
         Writeln('x<',b/a)
            Else If b>=0 Then
       Writeln ('нет решений')
     Else
  Writeln('x — любое число');
End.
```

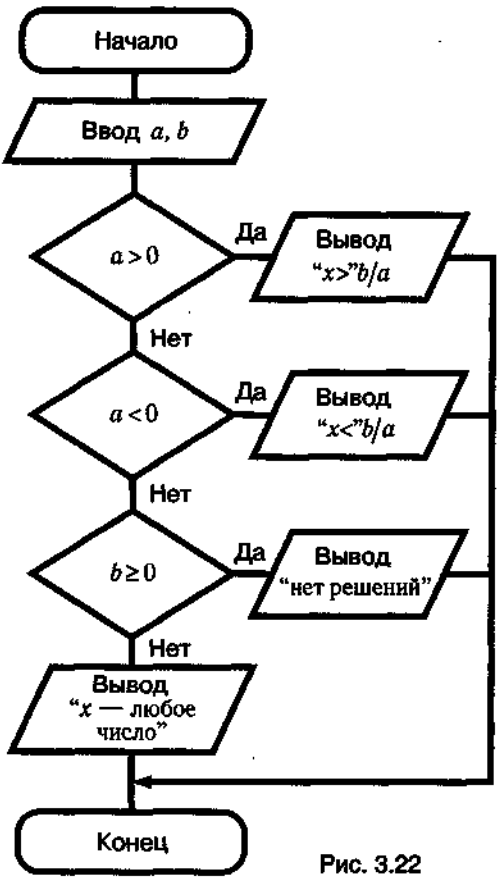


Рис. 3.22

В случае вложенных ветвлений каждое новое ключевое слово `Else` следует считать относящимся к ближайшему неиспользованному `If`. Например,

```
If x>-5 Then
   If x<=6 Then y:=x*x
           Else y:=x*x*x
     Else y:=x;
```

Этот фрагмент позволяет вычислить значение y по формуле:

$$y = \begin{cases} x, & \text{если } x \leqslant -5, \\ x^2, & \text{если } -5 < x \leqslant 6, \\ x^3, & \text{если } x > 6. \end{cases}$$

?

1. **Какие значения могут принимать переменные типа Boolean?**
2. **Какие операции сравнения и логические операции допустимы в языке Паскаль?**
3. **Что может использоваться в качестве условия в условном операторе?**

Упражнение

Напишите программы решения задач:

а) найти лучший результат в забеге на 100 м трех спортсменов;

б) решить квадратное уравнение $ax^2 + bx + c = 0$ $(a \neq 0)$;

*в) заданы три действительных числа. Выяснить, могут ли они оказаться длинами сторон некоторого треугольника;

*г) определить, принадлежит ли точка (x, y) заданной области:

$-3 \leqslant x \leqslant 3$, $y \leqslant x + 3$, $y \leqslant -x + 3$, $y \geqslant -4$ (рис. 3.23, *а*);

$-3 \leqslant x \leqslant 3$, $y \leqslant \frac{2}{3}x + 2$, $y \geqslant -4$ (рис. 3.23, *б*);

$2x - 4 \leqslant y \leqslant 2x + 4$, $-2x - 4 \leqslant y \leqslant -2x + 4$ (рис. 3.23, *в*).

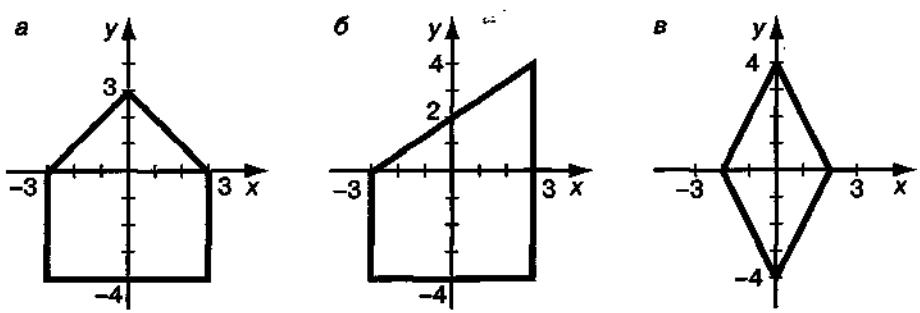


Рис. 3.23

*15.3. Оператор выбора CASE

Оператор выбора (варианта) CASE представляет собой частный случай реализации структуры ВЕТВЛЕНИЕ, когда возникает необходимость выбора одного из нескольких возможных вариантов вычислений в зависимости от значений некоторого выражения (ключа, селектора).

Порядок выполнения оператора CASE следующий: вычисляется значение выражения *K*; полученное значение сравнивается со значениями *K*1, *K*2, ..., *KN*; если оно совпадает с одним из этих значений, то управление передается соответствующему оператору и выполнение оператора CASE завершается. Если значение выражения *K* не совпадает ни с одним из возможных значений, далее все зависит от типа оператора CASE: если он полный (в нем присутствует служебное слово ELSE), то управление передается команде *S*; в противном случае выполнение оператора завершается.

Вариант записи, полученный после зачеркивания строки и блока в приведенной на рисунке 3.24 схеме, соответствует оператору выбора (или, как его еще называют, оператору варианта) в неполной (сокращенной) форме.

```
CASE K OF
   K1: S1;
   K2: S2;
   ...
   KN: SN;
   ELSE S;
END;
```

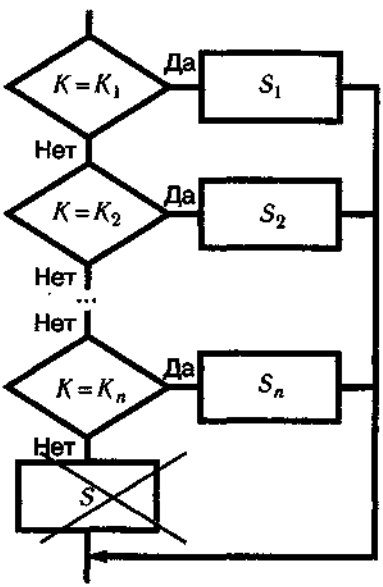


Рис. 3.24

K — выражение, определяющее значение ключа;

*K*1, *K*2, ..., *KN* — возможные значения ключа;

*S*1, *S*2, ..., *SN*, *S* — выполняемые операторы (простые или составные).

Селектор *K* представляет собой выражение порядкового типа. К порядковому типу относятся целочисленный, логический и символьный типы. С последним мы познакомимся в 12-м классе.

В языке Паскаль допускается использование нескольких возможных значений ключей, разделенных запятой (перечисление значений) или двумя точками ".." (диапазон значений); например:

```
Var I: Integer;
   ...
If I=1 Then A:=X+Y;
If I=2 Then A:=X-Y;
If (I>=3) and (I<=5)
   Then  A:=X*Y;
If (I>5) and (I<8)
    Then A:=X/Y;
```

```
Case I of
   1:      A:=X+Y;
   2:      A:=X-Y;
   3..5:   A:=X*Y;
   6,7:    A:=X/Y;
End;
```

Пример 1. Составить программу, которая проверит, делится ли заданное натуральное число на 7.

Пусть x — заданное натуральное число. Составим выражение сравнения: `x mod 7 = 0`. Используем это выражение в качестве селектора; его возможные значения: True, False. Ниже приводятся два варианта использования оператора выбора (в полной и неполной формах):

```
Program PRIM_1;
   Var x : LongInt;
Begin
   Write('Введите натуральное число ');
   Readln(x);
   Case x mod 7 = 0 of
   True: Writeln
      ('делится на 7');
   False: Writeln
      ('не делится на 7');
   End;
End.
```

```
Case x mod 7 = 0 of
   True: Writeln
      ('делится на 7')
   Else Writeln
      ('не делится на 7');
End;
```

Пример 2. Составить программу, которая по номеру дня недели определит его название.

При написании программы на языке Паскаль можно использовать условный оператор IF:

```
If Num=1 Then
Writeln('Понедельник')
   Else if Num=2 Then
   Writeln('Вторник')
      Else if Num=3 Then
      Writeln('Среда')
```

```
Else if Num=4 Then
Writeln('Четверг')
   Else if Num=5 Then
   Writeln('Пятница')
      Else if Num=6 Then
      Writeln('Суббота')
         Else if Num=7 then
         Writeln('Воскресенье')
            Else
            Writeln ('Это не номер дня недели');
```

Использование оператора Case делает программу более компактной:

```
Program Day;
  Var Num: Byte;
Begin
  Writeln('Введите номер дня недели');
  Readln(Num);
  Case Num of
    1: Writeln ('Понедельник');
    2: Writeln ('Вторник');
    3: Writeln ('Среда');
    4: Writeln ('Четверг');
    5: Writeln ('Пятница');
    6: Writeln ('Суббота');
    7: Writeln ('Воскресенье')
    Else Writeln ('Это не номер дня недели');
  End;
End.
```

Пример 3. Составить программу, которая для заданных номеров месяца и года позволит определить количество дней в этом месяце.

Дано: *month, year* — номера месяца и года.
Найти: *kol_days* — количество дней в этом месяце.

$$\textit{Связь: } kol\_days = \begin{cases} 31, \text{ если } month = 1 \text{ или } 3, \text{ или } 5, \text{ или } 7, \text{ или } 8, \text{ или } 10, \text{ или } 12, \\ 30, \text{ если } month = 4 \text{ или } 6, \text{ или } 9, \text{ или } 11, \\ 29, \text{ если } month = 2 \text{ и } year \text{ — високосный}, \\ 28, \text{ если } month = 2 \text{ и } year \text{ — невисокосный}, \end{cases}$$

year — високосный, если year делится без остатка на 4, кроме тех, которые делятся на 100 и не делятся на 400. Например, 1900 год — невисокосный, 2000 год — високосный.

Будем считать, что данные корректны.

```
Program Days;
   Var  Year: Integer;
        month,kol_days: Byte;
        Sto: Boolean;
Begin
   Write('Введите номер месяца ');
   Readln(month);
   Case month of
      1,3,5,7,8,10,12: kol_days:=31;
      4,6,9,11       : kol_days:=30;
      2              : Begin          {февраль}
            Write('Введите номер года ');
            Readln(Year);
            Sto:=(Year mod 100)=0;
            if ((Sto=True) and (Year mod 400 =0)) or
               ((Sto=False) and (Year mod 4=0)) Then
            kol_days:=29
               Else kol_days:=28;
                          End;
   End;
   Writeln(kol_days);
End.
```

1. Какова структура условного оператора? * Оператора выбора?
2. Каков порядок выполнения условного оператора? * Оператора выбора?

Упражнение

Напишите программы решения следующих задач:

а) программа предлагает ввести возраст человека и выводит, к какой группе он относится: дошкольник, ученик, работник, пенсионер;

б) программа предлагает ввести число k полных лет человека (от 1 до 120) и выводит фразу: «Вам k лет». Например, при $k = 3$ фраза будет такой: «Вам три года», при $k = 18$ — «Вам 18 лет», при $k = 101$ — «Вам 101 год»;

в) программа предлагает ввести номер года и выводит в римской системе нумерации век, к которому относится заданный год;

г) программа предлагает ввести текущее время и выводит период суток (день, ночь, утро, вечер);

*д) таблица К. Купера позволяет определить степень физической подготовленности человека в зависимости от суммы очков, набранных им на занятиях по физической культуре за неделю:

Сумма очков, набранных за неделю		Степень физической подготовленности
Юноша	Девушка	
Не менее 75	Не менее 65	Превосходная
51—74	41—64	Отличная
32—50	27—40	Хорошая
21—31	16—26	Удовлетворительная
10—20	8—15	Плохая
Меньше 10	Меньше 8	Очень плохая

Программа предлагает ввести сумму набранных очков, уточняет, кто их набрал (юноша или девушка), и выводит на экран компьютера степень физической подготовленности.

§ 16. Алгоритмическая структура ПОВТОРЕНИЕ

16.1. Цикл с предусловием WHILE

Мы часто встречаем алгоритмы, в которых одно и то же действие или группа действий повторяется до тех пор, пока выполняется некоторое условие. При составлении программ на языке Паскаль повторяющиеся действия можно реализовать с помощью операторов цикла. Одним из них является цикл `While`: `While P do S;`.

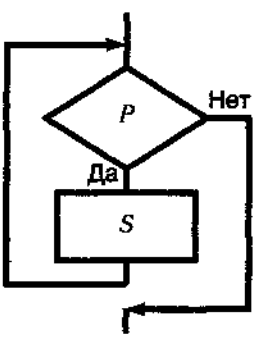


Рис. 3.25

Его принято называть «цикл с предусловием», так как проверка условия выполнения цикла осуществляется *до* выполнения операторов, входящих в состав тела цикла.

В общем виде графически работа оператора представлена на рисунке 3.25. Здесь P — логическое выражение (условие выполнения цикла); S — оператор (простой или составной), который выполняется, если P истинно, — тело цикла.

Использование цикла `WHILE` оправдано, когда количество повторений операторов тела цикла заранее неизвест-

но и зависит от истинности условия P, которое проверяется в начале тела цикла. Это количество может быть равно нулю (если еще до первого попадания в цикл условие P оказалось ложным).

Например, согласно фрагменту программы

```
Readln(K);
While K<=5 do
Begin
   Writeln('Делай ',K);
   K:=K+1;
End;
```

Тело цикла

на экран компьютера при $K = 3$ будет выведено:

Делай 3
Делай 4
Делай 5

При $K = 5$ тело цикла будет выполняться один раз, а при $K = 10$ ($K > 5$) — ни одного раза.

Пример 1. Написать программу нахождения первой (старшей) цифры заданного натурального числа n.

Идея алгоритма основана на последовательном делении на 10 числа n и получаемых частных от деления (рис. 3.26); процесс деления закончится, когда будет получено частное, меньшее 10.

На языке Паскаль программа может выглядеть так:

```
Program ONE;
   Var   n: LongInt;    {заданное число}
         C: LongInt;    {искомая цифра}
Begin
   Write('n='); Readln(n);
   C:=n;
   While C>9 do C:=C div 10;
   Writeln('Первая цифра=',C);
End.
```

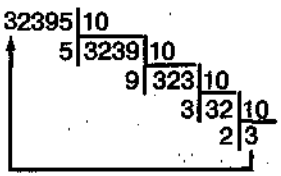


Рис. 3.26

Анализируя программу, можно заметить, что тело цикла может выполняться различное количество раз: ни одного (например, при вводе чисел 2, 8 или другого, не большего 9), 1 (для двузначных чисел), 2 (для трехзначных) и т. д. Использование цикла While в данном случае вполне уместно.

Пример 2. Написать программу вычисления значений функции $y = \frac{1}{x}$ при $x = 1; 1{,}1; 1{,}2; 1{,}3; \ldots; 2$.

Графическое представление алгоритма решения задачи показано в виде блок-схемы на рисунке 3.27.

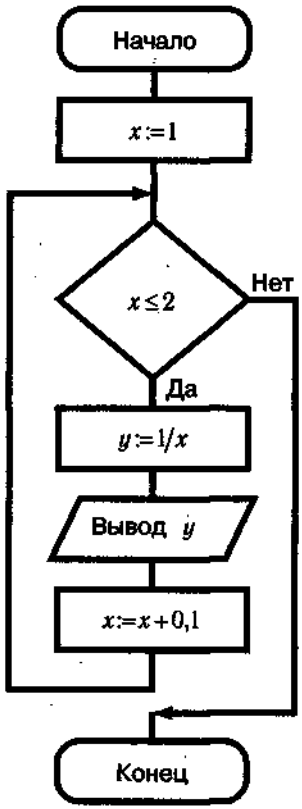


Рис. 3.27

Будем последовательно подставлять в формулу $y = \frac{1}{x}$ вместо x значения 1; 1,1; ...; 2, вычислять соответствующие значения y, каждое полученное значение y выводить на экран.

Программа на языке Паскаль может быть такой:

```
Program FUNCT_1;
  Var x,y: Real;
Begin
  x:=1;
  While x<=2 do
  Begin
    y:=1/x;
    Writeln(y:0:5);
    x:=x+0.1;
  End;
End.
```

Переменной x вначале присваивается значение 1. Так как условие $1 \leqslant 2$ истинно, выполняется тело цикла: вычисляется и выводится на экран значение y, значение переменной x увеличивается на 0,1. Условие $1{,}1 \leqslant 2$ истинно, поэтому выполняется тело цикла для значения $x = 1{,}1$; x становится равным 1,2; и т. д. Последний раз тело цикла выполнится при $x = 2$. Затем значение переменной x становится равным 2,1; условие $2{,}1 \leqslant 2$ ложно, выполнение цикла завершается.

! Обратите внимание на то, что на экран не выводится значение функции в последней точке $x = 2$. Это связано с особенностью представления числовой информации (в частности, вещественных чисел) в памяти компьютера. Для получения требуемого вывода рекомендуется записать в заголовке цикла условие в виде: `While x<=b+h/2 do ...`, где h — дробный шаг изменения значения x (в данном случае условие: `x<=2.05`).

***Пример 3.** Написать программу проверки того, можно ли разделить поровну между тремя первоклассниками имеющиеся конфеты (ребята еще не умеют ни умножать, ни делить).

Пусть n — количество купленных конфет, которые необходимо разделить на 3.

Это можно сделать, если остаток от деления n на 3 равен нулю, и нельзя в противном случае.

Будем считать, что данные корректны: $n > 0$. Так как дети не умеют делить, предложим им раскладывать конфеты в три кучки, добавляя в каждую по очереди по одной конфете (количество конфет при этом будет уменьшаться на 3). Если после очередного этапа раскладывания не останется ни одной конфеты, значит, деление возможно, если кому-то не хватило конфеты — невозможно.

Запишем программу на языке Паскаль. Здесь тело цикла содержит единственную команду n:=n-3:

```
Program VKUSNO;
  Var n: Integer;
Begin
  Write('Число конфет=');
  Readln(n);
  While n>0 do n:=n-3;
  if n=0 Then Write('можно ')
       Else Write('нельзя ');
  Writeln('разделить');
End.
```

Пример 4. Имеется фрагмент программы суммирования квадратов десяти последовательных натуральных чисел (рис. 3.28), в котором Женя допустил ошибки. При запуске программа зависает, и Евгений вынужден прерывать ее работу принудительно (Ctrl + Break). Разобраться в причинах зависания.

```
S:=0; n:=1;
While n<=10 do
   S:=S+k;
Writeln(S);
```

Рис. 3.28

Анализ 1. Зависание программы свидетельствует о наличии так называемого «зацикливания»: мы имеем дело с бесконечным циклом, в котором условие выхода из цикла, в данном случае $n > 10$, не наступает, пока не будет выполнено принудительное завершение работы программы. Проследим изменение значений переменной n в заголовке цикла. Первоначально $n = 1$. Условие 1<=10 истин-

но, поэтому происходит вхождение в цикл и выполняется команда присваивания S:=S+k;. Значение *n* не изменяется, и компьютер постоянно проверяет справедливость истинного условия 1<=10. Итак, первая ошибка связана с неправильной организацией работы счетчика *n*: автор задал начальное и конечное значения, а значение шага не определил.

Устранение ошибки 1. Добавим в тело цикла команду присваивания для увеличения значения *n* на 1: n:=n+1. Это приведет к необходимости организации составного оператора Begin ... End (рис. 3.29). После внесения изменений и запуска программы зацикливание не наблюдается, однако на экран выводится неверный результат (число 0).

```
S:=0; n:=1;
While n<=10 do
Begin
  S:=S+k;
  n:=n+1;
End;
Writeln(S);
```

Рис. 3.29

Анализ 2. Обратим внимание, как осуществляется суммирование. Вначале S = 0 и затем в цикле должна накапливаться сумма путем добавления квадратов чисел 1, 2, ..., 10. Запись команды S:=S+k; говорит о том, что в переменной *S* накапливаются значения переменной *k*, которая не задается и не изменяется. Сами суммируемые числа обозначены в программе переменной *n*, которую и следует записать в теле цикла вместо *k*.

Устранение ошибки 2. Заменим команду S:=S+k; на S:=S+n; (рис. 3.30).

```
S:=0; n:=1;
While n<=10 do
Begin
   S:=S+n;
   n:=n+1;
End;
Writeln (S);
```

Рис. 3.30

Анализ 3. В программе есть еще одна ошибка, связанная с несоответствием алгоритма условию задачи: программа вычисляет сумму десяти последовательных чисел вместо суммы их квадратов.

Устранение ошибки 3. Заменим команду `S:=S+n;` на `S:=S+n*n;` (рис. 3. 31).

```
S:=0; n:=1;
While n<=10 do
Begin
  S:=S+n*n;
  n:=n+1;
End;
Writeln (S);
```

Рис. 3.31

Пример 5. Написать фрагмент программы, который выполняется до тех пор, пока не будет введено нужное число, например *A*, удовлетворяющее условию `0<A<pi/2`.

Этот фрагмент может быть таким:

```
P:=False;        {присвоим логической переменной P значение False: нужное условие не выполнено}
While P=False do{пока не выполнено нужное условие}
Begin            {будем повторять ввод}
  Writeln ('Введите число 0<A<pi/2');
  Readln(A);
  If (A>0) and (A<pi/2) Then
  P:=True;       {условие выполнено}
End;
```

Такую конструкцию удобно использовать, например, для контроля ввода исходных данных.*

Упражнения

1. Определите, сколько раз будет выполняться тело цикла:

а)
```
i:=1;
While i<=50 do
Begin
  Writeln (i);
  i:=i+2;
End;
```

б)
```
k:=12;
While k<=356 do
Begin
  H:=k/5;
  k:=k+3;
End;
```

в)
```
N:=100;
While N>=0 do
Begin
  N:=N-3;
  s:=s+1;
End;
```

2. Напишите программы решения следующих задач:
   а) найти сумму цифр заданного натурального числа;
   б) определить количество цифр заданного натурального числа;

в) вычислить сумму $1+1/2+1/3+ \ldots + 1/n$ (значение n вводится с клавиатуры);
г) вычислить значения функций $y = \sin x$ и $y = \cos x$ на отрезке $[-1; 1]$ с шагом 0,1. Результат вывести в таблицу вида

```
-----------------------------
! x ! sin x ! cos x !
-----------------------------
```

*д) подсчитать количество ковриков квадратной формы, которые понадобятся, чтобы покрыть ими пол комнаты прямоугольной формы (размеры комнаты вводятся с клавиатуры). Например, если размеры комнаты 300 см × 750 см, то понадобится 4 коврика (рис. 3.32).

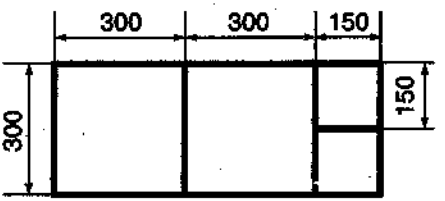


Рис. 3.32

16.2. Цикл с параметром FOR

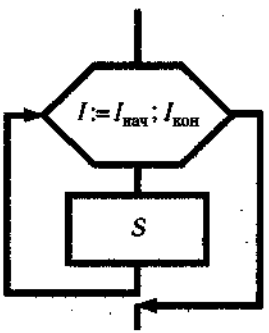


Рис. 3.33

Для организации циклических действий в программах можно использовать оператор FOR.

Использование цикла FOR оправдано, когда количество повторений цикла заранее (т. е. еще до начала выполнения цикла) известно.

В общем виде графически работа оператора представлена на рисунке 3.33.

Форма записи этого оператора такова:

FOR I:=$I_{нач}$ TO $I_{кон}$ do S;

или

FOR I:=$I_{нач}$ DOWNTO $I_{кон}$ do S;

где I — идентификатор (параметр цикла), счетчик количества повторений цикла;

$I_{нач}$ — выражение, позволяющее определить начальное (первое) значение I;

$I_{кон}$ — выражение для определения последнего (конечного) значения I;

TO — служебное слово, определяющее шаг изменения параметра цикла I равным 1;

DOWNTO — служебное слово, определяющее шаг изменения параметра цикла I равным −1;

S — оператор (простой или составной) — тело цикла;

:= — составной символ (имеет смысл «от»).

Переменная I должна быть порядкового типа; $I_{нач}$, $I_{кон}$ — выражения, совместимые по типу с переменной I.

FOR — оператор цикла с параметром (шагом). Здесь тело цикла повторяется для всех значений параметра I, определенного в границах [$I_{нач}$; $I_{кон}$] (отсюда и название цикла). Порядок его выполнения: вычисляются значения выражений $I_{нач}$ и $I_{кон}$; параметр цикла принимает последовательно все значения от начального до конечного; для каждого из них выполняется тело цикла.

Служебное слово TO в записи оператора цикла предполагает, что $I_{нач} \leqslant I_{кон}$, при этом значение параметра цикла на каждом шаге увеличивается на 1. Служебное слово DOWNTO предполагает, что $I_{нач} \geqslant I_{кон}$ и значение параметра уменьшается на 1. При нарушении этих условий цикл выполняться не будет.

После выхода из цикла значение параметра цикла не всегда определено, что следует учитывать при разработке циклических программ.

Пример 1. Смоделировать работу сберегательного банка. Известно, что банк ежемесячно начисляет P % на сумму вклада, причем начиная со второго месяца проценты начисляются на накопленную сумму с учетом процентов. Написать программу вычисления денежной суммы, которая окажется на счету по истечении года (первоначальная сумма вводится с клавиатуры). Считать, что данные корректны.

Дано: S — первоначальная сумма,

P — ежемесячная процентная ставка.

Найти: S_1, S_2, ..., S_{12} — сумма на счету в конце 1-го, 2-го, ..., 12-го месяцев.

Связь: $S_n = S_{n-1} \cdot \left(1 + \frac{P}{100}\right)$, $S_0 = S$, где $n = 1, 2, 3, ..., 12$.

При: $S > 0$; $3 < P < 50$. Будем считать, что данные корректны.

Метод: Первоначально подставим в правую часть формулы значение S и найдем S_1 (сумму по истечении одного месяца). При подстановке S_1 в правую часть формулы найдем значение S_2 (сумму по истечении двух месяцев) и т. д. Такие действия проделаем 12 раз. Для подсчета количества прошедших месяцев введем переменную g.

Анализ алгоритма свидетельствует о возможности его реализации с помощью цикла FOR (в алгоритме присутствует целочисленная переменная g, для которой известны начальное значение 1, конечное — 12, шаг изменения 1).

Графическое представление алгоритма решения задачи показано на рисунке 3.34.

Программа на языке Паскаль может быть такой:

```
Program Pribyl;
  Var S: Real;
      P,g: Integer;
```

```
Begin
   Write('Начальная сумма=');
   Readln(S);
   Write('Ставка P(%)=');
   Readln(P);
   For g:=1 To 12 do
   Begin
      S:=S*(1+P/100);
      Writeln(g,':',S:0:2);
   End;
End.
```

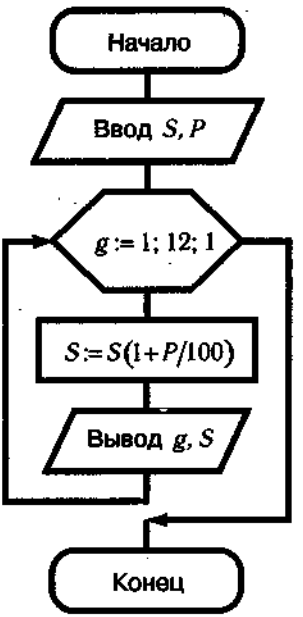


Рис. 3.34

Пример 2. Составить программу отсчета времени, оставшегося до начала спектакля: программа должна предложить ввести оставшееся время (в секундах) и выводить на экран отсчет времени в обратном порядке. Когда отсчет окажется равным нулю, программа должна вывести текст «Начало спектакля. Вход воспрещен!!!».

Программа на языке Паскаль может быть такой:

```
Program Nachalo;
   Uses CRT;
   Var T,S: Integer;
Begin
   Write ('Время до начала (с)=');
   Readln(T);
   For S:=T downto 0 do
   Begin
      ClrScr; GotoXY(40,10);
      Write(S); Delay(1000);
   End;
   GotoXY(35,15); TextColor(4);
   Writeln ('Начало спектакля.');
   Writeln ('Вход воспрещен!!!');
End.
```

Пример 3. Известен возраст каждого конкурсанта фестиваля «Славянский базар». Составить программу, определяющую средний возраст конкурсантов.

Пусть возраст конкурсантов задается как число полных лет. Введем обозначения: K — общее количество конкурсантов, V — возраст очередного конкурсанта, S — суммарный возраст, SV — средний возраст конкурсантов. Тогда $SV = \frac{S}{K}$.

Найдем вначале суммарный возраст, выполняя постепенное суммирование с переменной S значения возраста V очередного конкурсанта. Первоначально $S = 0$. Переменная S играет роль сумматора, куда по очереди добавляются значения V.

Получим программу:

```
Program FEST;
   Var  K,V,i: Integer;
        S: LongInt;
        SV: Real;
Begin
   Write ('Сколько конкурсантов=');
   Readln(K);
   S:=0;                  {Пока S=0}
   For i:=1 to K do       {Для каждого конкурсанта}
   Begin
      Readln(V);          {вводим возраст}
     S:=S+V;              {и суммируем его}
   End;
   SV:=S/K;
   Writeln('SV=',SV:0:1);
End.
```

При программировании можно выбирать различные варианты алгоритмических структур. Например, вычисление значений функции $y = \sin x$ на отрезке [0; 2] с шагом 0,1 можно задать такими вариантами записи операторов:

```
1)x:=0; while x<=2.05 do begin y:=sin(x); writeln
(x:0:1,y:5:1); x:=x+0.1; end;
2)x:=0; for k:=1 to 21 do begin y:=sin(x); writeln
(x:0:1,y:5:1); x:=x+0.1; end;
3)a:=0.1; for k:=0 to 20 do begin x:=a*k; y:=sin(x);
writeln (x:0:1,y:5:1); end;
```

Упражнение

Напишите программы решения следующих задач:

а) вывести на экран квадраты натуральных чисел от 1 до 50;

б) определить среднее значение температуры воздуха в некоторой местности за n дней;

*в) вывести на экран монитора значения выражения 2^n, где $n = 1, 2, 3, 4, ..., 17$;

г) известны результаты соревнования по прыжкам в высоту n учеников. Найдите суммарный результат.

16.3. Смешанные алгоритмы

Алгоритм, содержащий алгоритмические структуры ВЕТВЛЕНИЕ и ЦИКЛ, называют **смешанным**. В языке Паскаль тело цикла может содержать любые операторы (как простые, так и структурные). При программировании таких алгоритмов следует следить за сохранением вложенности алгоритмических структур.

***Пример** 1. Настя выбрала в магазине открытки двух типов: по цене *M* р. для поздравления мальчиков и по цене *D* р. для поздравления девочек. Всего у Насти *R* р. Составить программу, которая позволит получить все возможные варианты покупок, где будут истрачены все имеющиеся у Насти деньги.

Дано: *R* (р.) — общая денежная сумма;

M (р.), *D* (р.) — стоимости открыток для мальчиков и девочек.

Найти: *KM*, *KD* — количество открыток для мальчиков и девочек соответственно.

Связь: `M*KM+D*KD=R`.

Метод: Будем перебирать значения *KM* от 0 (наименьшее количество открыток; все деньги тратятся на открытки для девочек) до $\left[\frac{R}{M}\right]$ (наибольшее количество открыток; все деньги тратятся на покупку открыток для мальчиков). Тогда `KD=(R-M*KM)/D`. Если *KD* — целое число, получен вариант покупки.

Так как Настя на сумму *R* может не купить ни одной открытки, введем логическую переменную `Yes`, которой присвоим значение `True`, если покупка возможна.

Получим программу:

```
Program New_Year;
  Var  M,D,R,KM: Integer;
       KD: Real;
       Yes: Boolean;
Begin
  Write('Всего денег='); Readln(R);
  Writeln('Стоимости открыток:');
  Write('для мальчиков '); Readln(M);
  Write('для девочек '); Readln(D);
  Yes:=False;
  For KM:=0 To R div M do
  Begin
    KD:=(R-M*KM)/D;
    If KD=Trunc(KD) Then
    Begin
```

```
        Yes:=True;
        Writeln(KM,', ', Trunc(KD));
      End;
   End;
   If Yes=False Then
      Writeln ('Покупка невозможна');
End.*
```

Пример 2. Написать программу нахождения среди трехзначных чисел таких чисел, сумма старшей и младшей цифры каждого из которых равна вводимому с клавиатуры числу.

Пусть X — заданное число. Трехзначное число C удовлетворяет условию $100 \leqslant C \leqslant 999$.

Будем перебирать все числа, удовлетворяющие этому условию, т. е. искать сумму $m + s$, где m — младшая цифра, равная остатку от деления C на 10, s — старшая цифра, равная целой части от деления C на 100. Если $m + s = X$, выведем число C на экран.

Логическая переменная Yes предназначена для отслеживания случая отсутствия искомых чисел.

Получим программу:

```
Program _3;
   Var  C,X,m,s: Integer;
        Yes: Boolean;
Begin
   Write('Заданное число=');
   Readln(X);
   Yes:=False;
   For C:=100 To 999 do
   Begin
      m:=C mod 10;
      s:=C div 100;
      If m+s=X Then
      Begin
         Yes:=True; Writeln(C);
      End;
   End;
   If Yes=False Then
      Writeln ('Таких чисел нет');
End.
```

Для решения задач часто бывает необходимо организовать перебор нескольких значений переменных, что приводит к так называемым *вложенным циклам*. При этом каждый внутренний цикл должен быть полностью вложен во все внешние по отношению к нему циклы (рис. 3.35).

