С. Г. Никитенко

СВОБОДНОЕ ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ BASIC-256 для школы

Санкт-Петербург
«БХВ-Петербург»
2011

УДК 681.3.068(075.3)+800.92Basic-256
ББК 32.973.26–018.1я721
Н62

Никитенко С. Г.

Н62 Свободное программное обеспечение. BASIC-256 для школы. — СПб.: БХВ-Петербург, 2011. — 224 с.: ил. — (ИиИКТ)

ISBN 978-5-9775-0740-0

Для свободно распространяемой кроссплатформенной среды BASIC-256 рассматриваются: интерфейс пользователя, основные операторы, правила написания программ, примеры решения типовых школьных задач из курса информатики и ИКТ, методика решения заданий ЕГЭ. Подробно рассмотрены часто встречающиеся на олимпиадах различного уровня темы: ввод/вывод числовых и строковых данных посредством текстовых файлов и построение графиков функций. Приведены задания для тренировки учащихся, проведения зачетных работ и выполнения проектов. Все зачетные работы и проекты ориентированы на выполнение по строго индивидуальным заданиям. Также книга содержит справочную информацию по BASIC-256.

Для образовательных учреждений

УДК 681.3.068(075.3)+800.92Basic-256
ББК 32.973.26–018.1я721

ISBN 978-5-9775-0740-0

Оглавление

Введение

Свободное программное обеспечение в школе

Впервые идея о необходимости разработки свободной отечественной операционной системы (ОС) для использования в учреждениях бюджетной сферы на государственном уровне была высказана в послании Президента Федеральному собранию РФ в 2006 г. В конце 2007 г. Правительство России приняло решение о выделении 100 миллионов долларов для Министерства образования РФ на закупку комплекта лицензионного программного обеспечения (ПО) и поставку его в 52 000 школ России с лицензией до 31.12.2010 г. и о разработке комплекта свободного ПО (СПО) для школ[1]. Исполнение проекта было поручено по конкурсу ряду фирм: группе компаний "Армада", ООО "Альт Линукс", ОАО "Линукс Инк".

В 2008 г. в 1107 школах Татарстана, Пермского края и Томской области и еще в 770 школах из других регионов были установлены школьные версии ОС ALT Linux, проведено обучение педагогов и начата опытная эксплуатация комплекта СПО в учебном процессе и документообороте. В этом же году Министерством информационных технологий и связи РФ разработана "Концепция развития, разработки и использования свободного программного обеспечения в РФ". В марте 2009 г. Минкомсвязи опубликовало план перехода государственных органов на СПО. Летом 2009 г. во все школы России был поставлен комплект ПО "Первая Помощь 2.0" из 27 компакт-дисков и DVD, в состав которого входило 9 дисков с различными версиями школьных дистрибутивов ОС ALT Linux и с дидактическими материалами.

---

[1] Распоряжение Правительства РФ № 1147-р от 18.10.2007 г.

Экономическое положение разных регионов, возможности по обновлению парка компьютеров в школах очень разные. По данным автора (из бесед с учителями информатики и заместителями директоров по информационным технологиям московских школ) многие школы вообще не озабочиваются проблемой перехода на СПО: "Директор обещал, что найдет нужные 800 долларов на лицензионную Windows и прочие программы". Из бесед с директорами сельских школ Рязанской области: "Денег нет и не будет. Альтернативы переходу на Linux нет". Это крайние точки зрения. На самом деле есть школы — обладатели грантов приоритетного национального проекта "Образование" и других проектов, которые приобрели компьютерные классы с предустановленной лицензионной версией Windows 7/Vista. Есть школы, которые имеют по два и более компьютерных класса. Причем самые новые из них укомплектованы лицензионной Windows 7, а в остальных стоит Windows XP с истекшим 31.12.2010 г. сроком лицензии. Вдобавок практически во всех регионах есть школы с классами, оснащенными компьютерами фирмы Apple. На них в качестве базовой ОС установлена одна из версий лицензионной MAC OS X, в дополнение к которой обязательно установлена лицензионная Windows XP/Vista/7.

В упоминавшемся Распоряжении Правительства РФ рекомендуется дать школе возможность выбора. Возможные варианты представлены далее. Вариативность обеспечивается еще и тем, что наиболее популярные свободные программы (типа свободного офиса OpenOffice.org) имеют версии и для Windows, и для Linux, и для MAC OS X:

- ❒ лицензионные Windows XP/Vista/7, Office, графика, среды программирования;
- ❒ лицензионные Windows/MAC OS X, свободные Office, графика, среды программирования;
- ❒ полный переход на свободное ПО.

Возможны и промежуточные решения. Так, компьютеры бухгалтерии и администрации, использующие специальные программы, не имеющие аналогов в Linux (например, программы печати аттестатов, базы данных для отчетов по ЕГЭ), остаются с ОС

Windows, а компьютерные классы и учебные кабинеты, обеспечивающие учебный процесс, переводятся на Linux полностью или сохраняют лицензионную Windows или MAC OS X с необходимым набором свободного ПО. Возможен вариант, когда в современных компьютерных классах, обеспечивающих учебный процесс, устанавливаются и Windows, и Linux. Учащихся физико-математических, экономических, информационно-технологических классов есть прямой смысл знакомить и с Windows, и с Linux, и с соответствующими наборами приложений. Профессиональное образование в вузах и колледжах уже несколько лет постепенно переводится на двух- или даже трехплатформенное обучение.

Одна из главных проблем перехода на СПО в школе в сложившихся условиях — обеспечение методического единства в преподавании всех дисциплин, связанных с применением компьютеров в условиях использования разных операционных систем и комплектов приложений. Положение облегчается тем, что большинство популярных программ, включая среды программирования, имеют свободные аналоги для работы во всех перечисленных ОС: Windows, Linux, Mac OS X. Данные о соответствии сред программирования, использовавшихся в школах для изучения раздела "Основы алгоритмизации и элементы программирования" курса информатики и ИКТ, их свободным аналогам в составе школьных дистрибутивов ALT Linux и для ОС Windows приведены в табл. В1.

В этой книге мы предлагаем в качестве инструментального средства для ознакомления с элементарными основами программирования в курсе информатики и ИКТ использовать среду BASIC-256. Выбор обоснован следующими обстоятельствами. BASIC-256 уже поставлен во все школы России в составе школьных дистрибутивов ALT Linux. Последняя серия дистрибутивов ALT Linux 5.0.2 Школьный представлена в свободном доступе на сайте фирмы и содержит работоспособные версии приведенных в табл. В1 сред программирования. Существует версия BASIC-256 и для Windows. Поэтому есть возможность в школьных компьютерных классах с тремя упомянутыми ОС (Windows, ALT Linux, Mac OS X) обеспечить методическое единство в преподавании основ алгоритмизации и программирования за счет использова-

Таблица В1. Среды программирования, поставлявшиеся в школы, и их свободные аналоги

Назначение приложения	Среды, поставлявшиеся в школы в 1985–2009 гг.	Дистрибутивы Alt Linux 4.0–5.0.2 Школьный	Свободные или бесплатные приложения Windows
Школьный алгоритмический язык	Кумир	Кумир	
Исполнители для начального обучения основам алгоритмизации	Кенгуренок Пылесосик Муравей Машинист Кукарача (Таракан) в составе пакета Роботландия	Scratch	
LOGO, начальное обучение основам алгоритмизации и программирования	LogoWriter LogoMiry DLOGO	KTurtle	KTurtle, DLOGO
BASIC	QBasic Quick Basic Turbo Basic	BASIC-256	BASIC-256, Small BASIC
Pascal	Borland Pascal 7.0 Turbo Pascal 7.0	Free Pascal	Free Pascal Pascal ABC BlackBox Component Builder
Visual Basic	Visual Basic 2005 в составе Visual Studio 2005	Gambas	Gambas
Delphi (Object Pascal)	Borland Delphi 2006 в составе Borland Developer Studio 2006	Lazarus	Lazarus Borland Delphi 7 Lite
C/C++, Java	Borland Developer Studio 2006 Visual C++ 2005 в составе Visual Studio 2005	Eclipse Code::Blocks IDE	Eclipse Code::Blocks IDE

ния одного и того же продукта. Вдобавок, BASIC-256, пожалуй, ближе всех к популярной в прошлые годы среди учителей информатики среде программирования QBasic от Microsoft — ее использовали от 60 до 80 % учителей информатики вплоть до конца 2010 г.