```
For i:=1 to 5 do  Begin...
   For j:=1 to 9 do Begin...
      For k:=1 to 12 do Begin...
                        End;...
                  End;...
            End;
```

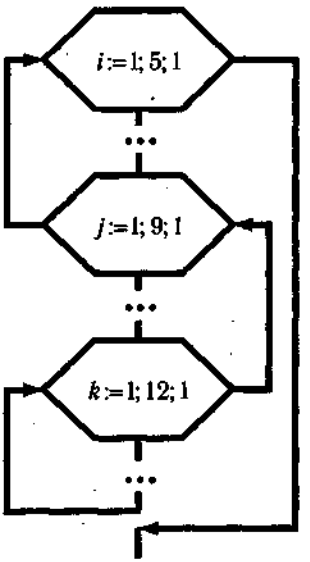


Рис. 3.35

Пример 3. Составить программу вывода таблицы умножения каждого натурального числа от 1 до 9 на 2, 3, 4, ..., 9 (таблицы Пифагора) (рис. 3.36).

Программа на языке Паскаль может выглядеть следующим образом:

```
Program Tablica;
   Var x,y: byte;
Begin
   For x:=1 to 9 do
            {1-й сомножитель}
   Begin
      For y:=1 to 9 do
            {2-й сомножитель}
      Write(x*y:3);
            {вывод x*y}
      Writeln;
            {переход на новую строку}
   End;
End.
```

вложенный цикл

внешний цикл

1	2	3	4	5	6	7	8	9
2	4	6	8	10	12	14	16	18
3	6	9	12	15	18	21	24	27
...	...	...	...	...	...	...	...	...
9	18	27	36	45	54	63	72	81

Рис. 3.36

Пример 4. Написать программу подсчета количества двузначных чисел, которые делятся на сумму своих цифр.

Будем рассматривать искомые числа в виде совокупности цифр $\overline{ab}$ и, перебирая их различные возможные значения, формировать число $x = 10a + b$, а также сумму цифр $S = a + b$. Количество искомых чисел обозначим K.

Графическое представление алгоритма решения задачи показано в виде блок-схемы на рисунке 3.37.

Получим программу:

```
Program Liki;
   Var  a,b,x,S: Byte;
        K: Integer;
Begin
   K:=0;
        {a – старшая цифра,
        b – младшая цифра}
   For a:=1 to 9 do
      For b:=0 to 9 do
      Begin
         x:=10*a+b;
         {x – двузначное число}
         S:=a+b;
         {S – сумма цифр числа}
         If x mod S=0 then
         Begin
            {если число x делится на
            сумму цифр, считаем его}
            K := K+1;
            {и выводим на экран}
            Write (x,' ');
         End;
      End;
      {перевод курсора на новую строку}
      Writeln;
      {вывод общего количества чисел}
      Writeln('Всего чисел: ',K);
End.
```

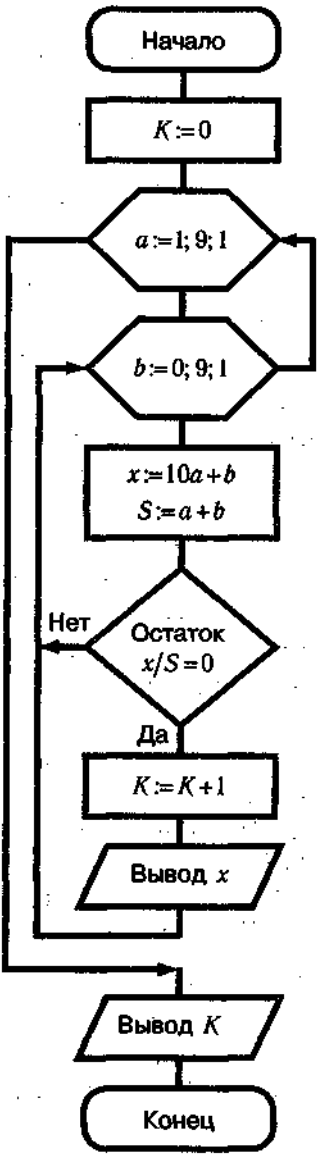


Рис. 3.37

***Пример 5.** Написать программу решения задачи: «Хозяйство должно разлить весь собранный березовый сок в банки объемом 3 л, 5 л, 10 л. Определить, можно ли это сделать и сколько банок каждого вида понадобится (считать, что количество банок неограниченно). Получить вариант разлива с наименьшим количеством использованных банок».

Дано: L (л) — объем собранного сока.

Найти: a, b, c — количество банок соответственно объемом 3 л, 5 л, 10 л.

Связь: $\begin{cases} 3a + 5b + 10c = L, \\ a + b + c \to \min. \end{cases}$

Разлить весь сок в банки полностью нельзя, если данное уравнение не имеет решений в целых неотрицательных числах.

При: L — целое число.

Метод:

1. Для получения варианта разлива сока в десятилитровые банки будем перебирать значения переменной z от минимального 0 до максимального $\left[\frac{L}{10}\right]$. Останутся не разлитыми $P = L - 10z$ (л) сока, которые следует разлить в банки объемом 5 л; для этого будем перебирать значения переменной y от 0 до $\left[\frac{P}{5}\right]$. Для разлива в трехлитровые банки останется $x = P - 5y$ (л) сока. Пусть $t = \frac{x}{3}$. Если t — целое число, одно из решений задачи получено; его можно вывести на экран компьютера. Факт получения решения будем фиксировать логической переменной `Yes`, которой присвоим значение `True`. По значению `Yes` при завершении работы программы будем судить, требуется ли вывести сообщение «Разлив невозможен».

2. Для получения наилучшего варианта разлива будем сравнивать общее количество использованных банок $K = t + y + z$ со значением некоторой переменной *min*. Эту переменную будем использовать для хранения наименьшего значения суммы: $min = K$, если справедливо неравенство: $K < min$. Одновременно запомним в переменных a, b, c соответствующие значения t, y, z. Первоначально будем полагать значение *min* равным достаточно большому целому числу (*MaxInt*).

Получим следующую программу:

```
Program Rasliv;
  Var L,y,z,P,min,K,t,a,b,c,x: Integer;
      x3: Real;
      Yes: Boolean;
Begin
  Write('Всего сока=');
  Readln(L);
```

```
   Yes:=False;
   min:=MaxInt;{Max Int=32767}
   For z:=0 To L div 10 do
   Begin
      P:=L-10*z;
      For y:=0 To P div 5 do
      Begin
         x:=P-5*y; x3:=x/3; t:=Trunc(x3);
         If x3=t Then
         Begin
            Yes:=True;
            Writeln(t,',',y,',',z);
            K:=t+y+z;
            If K<min Then
            Begin
               min:=K; a:=t;
               b:=y; c:=z;
            End;
         End;
      End;
   End; Writeln;
   If Yes=False Then
      Writeln('Разлив невозможен')
         Else Writeln(a,',',b,',',c);
End.*
```

?

1. Что такое «условный оператор» и как он применяется?
2. В каких случаях желательно использовать цикл For? While?

Упражнения

1. Напишите программу проверки, является ли четной первая цифра заданного натурального числа.

*2. Напишите программу, проверяющую, хорошо ли пользователь умеет складывать в уме числа. Программа должна 10 раз запросить два числа (слагаемых); вывести их на экран; предложить ввести сумму этих чисел; проверить, правильна ли эта сумма; вывести на экран компьютера соответствующее сообщение («Правильно», «Неправильно»); вывести количество правильных и неправильных ответов.

3. Напишите программу, которая среди двузначных чисел найдет все числа, делящиеся на число, вводимое с клавиатуры.

4. Пусть натуральное число является «счастливым», если оно делится на 7. Получите на заданном числовом промежутке $[A; B]$ все «счастливые» натуральные числа (значения A и B вводятся с клавиатуры).

*5. Известны результаты соревнования по прыжкам в высоту n учеников. Найдите самый лучший результат.

§ 17. Использование алгоритмических структур в графике

17.1. Графические построения

При построении графических изображений мы использовали только одну из известных нам алгоритмических структур — СЛЕДОВАНИЕ. Освоение других структур позволяет использовать их для получения сложных графических изображений.

Рассматривая некоторое изображение в виде совокупности одноименных объектов (точек, отрезков и т. д.), можно прийти к пониманию возможности использования алгоритмической структуры ПОВТОРЕНИЕ для получения всего изображения.

Пример 1. Написать программу получения изображения горизонтального отрезка с началом в точке (0, 240).

Будем рассматривать отрезок как совокупность точек и использовать для получения его изображения процедуру `PutPixel`. Поместим ее в тело цикла с параметром `For`. Таким образом, получим изображение 640 точек с координатами (k, 240) при $k = 0, 1, 2, 3, ..., 639$; в результате соседние пиксели сольются в горизонтальный отрезок.

Получим следующую программу:

```
Program PUT_LINE;
  Uses Graph;
  Var a,b,k: Integer;
Begin
  a:=Detect; InitGraph(a,b,'');
  {горизонтальная прямая}
  For k:=0 To 639 do
    PutPixel(k,240,14);
          {14 — желтый цвет}
  Readln; CloseGraph;
End.
```

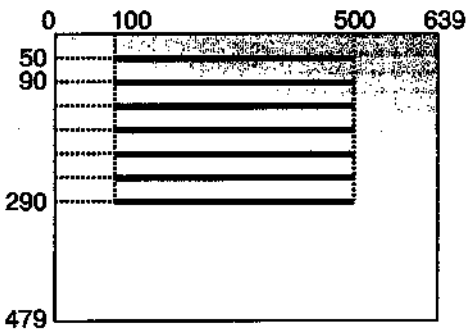


Рис. 3.38

Пример 2. Написать программу получения изображения семи параллельных отрезков равной длины (рис. 3.38).

Требуемое изображение можно рассматривать как совокупность отрезков. Концы отрезков: (100, y), (500, y), где $y = 50, 90, ..., 290$, значение y изменяется с шагом 40. Таким образом, можно организовать цикл While с условием выполнения цикла $y \leqslant 290$. В цикле будем использовать процедуру вывода отрезка прямой и оператор изменения значения y на 40 (величину шага). Эта идея реализована в программе LINE7_1.

```
Program LINE7_1;
   Uses Graph;
   Var a,b,y: Integer;
Begin
   a:=Detect;
   InitGraph (a,b,'');
   y:=50;
   While y<=290 Do
   Begin
      Line(100,y,500,y);
      y:=y+40;
   End;
   Readln; CloseGraph;
End.
```

При решении этой задачи можно также использовать цикл с параметром For, где параметр цикла (например, переменная k) указывает количество полученных изображений отрезка: $k = 1, 2, ..., 7$. Тело цикла имеет такой же вид, как и в программе LINE7_1. Данная идея реализована в программе LINE7_2.

```
Program LINE7_2;
   Uses Graph;
   Var a,b,k,y: Integer;
Begin
   a:=Detect; InitGraph(a,b,'');
   y:=50;
   For k:=1 to 7 do
   Begin
      Line(100,y,500,y);
      y:=y+40;
   End;
   Readln; CloseGraph;
End.
```

Использование счетчика случайных чисел позволяет получать случайно расположенные на экране изображения. Покажем это на примере.

Пример 3. Составить программу вывода на экран заданного количества случайно расположенных точек. Программа должна:

1) предложить ввести количество требуемых точек (N);

2) случайным образом сформировать координаты точки $0 \leqslant x \leqslant 639$, $0 \leqslant y \leqslant 479$ и цвет точки $1 \leqslant C \leqslant 15$;

3) изобразить точку на экране, замедляя на миллисекунды процесс перехода к получению изображения следующей точки (регулировать скорость рисования).

Переменная k — параметр цикла (счетчик), целочисленная переменная, которая используется для подсчета количества выводимых точек.

Программа на языке Паскаль может выглядеть следующим образом:

```
Program STARS; {Звездное небо}
  Uses Crt,Graph;
  Var a,b,N,k,x,y,C:Integer;
Begin
  ClrScr;
  Write ('Количество точек=');
  Readln(N);
  Randomize;
  a:=Detect; InitGraph(a,b,'');
  For k:=1 To N do
  Begin
    x:=Random(640); y:=Random(480);
    C:=Random(15)+1; PutPixel(x,y,C);
    Delay(1000);
  End;
  Readln; CloseGraph;
End.
```

Пример 4. Написать программу закраски прямоугольника различными цветами: смена цвета должна осуществляться после нажатия клавиши Enter.

Пример основан на выполнении многократных действий: получение изображения прямоугольника, заполненного некоторым цветом.

Пусть i — номер цвета заполнения; $i = 0, 1, 2, ..., 15$. Таким образом, уместно использование цикла `For`. В цикле следует задать стиль заполнения области (1 — сплошной текущий цвет) и изобразить закрашенный прямоугольник, например, с диагональными точками (220, 200), (450, 300).

Получим программу:

```
Program BAR_16;
   Uses Graph;
   Var a,b,i: Integer;
Begin
   a:=Detect;
   InitGraph(a,b,'');
   For i:=0 To 15 do
   Begin
      SetFillStyle(1,i);
      Bar(220,200,450,300);
      Readln;
   End;
   CloseGraph;
End.
```

Возможность выбора того или иного изображения для построения, представленная в алгоритме (и соответственно в программе) в виде некоторой вариативности, обеспечивается использованием алгоритмической структуры ВЕТВЛЕНИЕ.

Пример 5. Написать программу, которая по желанию пользователя выведет на экран изображение треугольника или прямоугольника.

Пусть P — переменная, по значению которой пользователь будет принимать решение о выводе требуемого изображения: треугольника (при $P = 1$) или прямоугольника (при $P \neq 1$).

Будем получать изображение треугольника с вершинами в точках (240, 70), (400, 370), (180, 300) и изображение прямоугольника с вершинами в диагональных точках (240, 70) и (480, 300) (рис. 3.39).

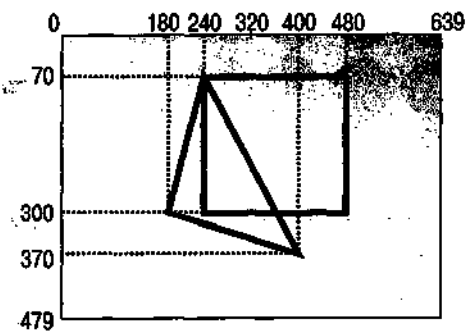


Рис. 3.39

На языке Паскаль получим такую программу:

```
Program TRE_PR;
   Uses Graph;
   Var a,b: Integer;P: Byte;
Begin
   Writeln('Сделайте выбор:');
   Writeln('1 — треугольник');
```

```
   Writeln
      ('не 1 - прямоугольник');
   Readln(P);
   a:=Detect;
   InitGraph(a,b,'');
   If P=1 Then
   Begin
      Line(240,70,400,370);
      Line(400,370,180,300);
      Line(180,300,240,70);
   End
      Else Rectangle (240,70,480,300);
   Readln; CloseGraph;
End.
```

Упражнения

1. Напишите программы получения изображений:

   а) десяти параллельных вертикальных отрезков одинаковой длины;

   *б) случайного количества случайно расположенных окружностей (треугольников, отрезков).

2. Напишите программу закраски произвольного треугольника различными цветами (смена цвета должна осуществляться после нажатия клавиши Enter).

*3. Напишите программу, которая по желанию пользователя выведет на экран изображение треугольника (равнобедренного, равностороннего или разностороннего), квадрата, прямоугольника, параллелограмма или ромба.

17.2. Программирование движущихся объектов

Рассмотрим два способа решения задач моделирования движений средствами языка программирования:

1. «Нарисуй — сотри — и нарисуй в новом месте». Этот способ реализовать легко: нужно организовать цикл, задав координаты для перемещения объекта в требуемом направлении; в теле цикла нужно предусмотреть очистку экрана, изображение объекта в новом месте и приостановку выполнения программы.

2. «Нарисуй — закрась цветом фона — и нарисуй в новом месте». Здесь объект прорисовывается на старом месте дважды: вначале требуемым цветом, а затем цветом фона.

Пример 1. Написать программу перемещения красного квадрата по черному экрану слева направо.

Способ 1

```
Program DVIG_1;
   Uses Crt, Graph;
   Var a,b,x: Integer;
Begin
   a:=Detect;
   InitGraph(a,b,'');
   For x:=0 to 639 do
   Begin
      SetFillStyle (1,4);     {красный}
      Bar(x,200,x+10,210);
      Delay (1000);
      ClearDevice;            {очистка экрана}
   End;
   Readln;
   CloseGraph;
End.
```

Способ 2

```
Program DVIG_2;
   Uses Crt, Graph;
   Var a,b,x: Integer;
Begin
   a:=Detect;
   InitGraph(a,b,'');
   For x:=0 to 639 do
   Begin
      SetFillStyle(1,4);      {красный}
      Bar(x,200,x+10,210);
      Delay (1000);
      SetFillStyle(1,0);      {черный}
      Bar(x,200,x+10,210);
   End;
   Readln; CloseGraph;
End.
```

*Если завершение перемещения объекта нужно связать с действиями пользователя (например, с нажатием на клавишу), удобно использовать функцию **KeyPressed** модуля Crt. Функция возвращает значение TRUE, если нажата какая-либо клавиша: While not KeyPressed Do ...; (пока не нажата клавиша, делать ...).

Пример 2. Написать программу вертикального перемещения шарика (от верхней границы экрана к нижней и обратно и т. д.), пока не будет нажата любая клавиша.

Пока не нажата какая-либо клавиша, программа должна многократно выполнять следующие действия: рисовать красный круг, закрашивать его на старом месте цветом фона (черным), ординату y (центра круга) увеличивать на $dy = 1$, рисовать новый круг и т. д. Когда круг коснется нижней или верхней границы экрана, значение dy изменится на противоположное и круг изменит направление движения.

После нажатия любой клавиши работа программы будет завершена.

Программа на языке Паскаль может выглядеть следующим образом:

```
Program PADENIE;
   Uses Crt, Graph;
   Const R=10; x=320;
   Var a,b,y,dy: Integer;
Begin
   a:=Detect;
   InitGraph(a,b,'');
   y:=R; dy:=1;
   While not KeyPressed do
   Begin
      SetFillStyle(1,4);          {красный}
      PieSlice(x,y,0,360,R);
      Delay(100);
      SetFillStyle(1,0);            {черный}
      PieSlice(x,y,0,360,R);
      y:=y+dy;
      If (y=640-R) or (y=R) Then dy:=-dy;
   End;
   CloseGraph;
End.*
```

Упражнения

1. Испытайте программы из п. 17.2, выберите подходящие цвета, скорости перемещения. Обратите внимание, что программа PADENIE моделирует движение шарика в туннеле белого цвета. Измените программу для перемещения шарика в однородной среде (например, на синем фоне).
2. Примените наиболее понятный Вам метод для моделирования:

   а) горизонтального перемещения отрезка;

   б) вертикального падения теннисного мячика.

*17.3. Решение практических задач. Деловая графика

С целью автоматизации построения различных видов графиков и диаграмм, которые представляют интерес для коммерческих пользователей, разрабатываются системы **деловой графики**, представляющие собой технологию создания изображений с сопровождающим их текстом. Программы деловой графики могут входить в интегрированные программные пакеты (например, в Microsoft Excel), а могут использоваться и самостоятельно (PowerPoint, пакет «Галактика»).

Рассмотрим примеры решения практических задач с использованием графических возможностей языка программирования Паскаль.

Пример 1. Магазин канцелярских товаров за день продал известное количество тетрадей, карандашей и ручек. Построить круговую диаграмму, позволяющую определить, какой товар пользовался в течение дня наибольшим спросом.

Пусть T, K, R — количество проданных тетрадей, карандашей и ручек соответственно. Суммарное количество проданных товаров составляет $S = T + K + R$ единиц товара. Тогда на круге, который составляет 360 градусов, на одну единицу товара приходится $g = \frac{360}{S}$ градусов, на проданные тетради — $gT = g*T$ градусов, на карандаши — $gK = g*K$ градусов, на ручки — остальные. Таким образом, для построения диаграммы нужно разбить круг на три сектора в соответствии с произведенными расчетами. R_o — радиус окружности, на которой будем выполнять построения.

Получим следующую программу:

```
Program MAGAZIN;
  Uses Crt, Graph;
  Const Ro=100;                          {радиус окружности}
  Var   a,b,T,K,R,S,gT,gK: Integer;
        g: Real;
Begin
```

```
  ClrScr; GotoXY(35,5);
  Writeln ('Сколько продано тетрадей, карандашей, ручек?');
  GotoXY(45,6); Readln(T,K,R);
  S:=T+K+R; g:=360/S;
  gT:=Round(G*T); gK:=Round(G*K);
  a:=Detect; InitGraph(a,b,'');
  SetFillStyle(1,14);                    {желтый}
  PieSlice(300,200,0,gT,Ro);
  Bar(450,150,500,200);
  OutTextXY(510,170,'тетради');
  SetFillStyle(1,2);                     {зеленый}
  PieSlice(300,200,gT,gT+gK,Ro);
  Bar(450,230,500,280);
  OutTextXY(510,250,'карандаши');
  SetFillStyle(1,4);                     {красный}
  PieSlice(300,200,gT+gK,360,Ro);
  Bar(450,310,500,360);
  OutTextXY(510,330,'ручки');
  Readln; CloseGraph;
End.
```

Пример 2. Написать программу построения диаграмм роста вкладов с простыми и сложными процентами. Программа должна запросить начальную денежную сумму и вывести диаграммы.

При начислении простых процентов будем пользоваться формулой

$$S_n = S_{n-1} + S_0 \frac{p}{100},$$

а при начислении сложных процентов — формулой

$$S_n = S_{n-1} \cdot \left(1 + \frac{p}{100}\right),$$

где S_0 — первоначальная сумма, p — ежемесячная процентная ставка.

Будем изображать результаты расчетов в виде столбчатых диаграмм — прямоугольников (для простых процентов — желтого цвета, а для сложных — зеленого цвета). В программе будем использовать одинаковую процентную ставку P % по каждому виду вклада и проследим перспективу роста вклада на один год (12 месяцев).

Программа будет такой:

```
Program Pribil;
   Uses Crt, Graph;
   ConstK=12;                  {12 месяцев}
        P=30;                  {30%}
   Var  S,S0,SK: Real;
        h,m,a,b: Integer;
Begin
   ClrScr; GotoXY(10,3);
   Write ('Начальная сумма(>=20000)=');
   GotoXY(40,3);
   Readln(S0);
   If S0<=100000 Then
      S0:=int(S0/10000)
        Else S0:=int(S0/100000);
                               {масштаб}
   S:=S0; SK:=S0;
   a:=Detect; InitGraph(a,b,'');
   ClearDevice;
   Line(0,400,640,400); h:=10;
   For m:=1 to K do            {m-номер месяца}
   Begin
      S:=S+S0*P/100; h:=h+10;
      SetFillStyle(1,14);      {желтый}
      Bar(h,400-Round(S),h+10,400);
      SK:=SK*(1+P/100); h:=h+10;
      SetFillStyle(1,2);       {зеленый}
      Bar(h,400-Round(SK),h+10,400);
   End;
   Readln;
   CloseGraph;
End.
```

На рисунке 3.40 приводятся результаты работы программы Pribil для

$$S = 1\,000\,000 \text{ р.}$$

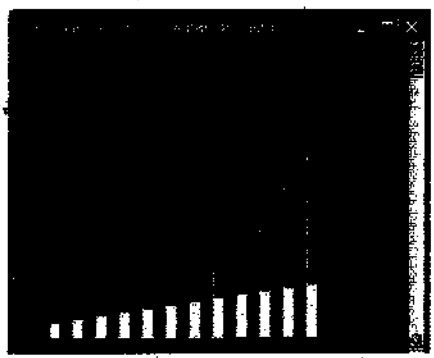


Рис. 3.40

При построении изображений на графическом экране возникают проблемы, связанные с его ограниченными размерами и

ориентацией осей, отличающейся от декартовой системы координат. Это требует масштабирования изображений и перерасчета реальных координат в координаты графического экрана.

Пример 3. Написать программу построения графика функции $y = ax + b$.
Решение этой задачи распадается на следующие этапы:

- получение изображения системы координат на экране (рис. 3.41);
- выбор на осях единиц масштаба и градуирование осей с указанием предельных (наибольшего и наименьшего) значений аргумента и результата;
- вычисление значений функции y на отрезке $[x_{min}; x_{max}]$ и получение координат точек графика (x, y);
- преобразование точек графика (x, y) в систему координат экрана (x_1, y_1);
- вывод точек графика на экран.

Этап 1. 1) Пусть точка экрана (320, 240) соответствует началу системы координат. Тогда для получения изображения оси абсцисс достаточно провести отрезок через точки (0, 240) и (639, 240), оси ординат — через точки (320, 0) и (320, 479).

2) Укажем направление оси абсцисс *OX* путем соединения точки (639, 240) и точек (635, 235) и (635, 245); направление оси ординат *OY* укажем, соединив точку (320, 0) с точками (315, 5) и (325, 5).

3) Укажем начало координат путем вывода текста «O» в точку (300, 250), подпишем оси координат: «X» в точке (625, 225) и «Y» в точке (330, 5).

Этап 2. Выберем масштаб по оси *OX* равным восьми точкам (пикселям) экрана. Так как общее количество точек экрана по оси *OX* равно 640, то мы можем пометить на этой оси 80 точек: будем делать это с помощью вертикальных отрезков длиной 4 пикселя. Предельные значения аргумента обозначим путем вывода на экран текста: «−39» — возле крайней точки слева вдоль оси *OX* и «39» — возле крайней точки справа. Таким образом, $x_{min} = -39$, $x_{max} = 39$.

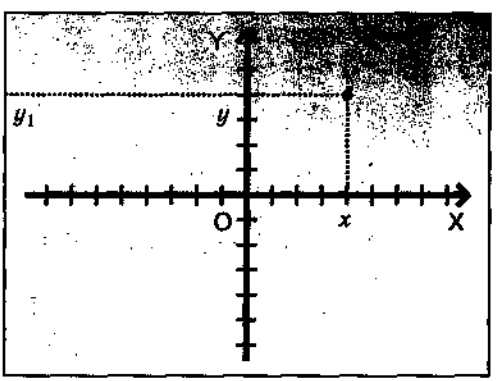


Рис. 3.41

Масштаб по оси *OY* примем равным шести пикселям. Пометим 80 точек на этой оси горизонтальными отрезками длиной 6 пикселей. Выделим путем вывода текста «−39» и «39» граничные точки, соответствующие наименьшему и наибольшему значениям функции в области экрана.

Этап 3. Будем перебирать значения аргумента x_2 от −39 до 39 с шагом 0,1, вычислять значения функции $y_2 = ax_2 + b$.

Этап 4. Полученные координаты точки графика функции (x_2, y_2) следует преобразовать в координаты точки (x_1, y_1) относительно координат экрана, смещая значения x_2 и y_2 относительно начала координат (320, 240) с учетом масштаба по каждой оси:

```
x1=x2*8+320;  y1=240-6*y2.
```

При реализации описанного алгоритма на языке Паскаль следует выполнить правильный выбор типов данных. Так, например, переменные x2, y2 — вещественного типа, x1, y1 — целого типа.

Получим программу:

```
Program LINE_AXB;
  Uses Graph;
  Var  m,n,X,Y,x1,y1: Integer;
       x2,y2,a,b: Real;
  BEGIN
    Writeln('Построение графика функции y=ax+b');
    Write('a=');
    Readln(a);
    Write('b=');
    Readln(b);
    m:=Detect;
    InitGraph(m,n,'');
    Line(0,240,639,240);          {Ось OX}
    Line(320,0,320,479);          {Ось OY}
    Line (639,240,635,235);       {Стрелка \ для оси OX}
    Line (639,240,635,245);       {Стрелка / для оси OX}
    OutTextXY(625,225,'X');       {Вывод текста 'X'}
    Line(320,0,315,5);            {Стрелка / для оси OY}
    Line(320,0,325,5);            {Стрелка \ для оси OY}
    OutTextXY(330,5,'Y');         {Вывод текста 'Y'}
    OutTextXY(300,250,'O');       {Вывод текста 'O'}
    X:=8;                         {8 — масштаб по оси OX}
    While X<=639 Do               {Градуирование оси OX}
    Begin
      For Y:=-2 To 2 Do Putpixel(X,240+Y,7);
      X:=X+8;
```

```
    End;
    Y:=6;                          {6 — масштаб по оси OY}
    While Y<=479 Do                {Градуирование оси OY}
    Begin
       For X:=-3 To 3 Do Putpixel(320+X,Y,7);
       Y:=Y+6;
    End;
    {Вывод по оси OX предельных значений аргумента}
    OutTextXY(8,250,'-39');
    OutTextXY(620,250,'39');
    {Вывод по оси OY предельных значений функции}
    OutTextXY(290,470,'-39');
    OutTextXY(295,6,'39');
    x2:=-39;
    While X2<=39 Do                {(x2,y2) — точка графика}
    Begin                          {функции}
       y2:=a*x2+b;
       x1:=Round(x2*8+320);        {(x1,y1) — точка графика}
       y1:=Round(240-6*y2);        {в системе координат}
       Putpixel(x1,y1,7);          {экрана}
       x2:=x2+0.1;
    End;
    Readln;
    CloseGraph;
End.
```

В результате выполнения программы при $a = -2$, $b = 17$ получится следующий график (рис. 3.42).

Упражнение

Семейный бюджет Даниила на октябрь составил X р. Статьи расходов за месяц следующие: K (р.) — квартплата, P (р.) — питание членов семьи, U (р.) — культурная программа, R (р.) — прочие расходы. Неиспользованная сумма — N (р.).

Изобразите расходы семьи в виде диаграммы:

а) круговой;

б) столбчатой.

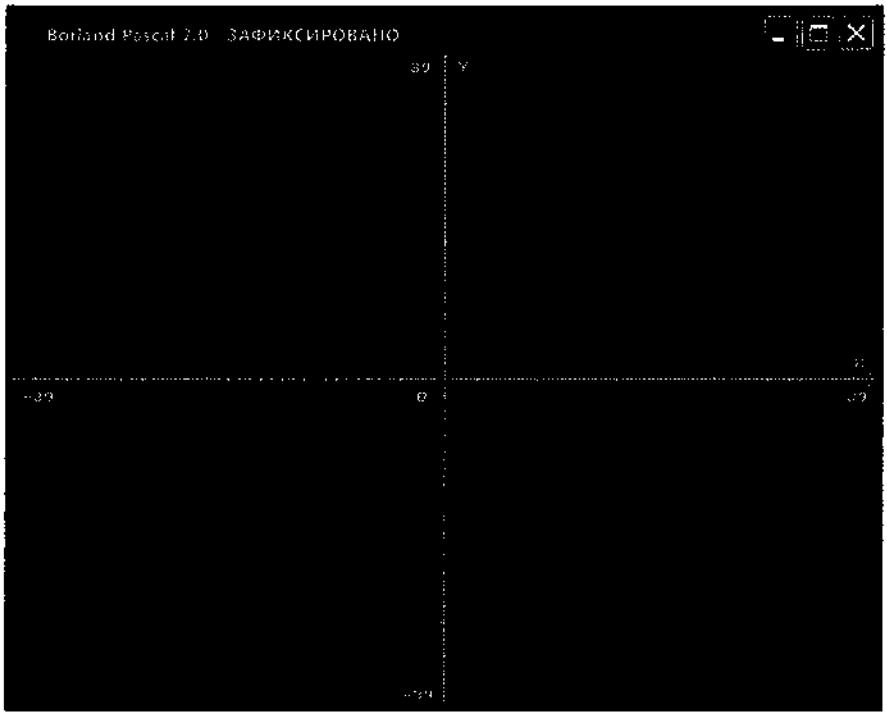


Рис. 3.42

* § 18. Подпрограммы пользователя

Часто в процессе разработки алгоритма обнаруживаются одинаковые части (фрагменты). Оформив их в виде вспомогательных алгоритмов, Вы можете неоднократно в случае необходимости обращаться к этим алгоритмам и выполнять требуемые действия. В виде вспомогательных алгоритмов удобно оформлять также отдельные самостоятельные части алгоритма, полученные при разбиении большой задачи на более мелкие подзадачи. Такой подход носит название **структурного** или **блочного** (**модульного**) программирования и является одной из концептуальных особенностей языка Паскаль. Объем программы благодаря этому сокращается, а сама программа становится более наглядной («читабельной»), упрощается ее отладка.

Вы уже знакомы с некоторыми стандартными подпрограммами языка Паскаль. Существует возможность создания подпрограмм самим программистом. Такие подпрограммы называют **подпрограммами пользователя**.

18.1. Описание подпрограмм пользователя

Правила языка Паскаль требуют обязательного описания всех используемых в программе подпрограмм пользователя.

При первоначальном ознакомлении со структурой программы на языке Паскаль в § 9 было отмечено, что данная структура не является полной. Мы уже расширяли ее при изучении графических возможностей языка. И вновь дополнение: в конце описательной части дается описание используемых в программе подпрограмм пользователя. Уточненная структура программы на языке Паскаль с учетом описания подпрограмм пользователя имеет вид, показанный на рисунке 3.43.

Подпрограммы пользователя бывают двух видов: **процедуры** и **функции**.

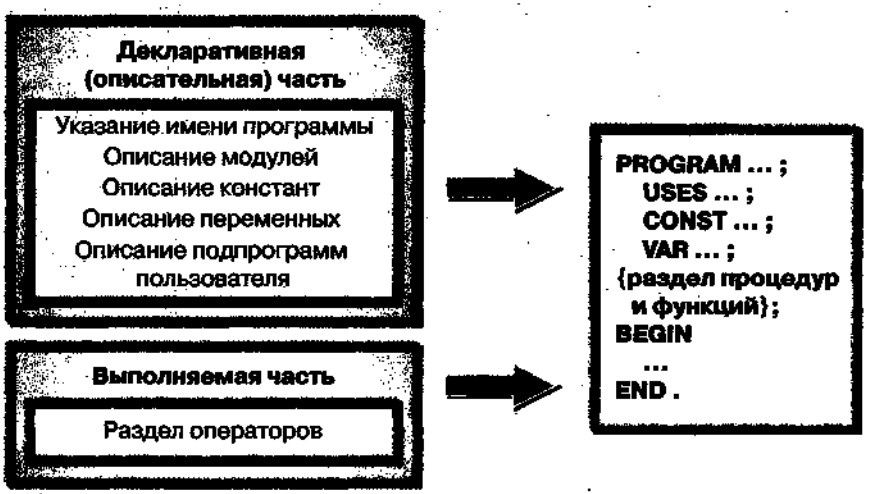


Рис. 3.43

18.2. Процедуры

Описание процедуры пользователя имеет вид, показанный на рисунке 3.44. При описании в заголовке процедуры можно указать список переменных (параметров), передаваемых в нее извне. Эти параметры называют **формальными**, так как они не связаны ни с какими конкретными значениями.

Имя_процедуры — уникальный идентификатор.

Параметры-аргументы (их называют параметрами-значениями) описываются так: `Аргумент : тип ;`. Их значения задаются на входе в процедуру.

Получить из процедуры значения результатов можно с помощью параметров-переменных. Для указания того, что некоторый параметр является результирующим (параметром-переменной), перед его именем записывают служебное слово (директиву) VAR.

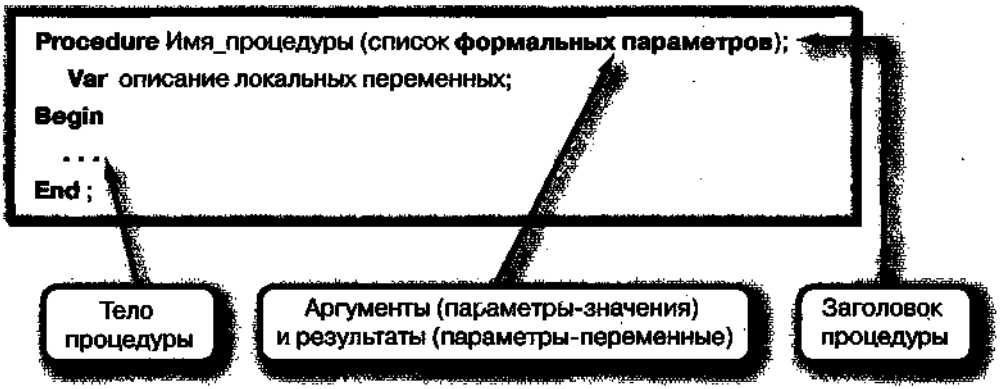


Рис. 3.44

Например, в заголовке процедуры

```
Procedure PRIMA(X:Integer;A:Real;
    Var P:Real;Var k:Integer);
```

подразумевается, что в процедуре используются два входных параметра (параметра-значения): *X* целого типа и *A* вещественного типа; в результате выполнения процедуры будут получены значения двух переменных (параметров-переменных): *P* вещественного типа и *k* целого типа.

При использовании процедур различают *глобальные* и *локальные* переменные: **глобальные** описываются в основной программе (они доступны из любого объекта программы), **локальные** — описываются в процедуре (они доступны только в пределах той подпрограммы, где они описаны).

Заголовок процедуры может иметь вид:

```
Procedure Имя_процедуры;
```

Здесь формальные параметры отсутствуют. Например,

```
Procedure Print;
Begin
  Writeln ('-------------');
End;
```

Обращение к процедуре (вызов) осуществляется путем указания ее имени и списка фактических параметров:

```
Имя_процедуры (список фактических параметров);
```

Фактические параметры — это конкретные величины, подставляемые вместо формальных параметров.

Процедура вызывается как отдельный оператор.

При использовании процедуры необходимо следовать определенным требованиям:

1. Количество фактических параметров должно совпадать с количеством формальных параметров в описании процедуры.
2. Порядок записи фактических параметров должен соответствовать порядку записи формальных параметров.
3. Типы фактических и формальных параметров-переменных должны совпадать. Типы фактических и формальных параметров-значений должны быть совместимыми.

Фактические параметры-значения представляют собой константы, имена переменных, выражения соответствующего типа; фактические параметры-переменные — имена переменных соответствующего типа.

Примеры обращения к процедуре PRIMA:

```
PRIMA(100, -1.3, Y, M);
PRIMA(V, L, G, KOL);
PRIMA(0, S*S, FACT, T);
PRIMA(N div 2, OPEL, RENAULT, MARKA);
```

Фактические параметры-значения V, N должны быть описаны в программе как целые; L, S, OPEL — как целые или вещественные (`Byte`, `Word`, `Integer`, `LongInt`, `Real`); фактические параметры-переменные Y, G, FACT, RENAULT — как вещественные (`Real`); M, KOL, T, MARKA — как целые (`Integer`).

Схема обращения к процедуре PRIMA представлена на рисунке 3.45.

① Из основной программы в процедуру передаются значения параметров-значений и параметров-переменных. Например, при обращении `PRIMA(100, -1.3, Y, M);` выполняется присваивание: `X:=100; A:=-1.3; P:=Y; k:=M;`

② Выполняются операторы, входящие в состав тела процедуры (для нашего примера в процедуре формируются значения переменных P, k).

③ Полученные значения параметров-результатов передаются в основную программу соответствующим переменным. В нашем случае выполняется присваивание: `Y:=P; M:=k;`

④ Продолжается выполнение основной программы.

Пример 1. Требуется сделать ограду для двух земельных участков, имеющих форму прямоугольников, если для каждого участка заданы его площадь и периметр. Написать программу определения длины секций ограды.

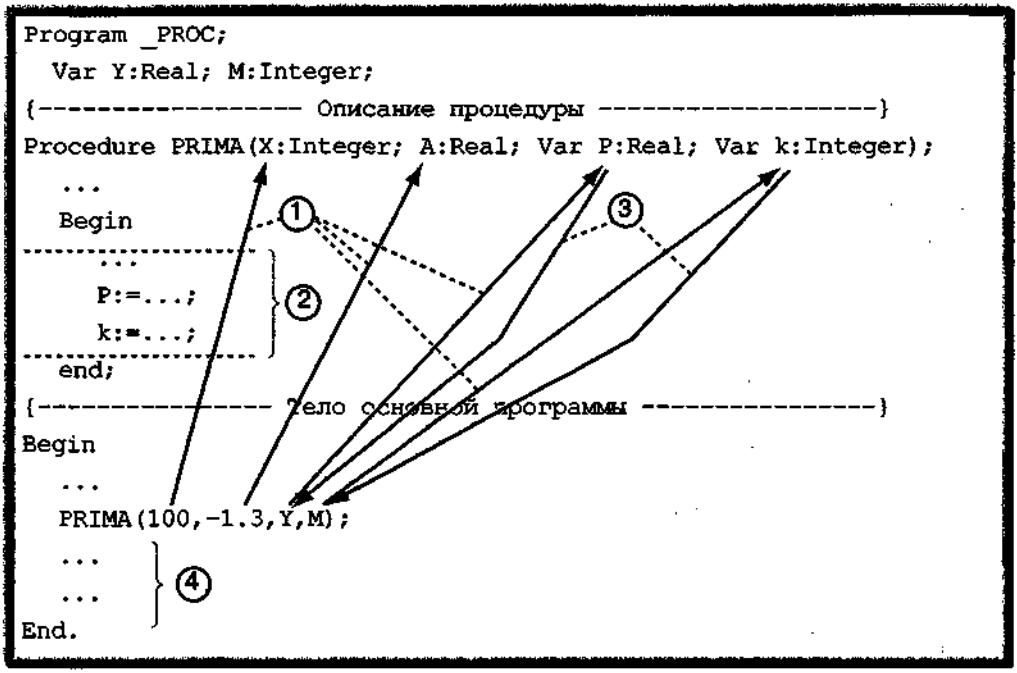


Рис. 3.45

Дано: S_1 , P_1 — площадь и периметр первого участка,
S_2 , P_2 — площадь и периметр второго участка.

Найти: a_1 , b_1 — длины границ первого участка,
a_2 , b_2 — длины границ второго участка.

Связь: Для участка прямоугольной формы с площадью S и периметром P числовые значения длин границ a и b определяются из системы

$$\begin{cases} ab = S, \\ 2(a+b) = P, \end{cases}$$

которая сводится к решению квадратного уравнения $x^2 - \frac{P}{2}x + S = 0$.

При: $\frac{P^2}{4} - 4S \geqslant 0$.

Как видно, решение задачи сводится к двукратному вычислению корней квадратного уравнения. Входные данные: S, P — площадь и периметр прямоугольника (вещественные числа). Так как в качестве результатов будут получены значения двух переменных (a, b — длины сторон прямоугольника, вещественные числа), возможно использование процедуры пользователя.

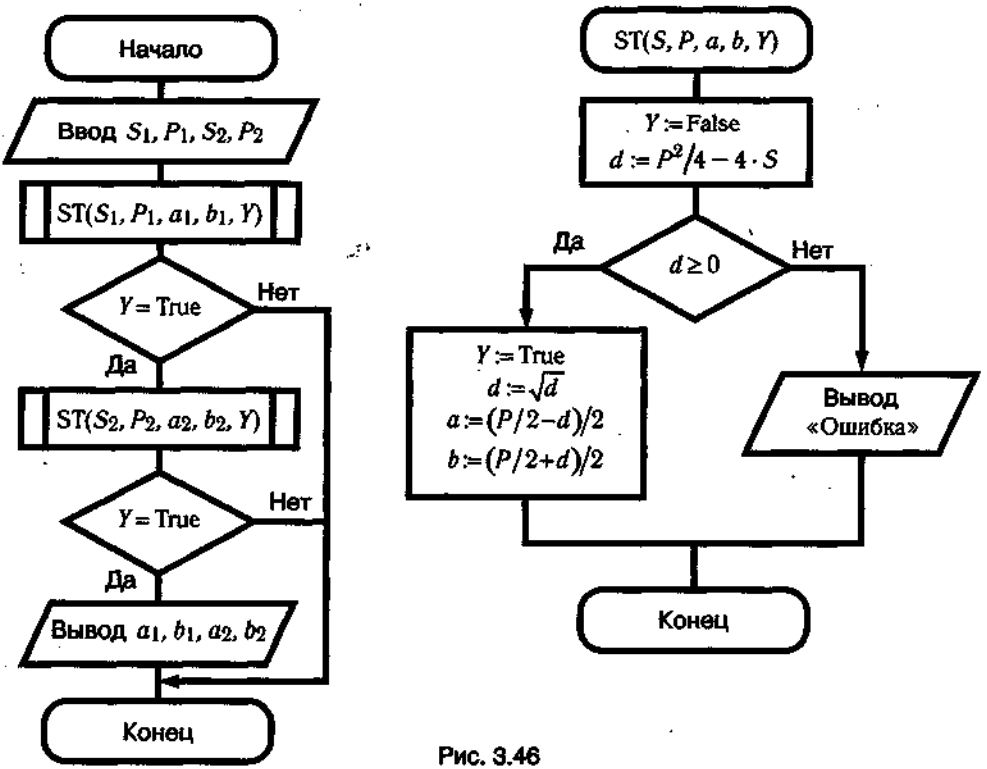


Рис. 3.46

Графическое представление алгоритма решения задачи показано в виде блок-схемы на рисунке 3.46.

Программа будет такой:

```
Program OGRADA;
  Var  S1,P1,S2,P2,a1,b1,a2,b2: Real;
       Y: Boolean;
       {Описание процедуры}
  Procedure ST (S,P: Real; Var a,b: Real;
    Var Y: Boolean);
    Var d: Real;
  Begin
    Y:=False; d:=P*P/4-4*S;
    If d>=0 Then
    Begin
      Y:=True; d:=sqrt(d);
```

```
        a:=(P/2-d)/2;
        b:=(P/2+d)/2;
      End
        Else Writeln('Ошибка в данных');
   End;
{Основная программа}
Begin
   Write ('S1 P1='); Readln(S1,P1);
   Write ('S2 P2='); Readln(S2,P2);
   ST(S1,P1,a1,b1,Y);
   If Y=True Then
   Begin
      ST(S2,P2,a2,b2,Y);
      If Y=True Then
        Writeln(a1:7:1, b1:7:1,a2:7:1,b2:7:1);
   End;
End.
```

Упражнение

Напишите программу определения наибольшего числа среди четырех заданных чисел.

18.3. Функции

Описание функций пользователя имеет вид, показанный на рисунке 3.47.

В заголовке функции формальные параметры могут отсутствовать.

Параметры-значения (аргументы) описываются так же, как и в процедуре: `Аргумент:тип;`. Их значения задаются на входе в функцию.

Результирующей переменной является идентификатор `Имя_функции`. Тип значения, присваиваемого идентификатору функции, называют типом функции (он указывается в конце заголовка функции после символа «:»).

Например, в заголовке функции

```
Function FUNC (X: Integer; A: Real) : Integer;
```

подразумевается, что в процедуре-функции используются два входных параметра (параметры-значения): X целого типа и A вещественного типа; в результате ее выполнения будет получено значение переменной FUNC (параметр-переменная) целого типа. Само значение результата FUNC формируется в теле функции с помощью оператора присваивания.

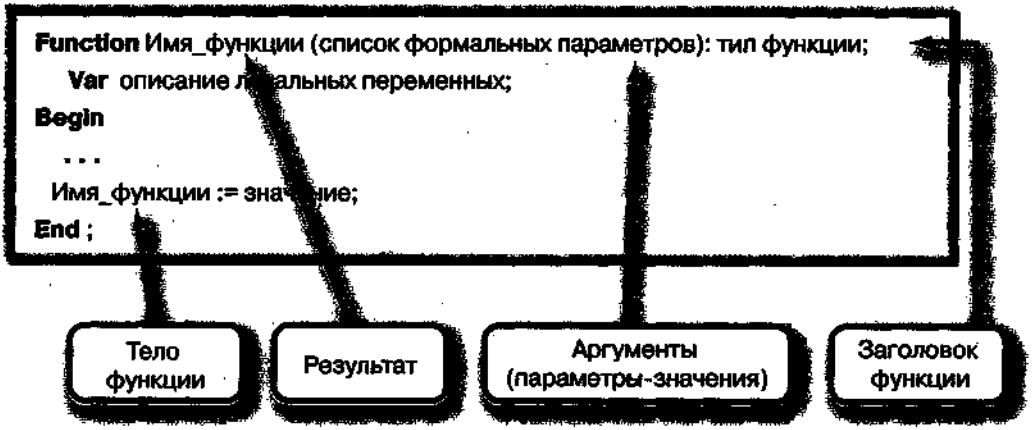


Рис. 3.47

Обращение к функции (вызов) осуществляется путем указания ее имени и списка фактических параметров: `Имя_функции (список фактических параметров)`. Функция, в отличие от процедуры, может быть вызвана только в составе некоторого выражения. При этом необходимо следовать определенным требованиям (см. требования, описанные в п. 18.2).

Фактические параметры-значения представляют собой константы, имена переменных, выражения соответствующего типа, фактический результат — значение идентификатора функции.

Примеры обращения к функции FUNC:

```
M:=FUNC(5, -1.3);
Writeln(FUNC (V, L));
P:=FUNC(0, S*S)+FUNC(K,2);
Write (FUNC (N div 2, Info) : 5);
```

Переменные K, V, N должны быть описаны в программе как целые (`Integer`, `Byte` и др.), M, P, L, S, Info — как целые или вещественные.

Схема выполнения функции пользователя дана на рисунке 3.48.

① Из основной программы в функцию передаются значения параметров-аргументов. Например, при обращении `M:=FUNC (5, −1.3);` выполняется присваивание: `X:=5; A:=-1.3;`

② Выполняются операторы, входящие в состав тела функции (для нашего примера в функции формируется значение переменной `FUNC`).

③ Полученное значение параметра-результата передается в основную программу в вызывающее выражение (это значение передается переменной M).

④ Продолжается выполнение основной программы.

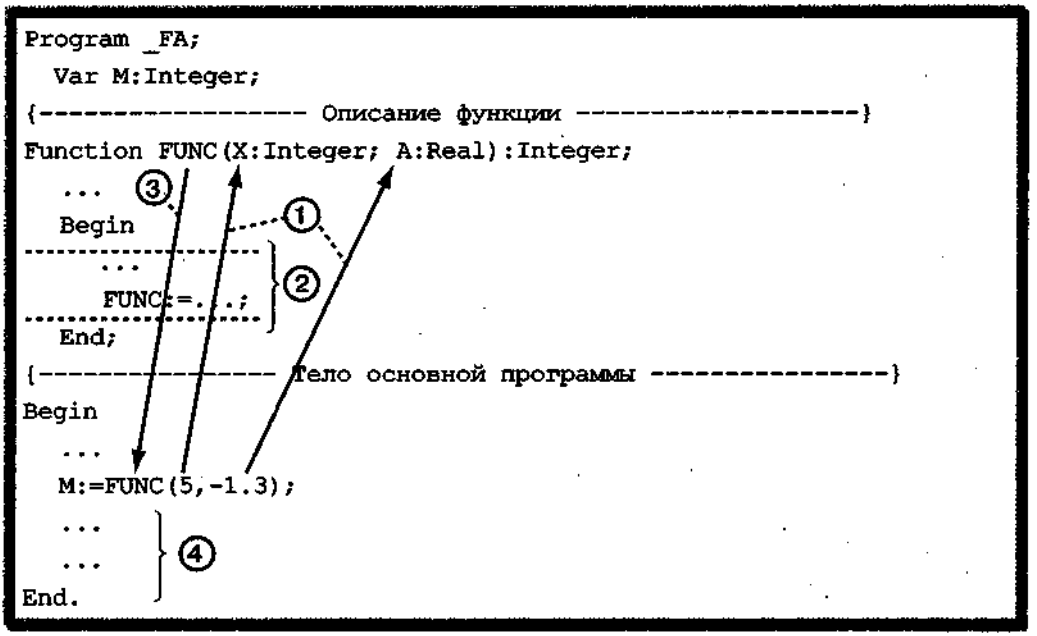


Рис. 3.48

Пример 1. Для создания парковой зоны предполагается объединить три земельных участка. Составить программу определения площади зоны отдыха, если длины границ участков заданы (данные корректны) (рис. 3.49).

Дано: a, b, c, d, e, f, g — длины границ участков.

Найти: S — суммарная площадь участков.

Связь: $S = S_1 + S_2 + S_3$, где S_1, S_2, S_3 — площади участков.

Как видно из рисунка, решение задачи сводится к вычислению площадей треугольников по формуле Герона. Имеет смысл вычисление площади оформить в виде подпрограммы пользователя. Входные данные для подпрограммы: x, y, z — длины сторон треугольника, результат

$$S = \sqrt{p(p-x)(p-y)(p-z)},$$

где $p = \frac{x+y+z}{2}$. Так как результатом выполнения подпрограммы является единственное значение (площадь), относящееся к простому типу, будем использовать функцию пользователя.

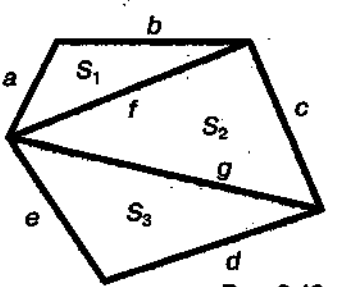


Рис. 3.49

Графическое представление алгоритма решения задачи показано в виде блок-схемы на рисунке 3.50.

Программа будет такой:

```
Program Plosha;
   Var   a,b,c,d,e,f,g,
         S,S1,S2,S3: Real;
   Function PL(x,y,z: Real): Real;
      Var P: Real;
   Begin
      P:=(x+y+z)/2;
      PL:=sqrt(P*(P-x)*(P-y)*(P-z));
   End;
Begin
   Writeln
      ('Введите длины границ участков');
   Readln(a,b,c,d,e,f,g);
   S1:=PL(a,b,f);
   S2:=PL(f,c,g);
   S3:=PL(g,d,e);
   S:=S1+S2+S3;
   Writeln(S:0:2);
End.
```

⇒