Далее для BASIC-256 рассматриваются: интерфейс пользователя, основные операторы, правила написания программ, примеры решения типовых школьных задач из курса информатики и ИКТ, методика решения заданий ЕГЭ, приведены также задания для тренировки учащихся, проведения зачетных работ и проектов. Все зачетные работы и проекты ориентированы на выполнение их строго по индивидуальным заданиям — никаких традиционных среди многих учителей контрольных работ с 2–4 вариантами заданий автор, имеющий 10-летний опыт преподавания информатики в различных школах, ПТУ и колледжах г. Рязани, не признает: 30 учащихся — 30 пусть однотипных, но разных вариантов заданий по каждой зачетной теме.

ГЛАВА 1

Начальные сведения о BASIC-256

1.1. Что такое BASIC-256?

BASIC-256 — свободная версия популярного среди учителей информатики интерпретатора языка BASIC, используемая в соответствии с правилами GNU General Public License. Авторское право на эту программу принадлежит с 2006 г. группе The BASIC-256 Team. Первоначальное название языка — KidBasic.

Разработаны версии интерпретаторов языка для ряда платформ, в частности, для Linux и для 32-битных версий Windows. В состав серии дистрибутивов ALT Linux 5.0.2 Школьный (январь 2011 г.) включен BASIC-256 версии 0.9.6.58 со встроенным справочником в переводе В. Чёрного и С. Ирюпина. В июне 2010 г. для 32-битных версий Windows, включая соответствующие версии Windows Vista и Windows 7, разработан BASIC-256 версии 0.9.6p.

Сайт группы разработчиков (англоязычный) размещен по адресу [1]. Дистрибутив BASIC-256 версии 0.9.6p можно загрузить со страницы [2]. Руководство по основным структурам данных и операторам BASIC-256 версии 0.9.5 на английском языке доступно на странице [3], а на немецком языке — на странице [4].

В BASIC-256 версии 0.9.6p имеется встроенный англоязычный справочник RU-BASIC-256 Reference 0.9.5*x*, вызываемый по клавише <F1> [5]. Справочник [5] и руководство [7] в наибольшей степени соответствуют взятым за основу в пособии версиям

0.9.6p (2010-06-10) для Windows и 0.9.6.58 (2011-01-07) для ALT Linux 5.0.2 Школьный.

1.2. Интерфейс пользователя BASIC-256

После установки русской версии BASIC-256 (например, v.0.9.6p) запуск ее выполняется из меню Windows командой **Пуск | Программы | BASIC-256 | BASIC-256**. Открывшееся окно программы (рис. 1.1) имеет обычный для приложений Windows вид. Заголовок окна имеет стандартную структуру. Меню программы содер-