```
S:=PL(a,b,f)+PL(f,c,g)+
PL(g,d,e);
```

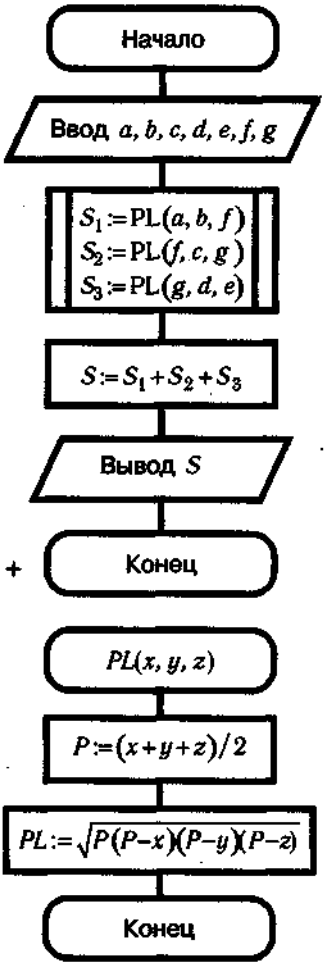


Рис. 3.50

Упражнение

Напишите программы решения следующих задач:

а) случайным образом задаются координаты трех точек на плоскости. Требуется определить: образуют ли они треугольник, и если образуют, то его тип (равнобедренный, равносторонний, разносторонний);

б) для заданных вещественных чисел x и y необходимо вычислить $\max(x, y) + \max(x + y, x - y)$, где $\max(a, b)$ — наибольшее значение чисел a и b;

в) назовем автобусный билет (это шестизначное число) «счастливым», если сумма его цифр делится на 7. Определить, являются ли билеты, которые компостируют два друга, «счастливыми».

ТЕХНОЛОГИЯ ОБРАБОТКИ ИНФОРМАЦИИ В СИСТЕМЕ УПРАВЛЕНИЯ БАЗАМИ ДАННЫХ

§ 19. Системы управления базами данных

В современном мире сложность информации постоянно возрастает. Для автоматизации хранения и поиска необходимой информации создаются и применяются системы управления базами данных и базы данных.

Из курса информатики вам уже известны эти понятия.

Напомним, что базы данных (БД) — это взаимосвязанные и организованные определенным образом данные, отображающие состояние объектов и отношения между ними в определенной предметной области.

Под системой управления базами данных (СУБД) понимается совокупность программных и языковых средств, предназначенных для создания и обработки баз данных.

СУБД и БД применяются во всех сферах деятельности человека для обработки и хранения информации.

В различных отраслях производства используются базы данных, содержащие информацию о промышленных и сельскохозяйственных изделиях, товарах, договорных обязательствах, поставщиках и заказчиках.

В научно-исследовательских институтах и конструкторских бюро создаются базы данных, хранящие информацию о физических свойствах материалов, проектных решениях и др.

В современных поликлиниках и больницах используются СУБД для создания и обработки баз данных, описывающих истории болезней пациентов, результаты проведенных исследований состояния их здоровья.

В настоящее время разработано около сотни разных СУБД.

Нам предстоит познакомиться с возможностями СУБД Access, которая предназначена для работы с реляционной моделью организации данных (от английского *relation — отношение*). В реляционной модели данные организованы в виде совокупности таблиц, между которыми устанавливаются связи. Назначение таких связей и порядок их задания мы рассмотрим в дальнейшем в § 22.

Реляционная модель баз данных была предложена в конце 60-х годов XX в. Эдгаром Коддом.

Первая версия СУБД Access была создана в начале 90-х годов XX в.

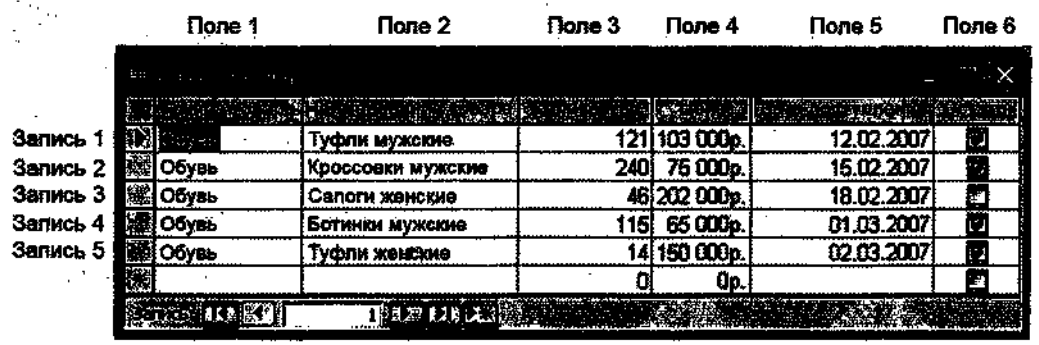


Рис. 4.1

Каждая таблица состоит из записей и полей, например таблица «Товар» базы данных «Магазин» (рис. 4.1).

Запись представляет собой строку таблицы базы данных, а поле — это столбец такой таблицы.

Для описания поля используются следующие характеристики: имя, тип, размер, формат данных поля.

В таблице «Товар» представлены шесть полей и пять записей. Каждое поле имеет свое имя, в нем хранятся данные определенного типа.

Поле *Количество* содержит **числовые** данные.

Поля *Группа товара* и *Наименование товара* являются **текстовыми** данными.

Поле *Цена* является **денежным** типом данных и содержит цену товара в белорусских рублях.

Поле *Дата поступления* содержит информацию о **дате** поступления товара в магазин.

Поле *Остаток* является **логическим** типом данных и принимает значение *Да (Истина)*, если товар присутствует на складе магазина, *Нет (Ложь)* — товар не присутствует.

Более полное описание типов полей рассмотрим в § 21.

Информация, хранимая в базах данных, должна быть структурирована, актуальна, обновляема и доступна пользователю.

1. Что называют базами данных?
2. Какие системы называют системами управления базами данных?
3. Что называют записью и полем в таблице базы данных?
4. Как описывается поле в таблице базы данных?

§ 20. Основные элементы интерфейса СУБД Access

СУБД Access после ее установки может загружаться с **Рабочего стола** двойным щелчком левой клавишей мыши по ярлыку  или через систему меню: кнопка **Пуск → Программы → Microsoft office → Microsoft Access**.

После запуска СУБД Access откроем базу данных. Для этого выполним цепочку команд: **Файл → Открыть →** в окне **Открытие файла базы данных** укажем файл базы данных, например Магазин.mdb **→ Открыть**. Основные элементы интерфейса окна показаны на рисунке 4.2.

На панели **Стандартная** размещаются основные инструменты, используемые при работе с базой данных. Часть инструментов этой панели аналогична панели **Стандартная** текстового редактора Word, а вторая часть предназначена для работы с базой данных. Применение этих инструментов мы рассмотрим в дальнейшем.

В окне **База данных** расположены ее объекты: **Таблицы, Запросы, Формы, Отчеты** и др. Все эти объекты, если они создавались, хранятся в общем файле базы данных на диске с расширением **.mdb**. При работе с каждым из объектов они имеют свое окно, как показано на рисунке 4.2.

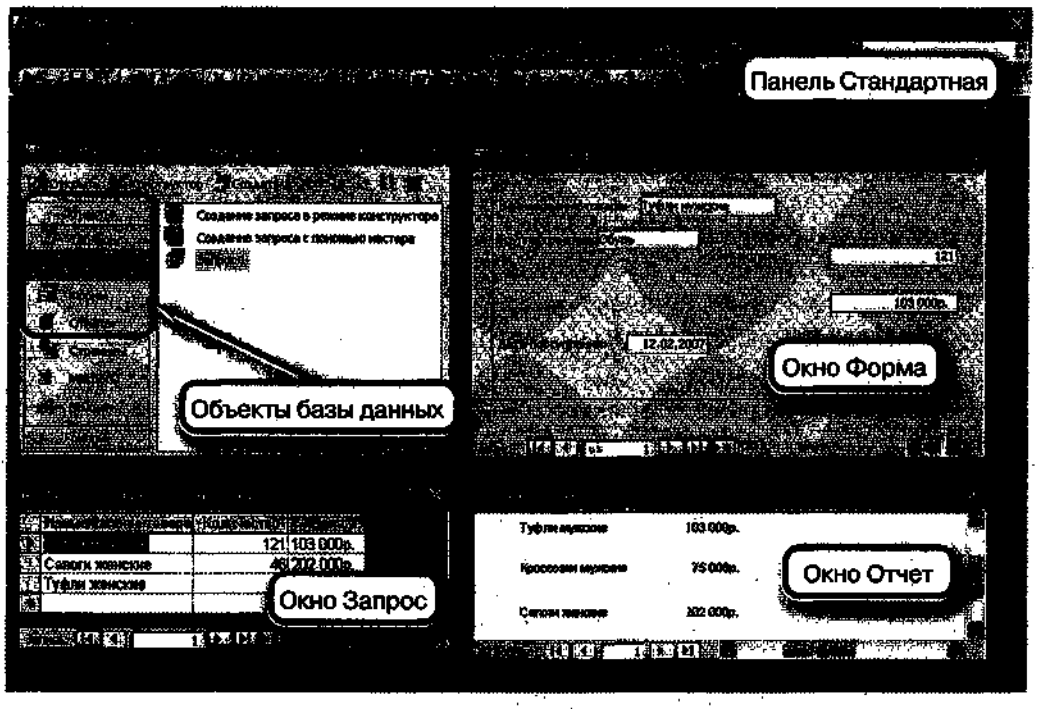


Рис. 4.2

Рассмотрим назначение каждого объекта.

Таблицы являются основным объектом базы данных, который предназначен для хранения данных. Таблицы состоят из записей и полей. На основе таблиц создаются остальные объекты базы данных. Как правило, для решения сложных задач одной таблицы недостаточно. Поэтому в базе данных присутствует сразу несколько таблиц, связанных между собой.

Запросы предоставляют возможность отобрать данные из таблиц на основании определенных условий.

Формы отображают данные из таблиц или запросов. С помощью форм удобно вводить данные в таблицы.

Отчеты предназначены для создания документа, построенного на информации, отобранной из базы данных. Отчеты можно просмотреть на экране, распечатать на принтере.

?

1. Как загрузить СУБД Access и открыть готовую базу данных?
2. Перечислите основные элементы интерфейса СУБД.
3. Какие объекты присутствуют в окне **База данных**?
4. Для чего предназначен объект **Таблица**?

§ 21. Создание таблицы базы данных

21.1. Проектирование баз данных

Изучение возможностей СУБД Access начнем с проектирования базы данных. Таблица является основным объектом базы данных. База данных может содержать одну или несколько связанных таблиц в зависимости от требований, предъявляемых разработчиком. На основании таблиц создаются остальные объекты базы данных.

Пример. Необходимо спроектировать базу данных «Библиотека дисков» (файл Library_disk.mdb), содержащую сведения об использовании CD и DVD дисков из личной библиотеки пользователя.

Основной задачей использования базы данных «Библиотека дисков» является отслеживание выдачи дисков всем желающим клиентам.

Проведя анализ необходимой для хранения информации, попытаемся сначала расположить ее в одной таблице, поля которой разделим на три группы: сведения о клиентах, сведения о наличии дисков и сведения о выдаче дисков. Структура таблицы базы данных в этом случае должна иметь следующий вид:

Имя поля	Тип данных	
Фамилия	Текстовый	Сведения о клиентах
Имя	Текстовый	
Адрес	Текстовый	
Телефон	Текстовый	
Электронная почта	Текстовый	
Название диска	Текстовый	Сведения о дисках
Тип диска	Текстовый	
Стоимость диска	Денежный	
Дата выдачи	Дата/время	Сведения о выдаче дисков
Отметка о возврате	Логический	

Наполнять данными таблицу, имеющую представленную структуру, достаточно неудобно. Например, при выдаче нескольких дисков для одного клиента будет необходимо многократно повторять информацию о нем: фамилию, имя, адрес и т. д., что приведет к неоправданному увеличению размера таблицы и может повысить вероятность ошибок при вводе информации.

! Для повышения эффективности при работе с создаваемой базой данных необходимо выполнить процесс *нормализации*.

Для этого разделим одну таблицу на три: «Клиенты», «Диски», «Выдача дисков».

Опишем структуру каждой таблицы.

Таблица «Клиенты»

Имя поля	Тип данных	Имя поля	Тип данных
Код клиента	Счетчик	Адрес	Текстовый
Фамилия	Текстовый	Телефон	Текстовый
Имя	Текстовый	Электронная почта	Текстовый

Таблица «Диски»

Имя поля	Тип данных	Имя поля	Тип данных
Код диска	Счетчик	Тип диска	Текстовый
Название диска	Текстовый	Стоимость диска	Денежный

Таблица «Выдача дисков»

Имя поля	Тип данных	Имя поля	Тип данных
Код выдачи	Счетчик	Дата выдачи	Дата/время
Код клиента	Числовой	Отметка о возврате	Логический
Код диска	Числовой		

В представленных таблицах впервые присутствуют данные **Счетчик**. Он применяется для хранения целых числовых значений, которые СУБД Access увеличивает при переходе к каждой новой записи. Счетчик может использоваться в качестве уникального идентификатора записи таблицы, в которой не предусмотрено другой такой величины.

В нашем случае *Код клиента, Код диска* и *Код выдачи* будут уникальными идентификаторами, которые позволят легко отличить одного клиента от другого, один диск от другого и присвоить уникальные специальные значения каждой выдаче дисков.

Поля *Код клиента, Код диска* в дальнейшем будут использованы для установки связей между таблицами, которые мы рассмотрим позже в § 22.

Перед началом создания таблиц необходимо с помощью цепочки команд в СУБД Access создать новую базу данных: **Файл → Создать Новая база данных →** в окне **Файл новой базы данных** указать имя файла базы, например Library_disk → **Открыть**.

Создание таблицы логически разделяется на два этапа:

- создание структуры таблицы: имена полей, типы данных, размер и формат полей;
- ввод в таблицу необходимой информации.

СУБД Access позволяет создавать структуру таблицы тремя способами в режиме **Конструктора**, **Мастера** или **Таблицы** (путем ввода данных) (рис. 4.3).

Рассмотрим работу режимов **Мастер** и **Конструктор**. В режиме **Таблицы** ввод данных производится в пустую таблицу.

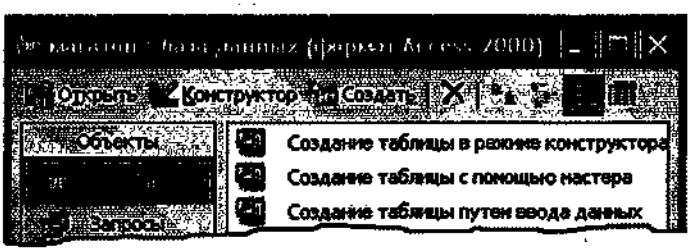


Рис. 4.3

21.2. Создание структуры таблицы в режиме Мастера

В режиме **Мастера** удобно создавать структуру таблицы, в которой используются стандартные имена полей и типы данных в этих полях. **Мастер** вначале предлагает выбрать образец таблицы, а затем отобрать поля для нее.

Пример. Создать в режиме **Мастера** структуру таблицы «Клиенты». Для выполнения этого задания необходимо:

1. Выбрать объект **Таблицы** в окне **База данных** и щелкнуть два раза левой клавишей мыши по тексту **Создание таблицы с помощью мастера**.

2. Выбрать образец таблицы «Клиенты», образцы полей перенести с помощью стрелок из окна **Образцы полей** в окно **Поля новой таблицы**, как показано на рисунке 4.4. Имя перенесенного поля может быть изменено с помощью кнопки **Переименовать поле...** .

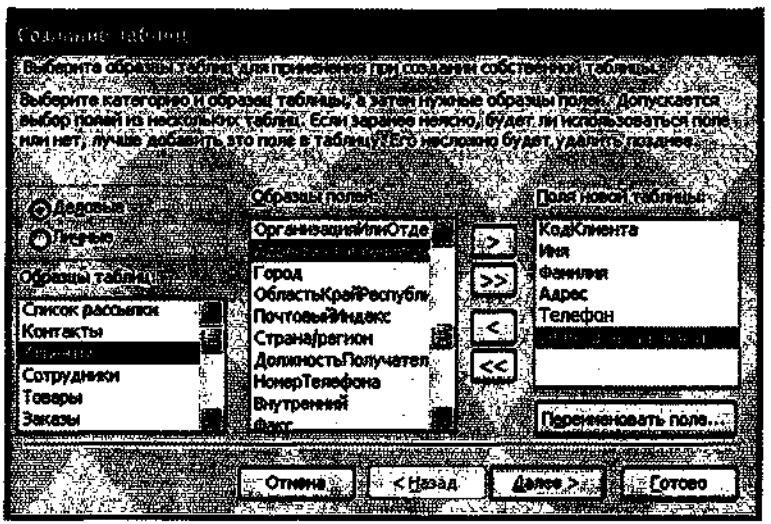


Рис. 4.4

3. Щелкнуть левой клавишей мыши по кнопке **Далее** → задать имя таблицы, например «Клиенты», и установить флажок **Пользователь определяет ключ самостоятельно** → выбрать поле с уникальными для каждой записи данными, для нашего примера — это **КодКлиента** → **Готово**. В результате откроется окно сформированной таблицы «Клиенты», которую можно заполнять информацией (рис. 4.5).

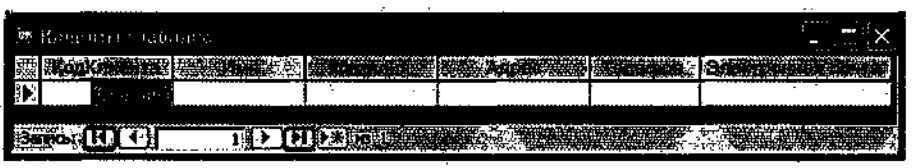


Рис. 4.5

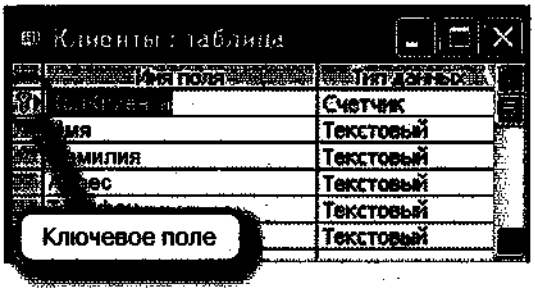


Рис. 4.6

Созданная **Мастером** структура таблицы «Клиенты» представлена на рисунке 4.6.

Желательно, чтобы каждая таблица имела **ключ** — одно или несколько полей, содержимое которых уникально для каждой записи. В случае отсутствия ключевого поля затрудняется поиск записей в таблице, так как в ней могут встречаться клиенты с одинаковыми фамилиями, адресами и даже телефонными номерами. Для таблицы «Клиенты» таким полем является **КодКлиента**. Тип данных этого поля — **Счетчик**. В нашей таблице каждый клиент будет иметь уникальный номер, и никакие две записи в этой таблице не будут одинаковыми.

Мастер определяет тип данных полей самостоятельно в зависимости от названия без предварительного определения этих типов пользователем. В дальнейшем в режиме **Конструктора** тип поля может быть изменен.

21.3. Создание структуры таблицы в режиме Конструктора

Режим **Конструктора** предоставляет пользователю возможность самостоятельно описывать и изменять структуру таблицы. После двойного щелчка левой клавишей мыши по строке **Создание таблицы** в режиме **Конструктора** (см. рис. 4.3) **Конструктор** открывает специальное окно, в котором необходимо описать поля.

Пример. Создать с помощью **Конструктора** таблицу «Диски», которая должна содержать четыре поля, для каждого поля требуется ввести имя, тип данных и в нижней части окна определить свойства поля, как показано на рисунке 4.7.

При вводе типов данных и свойств полей пользователю необходимо использовать раскрывающиеся списки (см. рис. 4.7).

В таблице «Диски» ключевым полем является **Код диска**. Чтобы это поле ста-

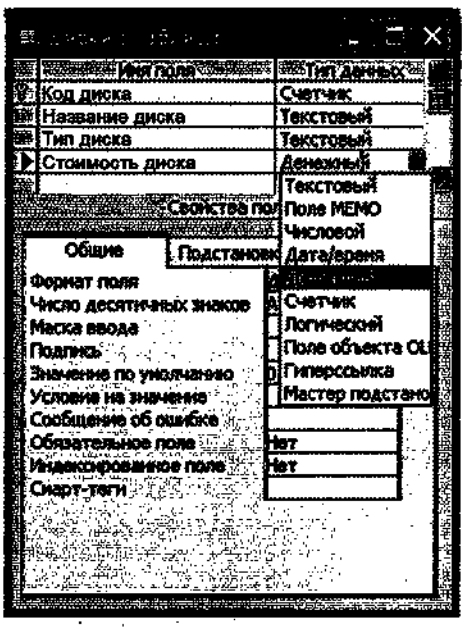


Рис. 4.7

ло ключевым, достаточно в режиме **Конструктора** поместить курсор в выбранное поле и нажать кнопку **Ключевое поле** на панели **Базы данных** или выполнить команды меню **Правка → Ключевое поле**. Если необходимо определить сразу несколько ключевых полей, то следует выделить нужные строки и нажать кнопку **Ключевое поле**. Повторные аналогичные действия отменяют признак ключевого поля.

СУБД Access допускает использование различных типов данных. Основные типы данных и их описание приведены в следующей таблице.

Тип данных	Описание типа данных
Числовой	Содержит произвольные числовые значения
Текстовый	Текстовые, числовые данные, не требующие вычислений. Длина поля не превосходит 255 символов
Поле MEMO	Текстовые данные большого объема. Длина поля не превосходит 65 535 символов
Дата/время	Представление даты и времени в различных форматах
Денежный	Числовые денежные значения
Логический	Логические значения
Гиперссылка	Ссылка на некоторый документ или файл
Объект OLE	Документы различных типов, подготовленные в Word, Excel, точечные рисунки и др.

Назначение поля **Счетчик** уже рассматривалось нами ранее.

После завершения описания таблицы она сохраняется с помощью цепочки команд: **Файл → Сохранить как →** указать имя таблицы, например **Диски → Ок**. Созданные в **Мастере** таблица «Клиенты» и в **Конструкторе** таблица «Диски» добавляются в окно **Базы данных**.

21.4. Ввод и редактирование данных в таблице

После завершения работы по созданию структуры таблицы пользователь может в режиме **Таблицы** приступить к заполнению таблицы данными. Для этого необходимо сделать двойной щелчок левой клавишей мыши по названию таблицы в окне **Базы данных**, например «Клиенты».

Ввод данных в таблицу базы данных и их редактирование осуществляется непосредственно в соответствующей ячейке таблицы. Действия по вводу и редакти-

	1	Сидоров	Вася	ул. Плеханова, д. 20	2954321	Sidor@tut.by
	2	Симончик	Валя	ул. Васнецова, д.3, кв.89	2961736	sim@mail.ru
	3	Титов	Алексей	ул. Жилуновича, д.6	2474331	Titov@tut.by
	4	Смирнов	Александр	ул. Ванеева, д.6, кв. 5	2912345	mal@tut.by
	5	Иванов	Павел	ул. Кошевого, д.13, кв.5	2916754	Pivan@mail.ru
	(Счетчик)					

Рис. 4.8

рованию аналогичны заполнению и редактированию данных в таблице текстового редактора Word (рис. 4.8).

Для удаления в таблице одной записи необходимо выбрать режим **Таблица**, установить курсор в любое поле удаляемой записи и сделать щелчок левой клавишей мыши по кнопке **Удалить запись**. Для удаления нескольких записей одновременно их необходимо вначале выделить, а затем выполнить последовательность действий, описанных выше.

Добавление записей в таблицу осуществляется в режиме **Таблица**. При этом новая запись добавляется в конец таблицы.

После завершения ввода данных в таблицу или их редактирования таблица сохраняется с помощью цепочки команд **Файл → Сохранить**.

1. **Как создать структуру таблицы в режиме Мастера?**
2. **Можно ли изменить имя поля в режиме Мастера?**
3. **Как создать структуру таблицы в режиме Конструктора?**
4. **Что понимается под ключевым полем в таблице? Для чего применяется ключевое поле?**
5. **Какие типы данных допускается использовать в СУБД Access?**
6. **Как записи добавляются в таблицу и удаляются из нее?**

Упражнения

1. Создайте однотабличную базу данных «Страны», которая содержит таблицу «Страны 2002». Для этого:

   а) Опишите структуру таблицы (рис. 4.9).

   б) Введите в таблицу данные (рис. 4.10).

Страны 2002 : таблица

Имя поля	Тип данных
Страна	Текстовый
Столица	Текстовый
Население	Числовой
Площадь (кв км)	Числовой
Язык	Текстовый
Часть света	Текстовый

Рис. 4.9

Австралия	Канберра	19500000	7687000	Английский	Австралия и Океания
Австрия	Вена	8100000	83800	Немецкий	Европа
Алжир	Алжир	31400000	2382000	Арабский	Африка
Аргентина	Буэнос-Айрес	37900000	2767000	Испанский	Южная Америка
Беларусь	Минск	9900000	207600	Белорусский, русский	Европа
Бразилия	Бразилия	174700000	8512000	Португальский	Южная Америка
Великобритания	Лондон	59700000	244100	Английский	Европа
Вьетнам	Ханой	80200000	329600	Вьетнамский	Азия
Египет	Каир	70300000	1002000	Арабский	Африка
Индия	Дели	849600000	3288000	Хинди	Азия
Канада	Оттава	31300000	9976000	Английский, французский	Северная Америка
Китай	Пекин	12944000	9561000	Китайский	Азия
Люксембург	Люксембург	400000	2600	Немецкий, французский	Европа
Польша	Варшава	38500000	312700	Польский	Европа
Россия	Москва	148600000	17075400	Русский	Европа, Азия
США	Вашингтон	288500000	9364000	Английский	Северная Америка
Тунис	Тунис	9700000	164000	Арабский	Африка
Украина	Киев	48700000	603700	Украинский	Европа

Рис. 4.10

2. Создайте базу данных «Библиотека дисков» (файл Library disk.mdb), содержащую три таблицы «Клиенты», «Диски», «Выдача дисков», как показано на рисунке 4.11, *а*, *б*, *в*.

а

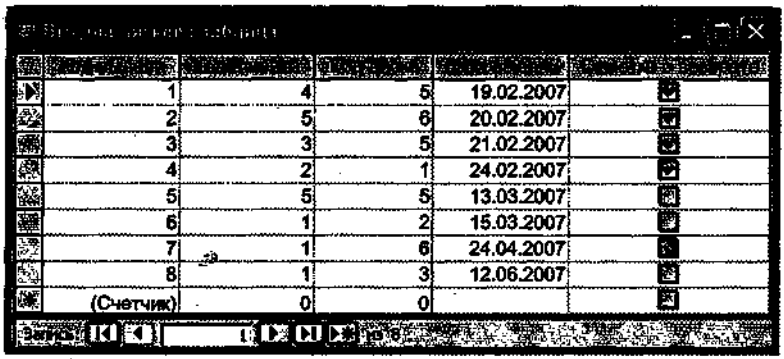
	Фамилия	Имя	Адрес	Телефон
1	Сидоров	Вася	ул. Плеханова, д. 20	2954321
2	Симончик	Валя	ул. Васнецова, д.3, кв.89	2961736
3	Титов	Алексей	ул. Жилуновича, д.6	2474331
4	Смирнов	Александр	ул. Ванеева, д.6, кв. 5	2912345
5	Иванов	Павел	ул. Кошевого, д.13, кв.5	2916754
(Счетчик)				

б

1	Комплект словарей	CD-R	15 000.00р.
2	Текстовые редакторы	CD-R	10 000.00р.
3	Пираты Карибского моря	DVD	18 000.00р.
4	Валерий Меладзе	DVD	7 000.00р.
5	Группа Сябры	DVD	6 000.00р.
6	Дима Колдун	DVD	9 000.00р.
(Счетчик)			

Рис. 4.11, *а*, *б*

в



1	4	5	19.02.2007	
2	5	6	20.02.2007	
3	3	5	21.02.2007	
4	2	1	24.02.2007	
5	5	5	13.03.2007	
6	1	2	15.03.2007	
7	1	6	24.04.2007	
8	1	3	12.06.2007	
(Счетчик)	0	0		

Рис. 4.11, *в*

§ 22. Связывание таблиц базы данных

Установление связей в СУБД Access дает возможность автоматически соединять данные из разных таблиц.

Процесс установки связей между таблицами называют построением схемы базы данных.

Для установления связей между двумя таблицами необходимо определить в каждой из них поля для этого связывания. Эти поля не обязательно могут иметь одинаковые имена, но должны содержать однотипные данные.

Пример. Установить связь между двумя таблицами «Покупатель» и «Заказы» в базе данных Trade.mdb. Структура каждой таблицы представлена на рисунке 4.12.

В таблице «Покупатель» ключевым полем является поле **Код покупателя**. Данное поле является счетчиком и содержит уникальные значения для каждой записи этой таблицы. Поле данной таблицы назовем *первичным ключом*.

Если каждый покупатель делает только один заказ, то в таблице «Заказы» поле **Код покупателя** будет содержать неповторяющиеся данные и должно быть определено как ключевое поле. В таблице «Заказы» ключевое поле **Код покупателя** будем называть *внешним ключом*.

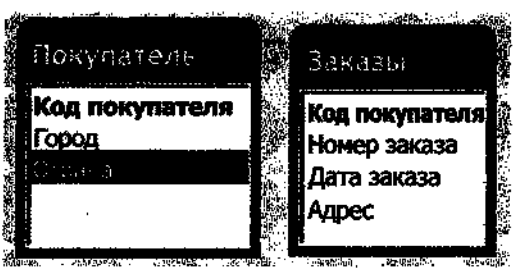


Рис. 4.12

В этом случае тип связи, установленный между первичным и внешним ключами, называют связью *один к одному*. Этот тип связи представлен на рисунке 4.13.

Рис. 4.13

Если в таблице «Заказы» один покупатель делает несколько заказов, то поле **Код покупателя** уже не будет уникальным. Получается, что данные в этом поле повторяются многократно. В этом случае ключевым полем с уникальными данными должно быть определено другое поле, например поле **Номер заказа**. В данном случае тип связи, установленный между одноименными полями **Код покупателя** в обеих таблицах, называют связью *один ко многим*, как показано на рисунке 4.14.

Рис. 4.14

Для создания **Схемы данных** в СУБД Access необходимо выполнить следующее:

1. Открыть многотабличную базу данных, для которой между таблицами устанавливаются связи, например базу данных trade.mdb.

2. Щелкнуть левой клавишей мыши по значку **Схема данных**  или выполнить цепочку команд **Сервис → Схема данных** на панели **Стандартная**.

3. Выделить первую таблицу, для которой устанавливается связь в окне **Добавление таблицы**. Для данного примера это таблица «Покупатель».

4. Щелкнуть левой клавишей мыши по кнопке **Добавить**. На экране будет отображена таблица, которую мы добавили.

5. Выделить таблицу, которую требуется добавить в **Схему данных** (в нашем случае это таблица «Заказы»), и щелкнуть левой клавишей мыши по кнопке **Закрыть**.

6. В окне таблицы «Покупатель» щелкнуть левой клавишей мыши на поле **Код покупателя**, которое будет использоваться для установки связи, и перетянуть его на совпадающее поле таблицы «Заказы».

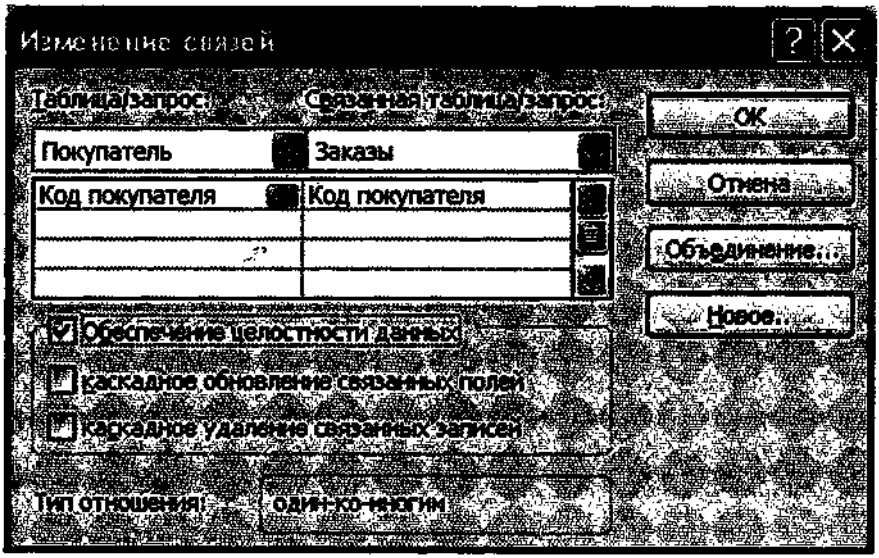


Рис. 4.15

7. В окне **Изменение связей** установить флажок **Обеспечение целостности данных**, затем щелкнуть левой клавишей мыши по кнопке **ОК** (рис. 4.15).

В результате выполненных действий в окне **Схема данных** два связанных поля соединятся линией, как показано на рисунке 4.14.

В дальнейшем для сохранения созданных связей необходимо щелкнуть левой клавишей мыши по кнопке **Сохранить** и закрыть окно **Схема данных**.

Созданные связи в дальнейшем можно удалить в окне **Схема данных**, щелкнув левой клавишей мыши по линии связи, чтобы выделить ее, а затем нажать клавишу Delete (Удалить) на клавиатуре. Двойной щелчок левой клавишей мыши по линии связи приведет к открытию окна **Изменение связей**, в котором схема базы данных может быть изменена.

1. Для чего необходимо связывать таблицы базы данных?
2. Какими свойствами должны обладать связываемые поля?
3. Какие типы связей существуют между таблицами?
4. Что понимается под схемой базы данных?

Упражнение

В базе данных «Библиотека дисков» свяжите таблицы «Клиенты», «Диски», «Выдача дисков» (см. рис. 4.11).

Результат выполнения задания дан на рисунке 4.16.

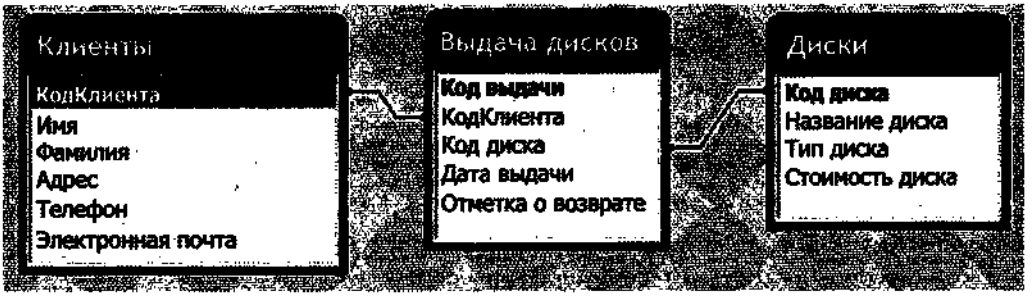


Рис. 4.16

§ 23. Модификация структуры таблицы

После создания базы данных и в процессе работы с ней проектировщик базы данных может обнаружить, что структура таблиц требует некоторых изменений. В этом случае ему необходимо выполнить модификацию ее структуры.

! Под **модификацией структуры** таблицы будем понимать удаление из нее существующих полей, добавление новых полей, изменение имени, типа, размера, формата данных поля.

В СУБД Access модификация структуры таблицы осуществляется в режиме **Конструктора**. Для модификации структуры таблицы необходимо:

1. Открыть базу данных, содержащую таблицу, структуру которой необходимо изменить.
2. Щелкнуть левой клавишей мыши по нужной таблице в окне **База данных**.
3. Щелкнуть левой клавишей мыши по кнопке **Конструктор** Конструктор, чтобы открыть описание структуры таблицы в режиме **Конструктора**.

Пример. Модифицировать структуру таблицы «Страны 2002» в режиме **Конструктора**, представленную на рисунке 4.17, в структуру, показанную на рисунке 4.18.

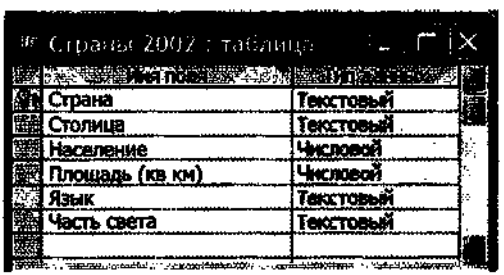


Рис. 4.17

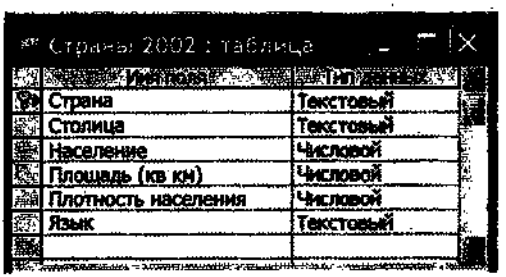


Рис. 4.18

Проведя анализ двух структур, заметим, что нам требуется удалить из старой структуры поле **Часть света**, а в модифицированной структуре описать новое числовое поле **Плотность населения.**

Для удаления поля в режиме **Конструктора** следует:

1. Установить текстовый курсор в удаляемую строку **Часть света**.

2. Щелкнуть левой клавишей мыши по кнопке **Удалить строки** на панели **Стандартная** или выполнить цепочку команд меню: **Правка → Удалить строки.**

Необходимо помнить, что при удалении поля все содержащиеся в нем данные удаляются.

Для вставки поля в режиме **Конструктора** требуется:

1. Установить текстовый курсор в поле строки, над которой будет расположено новое поле. Для нашего примера это будет поле **Язык**.

2. Щелкнуть левой клавишей мыши по кнопке **Добавить строки** на панели **Стандартная** или выполнить цепочку команд меню: **Вставка → Строки.**

3. После вставки новой (пустой) строки в нее вносятся необходимые данные.

Изменение имени поля осуществляется при редактировании соответствующей ячейки в столбце **Имя поля** (см. рис. 4.17). Изменение типа данных в режиме **Конструктора** выполняется с помощью выпадающих меню столбца **Тип данных**, а изменение подтипов, формата и размера данных производится в окне **Свойства поля.**

?

1. **Что понимается под модификацией структуры таблицы?**
2. **Как изменяются имя, тип, размер, формат данных поля в СУБД Access?**

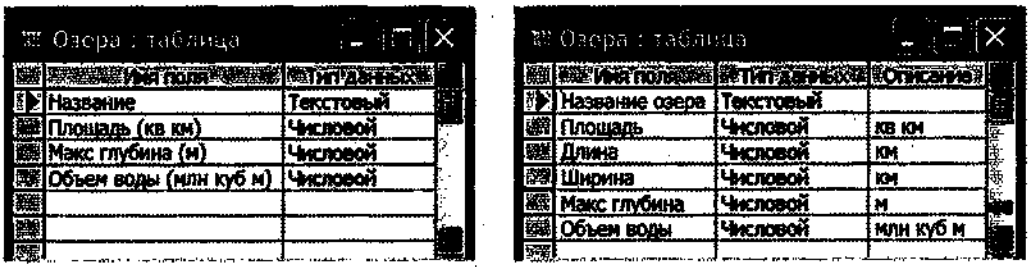
Упражнение

Откройте базу данных «Озера Беларуси», содержащую таблицу «Озера» (рис. 4.19).

Озера : таблица

Название	Площадь (кв.км)	Наибольшая (м)	Объем воды (млн.куб.м)
Нарочь	79,6	24,8	710
Свирь	22,28	8,7	104
Мядель	16,2	24,6	102
Дрывяты	36,1	12	224
Струсто	13	23	94
Снуды	22	16,5	108
Поцех	1,35	9,1	4
Свитязь	2,24	15	8
Недроно	3,72	12,2	18
Войсо	4,88	9,1	14

Рис. 4.19



Озера : таблица

Имя поля	Тип данных
Название	Текстовый
Площадь (кв км)	Числовой
Макс глубина (м)	Числовой
Объем воды (млн куб м)	Числовой

Озера : таблица

Имя поля	Тип данных	Описание
Название озера	Текстовый	
Площадь	Числовой	кв км
Длина	Числовой	км
Ширина	Числовой	км
Макс глубина	Числовой	м
Объем воды	Числовой	млн куб м

Рис. 4.20

В специальном столбце **Описание** укажите единицы измерения. Модифицируйте структуру таблицы «Озера»: измените имена полей, добавьте поля **Ширина, Длина**.

Результат выполнения задания показан на рисунке 4.20.

§ 24. Создание и заполнение формы

24.1. Возможности создания формы в СУБД Access

При работе с таблицей базы данных пользователь для удобства может представить данные таблицы на экране монитора в такой последовательности, в которой ему необходимо (удобно). Для этого создается форма, с помощью которой в дальнейшем можно вводить, редактировать и просматривать данные.

! Формы отображают данные из таблиц или запросов и создаются на их основе.

Рассмотрим создание формы по таблице. При этом достаточно иметь структуру таблицы. Впоследствии данные в таблицу могут быть введены через эту форму.

СУБД Access позволяет создавать формы в режиме **Мастера** и **Конструктора**.

При подготовке формы необходимо в окне **Базы данных** выполнить цепочку команд: **Формы → Создать** [Создать].

В открывшемся окне **Новая форма** предлагаются два основных способа: **Конструктор** и **Мастер форм**, а также **Автоформы в столбец, ленточная, табличная сводная таблица, сводная диаграмма** и др., как показано на рисунке 4.21. В окне **Новая форма** следует выбрать в качестве источника данных нужную таб-

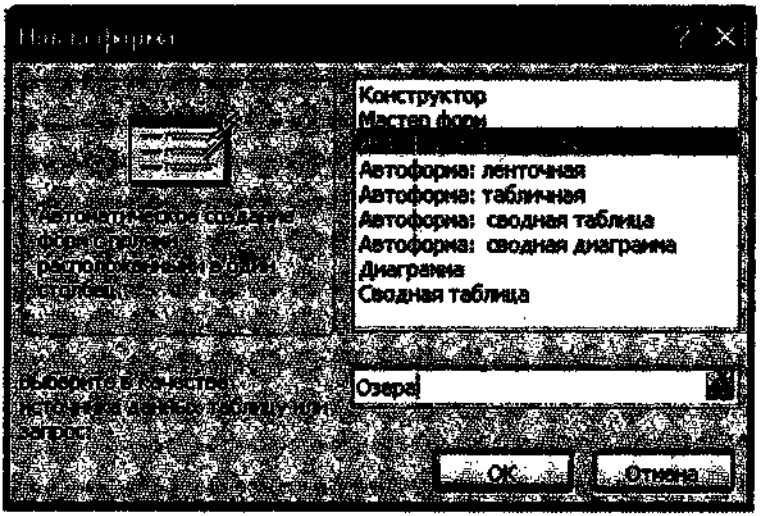


Рис. 4.21

лицу или запрос. Это особенно важно, если форма создается в многотабличной базе данных или в ней уже создано несколько запросов. Быстрое создание форм обеспечивают **Мастер форм** и **Автоформы**, однако этот способ не всегда устраивает пользователя и часто ограничивает возможности оформления форм.

24.2. Создание формы в Конструкторе

Создание формы с помощью **Конструктора** является более сложным процессом, чем с помощью **Мастера форм**.

Для создания формы в режиме **Конструктора** нужно выполнить цепочку команд:

1. Выбрать **Формы** → **Создать** в окне **База данных** → указать пункт меню **Конструктор** в окне **Новая форма**, а также выбрать источник данных — таблицу, например «Озера», на основании которой строится форма.

В результате выполненных действий откроется окно для конструирования формы и специальное окно со списком полей таблицы, как показано на рисунке 4.22.

2. Перетянуть с помощью мыши из **Списка полей** в **Область данных** необходимые поля таблицы, например **Название**. Если список полей не откроется одновременно с **Областью данных**, он может быть открыт с помощью специального значка **Список полей** на панели **Стандартная**. Этот значок представлен на рисунке 4.22.

При перетягивании каждого поля в **Область данных** в нем располагаются **Надпись** (имя) и **Текстовое поле** (данные).

Если размер **Области данных** недостаточен, его можно изменить аналогично тому, как это делается при изменении размера любого окна.

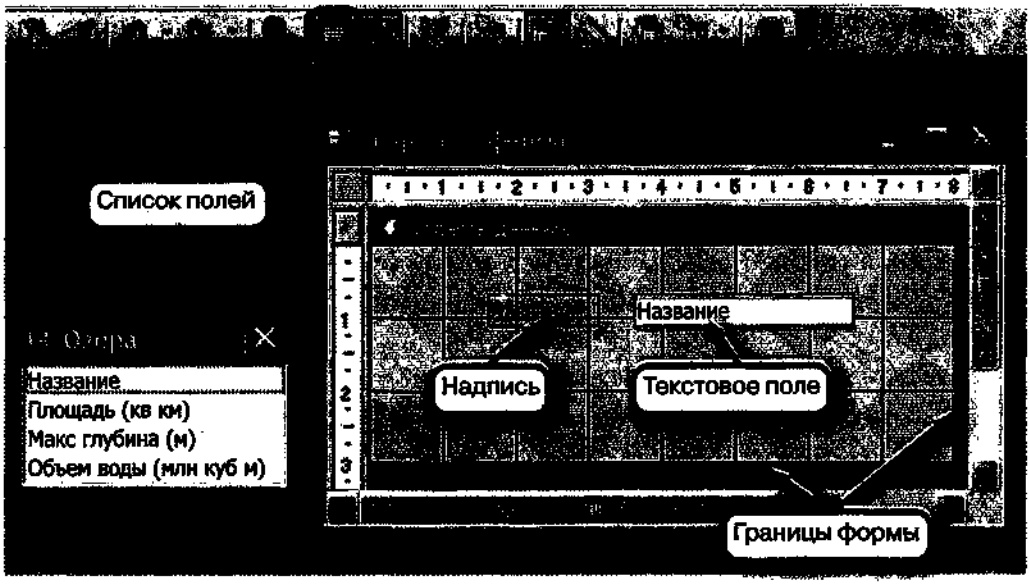


Рис. 4.22

Если элементы **Надпись** и **Текстовое поле** не полностью отображаются в форме, например «Объем воды (млн куб. м)», то могут быть изменены их размеры. Пользователь также может изменять положение этих элементов в **Области данных** и их расположение относительно друг друга. Эти изменения могут быть выполнены с помощью маркеров и мыши после выделения нужных элементов (рис. 4.23). Работать с элементами **Надпись** и **Текстовое поле** в области данных надо очень аккуратно. При перемещении элементов следует обращать внимание на сетку и линейки по горизонтали и вертикали.

Сделав щелчок левой клавишей мыши внутри **Надписи**, пользователь может отредактировать ее текст.

После завершения переноса полей из **Списка полей** и завершения их размещения с помощью кнопки **Автоформат** на панели **Стандартная** пользователь может выбрать стиль оформления формы.

Рис. 4.23

Завершение конструирования формы заканчивается ее сохранением. Формы, созданные в **Мастере форм**, могут быть изменены в **Конструкторе**.

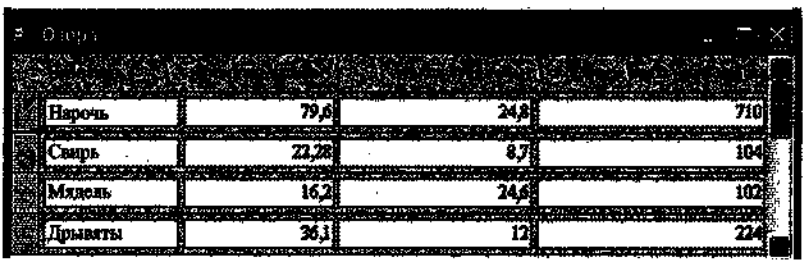
1. Для чего создаются и используются формы?
2. Какие способы создания формы предлагает СУБД Access?

Упражнение

Откройте готовые базы данных. Создайте формы, используя **Конструктор**.

а) База данных «Озера Беларуси», содержащая таблицу «Озера» (см. рис. 4.19).

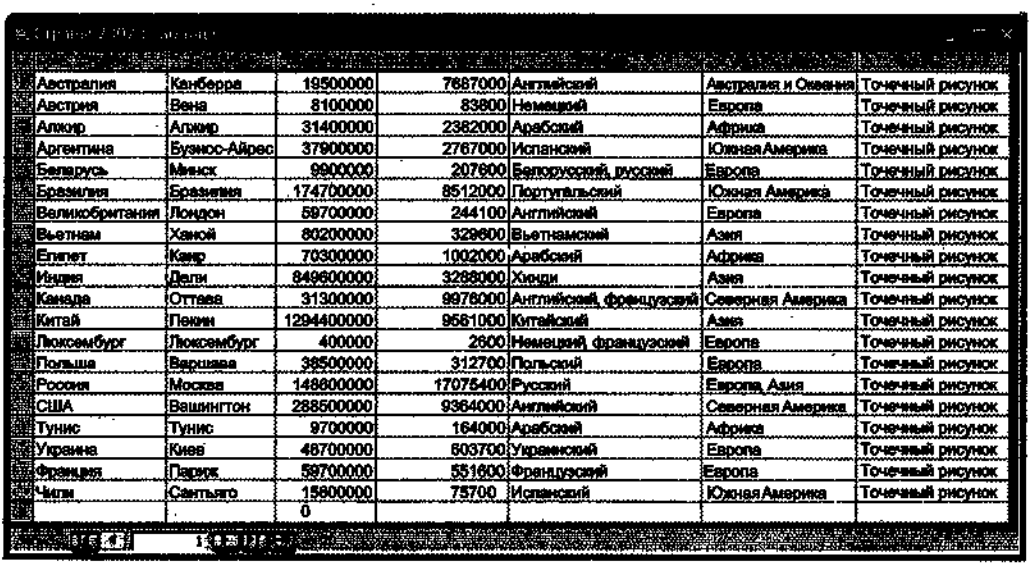
Результат выполнения задания дан на рисунке 4.24.



Нарочь	79,6	24,8	710	
Свирь	22,28	8,7	104	
Мядель	16,2	24,6	102	
Дрываты	36,1	12	224	

Рис. 4.24

б) База данных «Страны», содержащая таблицу «Страны 2002» (рис. 4.25).



Австралия	Канберра	19500000	7687000	Английский	Австралия и Океания	Точечный рисунок
Австрия	Вена	8100000	83800	Немецкий	Европа	Точечный рисунок
Алжир	Алжир	31400000	2382000	Арабский	Африка	Точечный рисунок
Аргентина	Буэнос-Айрес	37900000	2767000	Испанский	Южная Америка	Точечный рисунок
Беларусь	Минск	9900000	207600	Белорусский, русский	Европа	Точечный рисунок
Бразилия	Бразилиа	174700000	8512000	Португальский	Южная Америка	Точечный рисунок
Великобритания	Лондон	59700000	244100	Английский	Европа	Точечный рисунок
Вьетнам	Ханой	80200000	329600	Вьетнамский	Азия	Точечный рисунок
Египет	Каир	70300000	1002000	Арабский	Африка	Точечный рисунок
Индия	Дели	849600000	3288000	Хинди	Азия	Точечный рисунок
Канада	Оттава	31300000	9976000	Английский, французский	Северная Америка	Точечный рисунок
Китай	Пекин	1294400000	9561000	Китайский	Азия	Точечный рисунок
Люксембург	Люксембург	400000	2600	Немецкий, французский	Европа	Точечный рисунок
Польша	Варшава	38500000	312700	Польский	Европа	Точечный рисунок
Россия	Москва	148600000	17075400	Русский	Европа, Азия	Точечный рисунок
США	Вашингтон	288500000	9364000	Английский	Северная Америка	Точечный рисунок
Тунис	Тунис	9700000	164000	Арабский	Африка	Точечный рисунок
Украина	Киев	48700000	603700	Украинский	Европа	Точечный рисунок
Франция	Париж	59700000	551600	Французский	Европа	Точечный рисунок
Чили	Сантьяго	15800000	75700	Испанский	Южная Америка	Точечный рисунок
		0				

Рис. 4.25

Результат выполнения задания показан на рисунке 4.26.

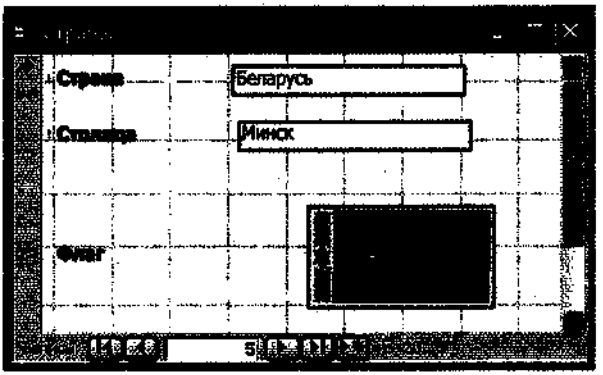
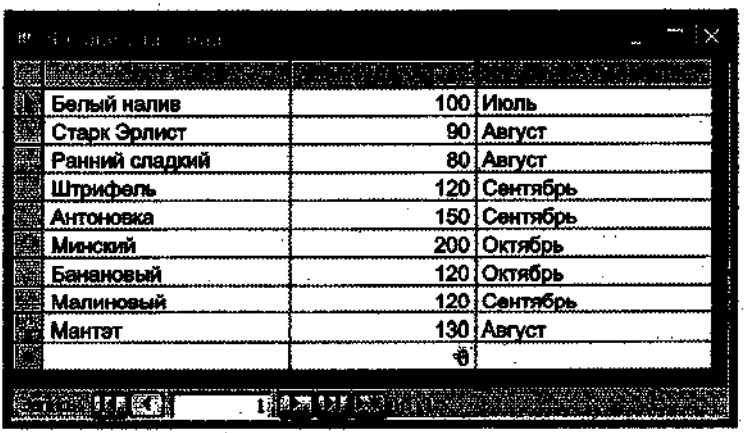


Рис. 4.26

§ 25. Использование фильтров

При работе с таблицами базы данных обычно на экране монитора необходимо отобразить не все записи, а отобранные по определенным критериям.

Для отбора записей таблицы по определенным критериям в СУБД Access используются **фильтры**. Рассмотрим применение фильтров на конкретной однотабличной учебной базе данных «Яблоки». Таблица этой базы представлена на рисунке 4.27.



Белый налив	100	Июль
Старк Эрлист	90	Август
Ранний сладкий	80	Август
Штрифель	120	Сентябрь
Антоновка	150	Сентябрь
Минский	200	Октябрь
Банановый	120	Октябрь
Малиновый	120	Сентябрь
Мантэт	130	Август

Рис. 4.27

СУБД Access предлагает пользователю различные способы фильтрации: *по выделенным данным, по заданному условию* и др.

Выполнение фильтрации по выделенным данным рассмотрим на примере.

Пример 1. Отобразить на экране монитора записи таблицы базы данных, которые содержат сведения о яблоках, созревающих в сентябре.

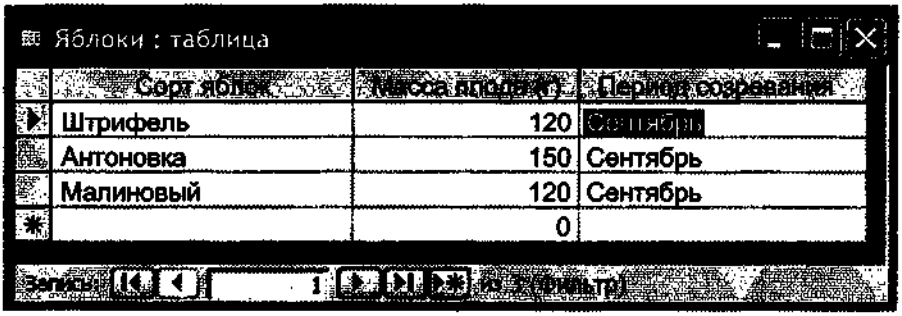


Рис. 4.28

Для выполнения задания необходимо:

1. Щелкнуть левой клавишей мыши на любой ячейке столбца **Период созревания** со значением «Сентябрь».

2. Щелкнуть левой клавишей мыши по кнопке **Фильтр по выделенному** или выполнить цепочку команд: **Записи → Фильтр → Фильтр по выделенному**.

В результате на экране монитора появится окно с таблицей, содержащей все записи, имеющие в поле **Период созревания** значения «Сентябрь», как показано на рисунке 4.28.

При сохранении полученной таблицы последний установленный фильтр запоминается СУБД Access и потом может быть снова применен.

Для отключения установленного фильтра можно воспользоваться кнопкой **Удалить фильтр** на панели **Стандартная**. Повторное нажатие этой кнопки приводит к применению к таблице последнего фильтра.

Фильтрацию по заданному условию удобно задавать после сделанного щелчка левой клавишей мыши по кнопке **Изменить фильтр** на панели **Стандартная** или выполнив цепочку команд: **Записи → Фильтр → Изменить фильтр**. В итоге откроется окно **Фильтр**, в котором задаются условия (рис. 4.29).

Фильтрацию данных по заданному условию покажем на примере.

Пример 2. Отобразить на экране монитора записи таблицы базы данных, которые содержат сведения о яблоках, созревающих в сентябре и имеющих массу 120 г.

С помощью выпадающих меню в окне **Фильтр** зададим условие «Масса плода (г) =120 и Период созревания = Сентябрь» (см. рис. 4.29).

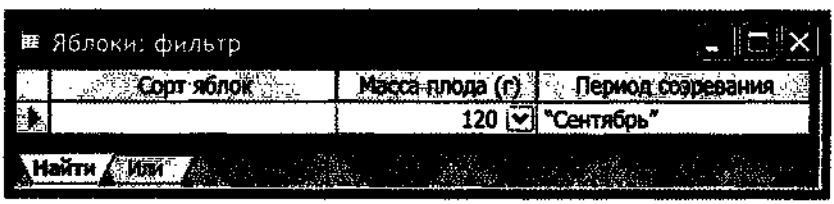


Рис. 4.29

Яблоки : таблица

Сорт яблок	Масса плода (г)	Период созревания
Антоновка	120	Сентябрь
Малиновый	120	Сентябрь
	0	

Запись: 1 из 3

Рис. 4.30

После применения фильтра получим таблицу, содержащую две записи, как показано на рисунке 4.30.

1. Для чего используются фильтры?
2. Какие операции фильтрации предлагает пользователю СУБД Access?

Упражнение

Создайте однотабличную базу данных «Аквариумные рыбки». Структура таблицы и ее содержание даны на рисунках 4.31 и 4.32.

Аквариумные рыбки : таблица

Имя поля	Тип данных
Название рыбки	Текстовый
Место происхождения	Текстовый
Длина (см)	Числовой

Рис. 4.31

Аквариумные рыбки : таблица

Название рыбки	Место происхождения	Длина (см)
Гамбузия	Южная Америка	5
Плятипецилиус черный	Мексика	4
Зеленый меченосец	Мексика	11
Тетра-фон-рио	Южная Америка	2
Бархатный гирардинус	Южная Америка	5
Макропод	Китай	9
Гурами жемчужный	Индия	10
Лялиус	Индия	5
Неон	Южная Америка	4
		0

Запись: 1 из 9

Рис. 4.32

Выполните отбор нужных записей таблицы, используя:

а) фильтрацию по выделенному «Южная Америка»;

б) фильтрацию по выделенному «Индия»;

в) фильтрацию по условию «Китай» или «Индия»;

г) фильтрацию по условию «Южная Америка» и Длина (см) >3.

§ 26. Поиск данных с помощью запросов

26.1. Запрос на выборку данных по одной таблице

Поиск информации в базах данных выполняется через запросы. С помощью **запроса** СУБД Access выбирает и отображает записи из таблиц базы данных, которые отвечают заданным условиям.

Запрос может формироваться на основе одной либо нескольких связанных таблиц или запросов, построенных ранее.

СУБД Access поддерживает создание запросов на выборку с помощью **Мастера** и **Конструктора**.

Покажем, как осуществляется поиск информации на примерах создания запросов на основе таблицы «Кольца» базы данных «Украшения из драгоценных металлов», которая представлена на рисунке 4.33.

Пример 1. Выполнить поиск информации в таблице базы данных, который покажет сведения о металле, камне и цене драгоценных колец.

Для создания соответствующего запроса необходимо выполнить следующие действия:

1. Щелкнуть левой клавишей мыши по кнопке **Запросы** в окне **Базы данных**.

2. Дважды щелкнуть левой клавишей мыши на строке **Создание запроса с помощью конструктора** или выполнить цепочку **Создать → Конструктор → Ок**.

Кольца : таблица

Металл	Проба	Камень	Цена
Белое золото	750	Бриллиант	3 060 000,00р.
Желтое золото	585	Бриллиант	2 025 000,00р.
Белое золото	585	Бриллиант	1 020 000,00р.
Серебро	925	Гранат	630 000,00р.
Красное золото	750	Сапфир	1 260 000,00р.
Желтое золото	585	Сапфир	1 710 000,00р.
Серебро	585	Оникс	900 000,00р.
Красное золото	585	Бриллиант	2 229 000,00р.
Белое золото	750	Бриллиант	1 680 000,00р.
Красное золото	585	Янтарь	468 000,00р.
Серебро	925	Гранат	702 000,00р.
Серебро	925	Гранат	576 000,00р.
Красное золото	925	Бриллиант	6 525 000,00р.
	0		0,00р

Запись: 1 из 13

Рис. 4.33

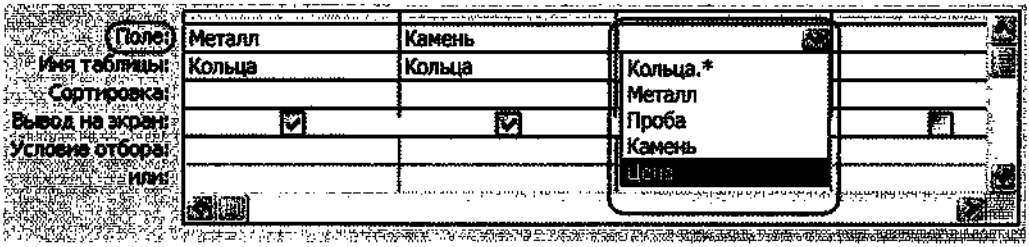


Рис. 4.34

3. В окне **Добавление таблицы** выбрать таблицу с именем **Кольца** и щелкнуть левой клавишей мыши по кнопке **Добавить.**

4. В результате выполненных действий откроется окно **Запрос на выборку Конструктора**, как показано на рисунке 4.34. В этом окне в строке **Поле** необходимо щелкнуть левой клавишей мыши по кнопке со стрелкой. Из открывающегося списка требуется последовательно выбирать поля, по которым будет формироваться запрос: **Металл, Камень, Цена.**

Установка флажка в выбранном столбце строки **Вывод на экран** позволит вывести на экран монитора информацию, которая содержится в соответствующих полях таблицы (в нашем примере: **Металл, Камень, Цена**).

В результате выполнения запроса на экране монитора появится временная таблица (рис. 4.35), которую при необходимости можно сохранить.

Пример 2. Осуществить поиск информации в таблице базы данных, который покажет сведения о камне всех серебряных колец, цена которых больше 660 000 р.

Кольца Запрос1 : запрос на выборку

Металл	Камень	Цена
Белое золото	Бриллиант	3 060 000,00р.
Желтое золото	Бриллиант	2 025 000,00р.
Белое золото	Бриллиант	1 020 000,00р.
Серебро	Гранат	630 000,00р.
Красное золото	Сапфир	1 260 000,00р.
Желтое золото	Сапфир	1 710 000,00р.
Серебро	Оникс	900 000,00р.
Красное золото	Бриллиант	2 229 000,00р.
Белое золото	Бриллиант	1 680 000,00р.
Красное золото	Янтарь	468 000,00р.
Серебро	Гранат	702 000,00р.

Рис. 4.35

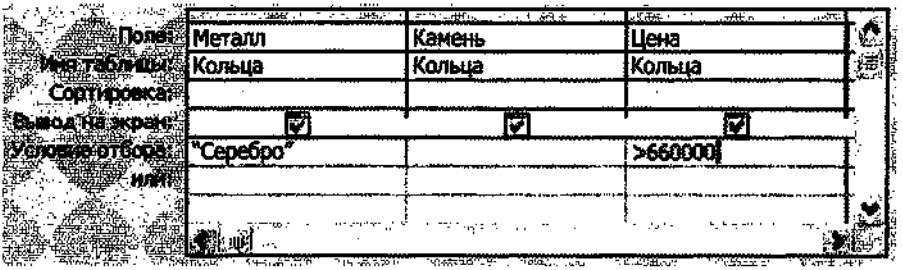


Рис. 4.36

В этом примере необходимо выполнить запрос **Металл=«Серебро»** и **Цена > 660 000**.

В окне **Конструктора** запроса в строке **Условие отбора** сформируем запрос, как показано на рисунке 4.36.

Для формирования данного запроса в **Конструкторе** необходимо:

1. В строке **Поле** щелкнуть левой клавишей мыши по кнопке со стрелкой. Из открывающегося списка последовательно выбрать поля, по которым будет формироваться запрос: **Металл, Камень, Цена**.

2. Установить флажок в выбранном столбце строки **Вывод на экран**. Это позволит вывести на экран монитора информацию, которая содержится в соответствующих полях таблицы базы.

3. Ввести необходимые значения в соответствующем столбце строки **Условие отбора**, как показано на рисунке 4.36.

В результате выполнения запроса на экране монитора появится временная таблица (рис. 4.37).

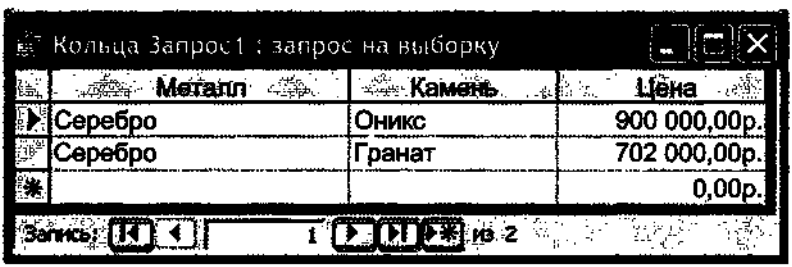


Рис. 4.37

Формирование запросов по двум либо нескольким связанным таблицам осуществляется аналогично. При этом в строке Имя таблицы необходимо указать таблицы, данные из которых будут использоваться в запросе.

1. Для чего используются запросы?
2. На основе каких объектов СУБД Access формируется запрос?

Упражнения

1. Откройте таблицу «Страны 2002» базы данных «Страны» (см. рис. 4.25) и создайте запросы по этой таблице:
   а) простой запрос, содержащий все записи полей **Страна, Столица, Население, Язык**;
   б) запрос, имеющий записи с полями **Страна** и **Население**<<10 млн человек»;
   в) запрос с записями **Столица** и **Язык** = «арабский».
2. Откройте таблицу «Озера» базы данных «Озера» (см. рис. 4.19) и создайте запросы по этой таблице:
   а) простой запрос, содержащий все записи полей **Название, Площадь (кв км), Макс глубина (м)**;
   б) запрос, имеющий записи с полями **Название, Объем воды (млн куб м)**>10 и **Объем воды (млн куб м)** < 20.

26.2. Параметрический запрос

Создание параметрического запроса облегчает работу пользователя, так как не требует постоянного изменения структуры запроса в окне **Конструктора**.

Параметрический запрос — это запрос, при выполнении которого пользователю необходимо ввести значение требуемого ему параметра.

Пример. Предположим, что у нас имеется база данных, содержащая таблицу «Водопад», которая представлена на рисунке 4.38. Необходимо осуществить поиск информации, содержащей сведения о всех водопадах конкретной страны.

Создание параметрического запроса начинается с конструирования простого запроса по выбору. В данном примере пользователю необходимо при каждом

Водопад : таблица

Код	Название водопада	Страна	Высота (м)
1	Анхель	Венесуэла	1054
2	Утигард	Норвегия	800
3	Мутарази	Зимбабве	762
4	Йосемитский	США	739
5	Эспеландс	Норвегия	703
6	Мардалсфосс	Норвегия	657
7	Кукенан	Венесуэла	610
8	Сатерленд	Новая Зеландия	580
9	Киле	Норвегия	561
(Счетчик)			0

Запись: 1 из 9

Рис. 4.38

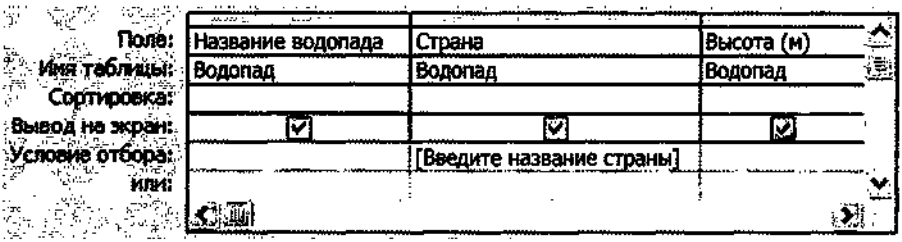


Рис. 4.39

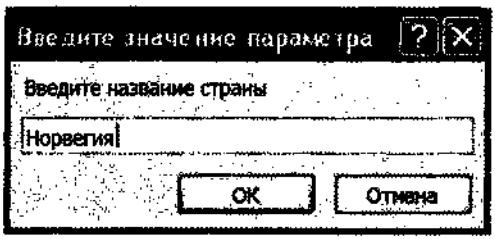


Рис. 4.40

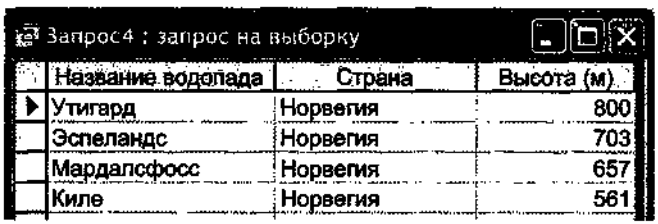


Рис. 4.41

выполнении запроса вводить название страны. Поэтому в строке **Условие отбора** указывается текст подсказки в квадратных скобках **[Введите название страны]**, как показано на рисунке 4.39.

При выполнении параметрического запроса открывается специальное окно для ввода названия страны, например «Норвегия» (рис. 4.40).

В результате выполнения запроса на экране монитора появляется таблица, содержащая сведения о водопадах Норвегии, как представлено на рисунке 4.41.

26.3. Вычисления в запросе

При формировании запросов иногда требуется получить информацию, которая содержит вычисления, полученные в результате выполнения арифметических операций.

Для выполнения таких запросов могут быть использованы арифметические Выражения.

Поле:	Название водопада	Страна	Высота(км): [Высота (м)]/1000
Имя таблицы:	Водопад	Водопад	
Групповая операция:	Группировка	Группировка	Группировка
Сортировка:			
Вывод на экран:	☑	☑	☑
Условие отбора:			
или:			

Рис. 4.42

Пример 3. В таблице «Водопад» (см. рис. 4.38) определить высоту всех водопадов в километрах.

Для создания такого запроса в строке **Поле** записывается арифметическое выражение «[Высота (м)]/1000» с пояснительным текстом «Высота (км)», как представлено на рисунке 4.42.

После выполнения запроса с выражением на экране монитора появляется таблица, представленная на рисунке 4.43.

При написании выражений пользователь может воспользоваться возможностями окна **Построитель выражений**, которое открывается после щелчка левой клавишей мыши по кнопке **Построить**.

Запрос1 : запрос на выборку

Название водопада	Страна	Высота(км)
Анхель	Венесуэла	1,054
Йосемитский	США	0,739
Кукенан	Венесуэла	0,61
Киле	Норвегия	0,561
Мутарази	Зимбабве	0,762
Мардалсфосс	Норвегия	0,657
Сатерленд	Новая Зеландия	0,58
Утигард	Норвегия	0,8
Эспеландс	Норвегия	0,703

Запись: 2 из 9

Рис. 4.43

1. Для чего используется параметрический запрос?
2. Могут ли в запросах использоваться вычисления?

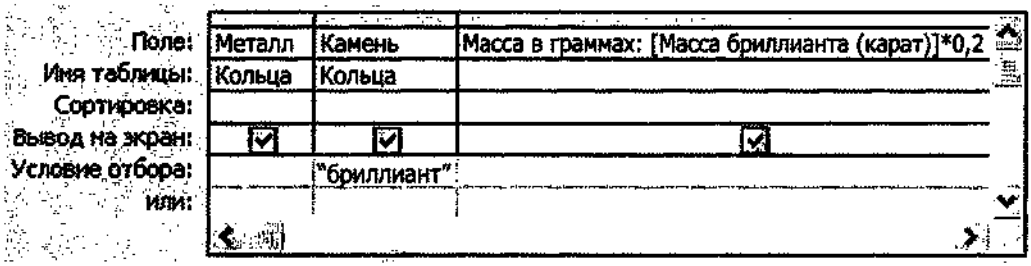


Рис. 4.44

Упражнение

Создайте запрос с **Выражением** на основе таблицы «Кольца» базы данных «Украшения из драгоценных металлов» (см. рис. 4.33). Переведите с помощью **Выражения** массу бриллиантов из каратов в граммы, как показано в окне **Конструктора запросов**.

Необходимый запрос показан на рисунке 4.44.

§ 27. Сортировка записей в таблице

Довольно часто при работе с таблицами базы данных возникает необходимость упорядочить их записи в определенной последовательности, т.е. отсортировать.

С понятием сортировки данных мы уже знакомились ранее.

Под сортировкой записей в таблице базы данных будем понимать процесс их упорядочения в определенной последовательности по значению одного из полей.

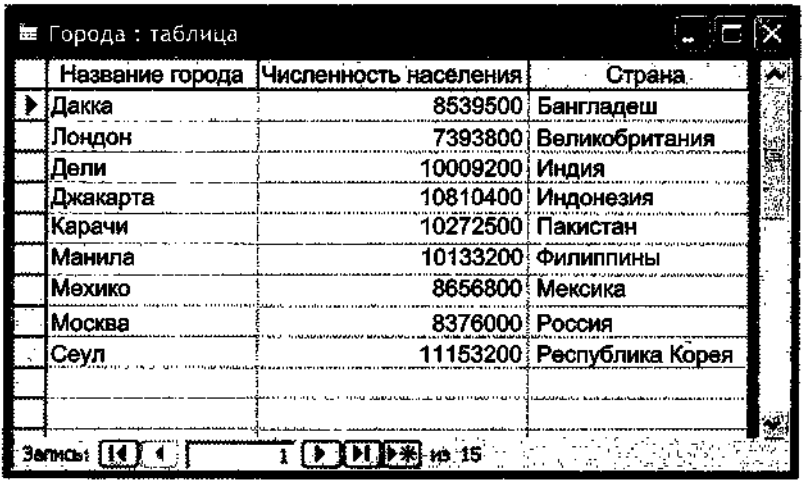
В зависимости от типа данных в поле (числовое, текстовое или др.), определенных для сортировки, все записи в таблице базы данных размещаются:

- по величине числа, если тип данных числовой;
- по алфавиту, если тип данных текстовый (символьный);
- по дате и времени, если данные в поле содержат значения даты и времени.

Записи в таблице базы данных сортируются по значению ключевого поля. Если ключевое поле не задано, то в таблице базы данных записи хранятся в том порядке, в каком они были введены.

СУБД Access предоставляет пользователям возможность сортировать записи в таблице. Для сортировки записей по данным конкретного поля необходимо установить курсор в любую строку соответствующего столбца и сделать щелчок левой клавишей мыши по одной из кнопок на панели **Стандартная**:

 — **Сортировка по возрастанию,**  — **Сортировка по убыванию.**



Города : таблица

Название города	Численность населения	Страна
Дакка	8539500	Бангладеш
Лондон	7393800	Великобритания
Дели	10009200	Индия
Джакарта	10810400	Индонезия
Карачи	10272500	Пакистан
Манила	10133200	Филиппины
Мехико	8656800	Мексика
Москва	8376000	Россия
Сеул	11153200	Республика Корея

Запись: 1 из 15

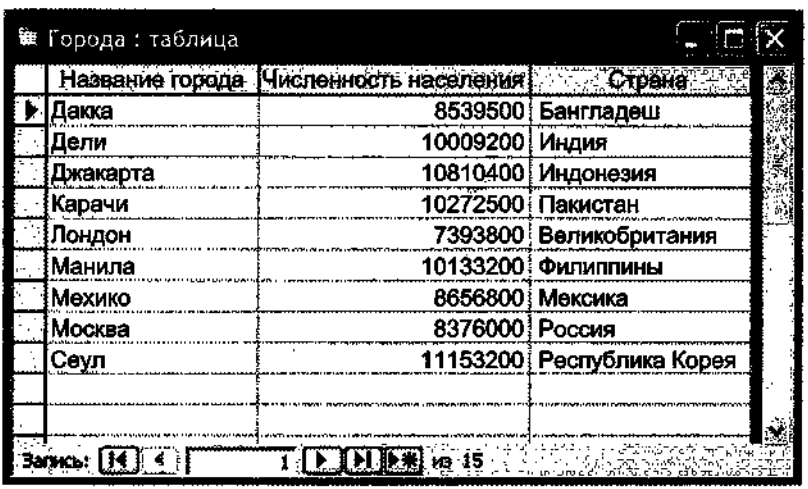
Рис. 4.45

Продемонстрируем процесс сортировки записей в таблице «Население городов мира», фрагмент которой представлен на рисунке 4.45.

Пример. Отсортировать записи таблицы «Города» по данным поля **Название города**, расположив записи в алфавитном порядке.

1. Щелкнем левой клавишей мыши в любой строке поля **Название города**.
2. Щелкнем левой клавишей мыши по кнопке **Сортировка по возрастанию**  или выполним цепочку команд: **Записи → Сортировка → Сортировка по возрастанию**.

Фрагмент результата сортировки представлен на рисунке 4.46.



Города : таблица

Название города	Численность населения	Страна
Дакка	8539500	Бангладеш
Дели	10009200	Индия
Джакарта	10810400	Индонезия
Карачи	10272500	Пакистан
Лондон	7393800	Великобритания
Манила	10133200	Филиппины
Мехико	8656800	Мексика
Москва	8376000	Россия
Сеул	11153200	Республика Корея

Запись: 1 из 15

Рис. 4.46

1. Для чего используется сортировка данных?
2. Какой процесс называют сортировкой записей таблицы базы данных?

Упражнения

1. Откройте таблицу «Страны 2002» базы данных «Страны» (см. рис. 4.25) и выполните сортировку записей в этой таблице:
   а) расположите записи поля **Страна** в алфавитном порядке;
   б) расположите записи поля **Население** в порядке возрастания;
   в) упорядочьте записи поля **Площадь** в порядке убывания.
2. Создайте базу данных «Реки Беларуси», фрагмент таблицы которой показан на рисунке 4.47:

Реки : таблица

Название	Площадь бассейна (кв км)	Длина (км)	Густота речной сети (км/кв км)
Ловать	21900	536	0,45
Днепр	504000	2201	0,39
Вилия	25100	498	0,44
Западный Буг	39400	772	0,42
Сож	42140	648	0,38
Припять	121000	761	0,42
Березина	24500	613	0,35
Неман	98200	937	0,47
Западная Двина	87900	1020	0,45
			0

Запись: 1 из 9

Рис. 4.47

   а) упорядочьте записи поля **Длина (км)** в порядке возрастания;
   б) расположите записи поля **Густота речной сети (км/кв км)** в порядке убывания.

§ 28. Создание отчетов

Одним из способов просмотра и распечатки итоговых сведений из базы данных является создание отчетов.

Под отчетом понимается документ, содержание которого формируется по определенному запросу на основе информации, размещенной в базе данных.

В отчетах данные представляются в удобном виде. Отчеты выводятся на экран монитора или печатаются на принтере.

СУБД Access предоставляет пользователю несколько способов создания **Отчета**: **Автоотчет, Мастер отчетов** и **Конструктор**. Отчеты являются самостоятельными объектами базы данных и формируются на основании таблиц или запросов.

После просмотра полученного отчета он может быть сохранен.

Рассмотрим создание отчета с помощью **Мастера** на примере многотабличной базы данных «Библиотека дисков».

Пример. Создать отчет, содержащий сведения из трех таблиц базы данных «Библиотека дисков» (см. рис. 4.11). Из таблицы «Клиенты» выбрать фамилию клиента, его имя, телефон, из таблицы «Диски» — название диска, а из таблицы «Выдача дисков» — дату выдачи и отметку о возврате.

Для этого необходимо выполнить следующие действия:

1. Открыть базу данных «Библиотека дисков» и дважды щелкнуть левой клавишей мыши по надписи **Создание отчета с помощью мастера** в окне **Базы данных.**

2. В окне **Создание отчетов**, поочередно активизируя название таблиц базы данных в списке **Таблицы и запросы**, перенести все необходимые поля из окна **Доступные поля** в окно **Выбранные поля**, как показано на рисунке 4.48.

3. В окне **Создание отчетов**, последовательно нажимая кнопку **Далее**, можно при необходимости указать вид представления данных, уровень их группировки,

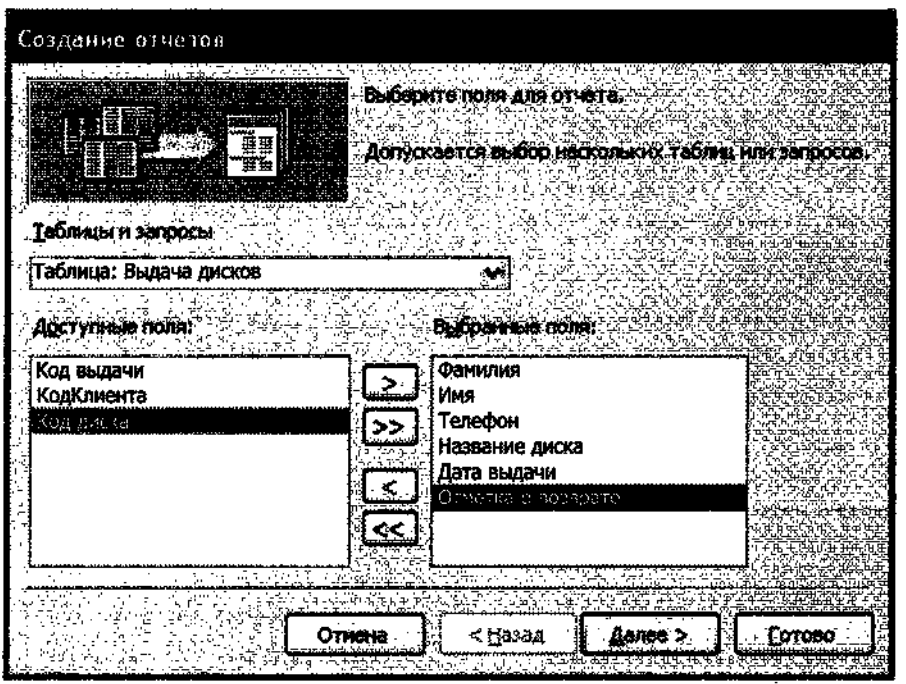
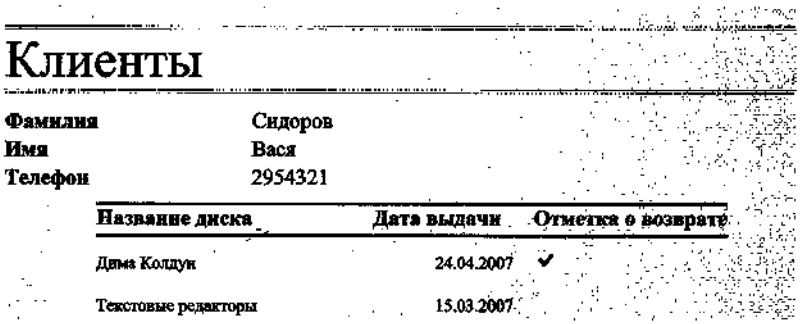


Рис. 4.48



Клиенты

Фамилия	Сидоров
Имя	Вася
Телефон	2954321

Название диска	Дата выдачи	Отметка о возврате
Дима Колдун	24.04.2007	✔
Текстовые редакторы	15.03.2007	

Рис. 4.49

выбрать порядок сортировки, например по дате выдачи дисков, макет размещения данных и стиль оформления данных, например **по левому краю 1**, **Строгий**, а в последнем окне нажать кнопку **Готово**.

В результате на экране монитора отображается отчет, фрагмент которого представлен на рисунке 4.49.

Созданный отчет может размещаться сразу на нескольких страницах и быть выведен в книжной либо альбомной форме, аналогично документу текстового редактора Word с помощью цепочки команд меню: **Файл → Печать → ...** .

Подготовка отчета с помощью **Конструктора** осуществляется несколько иначе. Вначале пользователь может создать и сохранить отчет с помощью **Мастера**. Затем в окне **Конструктора** можно изменить расположение элементов в отчете, удалить некоторые элементы, изменить или отредактировать надписи и т. д.

1. Для чего используются отчеты в базах данных?
2. Что называют отчетом в базе данных?
3. Какие объекты базы данных являются источниками для создания отчетов?

Упражнение

Откройте таблицу «Страны 2002» базы данных «Страны» (см. рис. 4.25) и создайте отчет в СУБД Access с помощью **Мастера**.

Результат выполнения задания показан на рисунке 4.50.

Страны 2002

Страна	Население	Площадь (кв км)
Австрия	8100000	83855
Беларусь	9900000	207600
Россия	148600000	17075400

Рис. 4.50

ВЫЧИСЛИТЕЛЬНЫЕ МЕТОДЫ И ТЕХНОЛОГИИ

§ 29. Основы работы в системе Mathcad

29.1. Основные элементы интерфейса

Mathcad является универсальной системой компьютерной математики. В ней сочетаются эффективные методы вычислений, символьные преобразования, программирование, визуализация и анимация результатов с простотой записи выражений. Интерфейс Mathcad достаточно прост, чтобы пользователь смог быстро освоить основные приемы работы. После запуска открывается окно, основные элементы которого представлены на рисунке 5.1.

Главное меню Mathcad отличается от аналогичного в Excel лишь двумя специфичными пунктами **Math (Математика)** и **Symbolics (Символьные операции)**, очень похожи и панели инструментов **Стандартная** и **Форматирование**. Конечно, Mathcad имеет гораздо большие возможности обработки и визуализации данных по сравнению с табличным процессором Excel.

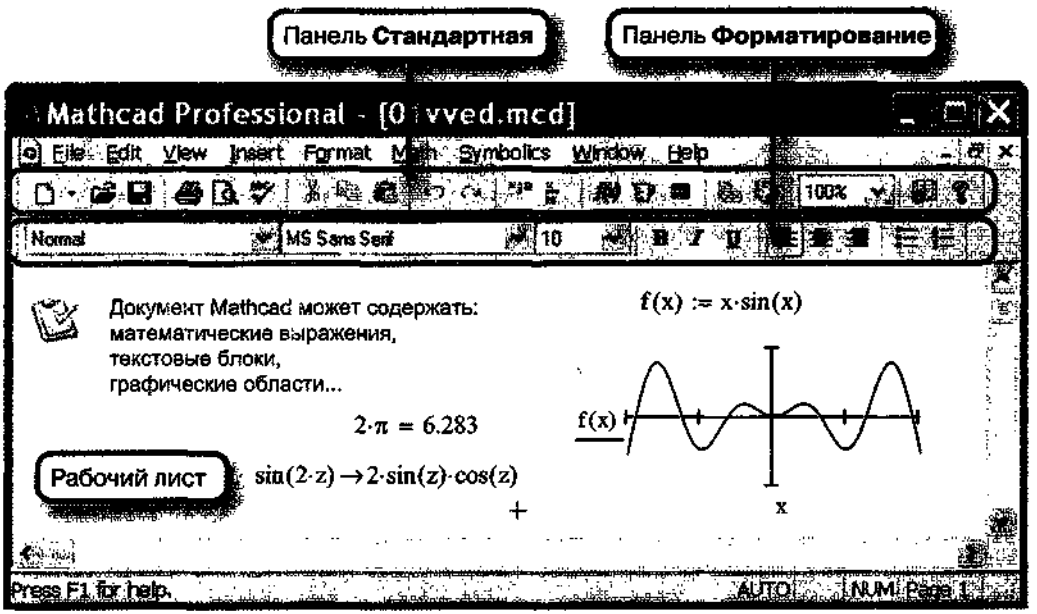


Рис. 5.1

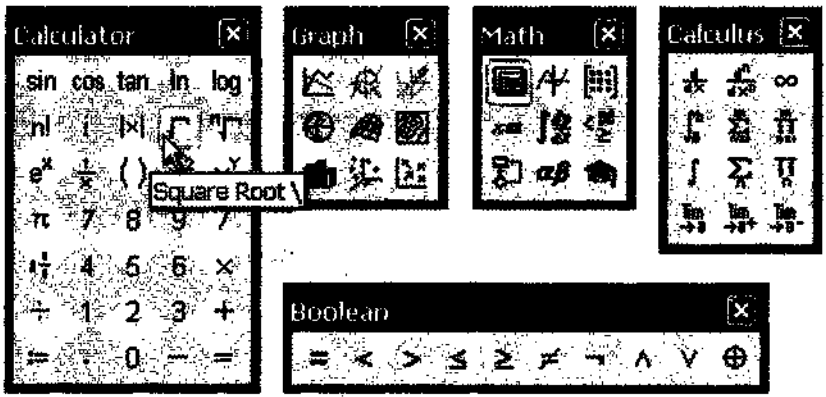


Рис. 5.2

Ввод математических выражений в системе Mathcad напоминает их запись в тетради. Упрощению ввода способствуют шаблоны и специализированные панели инструментов (рис. 5.2).

По мере необходимости они вызываются с помощью меню **View (Вид)** или панели **Math (Математика)**. Размещение панелей, их форму и размеры можно настраивать.

Панели содержат наиболее часто используемые функции, конструкции и отсутствующие на клавиатуре символы операций. Панель **Calculator (Калькулятор)** дает возможность вводить простейшие операции: арифметические действия, возведение в степень и извлечение корня, вычисление модуля и тригонометрических функций; панель **Calculus (Матанализ)** — производные, интегралы, суммы, произведения; панель **Boolean (Логические)** — логические операции. Панель **Graph (Графики)** помогает строить разнообразные графики. Многие кнопки панелей продублированы так называемыми «горячими клавишами», которые подсказываются при подведении курсора к символу на панели (см. рис. 5.2).

29.2. Основы работы в среде Mathcad

Документ Mathcad представляет собой рабочий лист, на котором размещаются блоки трех типов: математические выражения, фрагменты текста и графические области. Текстовые блоки размещаются произвольно. Их часто используют для комментариев. Расположение математических выражений и графических областей определяется порядком исполнения: *слева направо, сверху вниз*. Блоки можно выделять, перемещать по рабочему листу, копировать в буфер обмена, вставлять в документ. Место ввода очередного блока указывается курсором, который вне блоков имеет вид красного крестика (см. рис. 5.1).

Ввод текста начинают с кавычек «"», при этом курсор превращается в красную вертикальную черточку | (**текстовый курсор**). Текст, который вводится без начальных кавычек, воспринимается системой как математическое выражение, при этом курсор в зависимости от места в выражении принимает одну из форм ⌊ ⊥ ⌋ (**формульный курсор**).

Математические выражения содержат переменные, константы, операторы, функции и управляющие структуры. **Имена (идентификаторы)** переменных, констант и функций составляются из букв (латинских или греческих) и цифр. Первым символом должна быть буква. Mathcad различает строчные и прописные буквы: например, by, By и BY — разные имена.

Переменным до их использования необходимо *присвоить значения*. В качестве *оператора присваивания*, как и в языке Паскаль, используется знак «:=», который вводится нажатием клавиши «:» (двоеточие на клавиатуре). Такое присваивание называют **локальным**, оно действует до нового присваивания. Для *вычисления и вывода значения* выражения, константы или переменной предназначен знак «=» (равно). Попытка использовать переменную до присваивания значения приводит к ошибке. При этом будет выведено сообщение: «Переменная или функция не определена» (рис. 5.3).

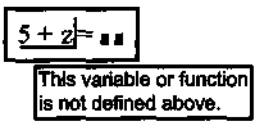


Рис. 5.3

Значения **глобальных** переменных известны в любом месте документа, в том числе и до точки их объявления. Для глобального присваивания предназначен знак «≡», который вводится с панели инструментов **Evaluation (Вычисления)**.

Некоторые переменные (их называют **системными** или **встроенными** — Build-In) определены в самой системе, например $\pi = 3{,}142$, $g = 9{,}807$ м/с^2, точность TOL. Конечно, их можно переопределить присвоением нового значения, например задать ускорение свободного падения на Луне $g = 1{,}62$ м/с^2.

Рассмотрим простейшие приемы работы в системе Mathcad.

Ввод и редактирование текстов и математических выражений напоминает аналогичные действия в текстовом редакторе Word.

! В Mathcad дробная часть от целой в десятичной дроби отделяется **точкой**. Отсутствующие на клавиатуре символы (например, π) вводятся с панелей инструментов.

Пример 1. Вычислить значение выражения $1{,}5 \cdot \frac{\pi}{7}$.

Наберем последовательность символов 1.5*π/7 и завершим ее знаком равно «=». Щелкнем левой клавишей мыши в любом месте рабочего листа. Получим результат в виде $1.5 \cdot \frac{\pi}{7} = 0.673$.

Для удаления или замены символа его выделяют формульным курсором с помощью мыши и клавиш управления курсором, при этом выделенный символ находится внутри синего уголка. Направление уголка изменяют клавишей Ins. На группу символов выделение распространяют клавишей Пробел.

Пример 2. Скопировать и отредактировать выражение из примера 1.

Выделим блок с созданным выражением, скопируем в буфер обмена (Ctrl+C), а затем три раза вставим на рабочий лист (Ctrl+V). Перемещая формульный курсор, выполним удаления, вставки и замены символов в копиях в соответствии с приведенными образцами:

$$2\cdot\frac{\pi}{5} = 1.257 \qquad \frac{2}{3}\cdot\frac{\pi+1}{5+1} = 0.46 \qquad \frac{(\pi+1)\cdot 5}{3\cdot(5+1)-3} = 1.381$$

Пример 3. Вычислить значения выражений $|a-b|$, a^5, $\sqrt{b}$, $\sqrt[3]{a^2+b^2}$, $\sin c + \operatorname{tg} c$ при $a=2$, $b=5$, $c=\pi/6$.

Присвоим значения переменным: $a := 2$ $b := 5$ $c := \frac{\pi}{6}$

Наберем выражения, вводя символы операций вычисления модуля, возведения в степень, извлечения корня и тригонометрические функции с панелей инструментов, и получим результаты вычислений:

$$|a-b| = 3 \qquad a^5 = 32 \qquad \sqrt{b} = 2.236 \qquad \sqrt[3]{a^2+b^2} = 3.072 \qquad \sin(c)+\tan(c) = 1.077$$

Пример 4. Ввести и отформатировать текст *Простейшие вычисления*.

Установим курсор (красный крестик) на свободное место. Введем кавычки «“» и наберем текст *Простейшие вычисления*. Выделим текст. С помощью панели **Форматирование** или меню **Format (Формат) → Text (Текст)** установим следующие параметры текста: шрифт Arial Cyr, курсивный, 12 пт, цвет синий.

?

1. Какие типы блоков может содержать документ Mathcad?
2. Какой вид может принимать курсор?
3. Какой вид принимает курсор в блоке математических выражений?
4. Какие клавиши используются при выделении частей выражения?

Упражнение

Вычислите значение выражения:

а) $\frac{5^4+3^3}{6^5-7^3}$;

б) $\frac{2{,}7^2-0{,}8^2}{3\cdot 0{,}4-7\cdot 1{,}5}$;

в) $\sqrt{\frac{2}{7}+\sqrt{3}}$;

г) $\frac{12\sqrt{75}}{5\sqrt{3}}-2\frac{3}{7}$;

д) $\left(2\sqrt{15}+\sqrt[4]{6}\right)^2-\sqrt[3]{7}\left(\sqrt{2}+\sqrt{5}\right)$;

е) $1{,}5^{-2}+2^{-3}+\sqrt{1{,}7}$;

ж) $\sqrt{6+\sqrt{20}}\cdot\sqrt{6-\sqrt{20}}$;

з) $\left|\sin(0{,}1\pi)-\sin(0{,}4\pi)\right|$;

и) $\cos(0{,}2\pi)+\left|0{,}5-\operatorname{tg}(0{,}8)\right|$;

к) $\operatorname{tg}(0{,}23)\cdot\sqrt{\sin(1{,}5)-\cos(0{,}2)}$.

§ 30. Функции. Символьные операции

В математике функцией называют закон, по которому каждому значению x ставится в соответствие определенное значение y. Основным способом задания функции является формула.

Mathcad предлагает большой набор встроенных функций, вызываемых командой **Insert → Function (Вставка → Функция)** или кнопкой ***f(x)*** на панели **Стандартная**.

Можно создавать и собственные функции. Они определяются так:

имя (аргументы) := выражение.

Слева после имени в скобках перечисляются аргументы, а справа после знака присваивания записывается выражение, которое может содержать переменные, константы, формулы и функции.

Пример 1. Определить функции $f(x)=x^2-7x-8$, $fm(x)=|f(x)|$ и $fp(x)=f'(x)+5$ и вычислить их значения при $x=5$ и $x=10$.

Введем имя и аргумент первой функции, знак присваивания и выражение:

$$f(x) := x^2 - 7\cdot x - 8$$

При вычислении значений функции возможны два способа передачи значений аргументов: непосредственная подстановка и предварительное присваивание:

$$f(5) = -18 \qquad x := 10 \qquad f(x) = 22$$

Во второй функции $fm(x)$ используем модуль, а в третьей $fp(x)$ — производную первой:

$$fm(x) := |f(x)| \qquad fm(5) = 18 \qquad fm(x) = 22$$

$$fp(x) := \frac{d}{dx}f(x) + 5 \qquad fp(5) = 8 \qquad fp(x) = 18$$

Шаблон производной $\frac{d}{dx}$ вызывается с панели **Calculus (Матанализ)**.

Пример 2. Определить функцию $y(t, x) = A\cos(\omega t - kx)$, описывающую зависимость смещения точки волны от времени t и расстояния x до источника. Вычислить ее значения при $t = 0$, $x = 0$ и $t = 1$, $x = 1$.

Переменные A, ω и k являются *параметрами*. Присвоим им значения до задания функции:

$$A := 10 \quad \omega := 2 \quad k := 3$$

Аргументами функции являются переменные t и x. Укажем их в левой части определения функции в скобках. Вычислим значения $y(t, x)$:

$$y(t,x) := A\cdot\cos(\omega\cdot t - k\cdot x) \quad y(0,0) = 10 \quad y(1,1) = 5.403$$

Система Mathcad позволяет не только производить разнообразные вычисления, но и выполнять **символьные операции**, результатом которых являются не числа, а формулы. Для *символьного преобразования* и *вывода* выражений предназначен знак «→» (стрелка), который вводится с панелей инструментов **Evaluation, Symbolic** или комбинацией клавиш Ctrl + . (точка).

Пример 3. Вывести формулы производных функций $f(x) = x^5$ и $f(x) = x \cdot \sin 3x$.

Вызовем шаблон производной и введем в него первую функцию: $\frac{d}{dx}x^5 \to 5\cdot x^4$

Вторую функцию сначала определим, а затем введем в шаблон производной:

$$f(x) := x\cdot\sin(3\cdot x) \qquad \frac{d}{dx}f(x) \to \sin(3\cdot x) + 3\cdot x\cdot\cos(3\cdot x)$$

Отметим, что символьные операции весьма полезны. Однако далеко не всегда вид полученных формул оправдывает наши ожидания (вместо упрощения может получиться усложнение). Поэтому следует указывать тип преобразования, выбирая подходящие варианты на панели **Symbolic (Символьные операции)**.

Преобразование **Simplify (Упростить)** предназначено для приведения подобных слагаемых, приведения дробей к общему знаменателю, использования тригонометрических тождеств:

$$\frac{3}{19} + \frac{47}{93}\ \text{simplify} \to \frac{1172}{1767} \qquad \frac{x^2 - 3\cdot x - 4}{x - 4} + 2\cdot x - 5\ \text{simplify} \to 3\cdot x - 4$$

Преобразование **Expand (Расширить)** используется для раскрытия скобок, разложения по степеням:

$$(x+y)^3 \text{ expand},x \rightarrow x^3+3\cdot x^2\cdot y+3\cdot x\cdot y^2+y^3 \qquad \sin(2\cdot x) \text{ expand},x \rightarrow 2\cdot\sin(x)\cdot\cos(x)$$

Преобразование **Collect (Собрать)** позволяет группировать подобные:

$$x^2-a\cdot y\cdot x^2+2\cdot y^2\cdot x-x \text{ collect},x \rightarrow (1-a\cdot y)\cdot x^2+\left(2\cdot y^2-1\right)\cdot x$$

Преобразование **Solve (Решить)** выражает указанную переменную через другие в виде формулы, т. е. аналитически решает уравнения или неравенства:

$$x^2-4\cdot x+1 \text{ solve},x \rightarrow \begin{pmatrix} 2+3^{\frac{1}{2}} \\ 2-3^{\frac{1}{2}} \end{pmatrix} \qquad x^2+2\cdot x-3>0 \text{ solve},x \rightarrow \begin{pmatrix} x<-3 \\ 1<x \end{pmatrix}$$

При использовании преобразований **Expand**, **Collect**, **Solve** необходимо указывать переменную, относительно которой производится операция.

К сожалению, решения имеют приемлемый вид только для простых уравнений и неравенств (например, линейных, квадратных, некоторых тригонометрических), а сложность формул часто делает их малопригодными на практике. Вместе с тем операция **Solve** может быть полезна для получения и анализа формул при решении уравнений и неравенств.

?

1. **Как определяется собственная функция в Mathcad?**
2. **Что понимают под символьными операциями?**
3. **Приведите примеры символьных операций Mathcad.**

Упражнения

1. Определите функцию $y(t)=A\cos(\omega t)$, описывающую смещение точки при колебаниях, и вычислите ее значения при $t=0$ и $t=2$. Амплитуда $A=5$, частота $\omega=3$.
2. Определите функции $y(x)=x^3-5x^2+12$ и $z(x)=x^3+\cos 2x$. Вычислите их производные при $x=1$ и $x=2$.
3. Введите и преобразуйте выражения. Сравните с образцами:

$$\frac{(a+1)}{15}+\frac{15\cdot\left(a^2-1\right)}{9} \text{ simplify} \rightarrow \frac{1}{15}\cdot a-\frac{8}{5}+\frac{5}{3}\cdot a^2 \qquad \frac{11}{19-3}+\frac{4}{7} \text{ simplify} \rightarrow \frac{141}{112}$$

$$\cos(3\cdot x)\ \text{expand},x \rightarrow 4\cdot\cos(x)^3 - 3\cdot\cos(x) \qquad x^2 > 4\ \text{solve},x \rightarrow \begin{pmatrix} x < -2 \\ 2 < x \end{pmatrix}$$

$$2\cdot\sin(x) - 1\ \text{solve},x \rightarrow \frac{1}{6}\cdot\pi \qquad \sin(2\cdot x) - 1\ \text{solve},x \rightarrow \frac{1}{4}\cdot\pi$$

$$x^2 - 2\cdot x - 5 > 0\ \text{solve},x \rightarrow \begin{pmatrix} x < 1 - 6^{\frac{1}{2}} \\ 1 + 6^{\frac{1}{2}} < x \end{pmatrix} \qquad x^2 + y^2 + z^2\ \text{solve},z \rightarrow \begin{bmatrix} \left(-x^2 - y^2\right)^{\frac{1}{2}} \\ -\left(-x^2 - y^2\right)^{\frac{1}{2}} \end{bmatrix}$$

§ 31. Построение графиков

Средства графического представления функций в системе Mathcad весьма разнообразны. Они выбираются в меню **Insert (Вставка)** или вызываются кнопкой на панели **Graph (График)** (см. рис. 5.2). Мы ограничимся графиками в прямоугольной (декартовой) системе координат (**XY-graph**).

Для построения графика зависимости $f(x)$ требуется сначала создать таблицы значений аргумента x_i и функции $y_i = f(x_i)$. Это довольно трудоемкая работа. В Mathcad для этого предусмотрен особый тип переменной — ранжированная переменная, которая представляет собой поименованную последовательность значений: от начального x_0 до конечного x_n с шагом d. Задается эта переменная так: x := x_0 , x_0+d .. x_n (первое значение, запятая, второе значение, «..», последнее значение). Символ последовательности «..» вводится нажатием клавиши «;» (точка с запятой). При $x_0 < x_n$ значения возрастают, при $x_0 > x_n$ — убывают. При шаге ±1 достаточно указать только первое и последнее значения.

Пример 1. Задать и вывести на экран последовательности чисел: x от 1 до 4 с шагом 0,5; j от 2 до 7 с шагом 1 и z от 0,6 до –5 с шагом –0,2.

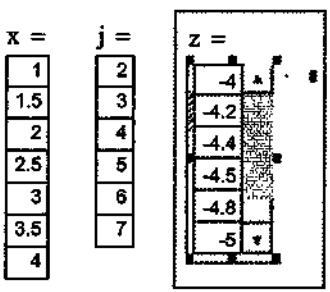


Рис. 5.4

Зададим ранжированные переменные:

x := 1, 1.5 .. 4; j := 2 .. 7 и z := 0.6, 0.4 .. –5.

Для их вывода будем набирать имя и знак «=». При этом на экране показывается несколько первых значений в форме таблицы. Для просмотра всех значений переменной z щелкнем мышью по таблице и воспользуемся появившейся полосой прокрутки (рис. 5.4).

Ранжированная переменная применяется в качестве аргумента при построении графиков и таблиц значений функций (например, n^2 на рисунке 5.5).

n := 1..5

n =	n^2 =
1	1
2	4
3	9
4	16
5	25

Рис. 5.5

Пример 2. Построить график функции $f(x) = x^3 - 12x - 5$ на промежутке $[-4; 4]$.

- Определим функцию: f(x):=x^3–12·x–5. Зададим область определения, используя ранжированную переменную с шагом 0.1: x:=–4, –3.9 .. 4.
- Укажем мышью (красным крестиком) на предполагаемое место левого верхнего угла будущего графика. Щелкнув по кнопке на панели **Graph**, вызовем шаблон построения графика в прямоугольной системе координат (рис. 5.6).
- Введем в нижнем черном квадратике аргумент x, а в левом — функцию $f(x)$.
- Щелкнем левой клавишей мыши за пределами области графика. Построенный график будет иметь заданный по умолчанию вид (рис. 5.7).
- Добавим оси, сетку и оцифровку шкал (рис. 5.8). Для этого двойным щелчком левой клавишей мыши по графику или командой **Format → Graph (Формат → → График)** вызовем окно форматирования. На вкладке **XY-axes (Оси XY)** снимем автоопределения (**Autoscale** и **Auto Grid**) и установим стиль осей

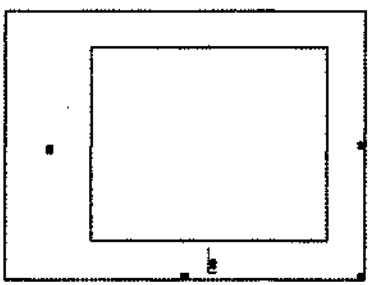


Рис. 5.6

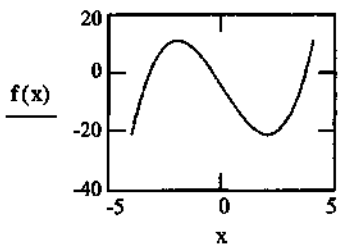


Рис. 5.7

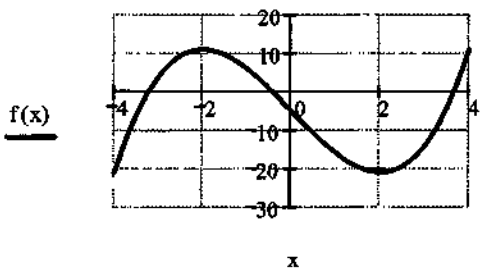


Рис. 5.8

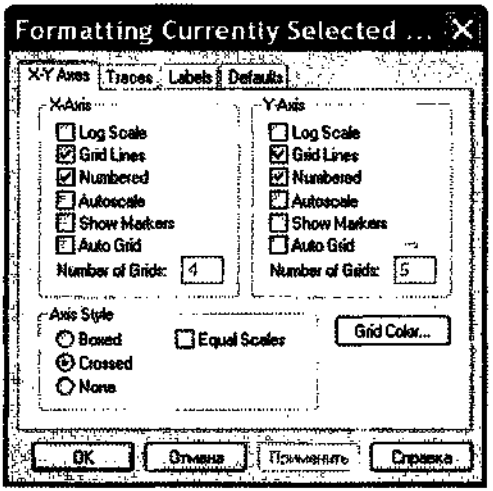


Рис. 5.9

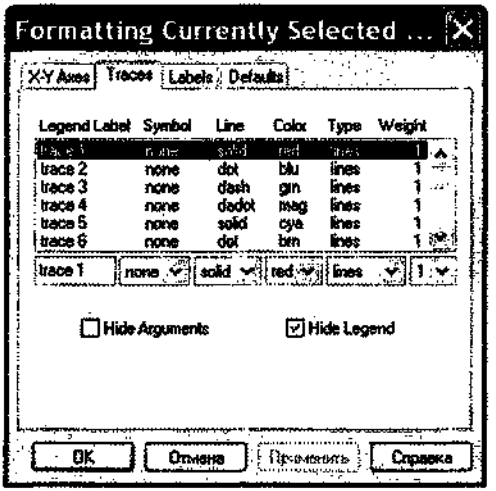


Рис. 5.10

(**Axis Style**) пересекающиеся (**Crossed**), сетку (**Grid Lines**) 4 × 5 и оцифровку шкал (**Numbered**) (рис. 5.9). На вкладке **Traces (Линии)** подберем цвет (**Color**) и толщину (**Weight**) линии (рис. 5.10).

Заметим, что график будет построен, даже если не задать область определения функции. В этом случае аргумент по умолчанию изменяется от –10 до 10 с шагом 1.

На одном шаблоне можно построить до 16 графиков функций, в том числе с параметрами и разными аргументами. Если аргументы разные, они вводятся через запятую в том же порядке, что и функции от них.

Пример 3. Построить на одном шаблоне графики функций $A(t)=10-bt$ и $x(t)=A(t)\sin(\omega t)$. Аргумент изменяется на промежутке [0; 20] с шагом 0,1. Параметры $b=0{,}4$ и $\omega=3$.

- Присвоим значения параметрам: b:=0.4 и ω:= 3.
- Определим функции: сначала A(t):=10–b·t, а затем x(t):=A(t)·sin(ω·t).
- Зададим область определения: x:= 0, 0.1 .. 20.
- Вызовем шаблон построения графика и в нижнем черном квадратике введем аргумент t, а в левом через запятую функции $A(t)$, $x(t)$.
- При форматировании графиков (рис. 5.11) на вкладке **Traces** подберем цвет и вид второй линии (**trase2**): вместо **dot** (точки) установим **solid** (сплошная).

Исследуйте самостоятельно изменения графиков, задав параметры $b=0{,}3$ и $\omega=5$.

$b := 0.4 \quad \omega := 3$

$A(t) := 10 - b \cdot t \quad x(t) := A(t) \cdot \sin(\omega \cdot t) \quad t := 0, 0.1 .. 20$

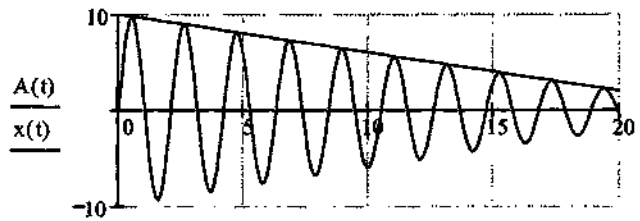


Рис. 5.11

Пример 4. На одном шаблоне построить график функции $f(x) = -x^2 + 2x - 3$ на промежутке $[-8; 8]$ с шагом 0,1 и касательную к нему в точке $a = -2$. Исследовать изменения наклона касательной в точках -3; -1; 0; 1; 2.

Построение касательной имеет следующие особенности.

- После задания функции, ее области определения и точки касания определим наклон касательной $k(x)$, т. е. найдем производную функции.
- Зададим касательную в точке a — прямую с угловым коэффициентом $k(a)$.
- В шаблон построения графика введем три аргумента x, x, a и три функции: исходную $f(x)$, касательную $ta(x)$ и ординату точки касания $f(a)$ (рис. 5.12).
- При форматировании графика на вкладке **Trases** подберем вид второй линии **(trase2)** сплошная **(solid)** и толщину **(Weight)** 2. Для точки касания **(trase3)** установим тип **(Type)** точки **(points)** и толщину **(Weight)** 6.

$f(x) := -x^2 + 2 \cdot x - 3 \quad x := -8, -7.9 .. 8 \quad a := -2$

$k(x) := \frac{d}{dx} f(x) \quad ta(x) := k(a) \cdot (x - a) + f(a)$

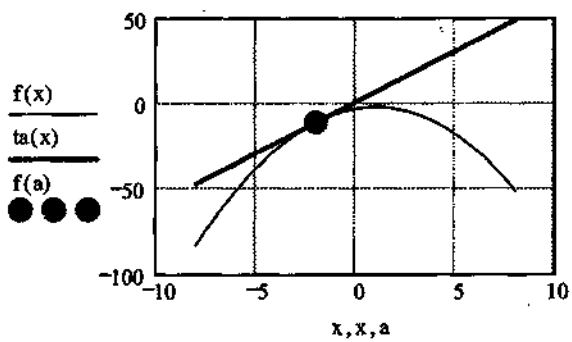


Рис. 5.12

1. Что называют ранжированной переменной?
2. Как задается ранжированная переменная?
3. Перечислите этапы построения графика в Mathcad.
4. Как построить несколько графиков на одном шаблоне?

Упражнения

1. Задайте и выведите на экран последовательности чисел:
   а) x от –3 до 7 с шагом 0,5; б) y от 2 до –2 с шагом –0,1.
2. Постройте график функции $f(t)=\sin 7t+\sin 8t$ на промежутке [0;15], шаг 0,1.
3. На одном шаблоне постройте график зависимости координаты точки от времени $x(t)=8+12t-2t^2$ и скорости $v(t)=\frac{dx}{dt}$ на промежутке [0; 5] с шагом 0,1.
4. На одном шаблоне постройте график функции $f(x)=x\sin x$ на промежутке [0; 12] с шагом 0,1 и касательную к нему в точке $a=5$.

§ 32. Решение уравнений

На практике Вам неоднократно приходилось решать различные уравнения, например линейные $ax+b=0$, квадратные $ax^2+bx+c=0$, тригонометрические.

Любое уравнение может быть приведено к виду $f(x)=0$ в результате переноса всех членов в левую часть. Решить уравнение — значит найти такие значения аргумента (**корни**), которые обращают его в тождество. Формулы для выражения корней в аналитическом виде существуют далеко не для всех уравнений и зачастую весьма громоздки. Поэтому на практике так велика роль вычислительных методов.

При нахождении корней уравнения выделяют два этапа:

1) **отделение** корней — поиск промежутков, которым они принадлежат;

2) **уточнение** корней — определение их значений с заданной точностью.

В основе отделения корней лежат следующие положения. Если непрерывная на отрезке $[a; b]$ функция $f(x)$ имеет на его концах противоположные знаки, то на этом отрезке находится хотя бы один корень (нуль функции), и график функции пересекает ось Ox. Если при этом функция монотонно возрастает или убывает (знак производной не изменяется), то корень на данном отрезке единственный.

Таким образом, решение уравнения сводится к отысканию промежутков локализации корней и последующему их уменьшению до тех пор, пока не будет получена требуемая точность. Наиболее просто и наглядно это делается с помощью таблиц и графиков. При этом область определения и шаг изменения аргумента выбирают так, чтобы график функции $f(x)$ отражал ее поведение на всем промежутке, содержащем искомые корни.

Рассмотрим некоторые способы решения уравнений в Mathcad. Начнем с графического, поскольку он позволяет в наиболее простом и наглядном виде отделять и уточнять корни.

Пример 1. Найти корни уравнения $x - 4\cos x = 0$ графическим способом.

- Определим функцию: f(x):=x–4·cos(x). Построим график функции $f(x)$, взяв сначала широкую область определения, чтобы увидеть особенности графика и обнаружить все корни (рис. 5.13), а затем сузив ее для уточнения корней (рис. 5.14).
- Для определения численных значений корней воспользуемся инструментом **Trace (Трассировка)**. Он вызывается щелчком правой клавишей мыши по графику и отображает координаты точек, попадающих на пересечение линий, которые можно перемещать при нажатой левой клавише мыши (рис. 5.15). Для следования по точкам графика необходимо установить флажок **Track Data Points (След точек данных)**.

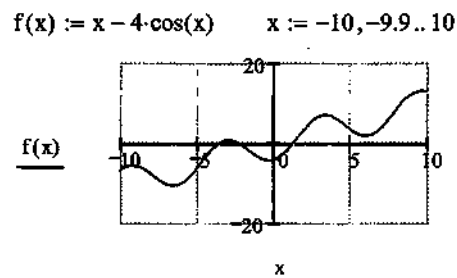


Рис. 5.13

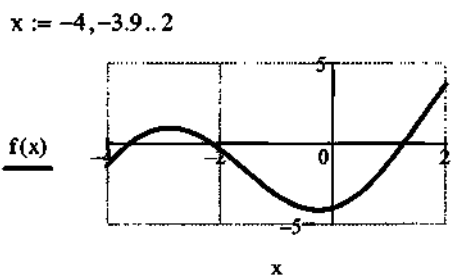


Рис. 5.14

Отслеживая в поле **Y-value** минимальное по модулю значение при переходе функции через нуль, в поле **X-value** последовательно определим все корни: –3.6; –2.1; 1.3. Уменьшим шаг изменения аргумента в 10 раз, задав x:= –4, –3.99..2 и

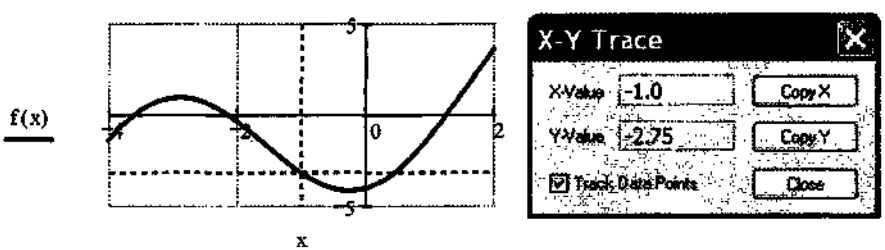


Рис. 5.15

растянем с помощью мыши график для удобства наблюдения. Корни будут определены с большей точностью: –3,59; –2,13; 1,25.

Mathcad содержит мощные средства решения уравнений. Некоторые из них собраны в категории **Solving (Решения)** и вызываются нажатием кнопки $f(x)$ на панели **Стандартная** или командой **Insert → Function (Вставка → → Функция)**.

Пример 2. Найти корни уравнения $x - 4\cos x = 0$ с помощью функции **root()**.

- Определим функцию: f(x):=x–4 · cos(x). Графическим способом отделим корни и возьмем их начальные приближения (например, –3,6; –2,1; 1,3 или даже –4; –2; 1).
- Будем последовательно задавать начальные приближения и применять функцию **root(f(x), x)**, как показано на рисунке 5.16.

$$f(x) := x - 4\cdot\cos(x)$$

$x := -4$ root(f(x),x) = –3.595
$x := -2$ root(f(x),x) = –2.133
$x := 1$ root(f(x),x) = 1.252

Рис. 5.16

$x := -4, -2..2$

x =	root(f(x),x) =
-4	-3.595
-2	-2.133
0	1.252
2	1.252

Рис. 5.17

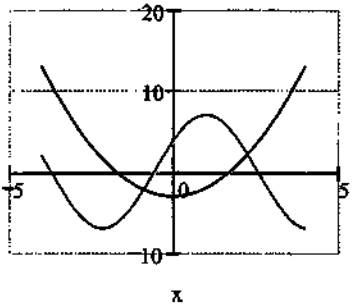
Множество начальных приближений можно задавать с помощью ранжированной переменной. При удачном выборе ее шага функция **root(f(x), x)** возвратит сразу все корни, причем некоторые из них могут повторяться (рис. 5.17).

Точность вычислений в Mathcad задается системной переменной TOL (по умолчанию TOL = 0.001). При необходимости требуемое значение может быть присвоено непосредственно на рабочем листе или установлено на вкладке **Built-in-Variables (Системные переменные)** окна **Math Options (Параметры)**, которое вызывается из меню **Math (Математика)**. Формат отображения результата должен соответствовать установленной точности (по умолчанию: общий, 3 десятичных знака). Настройки можно сделать в окне **Result Format (Формат результата)**, вызываемом из меню **Format (Формат)**.

Одним из самых мощных инструментов решения уравнений в Mathcad является конструкция **Given — find (Дано — найти)**, которая, как мы увидим в дальнейшем, позволяет решать не только уравнения, но и системы уравнений и неравенств.

$x := -4, -3.9 .. 4$

$\frac{7 \cdot \cos(x-1)}{x^2-3}$



$x := -1$

Given

$7 \cdot \cos(x - 1) = x^2 - 3$

$\text{find}(x) = -0.891$

Рис. 5.18

Пример 3. Найти корни уравнения $7\cos(x-1) = x^2 - 3$ с помощью конструкции **Given — find**.

- Построим графики функций (рис. 5.18), заданных выражениями в левой и правой частях уравнения, и найдем приближенные значения корней –0,9 и 2,3.
- Зададим начальное приближение первого корня, например –1.
- Наберем ключевое слово **Given**.
- Введем уравнение.

Две его части связаны жирным знаком равно «=», который вводится комбинацией клавиш Ctrl + = и является оператором логического сравнения.

- Применим функцию **find(x)**. Получим значение первого корня –0,891. Задав начальное приближение второго корня, например 2, получим 2,263.

1. С какой целью используют трассировку графика?
2. Назовите этапы решения уравнения с помощью функции **root()**.
3. Назовите этапы решения уравнения с помощью конструкции **Given — find**.
4. В каких случаях используется знак «=» (жирное равно)?

Упражнение

Найдите корни уравнения графическим способом, уточните корни с помощью функции **root()** и конструкции **Given — find**:

а) $2\sin x = x^2 - 1$ на промежутке $[-2; 2]$;

б) $\cos 2x = 5\sin(x - 1)$ на промежутке $[0; 10]$;

в) $3\sin x = \cos 3x + x$ на промежутке $[-3; 3]$;

г) $\cos(x + 1) = x \sin x$ на промежутке $[-2; 2]$.

§ 33. Операции с векторами и матрицами

При решении широкого спектра задач в различных областях: от физики и техники (расчет сил, скоростей, электрических цепей) до экономики (оптимизация ресурсов, затрат и прибылей) и графики (обработка изображений) — используются векторы и матрицы.

Матрицей называют поименованную совокупность элементов, которые размещаются в таблице и упорядочены по индексам (номерам строк и столбцов). Элементы матрицы обозначают буквами с двумя индексами a_{ij}. Первый индекс i означает номер строки, в которой расположен элемент, а второй индекс j — номер столбца.

Матрицы могут быть прямоугольными (n строк, m столбцов), квадратными (рис. 5.19, *а*) и даже занимать одну строку или столбец. **Вектор** рассматривается как частный случай матрицы, его компоненты (проекции) помещают в одну строку или столбец (рис. 5.19, *б*).

а

$$A = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{pmatrix}$$

б

$$B = \begin{pmatrix} b_1 \\ b_2 \\ \dots \\ b_n \end{pmatrix}$$

Рис. 5.19

Рис. 5.20

Работа с векторами и матрицами в Mathcad обеспечивается набором встроенных функций (категория **Vector and Matrix**), а также использованием панели инструментов **Matrix (Матрица)** (рис. 5.20).

Векторы и матрицы задаются либо заполнением шаблона вручную, либо с использованием в качестве индексов ранжированной переменной, если возможен способ вычисления значений элементов через их индексы.

33.1. Использование операций с векторами

Пример 1. На тело массой $m = 2$ кг действуют силы F_1 и F_2, проекции которых на оси координат $F_{1x} = -1$ Н, $F_{1y} = 6$ Н; $F_{2x} = 7$ Н, $F_{2y} = 2$ Н. Определить проекции и модуль результирующей силы, а также проекции и модуль ускорения тела.

Напомним, что проекции результирующего вектора равны сумме проекций слагаемых векторов: $F_x = F_{1x} + F_{2x}$; $F_y = F_{1y} + F_{2y}$ (рис. 5.21). Модуль вектора равен $F = \sqrt{F_x^2 + F_y^2}$. Ускорение прямо пропорционально действующей на тело силе и обратно пропорционально массе тела.

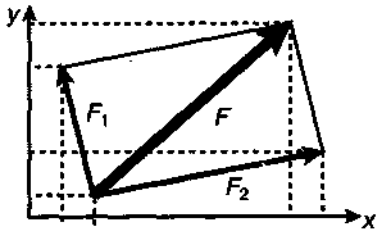


Рис. 5.21

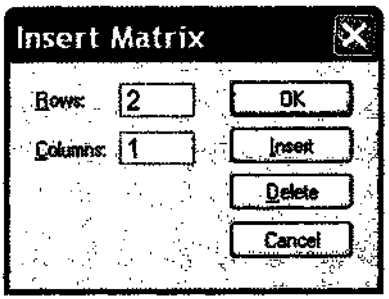


Рис. 5.22

Решение задачи в Mathcad сводится к вводу векторов и действиям с ними.

- Введем векторы F_1 и F_2. Для этого наберем имя, затем знак операции присваивания «:=». С помощью кнопки [⋮⋮] панели **Матрица** или команды **Insert → Matrix (Вставка → Матрица)** вызовем окно **Insert Matrix** (рис. 5.22). Укажем число строк **(Rows) 2** и столбцов **(Columns) 1.**
- Заполним шаблоны проекциями векторов:

$$F1 := \begin{pmatrix} \blacksquare \\ \blacksquare \end{pmatrix} \qquad F1 := \begin{pmatrix} -1 \\ 6 \end{pmatrix} \qquad F2 := \begin{pmatrix} 7 \\ 2 \end{pmatrix}$$

- Запишем сумму векторов и выведем ее значение.
- Для вычисления модуля вектора используем кнопку |x| панели **Матрица**.

$$F := F1 + F2 \qquad F = \begin{pmatrix} 6 \\ 8 \end{pmatrix} \qquad |F| = 10$$

- Результат несложно проверить расчетом по формуле $\sqrt{6^2 + 8^2} = 10$.
- Вектор ускорения получим делением вектора силы на массу:

$$m := 2 \qquad a := \frac{F}{m} \qquad a = \begin{pmatrix} 3 \\ 4 \end{pmatrix}$$

Наконец, применим операцию нахождения модуля к вектору a: $|a| = 5$

На практике часто приходится решать уравнения, левая часть которых является **многочленом** от неизвестного x:

$$a_0 + a_1x^1 + a_2x^2 + ... + a_{n-1}x^{n-1} + a_nx^n = 0,$$

где показатель степени n — целое неотрицательное число, а коэффициенты многочлена a_0, a_1, ..., a_n — действительные числа. Частными случаями таких уравнений являются квадратные и кубические.

Решение таких уравнений в Mathcad сводится к заполнению вектора-столбца V коэффициентами a_0, a_1, a_2, ..., a_n (начиная с a_0 и включая нули) и применению функции **polyroots(V)** из категории **Solving (Решения)**. Достоинство этого способа состоит в том, что он не требует определять начальные приближения неизвестных.

Пример 2. Найти корни уравнения $x^3 - 12x + 3 = 0$.

- Зададим шаблон вектора-столбца V с числом строк 4 (на одну больше степени многочлена). Заполним его коэффициентами a_0, a_1, a_2, ..., a_n (начиная с $a_0 = 3$ и включая нуль для отсутствующего a_2x^2).
- Применим функцию **polyroots(V)**:

$$V := \begin{pmatrix} 3 \\ -12 \\ 0 \\ 1 \end{pmatrix} \quad \text{polyroots}(V) = \begin{pmatrix} -3.583 \\ 0.251 \\ 3.332 \end{pmatrix}$$

Конечно, это уравнение можно решить и с помощью рассмотренной ранее функции **root()**, но при этом потребуется задать начальные приближения, например построив график.

33.2. Использование операций с матрицами. Обработка изображений

Ярким примером применения матричных операций служит обработка растровых изображений. Растровое изображение — это совокупность (матрица) пикселей, на каждый из которых в модели RGB отводится 24 бит: по 8 бит на каждый из трех основных цветов (красный, зеленый, синий). Наиболее просто обрабатывать серое изображение с глубиной цвета 8 бит, в котором каждому пикселю соответствует значение яркости от 0 до 255. Эти значения можно загрузить в матрицу и подвергнуть преобразованиям. Например, вырезание части изображения сводится к выделению фрагмента матрицы. Так получают различные графические эффекты.

Сначала потренируемся создавать и фрагментировать матрицы. Для присвоения элементам матрицы значений в качестве индексов (номеров строк и столбцов) будем использовать ранжированные переменные.

Индексы могут иметь только целочисленные значения и начинаться с нуля или единицы, что задается системной переменной **ORIGIN** (по умолчанию ее значение равно нулю). Во многих задачах удобнее, чтобы нумерация индексов начиналась с единицы (как и строк и столбцов матриц). Это значение следует установить в начале работы на вкладке **Built-in-Variables (Системные переменные)** окна **Math Options (Параметры)** или присвоить ORIGIN := 1 непосредственно на рабочем листе.

Пример 3. Создать матрицу A размером 3×4 (три строки, четыре столбца), элементы которой равны сумме номеров строк и столбцов. Выделить фрагменты размером 2×2 из левого верхнего и правого нижнего углов матрицы.

- Установим ORIGIN = 1.
- Зададим ранжированные переменные i := 1.. 3 и j := 1.. 4, поскольку матрица имеет три строки и четыре столбца. Для задания элемента на пересечении i-й строки и j-го столбца введем имя матрицы A, с помощью кнопки x_n панели **Матрица** введем индексы i, j и присвоим значение $A_{i,j} := i + j$. Выведем результат:

$$\text{ORIGIN} := 1 \qquad i := 1..3 \quad j := 1..4 \quad A_{i,j} := i + j \qquad A = \begin{pmatrix} 2 & 3 & 4 & 5 \\ 3 & 4 & 5 & 6 \\ 4 & 5 & 6 & 7 \end{pmatrix}$$

- Для выделения фрагмента матрицы используем функцию **submatrix** из категории **Vector and Matrix (Векторы и матрицы)**. Ее аргументы — имя исходной матрицы, индексы первой и последней строк и первого и последнего столбцов выделяемого фрагмента:

$$\text{submatrix}(A,1,2,1,2) = \begin{pmatrix} 2 & 3 \\ 3 & 4 \end{pmatrix} \quad \text{submatrix}(A,2,3,3,4) = \begin{pmatrix} 5 & 6 \\ 6 & 7 \end{pmatrix}$$

Функцию **submatrix** можно использовать для выделения фрагментов с обратным порядком следования строк (столбцов). Для этого нужно поменять местами аргументы (например, индексы начальной и конечной строк). Так, в примере 1 можно получить такой фрагмент:

$$\text{submatrix}(A,3,2,3,4) = \begin{pmatrix} 6 & 7 \\ 5 & 6 \end{pmatrix}$$

Пример 4. Загрузить изображение timer.bmp. Преобразовать его: получить фрагмент, повернуть на 180°, отразить зеркально и повернуть на 180° фрагмент изображения. Получить негативное и тонированные изображения.

- Установим ORIGIN=1.
- Применим функцию **READBMP("timer.bmp")** из категории **Image Processing (Обработка изображений)** для загрузки в матрицу ***M*** значений яркости пикселей изображения из файла timer.bmp.
- Для определения ширины w и высоты h изображения в пикселях (т. е. количества столбцов и строк матрицы) применим функции **cols** и **rows** из категории **Vector and Matrix (Векторы и матрицы)** (рис. 5.23).
- Выведем исходное изображение на экран (рис. 5.24). Для этого впишем имя матрицы ***M*** в шаблон, который вызовем командой **Insert → Picture (Вставка → Изображение)** или нажатием кнопки на панели **Матрица**.
- Для преобразования изображения будем выделять фрагменты матриц:
  M1:= submatrix(M, 1, 110, 10, 60) — обрезание справа;
  M2:= submatrix(M, h, 1, w, 1) — поворот на 180°;
  M3:= submatrix(M, 80, 10, 10, 90) — вырезание, поворот на 180° и зеркальное отражение.
- Выведем фрагменты изображения (рис. 5.25).
  Создадим матрицу фона ***F*** с яркостью пикселей, например 20, и матрицу негатива ***N*** (из максимальной яркости 255 вычтем значения для каждого пикселя).

$$i := 1..h \quad j := 1..w \quad F_{i,j} := 20 \quad N := (255 - M)$$

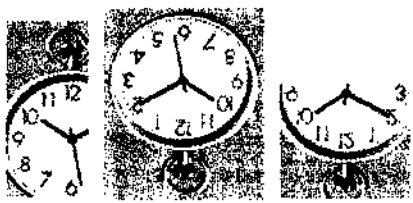
- Выведем негативное N и тонированные изображения (рис. 5.26). Эффект тонирования (одноцветной раскраски) изображения получается благодаря выводу матриц изображения и фона в каналы цвета RGB. Для этого в шаблон вы-

```
ORIGIN := 1
M := READBMP("timer.bmp")
w := cols(M)   w = 100
h := rows(M)   h = 120
```

Рис. 5.23

M

Рис. 5.24



M1 M2 M3

Рис. 5.25

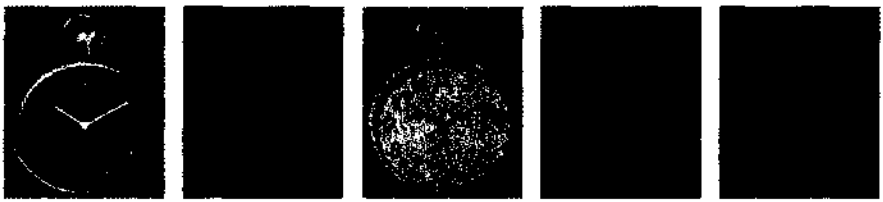


Рис. 5.26 N M,F,F F,M,F F,F,M M,F,M

вода изображения вписываются имена трех матриц: для красного, зеленого и синего цветов.

1. Как вводятся векторы и матрицы в системе Mathcad?
2. Какие уравнения можно решать с помощью функции **polyroots()**?
3. Что представляет собой растровое изображение?
4. Как вывести изображение на экран?
5. Как выделить фрагмент изображения?

Упражнения

1. Три вектора заданы проекциями: $\boldsymbol{A}(2; 6; -1)$, $\boldsymbol{B}(6; -2; 5)$ и $\boldsymbol{C}(12; 0; 6)$. Определите проекции и модули векторов: $\boldsymbol{D} = \boldsymbol{A} + \boldsymbol{B} + \boldsymbol{C}$; $\boldsymbol{E} = \boldsymbol{A} - \boldsymbol{B} - \boldsymbol{C}$.
2. Найдите корни уравнения с помощью функции **polyroots()**:

   а) $3x^3 + 5x^2 - 2 = 0$; б) $x^4 - 3x^3 - 9x^2 + 12x + 8 = 0$.

3. Загрузите матрицу изображения из файла. Определите ширину и высоту изображения в пикселях.
4. Выделите два фрагмента изображения из упражнения 3. Поверните первый фрагмент на 180°, превратите второй фрагмент в негатив. Выведите изображения на экран.

§ 34. Решение систем уравнений и неравенств

В курсе алгебры Вы познакомились с некоторыми способами решения систем уравнений с двумя неизвестными. Решение систем с большим количеством уравнений и неизвестных весьма трудоемко и требует применения вычислительной техники.

Рассмотрим сначала решение **систем линейных уравнений**. Напомним, что решением системы двух линейных уравнений с двумя неизвестными

$$\begin{cases} a_1 x + b_1 y = c_1, \\ a_2 x + b_2 y = c_2 \end{cases}$$

является упорядоченная пара значений x, y, которые обращают каждое уравнение системы в равенство.

Систему n линейных уравнений с n неизвестными в общем виде записывают так:

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + \ldots + a_{1n}x_n = b_1, \\ a_{21}x_1 + a_{22}x_2 + a_{23}x_3 + \ldots + a_{2n}x_n = b_2, \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots, \\ a_{n1}x_1 + a_{n2}x_2 + a_{n3}x_3 + \ldots + a_{nn}x_n = b_n. \end{cases}$$

Решением этой системы является упорядоченный набор значений неизвестных x_1, x_2, ..., x_n (обозначаются одной буквой x с индексом, который соответствует номеру неизвестного). Коэффициенты при неизвестных обозначены буквой a_{ij} с двумя индексами: первый индекс i соответствует номеру уравнения, а второй j — номеру неизвестного. Числа в правых частях уравнений обозначены буквой b с индексом, соответствующим номеру уравнения.

Символически систему уравнений можно записать в сокращенном виде:

$$\boldsymbol{AX = B}.$$

Эта запись по форме напоминает простейшее линейное уравнение $ax = b$. Существенная разница заключается в том, что символами $\boldsymbol{A}$, $\boldsymbol{X}$ и $\boldsymbol{B}$ здесь обозначены не числа, а более сложные математические объекты: упорядоченные наборы значений (чисел). Их удобно размещать в таблицах:

$$\boldsymbol{A} = \begin{pmatrix} a_{11} & a_{12} & \ldots & a_{1n} \\ a_{21} & a_{22} & \ldots & a_{2n} \\ \ldots & \ldots & \ldots & \ldots \\ a_{n1} & a_{n2} & \ldots & a_{nn} \end{pmatrix} \quad \boldsymbol{B} = \begin{pmatrix} b_1 \\ b_2 \\ \ldots \\ b_n \end{pmatrix} \quad \boldsymbol{X} = \begin{pmatrix} x_1 \\ x_2 \\ \ldots \\ x_n \end{pmatrix}$$

В математике такие таблицы называют матрицами (в программировании им соответствуют массивы).

Для решения системы n линейных уравнений с n неизвестными (матрица коэффициентов $\boldsymbol{A}$ такой системы квадратная) в Mathcad предусмотрена функция **Isolve (A, B)**, аргументами которой являются матрица коэффициентов $\boldsymbol{A}$ и вектор-столбец $\boldsymbol{B}$ правых частей системы уравнений.

Пример 1. Решить систему уравнений

$$\begin{cases} 12x - 5y + z = 6, \\ x - 2y - 7z = 2, \\ 5x + 3y - 2z = 4. \end{cases}$$

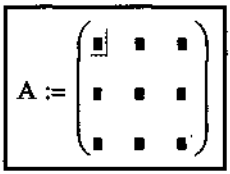


Рис. 5.27

- Введем матрицу коэффициентов при неизвестных ***A*** и вектор-столбец ***B*** правых частей системы уравнений. Для этого наберем имя матрицы, затем знак операции присваивания «:=». В окне **Insert Matrix** укажем число строк и столбцов и заполним появившийся шаблон (рис. 5.27). Для вектора ***B*** укажем число строк 3 и число столбцов 1.
- Применив функцию **Isolve (A, B)**, получим решение (рис. 5.28).

$$A := \begin{pmatrix} 12 & -5 & 1 \\ 1 & -2 & -7 \\ 5 & 3 & -2 \end{pmatrix} \qquad B := \begin{pmatrix} 6 \\ 2 \\ 4 \end{pmatrix} \qquad \text{lsolve}(A, B) = \begin{pmatrix} 0.594 \\ 0.176 \\ -0.251 \end{pmatrix}$$

Рис. 5.28

Для решения систем уравнений и неравенств с n неизвестными x_1, x_2, ..., x_n используется рассмотренная ранее конструкция **Given — find (Дано — найти)**. Наибольшая трудность при решении систем нелинейных уравнений состоит в определении начальных приближений. Они должны задаваться до блока решения, который начинается ключевым словом **Given (Дано)**. Внутри блока помещаются уравнения, а также неравенства. Завершает блок решения функция от n аргументов **find(x_1,x_2,...,x_n)**. Они перечисляются в том порядке, в котором будут выведены в вектор-столбец решения системы. Подчеркнем, что внутри блока решений используются знаки логического сравнения (<, ≤, >, ≥, жирный знак равно «=»), которые связывают левые и правые части уравнений и неравенств.

Пример 2. Решить систему уравнений

$$\begin{cases} y - 1 = x^3 - 5x, \\ x^2 = y + 1. \end{cases}$$

- Зададим начальные приближения (например, 1; 1), которые определим графическим способом (рис. 5.29).
- Наберем ключевое слово **Given**.
- Введем уравнения, используя операторы логического сравнения.
- Завершив блок функцией **find(x,y)**, получим решение системы (рис. 5.30, *а*).

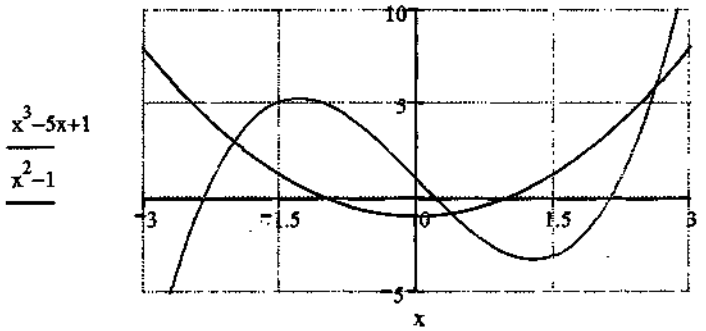


Рис. 5.29

а

x := 1 y := 1

Given

$y - 1 = x^3 - 5 \cdot x$

$x^2 = y + 1$

$\text{find}(x, y) = \begin{pmatrix} 0.382 \\ -0.854 \end{pmatrix}$

б

x := 1 y := 1

Given

$y - 1 = x^3 - 5 \cdot x$

$x^2 = y + 1$ $\boxed{x > 2}$

$\text{find}(x, y) = \begin{pmatrix} 2.618 \\ 5.854 \end{pmatrix}$

Рис. 5.30

- Введем в блок решения неравенство $x > 2$. Получим второе решение (рис. 5.30, *б*).
- Третье решение (–2; 3) получим, задав начальные приближения –2; 1 либо введя в блок решения неравенство $x < -1$.

Конструкция **Given — find** позволяет получать также решение простых алгебраических уравнений и систем (например, линейных, квадратных, кубических) в символьном (аналитическом) виде. В этом случае начальные приближения перед блоком решения не задаются. После символьных решений можно поставить знак вычислений «=» и получить численные решения (рис. 5.31).

Given

$y - 1 = x^3 - 5 \cdot x$

$x^2 = y + 1$

$$\text{find}(x, y) \rightarrow \begin{pmatrix} -2 & \frac{3}{2} + \frac{1}{2} \cdot 5^{\frac{1}{2}} & \frac{3}{2} - \frac{1}{2} \cdot 5^{\frac{1}{2}} \\ 3 & \frac{5}{2} + \frac{3}{2} \cdot 5^{\frac{1}{2}} & \frac{5}{2} - \frac{3}{2} \cdot 5^{\frac{1}{2}} \end{pmatrix} = \begin{pmatrix} -2 & 2.618 & 0.382 \\ 3 & 5.854 & -0.854 \end{pmatrix}$$

Рис. 5.31

1. Назовите этапы решения системы n линейных уравнений с n неизвестными.
2. Назовите этапы решения систем уравнений с помощью конструкции **Given — find**.
3. Как задается блок решения?
4. Как получить символьное решение системы алгебраических уравнений с помощью конструкции **Given — find**?

Упражнения

1. Найдите решения системы линейных уравнений:

а) $\begin{cases} x_1 - 3x_2 + 2x_3 + 3x_4 = 2, \\ 3x_1 - 2x_2 + 5x_3 - 4x_4 = 1, \\ 7x_1 + 5x_2 + 2x_3 + 3x_4 = 3, \\ 5x_1 + 2x_2 - 7x_3 + 2x_4 = 4; \end{cases}$ б) $\begin{cases} 2x_1 - 5x_2 + 3x_3 + 2x_4 = 7, \\ 6x_1 - 4x_2 + 3x_3 - 5x_4 = 12, \\ 3x_1 - 7x_2 + 5x_3 - 3x_4 = 5, \\ 2x_1 + 8x_2 - x_3 + 3x_4 = -9. \end{cases}$

2. Найдите решения системы уравнений и неравенств с помощью конструкции **Given — find** (начальные приближения определите графически):

а) $\begin{cases} \sin(4x - 1) = y - 1, \\ x^2 = \cos y; \end{cases}$ б) $\begin{cases} \sin(5x - 2) = y - 2, \\ x = \cos y; \end{cases}$

в) $\begin{cases} x^3 - 2x^2 = y - 3, \\ y^3 = 15x^2 - 7; \end{cases}$ г) $\begin{cases} \cos(3x - 1) = y - 1, \\ x = \operatorname{tg}(y - 1); \end{cases}$

д) $\begin{cases} x^2 + y^2 = 2, \\ y + 1 = x^2, \\ x > 0; \end{cases}$ е) $\begin{cases} x^2 = y + 3, \\ y - 1 = x^3 - 3x^2, \\ x > 3. \end{cases}$

ГЛАВА 6

КОММУНИКАЦИОННЫЕ ТЕХНОЛОГИИ

§ 35. Способы и скорость передачи информации

Человек, животные, технические устройства при передаче и приеме информации вступают в контакт друг с другом. Этот процесс называется **коммуникацией**.

Слово «*коммуникация*» произошло от латинского *communicatio* — *сообщение, передача.*

Коммуникационные технологии описывают процессы передачи информации от источника к приемнику.

Человек всегда стремился использовать различные способы передачи информации. Среди них выделим основные: *механические движения, звуковые и электромагнитные волны, электрические напряжение и токи, электронные лазерные пучки* и др.

Синхронизация процессов передачи информации предполагает согласование их во времени.

Передача информации в компьютерных сетях может осуществляться беспроводным способом или с помощью кабелей трех типов: *витой пары, коаксиального кабеля* и *оптоволоконного кабеля.*

Витая пара состоит из двух или более пар проводов, скрученных друг с другом. Скручивание проводов уменьшает влияние электромагнитных полей на передаваемые сигналы и повышает помехоустойчивость (рис. 6.1). Скорость передачи информации для данного типа кабеля составляет 0,25—1 Мбит/с.

Коаксиальный кабель состоит из центрального изолированного проводника, поверх которого расположен другой проводник, играющий роль экрана. Этот кабель обладает высокой механической прочностью и обеспечивает скорость передачи информации 10—50 Мбит/с (рис. 6.2). В настоящее время он интенсивно заменяется новыми видами связи.

Оптоволоконный кабель представляет собой тонкие стеклянные цилиндры. Скорость передачи информации для такого типа кабеля более 50 Мбит/с (рис. 6.3).

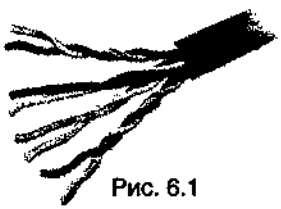


Рис. 6.1

Рис. 6.2

Рис. 6.3

При передаче информации важными являются такие показатели как *качество* и *скорость*.

Во время перемещения информации от источника к приемнику на нее воздействуют различные помехи. Эти помехи могут возникать из-за низкого качества линий связи, плохой работы передающей аппаратуры. В результате качество передаваемой информации ухудшается.

Скорость передачи информации в компьютерных сетях во многом зависит от способа подключения к сети Интернет. Среди существующих способов подключения выделим следующие:

- коммутируемый доступ по телефонной линии;
- беспроводное мобильное соединение (GPRS);
- доступ по ассиметричной абонентской линии (ADSL);
- беспроводная цифровая связь (Wi-Fi);
- доступ по выделенному каналу связи;
- спутниковый доступ и т. д.

Доступ по телефонной линии (или диалап-соединение — от английского *dial-up*, что означает «набрать номер») нам уже известен. Скорость работы при таком способе невысокая. Она зависит от скорости приема и передачи данных с помощью модема и скорости передачи информации по магистральному каналу связи провайдера. Например, при наличии на компьютере модема, работающего со скоростью 56 Кбит/с, скорость передачи информации по телефонной линии находится в диапазоне от 35 Кбит/с до 45 Кбит/с.

Беспроводное мобильное соединение с помощью специального протокола мобильной связи позволяет обеспечить среднюю скорость передачи данных от 30 Кбит/с до 40 Кбит/с. При таком соединении скорость передачи данных зависит также от типа мобильного телефона.

Доступ по асимметричной абонентской линии является специальным видом подключения к сети Интернет по обычной телефонной линии, но не мешает работе телефона. Такой доступ обеспечивает передачу данных по направлению к пользователю со скоростью до 8 Мбит/с, а от пользователя до 770 Кбит/с, в зависимости от протяженности и качества линии.

Беспроводная цифровая связь позволяет выполнить доступ к сети Интернет с помощью специального адаптера, который входит в стандартный комплект ноутбуков. Скорость передачи информации при таком соединении может достигать 11 Мбит/с.

1. Какие процессы описывают коммуникационные технологии?
2. Какие способы передачи информации вам известны?
3. От чего зависит скорость передачи информации в компьютерной сети?
4. Какие единицы измерения скорости передачи информации используются в компьютерной сети?

§ 36. Службы сети Интернет

36.1. Основные службы сети Интернет и протоколы

Основные возможности компьютерной сети Интернет реализуются через ее службы, которые можно разделить на: *Всемирную паутину, электронную почту, средства обмена сообщениями в режиме реального времени, форумы и телеконференции, сетевые новости* и др.

С назначением и возможностями Всемирной информационной паутины (World Wide Web — WWW) и электронной почтой (Electronic Mail — e-mail) мы уже знакомы. Назначение и возможности остальных служб рассмотрим в этом параграфе позже.

Функционирование компьютерной сети обеспечивают сетевые протоколы (наборы правил), определяющие формат и необходимые действия при обмене информацией между различными устройствами компьютерной сети.

Важнейшими протоколами сети Интернет являются протоколы TCP/IP.

Протокол управления передачей TCP (Transmission Control Protocol) предназначен для установки логического соединения между компьютерами клиентов и сервером, разбивает информацию на пакеты и контролирует доставку этих пакетов в пункты назначения.

Межсетевой Интернет-протокол IP (Internet Protocol) определяет адресацию при передаче информации и обеспечивает организацию транспортировки этой информации в пункты назначения по определенным маршрутам.

Протоколы TCP/IP тесно взаимосвязаны, поэтому в сети Интернет их объединяют и говорят о едином протоколе TCP/IP.

При работе с веб-страницами во Всемирной паутине используются протоколы передачи гипертекста HTTP (HyperText Transfer Protocol), который определяет формат и порядок обмена сообщениями между клиентом и сервером.

При работе с электронной почтой чаще других используются Простой Протокол Передачи электронной почты SMTP (Simple Mail Transfer Protocol) и Протокол Почтового отделения POP (Post Office Protocol).

С помощью протокола SMTP сообщения передаются от рабочего компьютера клиента к серверу провайдера, а также осуществляется передача сообщений между серверами отправителей и получателей.

Протокол POP используется для передачи электронных сообщений из почтовых ящиков клиентов, размещенных на сервере, на их рабочие компьютеры при помощи программ-клиентов.

36.2. Средства обмена информацией в режиме реального времени

К средствам обмена информацией в режиме реального времени относится целая группа *программ-мессенджеров* (от английского слова *messenger — связной, курьер*).

Программы-мессенджеры позволяют обмениваться через сеть Интернет текстовыми, голосовыми и даже видеосообщениями.

Известными программами-мессенджерами, которые популярны среди пользователей, являются программы ICQ и Skype.

Возможности программы Skype, имеющей простой интерфейс, позволяют:

- вести диалог с помощью текстовых сообщений (работа в чате);
- общаться голосом с компьютера на компьютер, с компьютера на телефон, и наоборот;
- пересылать файлы с одного компьютера на другой;
- вести телеконференцию.

Для голосового общения в программе Skype пользователю необходимо иметь на своем компьютере средства мультимедиа: звуковую карту, микрофон, колонки или наушники.

После установки программы Skype на вашем компьютере необходимо зарегистрироваться в окне **Создать пользователя**, указав имя, пароль и некоторые сведения о себе (рис. 6.4).

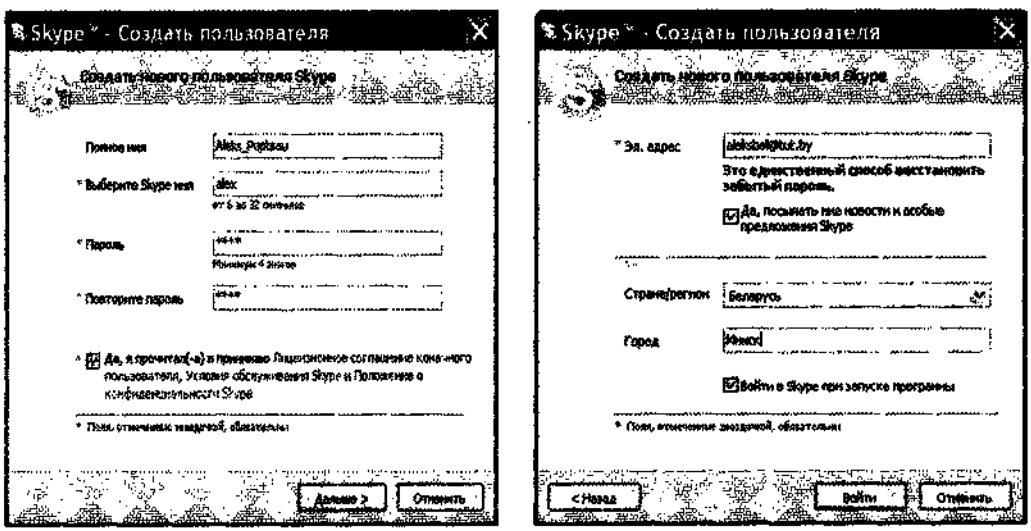


Рис. 6.4

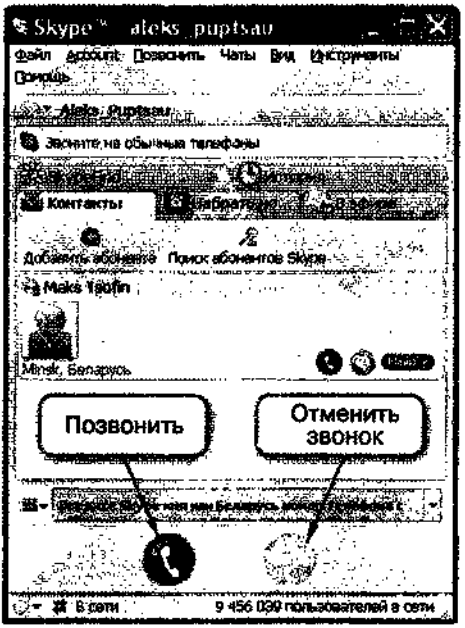


Рис. 6.5

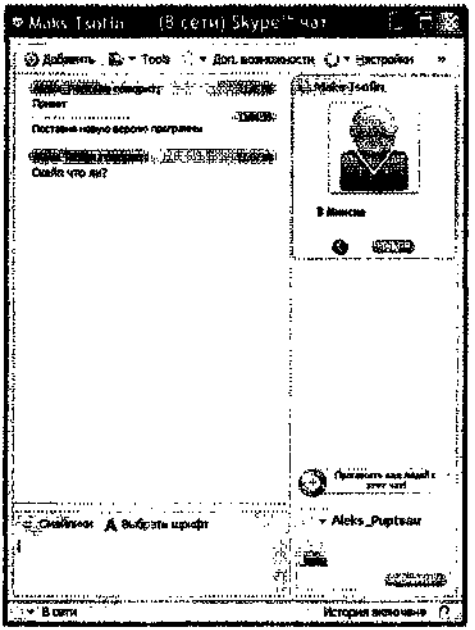


Рис. 6.6

Для добавления контакта с новым абонентом необходимо выполнить в основном окне **Skype** (рис. 6.5) последовательность действий **Инструменты → Добавить контакт,** а затем в окне **Добавить контакт** ввести нового абонента и нажать кнопку **Поиск**.

После этого программа Skype запросит абонента, которого Вы указали, с целью получить разрешение на общение с Вами.

Вызов абонента с компьютера в программе Skype выполняется двойным щелком левой клавишей мыши по необходимому контакту или одинарным щелчком этой клавишей по пиктограмме **Позвонить** (см. рис. 6.5). У вызываемого абонента появится сообщение о вызове. После завершения разговора необходимо щелкнуть левой клавишей мыши по кнопке **Отменить звонок** (см. рис. 6.5).

Используя контекстное меню основного окна программы Skype, пользователь может осуществить текстовый диалог (беседу в чате) (рис. 6.6) — пункт меню **Чаты,** выполнить звонок с компьютера на телефон — пункт меню **Позвонить** (см. рис. 6.5) и т. д.

Программа Skype позволяет вести телеконференцию в сети.

Под телеконференцией в сети Интернет понимается одновременное общение нескольких абонентов с помощью голоса или видео.

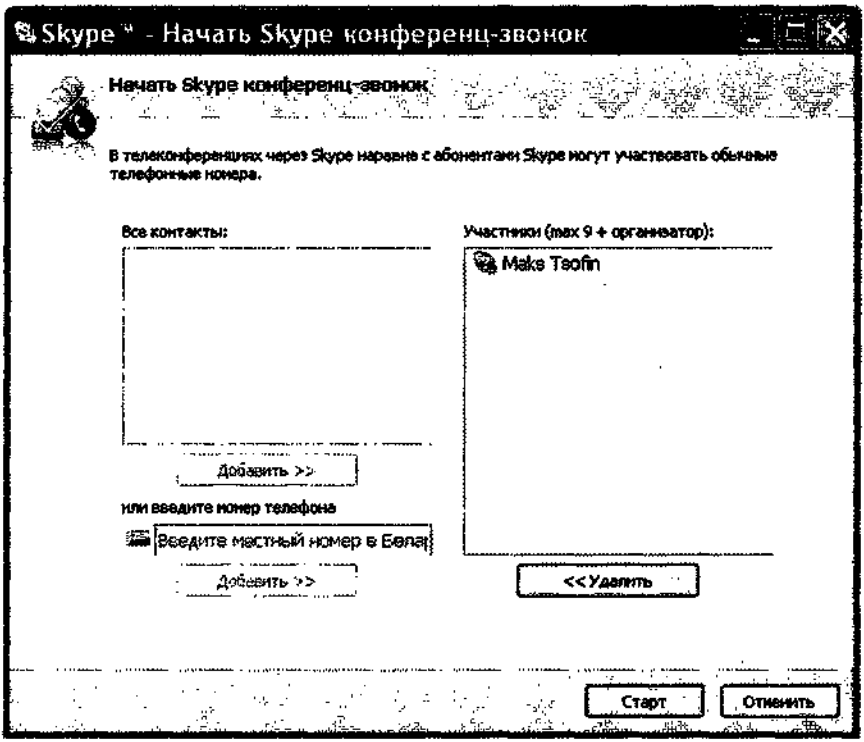


Рис. 6.7

Для начала телеконференции в программе Skype необходимо выполнить последовательность действий: **Позвонить** → **Начать конференц-звонок** → в окне **Начать Skype конференц-звонок** добавить абонентов для конференции → **Старт** (рис. 6.7).

Телеконференция может быть осуществлена сразу с десятью абонентами.

36.3. Форумы

Форумы (веб-форумы, www-форумы) — это сайты для публичного обмена сообщениями между пользователями.

В основу структуры форума положены треды (от английского *thread* — *нить*). Первое сообщение в треде задает тематику сообщений (тему для обсуждения), например **Кроссворд** (рис. 6.8). Затем идут первые комментарии к теме обсуждения и последовательность ответов, как показано на рисунке 6.9.

Для участия в форуме требуется регистрация, а при пользовании форумом необходимо выполнять свод правил, определенных в нем.

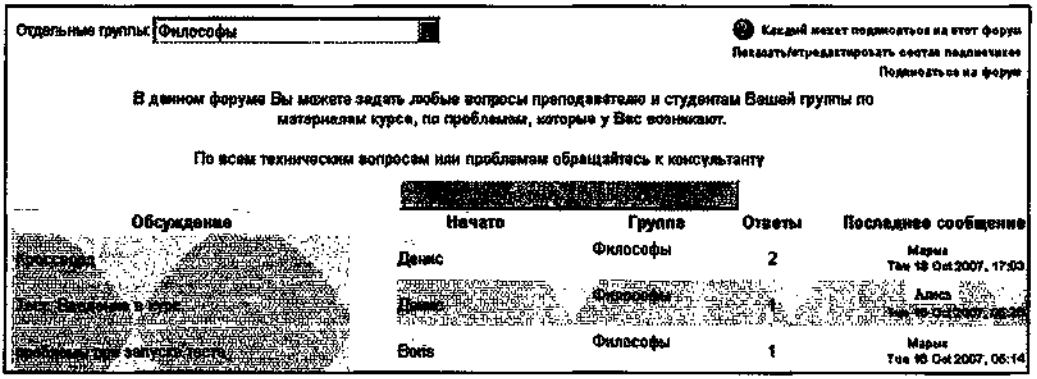


Рис. 6.8

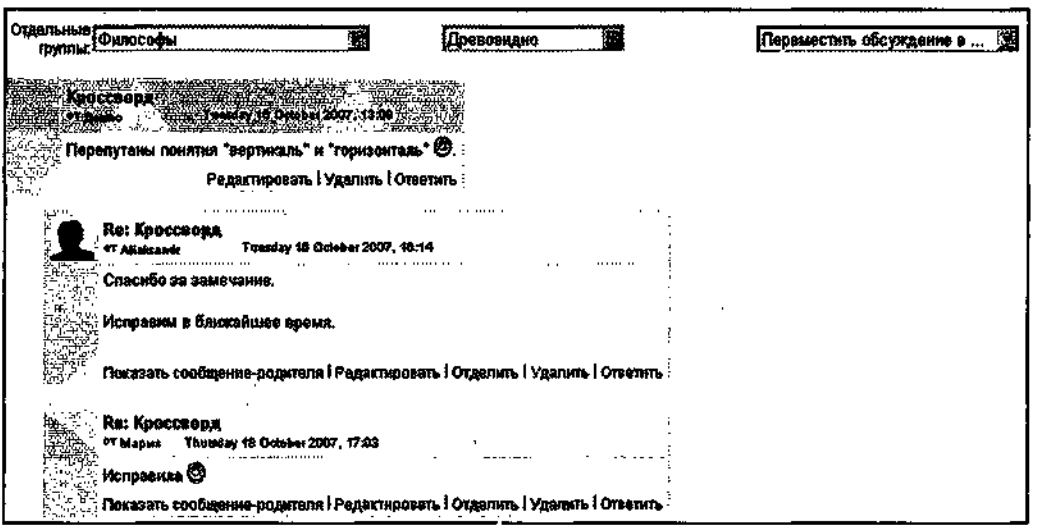


Рис. 6.9

36.4. Новости в сети Интернет

Сеть Интернет предоставляет пользователям разнообразную информацию, содержащую новости политики, культуры, науки, техники, здравоохранения и др.

Размещаться эти новости могут на сайтах ведущих телерадиокомпаний, телеграфных агентств, газет, журналов.

Например, новости нашей республики мы можем прочитать на сайте Белорусского телеграфного агентства (рис. 6.10).

Рис. 6.10

В сети Интернет существует служба сетевых новостей (Usenet), которая дает возможность пользователям пересылать сообщения и статьи на специальные электронные форумы и конференции. Посланное пользователем сообщение в соответствующую группу новостей становится доступным всем участникам дискуссии.

Служба сетевых новостей позволяет организовать списки рассылки. Электронное сообщение, попавшее в список рассылки, затем отправляется всем пользователям сети, подписавшимся на этот список.

Тематика группы новостей обычно определяется ее названием. Примеры некоторых таких названий приведены в таблице:

Название группы	Содержание темы новостей
Компьютеры (comp)	Обсуждение интересов компьютерных профессионалов и обычных пользователей
Отдых (rec)	Отдых, спорт, увлечения
Наука (sci)	Дискуссии, связанные с научной деятельностью
Общество (soc)	Обсуждение социальных и культурологических вопросов
Музыка (music)	Дискуссии на музыкальные темы
Общение (talk)	Обсуждение общих тем

Для чтения электронных сообщений и статей с сервера, обслуживающего сетевые новости, можно использовать программы-браузеры, например Netscape Communicator, или специальные программы для чтения новостей: NewsXpress, Forte Agent и др.

Специальный протокол передачи сетевых новостей NNTP (Net News Transfer Protocol) обеспечивает получение сетевых новостей и возможность помещения информации на доски объявлений сети.

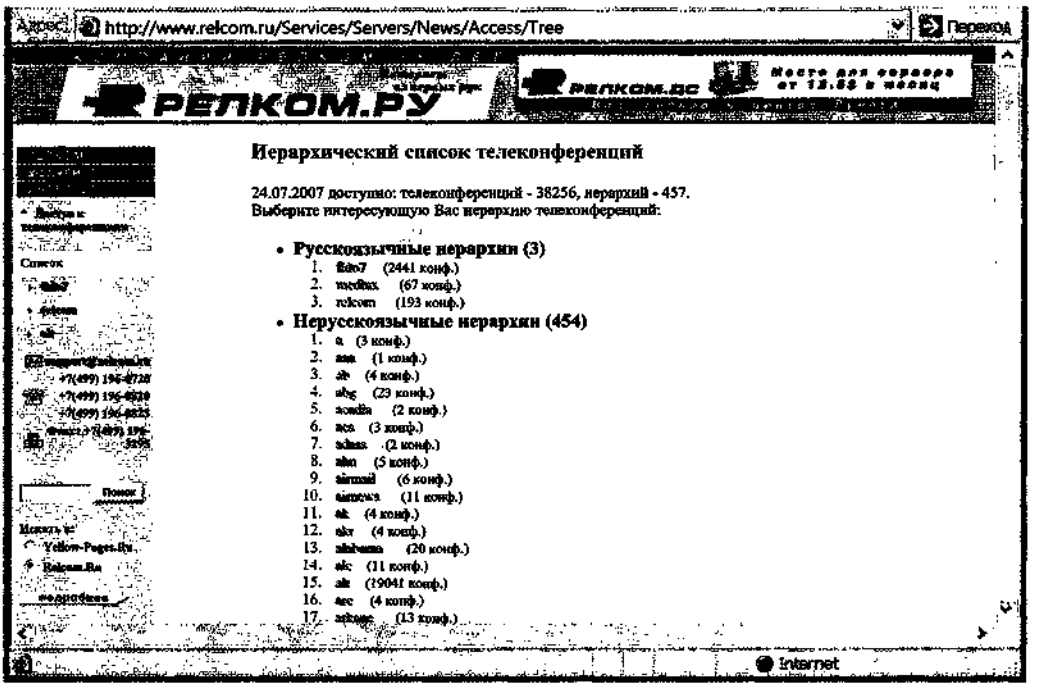


Рис. 6.11

Поддержку доступа пользователей к сетевым новостям осуществляют специализированные сайты, например сайт Релком.ру (рис. 6.11).

В настоящее время широкое распространение получила Интернет-технология RSS, которая позволяет осуществлять трансляцию материалов сайта. Эта технология, используя наборы стандартных схем разметки, облегчает компьютерным программам понимание структуры размещения информации на сайтах.

Технология RSS позволяет создавать список заголовков новостей, статей, анонсы. Пользователь, прочитав заголовок или анонс, решает, какой материал он будет смотреть. Это очень удобно для быстрого поиска информации на сайтах.

1. Какие службы сети Интернет Вам известны?
2. Какие функции выполняет протокол TCP/IP?
3. Какие протоколы используются при работе электронной почты?
4. Какие возможности предоставляют пользователям сети Интернет средства обмена информацией в режиме реального времени?
5. Что понимается под телеконференцией в сети Интернет?
6. Для чего используются Форумы в сети Интернет?
7. Какие типы новостей предоставляет сеть Интернет?

§ 37. Телекоммуникации в образовании и культуре

Нам уже известно, что коммуникационные технологии описывают процессы передачи информации.

Под телекоммуникацией понимается передача информации на большие расстояния, которая осуществляется с помощью телеграфа, телефона, радио, телевидения, компьютерных сетей.

Телекоммуникации в образовании и культуре, размещенные в сети Интернет, представляются *информационными ресурсами, проектами* в области образования и культуры, *системами дистанционного обучения* и т. д.

Под информационными ресурсами в области образования и культуры понимаются образовательные сайты, сайты учебных учреждений и учреждений культуры, электронные каталоги библиотек, издательств, электронные материалы (книги, журналы, газеты) и т. д.

Белорусские информационные ресурсы в области культуры широко представлены в сети Интернет. Среди них сайты театров, например сайт Национального академического театра им. Я. Купалы (рис. 6.12), сайты музеев и др.

Огромный интерес у пользователей сети Интернет вызывают сайты международных и национальных фестивалей в области литературы, искусства, музыки, народного творчества и др. Большой популярностью в мире пользует-

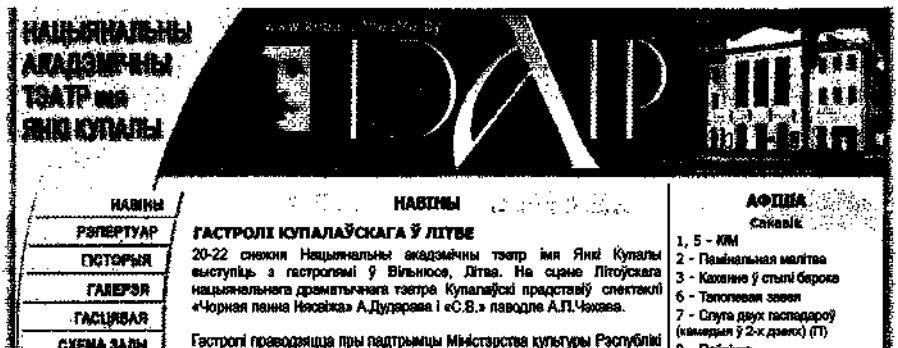


Рис. 6.12

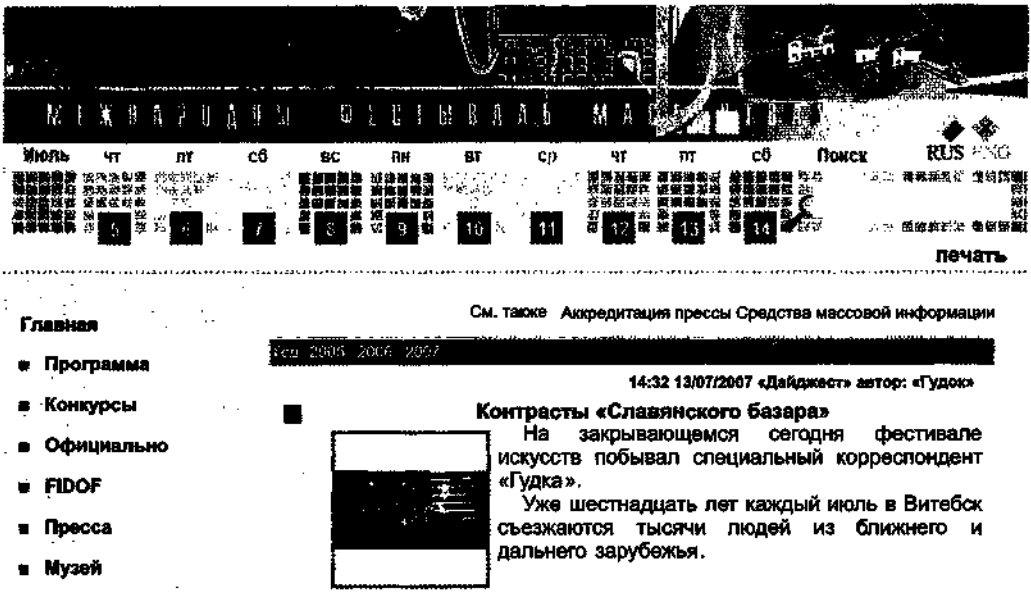


Рис. 6.13

ся сайт Международного фестиваля искусств «Славянский базар в Витебске» (рис. 6.13).

С некоторыми информационными ресурсами в области образования, такими, как сайты учреждений образования, электронными каталогами библиотек, мы уже знакомились ранее.

Посещение пользователями образовательных сайтов позволяет расширить и углубить их знания по различным учебным дисциплинам. Такие сайты обычно содержат большое количество учебных книг и материалов, тесты, рефераты, экзаменационные и олимпиадные задачи с решениями, анимацию различных природных явлений и процессов и др.

В настоящее время большое распространение в сети получили телекоммуникационные проекты. В основном такие проекты являются учебными.

Под учебным телекоммуникационным проектом понимается совместная творческая, исследовательская, учебно-познавательная, игровая деятельность участников проекта, организованная на основе компьютерной телекоммуникации.

Каждый телекоммуникационный проект создается с определенными целями для решения общих для участников проекта проблем и направлен на достижение совместного результата.

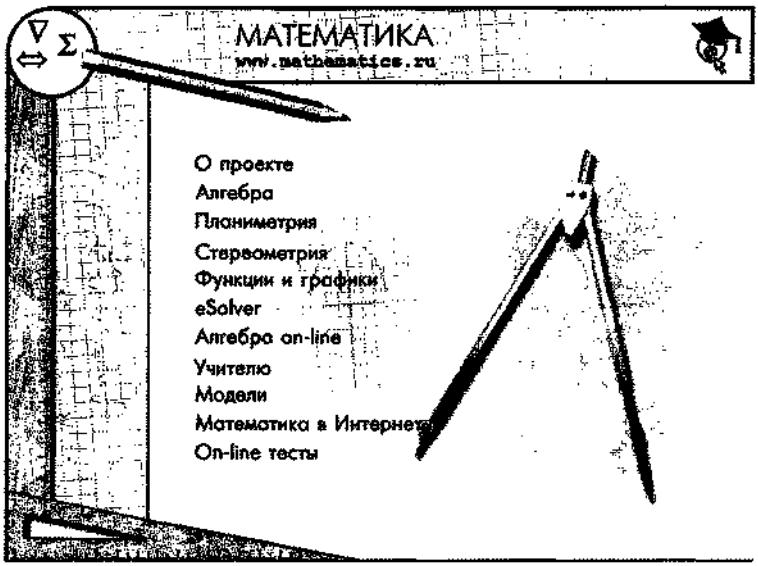


Рис. 6.14

Примером известного учебного проекта является проект «Открытый колледж», который предназначен для расширения и углубления знаний школьников по различным учебным дисциплинам: физика, математика и т. д. (рис. 6.14).

Телекоммуникационные проекты во многих странах мира являются неотъемлемой частью учебного процесса. Телекоммуникационная сеть Janet объединяет все университеты Англии, а в проекте Campus-2000 принимают участие 2000 школ из разных стран.

Темы телекоммуникационных проектов носят самый разнообразный характер, например культурологический — «Театр», «Праздники и обычаи народов мира» или узкопредметный — например, «Химический проект».

Широкое распространение в компьютерной сети получило дистанционное обучение.

Дистанционное обучение — это взаимодействие преподавателя и обучаемого на расстоянии, отражающее все присущие учебному процессу компоненты (цели, содержание, методы, организационные формы, средства обучения) и реализуемые средствами Интернет-технологий.

Наиболее распространенными системами дистанционного обучения являются Moodle, Web Tutor, Stellus и др.

Например, система Moodle — это система управления содержимым сайта (Course Management System — CMS), специально разработанная для создания качественных онлайн-курсов преподавателями.

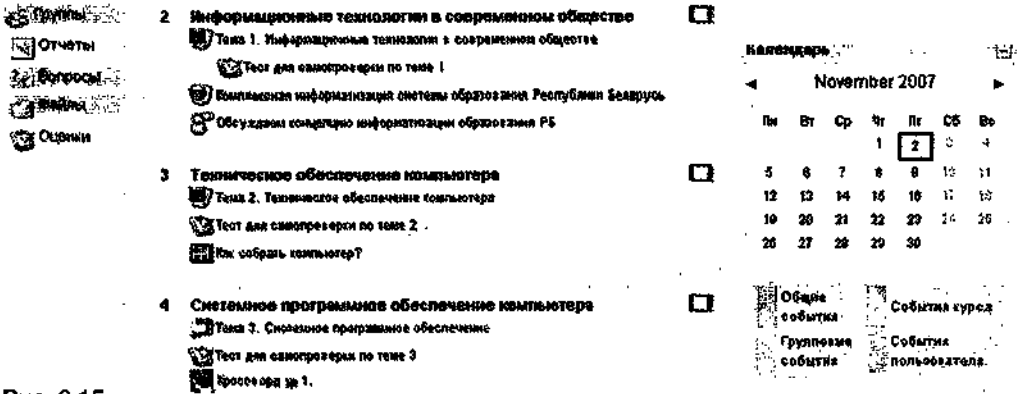


Рис. 6.15

Данная система состоит из отдельных учебных курсов. Под курсом в рамках системы не всегда понимается процесс обучения по какой-то заранее определенной программе. Курс может являться просто средой общения круга заинтересованных людей в рамках одной тематики.

В системе Moodle существуют пять типов пользователей: администраторы, создатели курсов, преподаватели, студенты и гости. Каждый из них имеет определенные права на доступ к ресурсам системы.

Например, учебный курс «Современные информационные технологии» в системе Moodle может содержать электронные учебные материалы, тематические форумы по обсуждению различных проблем, тесты для самопроверки, контрольные тесты, кроссворды (рис. 6.15). На курс обучаемые должны зарегистрироваться, координаторы курса формируют виртуальные группы обучаемых студентов или учащихся. Результаты выполнения тестов фиксируются в специальных оценочных таблицах.

1. Какие информационные ресурсы сети Интернет относятся к ресурсам в области образования и культуры?
2. Что содержат образовательные сайты?
3. Приведите примеры сайтов из области культуры.
4. Что понимается под учебным телекоммуникационным проектом?
5. Что представляет собой дистанционное обучение?

Упражнение

Откройте с помощью программы-браузера, сайты, содержащие информацию в области образования и культуры. Какая информация на них хранится?

а) образовательные сайты:

Астрономия	
http://www.astrolab.ru	Ресурс об объектах Вселенной
http://www.astrotop.ru/	Каталог астрономических ресурсов
Русский язык и литература	
http://www.gramma.ru/	Сайт «Культура письменной речи»
http://slovar.boom.ru/	Толковый словарь современной компьютерной лексики
http://bookz.ru/	Большая электронная литературная библиотека
Математика	
http://www.mathnet.spb.ru	Сайт «Элементарная математика»
http://www.mathtest.ru/	Тестирование по математике для школьников
Физика	
http://school.komi.com/	Дистанционная физическая школа
http://www.fizika.ru/	Сайт «Физика.ru: учебники, тесты, лабораторные занятия».
Химия	
http://www.chemistry.ssu.samara.ru/	Сайт «Органическая химия»
http://optimag.narod.ru/	Химия для любознательных
Английский язык	
http://www.study.ru/	Большой портал по изучению английского языка
http://www.translate.ru/	Онлайн-словарь и переводчик

б) информационные сайты в области культуры:

http://www.balet.by/	Национальный академический Большой театр балета Беларуси
http://www.theatre.vitebsk.by	Национальный академический драматический театр им. Я. Коласа

http://www.menka-museum.com/	Беларускі дзяржаўны музей дойлідства і побыту
http://www.russianmuseum.spb.ru/	Художественные музеи и картинные галереи России
http://www.hist.msu.ru/ER/museum.htm	Музеи мира в сети Интернет
http://forums.tut.by/	Форум «Культура и искусство Беларуси»
http://www.museumshome.com/	Музеи Беларуси

*§ 38. Коммерческая и рекламная деятельность в сети Интернет

Коммерческая деятельность в сети Интернет (электронная коммерция) является важнейшим составляющим элементом рынка товаров и услуг.

Формы электронной коммерции весьма разнообразны. К ним относятся электронные магазины, аукционы, корпоративные порталы и др.

В настоящее время организация коммерческой деятельности в сети Интернет подразделяется на несколько основных видов:

- предложение товаров в сетях;
- системы и средства платежей;
- электронная продажа товаров.

Предложение товаров предполагает размещение на веб-сайтах наборов товаров, формируемых по определенным признакам. Ассортимент товаров предназначен для удовлетворения потребностей покупателей.

Предлагаемые товары обычно размещаются на специальной электронной витрине. На этой витрине покупатель должен иметь возможность быстро находить необходимые товары. Многие электронные витрины содержат механизмы обработки заказов и позволяют пользователю получать итоговую квитанцию по отобранным товарам.

Примером электронной витрины является витрина белорусского торгового портала shop.open.by (рис. 6.16).

Системы платежей в компьютерной сети подразделяются на три основных вида: система с предоплатой, система с оплатой в момент совершения сделки и оплата по факту получения товара.

В качестве средств оплаты в сети могут выступать расчетные, кредитные, интеллектуальные карты клиентов и электронные деньги.

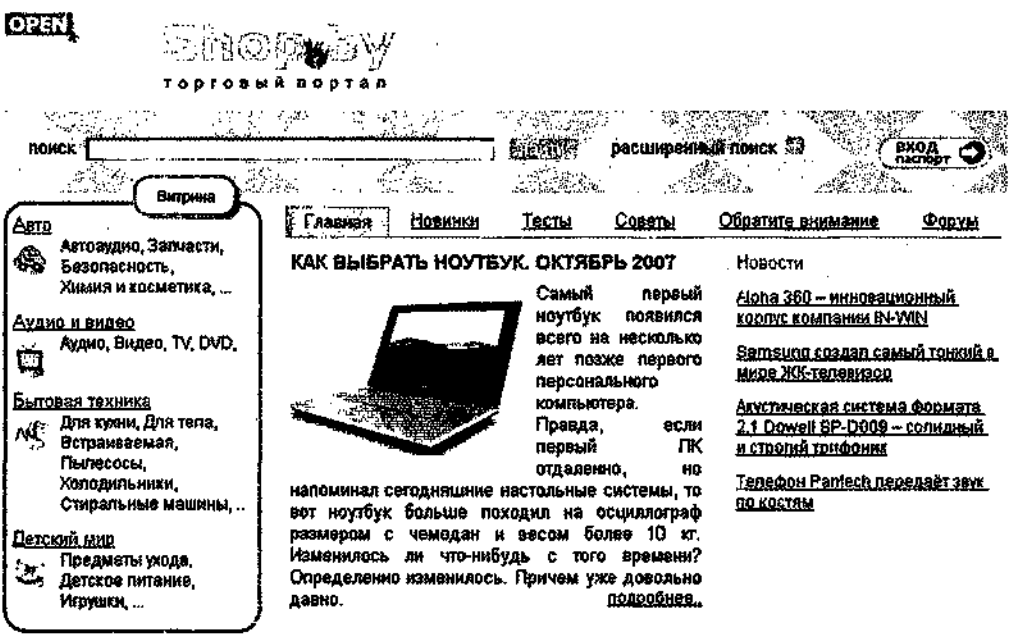


Рис. 6.16

Например, электронные деньги представляют собой чеки или сертификаты. Расчеты выполняются посредством списания определенного количества платежных единиц с одного счета и зачисления на другой.

В настоящее время широко используются электронные платежные средства на основе системы обозначений денежных переводов. Например, такой вид операций предоставляет клиентам Visa и MasterCard.

Большинство современных банков предоставляют возможность переводить денежные средства непосредственно с банковского счета клиента на счет торговых организаций или организаций, предоставляющих различные услуги. Например, используя банковскую карточку клиент Беларусбанка с помощью банкомата может оплатить коммунальные услуги и услуги мобильной связи.

Электронная продажа товаров является наиболее сложной системой продажи и состоит из нескольких этапов: **Просмотр каталога → Выбор товара → Оформление заказа** (рис. 6.17) **→ Формы оплаты и получения товара** (рис. 6.18).

Реклама в сети Интернет в основном размещается на веб-сайтах, баннерах, поисковых системах, каталогах и предназначена для информирования потенциальных покупателей о свойствах и качестве товаров и услуг.

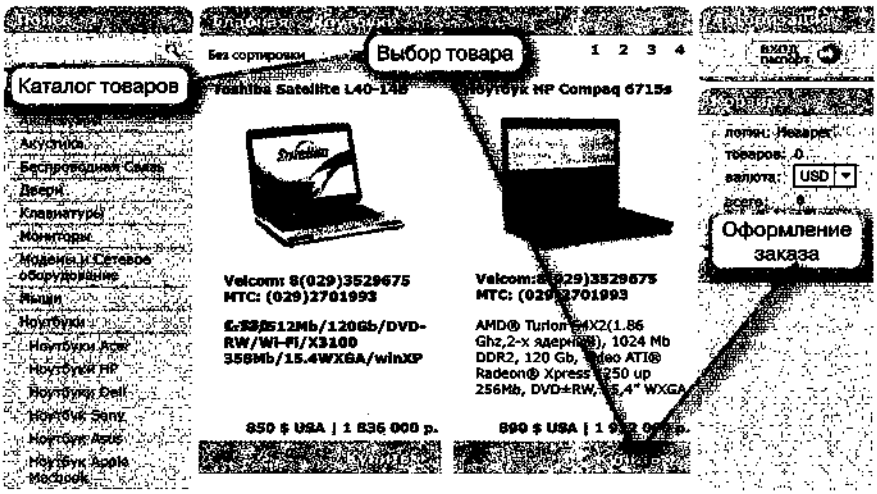


Рис. 6.17

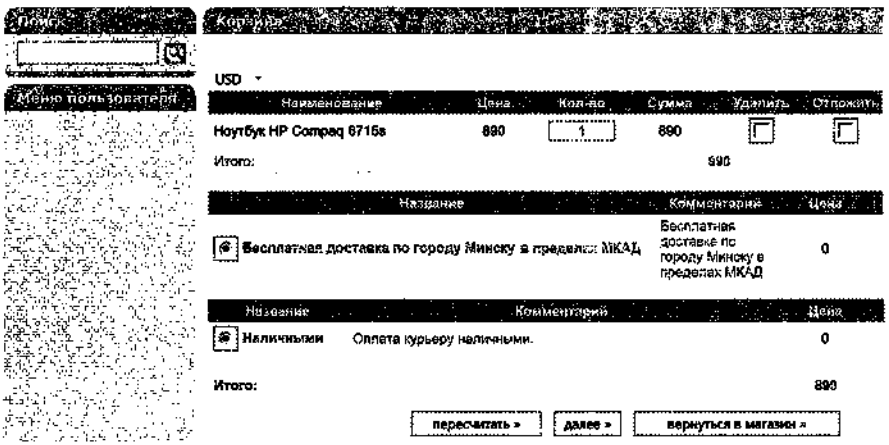


Рис. 6.18

Знакомство пользователей с рекламой в сети обычно осуществляется на двух уровнях.

На первом уровне пользователь взаимодействует с внешней рекламой на основании баннеров (рис. 6.19), текстовых и графических блоков, по рекламным каталогам и информации, размещенной в поисковых системах. Такое взаимодействие пользователь не контролирует. Он сталкивается с рекламой пассивно в результате просмотра веб-сайта, посещение которого часто совершенно не связано с рекламой.

На втором уровне пользователь просматривает непосредственно веб-сайт рекламодателя. Это происходит после щелчка левой клавишей мыши по баннеру или специальной рекламной ссылке. Просмотр пользователем веб-сайта рекламодателя является уже активным его знакомством с рекламой.

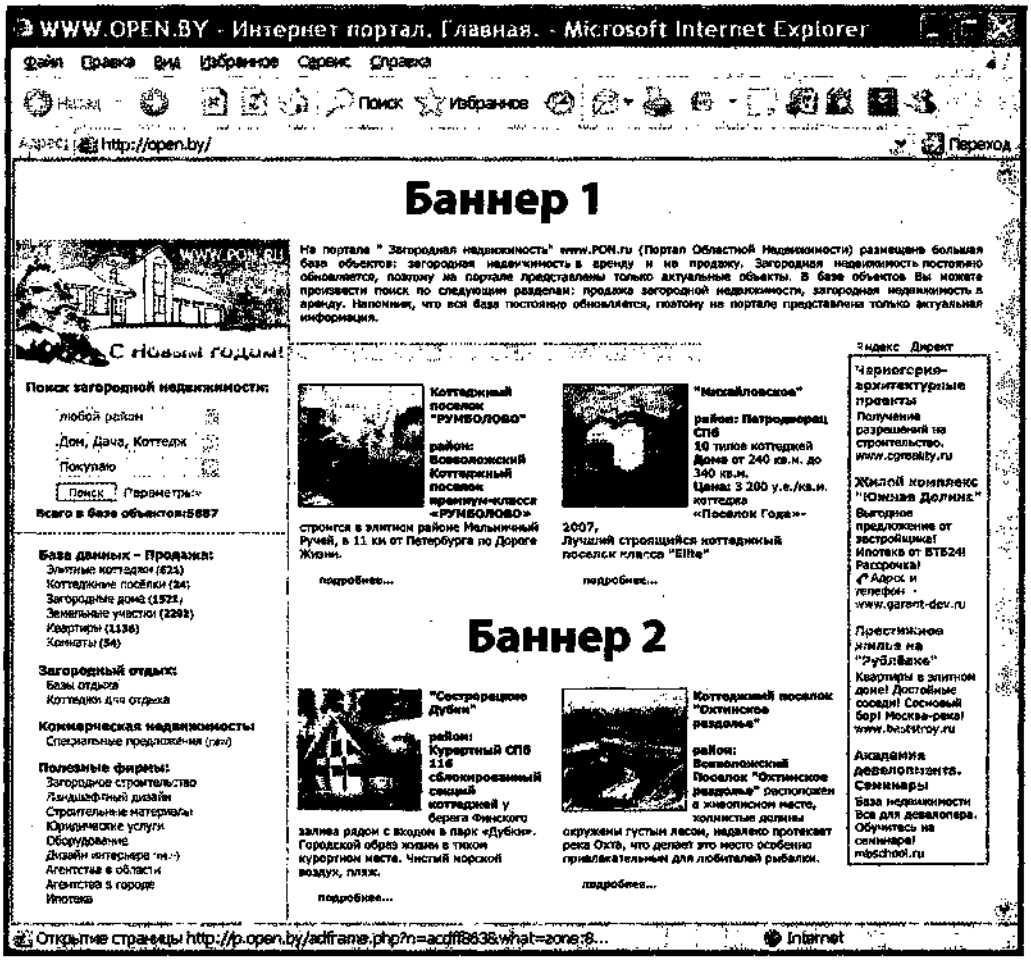


Рис. 6.19

Рекламные сайты обычно состоят из одной или нескольких веб-страниц и содержат информацию о государственных или коммерческих организациях, предлагаемой ими продукции, товарах, услугах, которые могут заинтересовать покупателей.

1. На какие виды подразделяется коммерческая деятельность в сети Интернет?
2. Из каких этапов состоит процесс электронной продажи товаров?
3. Где размещается электронная реклама в сети Интернет?
4. Что размещается на веб-сайтах рекламодателей?
5. Что представляют собой баннеры?

ПРИЛОЖЕНИЕ 1

Работа в интегрированной среде программирования Borland Pascal 7.0

Меню File

Назначение: выполнение операций с файлами текстов программ.

Команда	Назначение
New	Открытие окна для нового файла
Open (F3)	Открытие (загрузка) файла
Save (F2)	Сохранение файла с прежним именем
Save As	Сохранение файла с новым именем
Save All...	Сохранение файлов всех открытых окон
Change dir...	Установка текущего каталога
Exit (Alt+X)	Выход из среды программирования

Меню Edit

Назначение: выполнение операций редактирования текста программы.

Команда	Назначение
Undo (Alt+Backspace)	Отмена последней операции редактирования текста (откатка)
Cut (Shift+Delete)	Перемещение выделенного фрагмента текста из окна редактора в буфер обмена (промежуточную память)
Copy (Ctrl+Insert)	Копирование выделенного фрагмента из окна редактора в буфер обмена
Paste (Shift+Insert)	Копирование выделенного текста из буфера обмена в окно редактора
Clear (Ctrl+Delete)	Удаление выделенного фрагмента текста, минуя буфер обмена

Меню Search

Назначение: выполнение операций поиска и замены в тексте программы.

Команда	Назначение
Find	Поиск текста
Replace	Поиск текста и замена его новым текстом

Меню Run

Назначение: выполнение команд управления ходом выполнения программы.

Команда	Назначение
Run (Ctrl+F9)	Компиляция, компоновка и выполнение программы
Trace into (F7)	Построчное выполнение программы
Go to cursor (F4)	Выполнение программы до курсора

Меню Compile

Назначение: выполнение команд компиляции и компоновки программы.

Команда	Назначение
Compile (Alt+F9)	Компиляция программы
Information	Получение информации о программе и системе

Меню Debug

Назначение: выполнение команд, связанных с отладкой программы.

Команда	Назначение
User screen (Alt+F5)	Открытие окна пользователя (окна вывода результатов выполнения программы)

Меню Window

Назначение: выполнение команд управления окнами.

Команда	Назначение
Tile	Разместить окна мозаикой
Cascade	Разместить окна каскадом
Close all	Закрыть все окна
Next (F6)	Перейти к следующему окну
Close (Alt+F3)	Закрыть активное окно

Меню Help

Назначение: получение справочной информации.

Команда	Назначение
Contents	Получение содержания справочной системы
Index (Shift+F1)	Вызов тематического указателя
Topic search (Ctrl+F1)	Вызов помощи по заданной теме
Procedures and functions	Получение списка процедур и функций
Reserved words	Получение списка ключевых слов языка Паскаль
Standart units	Получение списка стандартных модулей
Borland Pascal Language	Получение перечня основных терминов языка Паскаль
Error massages	Получение информации о системе сообщений об ошибках

Окно текстового редактора

Назначение: создание и редактирование текстов программ (ввод текста программы, удаление и вставка символов и строк, «разрезание» и «склеивание» строк, поиск и замена текста, операции с фрагментами текста).

Для перемещения курсора используются курсорные клавиши, клавиши Home, End, PageUp, PageDown.

Смена режимов ввода текста (вставка/замена) — Insert.

Удаление текущей строки — Ctrl+Y.

Удаление символа слева от курсора (под курсором) — Backspace (Delete).

Получение отступов в строке — Tab.

Отмена последней операции редактирования текста (откатка) — Alt+Backspace.

Переключение с латинского шрифта на русский — Ctrl, или левая и правая клавиши Shift, или иначе (способ переключения устанавливается при загрузке русификатора среды программирования).

Максимальная ширина окна редактора составляет 78 знаков. При продолжении ввода текста он смещается влево.

ПРИЛОЖЕНИЕ 2

Ошибки компиляции

Причина возникновения: ошибки, связанные с нарушением синтаксиса языка Паскаль.

Код ошибки	Характер ошибки
1	Out of memory (Компилятору не хватает памяти)
2	Identifier expected (Требуется идентификатор) На месте позиционирования курсора должен находиться идентификатор; возможно, неправильно используется зарезервированное слово
3	Unknown identifier (Неизвестный идентификатор) Идентификатор не описан или неправильно написано зарезервированное слово
4	Duplicate identifier (Повторение идентификатора) Идентификатор описан дважды
5	Syntax error (Синтаксическая ошибка) Обнаружен недопустимый символ; возможно, строка не ограничена апострофами
6	Error in real constant (Ошибка в константе вещественного типа) Неверный синтаксис константы вещественного типа
7	Error in integer (Ошибка в константе целого типа) Неверный синтаксис константы целого типа
8	String constant exceeds line (Выход строки за допустимые границы) Возможно, не закрыт апостроф строки
10	Unexpected end of file (Неожиданный конец файла) Не закрыт комментарий, или нарушена вложенность операторов Begin .. End, или количество этих операторов различно, или нет точки в конце программы
11	Line too long (Строка слишком длинная) Превышена длина строки (она не может превышать 127 символов)
12	Type identifier expected (Требуется идентификатор типа) Не указан идентификатор типа

Продолжение

Код ошибки	Характер ошибки
16	Disk full (Диск заполнен) Отсутствует место на диске
20	Variable identifier expected (Требуется идентификатор переменной) В указанном месте должен находиться идентификатор переменной
26	Type mismatch (Несоответствие типов) Типы переменной и выражения, или операндов в выражении, или фактических и формальных параметров несовместимы
30	Integer constant expected (Требуется целая константа)
31	Constant expected (Требуется константа)
32	Integer or real constant expected (Требуется целая или вещественная константа)
33	Pointer Type identifier type (Требуется идентификатор типа)
34	Invalid function result type (Недопустимый тип функции)
36	BEGIN expected (Требуется Begin)
37	END expected (Требуется End)
38	Integer expression expected (Требуется выражение типа Integer)
40	Boolean expression expected (Требуется выражение типа Boolean)
41	Operand types do not match operator (Типы операндов не соответствуют типу знака операции)
42	Error in expression (Ошибка в выражении)
43	Illegal assignment (Неправильное употребление присваивания)
50	DO expected (Требуется зарезервированное слово DO)
54	OF expected (Требуется зарезервированное слово OF)
57	THEN expected (Требуется зарезервированное слово THEN)
58	TO or DOWNTO expected (Требуется зарезервированное слово TO или DOWNTO)
62	Division by zero (Деление на нуль)

Продолжение

Код ошибки	Характер ошибки
64	Cannot Read or Write variables of type (Нельзя вводить или выводить переменные этого типа)
66	String variable expected (Требуется строковая переменная)
67	String expression expected (Требуется выражение строкового типа)
71	Internal stack overflow (Внутреннее переполнение стека)
74	Constant and case types do not match (Типы констант и выражения оператора case не соответствуют друг другу)
76	Constant out of range (Выход константы за пределы допустимого диапазона)
79	Integer or real expression expected (Требуется выражение типа Real или Integer)
85	";" expected (Требуется ";")
86	":" expected (Требуется ":")
87	"," expected (Требуется ",")
88	"(" expected (Требуется "(")
89	")" expected (Требуется ")")
90	"=" expected (Требуется "=")
91	":=" expected (Требуется ":=")
94	"." expected (Требуется ".")
97	Invalid FOR control variable (Ошибочен тип переменной параметра цикла оператора FOR)
98	Integer variable expected (Требуется переменная целого типа)
100	String length mismatch (Ошибочна длина строковой переменной)
102	String constant expected (Требуется константа строкового типа)
103	Integer or real constant expected (Требуется константа типа integer или real)
106	Character expression expected (Требуется символьный тип выражения)

Продолжение

Код ошибки	Характер ошибки
108	Overflow in arithmetic operation (Переполнение при выполнении арифметических операций)
112	Case constant out of range (Выход значения селектора за пределы допустимых границ)
113	Error in statement (Ошибка в операторе)
140	Invalid floating point operation (Недопустимая операция с плавающей точкой)
143	Invalid procedure or function reference (Недопустимое обращение к процедуре или функции)
166	Procedure or function identifier expected (Требуется идентификатор процедуры или функции)

Ошибки выполнения

Причина возникновения: семантические ошибки, связанные с неправильным использованием элементов и конструкций языка Паскаль.

Код ошибки	Характер ошибки
106	Invalid numeric format (Ошибочное арифметическое выражение для ввода)
200	Division by zero (Деление на нуль)
201	Range check error (Выход за границы допустимого диапазона)
205	Floating point overflow (Переполнение при выполнении операции с плавающей точкой)
207	Invalid floating point operation (Недопустимая операция с вещественным числом) Использование отрицательного аргумента в функции sqrt, или неверное использование функций Trunc или Round, или неположительный аргумент в функции Ln
215	Arithmetic overflow error (Ошибка переполнения при выполнении арифметической операции)

Содержание

Глава 6
Коммуникационные технологии

(Название и номер школы)

Учебный год	Имя и фамилия ученика	Состояние учебного пособия при получении	Оценка ученику за пользование учебным пособием
20 /			
20 /			
20 /			
20 /			
20 /			

Учебное издание

Пупцев Александр Евгеньевич
Макарова Нина Петровна
Заборовский Георгий Александрович
Черняк Аркадий Александрович

ИНФОРМАТИКА

Учебное пособие для 11 класса
общеобразовательных учреждений
с русским языком обучения
с 12-летним сроком обучения
(базовый и повышенный уровни)

2-е издание, дополненное

Зав. редакцией *В. Г. Бехтина*. Редакторы *Н. М. Алганова, К. М. Лукашевич*. Оформление *Б. Г. Клюйко*. Художественный редактор *Л. В. Павленко*. Технический редактор *М. И. Чепловодская*. Компьютерная верстка *Г. А. Дудко*. Корректоры *Т. Н. Ведерникова, Е. П. Тхир, Д. Р. Лосик, В. С. Бабеня*.

Подписано в печать 12.03.2008. Формат $70 \times 90^1/_{16}$. Бумага офсетная. Гарнитура литературная. Офсетная печать. Усл. печ. л. 16,38. Уч.-изд. л. 11,9. Тираж 153 000 экз. Заказ 170.

Издательское республиканское унитарное предприятие «Народная асвета» Министерства информации Республики Беларусь. ЛИ 02330/0131732 от 01.04.2004.
220004, Минск, проспект Победителей, 11.

Республиканское унитарное предприятие «Минская фабрика цветной печати».
ЛП № 02330/0056853 от 30.04.2004. 220024, Минск, Корженевского, 20.