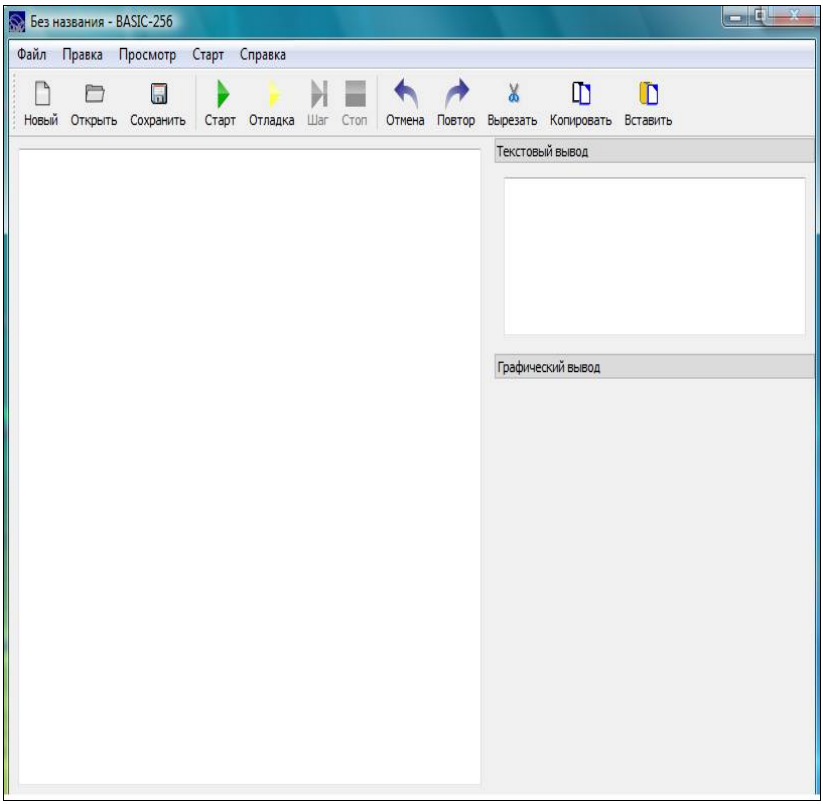

Рис. 1.1. Окно русских версий BASIC-256

жит пункты **Файл**, **Правка**, **Просмотр**, **Старт**, **Справка**. Смысл любого пункта интуитивно понятен при открытии его списка команд.

Расположенная под строкой меню панель инструментов содержит набор кнопок быстрого вызова наиболее употребительных команд: **Новый**, **Открыть**, **Сохранить**, **Старт**, **Отладка**, **Шаг**, **Стоп**, **Отмена**, **Повтор**, **Вырезать**, **Копировать**, **Вставить**. Кнопки формируют команды, позволяющие выполнять вызов программ из файлов, их сохранение, запуск, отладку и редактирование.

Рабочая область окна BASIC-256 выглядит несколько необычно по сравнению с более ранними версиями BASIC, такими как QBasic или Quick Basic. В формируемой по умолчанию рабочей области окна выделены три поля:

- ❒ слева находится основное поле — окно текста программы (окно редактора), управляемое редактором текста программы;
- ❒ справа в общем вертикальном столбце сформированы два поля (окна) исполнителя программы, управляемые отладчиком и средой исполнения:
  - верхнее поле называется **Текстовый вывод** и представляет собой экран исполнителя в текстовом режиме. На нем будут отображаться результаты вычислений и обработки данных, представляемые в виде текстовых (символьных) строк и выводимые чаще всего оператором `print`, а также сообщения об ошибках, формируемые самим интерпретатором на этапе синтаксического контроля программы пользователя;
  - нижнее поле называется **Графический вывод** и представляет собой графический экран исполнителя. На нем отображаются результаты выполнения графических операторов программы. По умолчанию графический экран имеет размеры 300×300 пикселов.

Характеристики текстового экрана (число строк, число символов в строке) зависят от типа и размера шрифта, используемого для вывода текстовой информации. Характеристики шрифта задаются специальными операторами BASIC-256 или командами меню.

Каждый из экранов очищается отдельной командой. Текстовый экран очищается обычной командой очистки текстового экрана `cls` (clear screen, очистить экран). Графический экран очищается специальной командой `clg` (clear graphic screen, очистить графический экран).

Окно текста программы присутствует на экране всегда. Экраны исполнителя (окно текста и окно графики) можно вызывать на экран или убирать с экрана командами меню **Просмотр | Окно текста** и **Просмотр | Окно графики**. По умолчанию эти функции включены — в соответствующем пункте меню стоит флажок ("галочка"). Одна подача команды включает функцию, повторная подача той же команды функцию отображения соответствующего окна на экране выключает. Отладчик позволяет при отладке программы вызвать на экран дополнительное окно для наблюдения за переменными программы — окно просмотра переменных. Оно вызывается на экран или убирается командой меню **Просмотр | Окно просмотра переменных**.

Размер шрифта, которым отображаются текст программы в окне редактора и результаты вычислений в окне текста, можно дискретно изменять командами меню **Просмотр | Размер шрифта | Мелкий / Средний / Большой / Огромный**. По умолчанию после загрузки BASIC-256 включен средний размер шрифта. Характеристики текстового экрана в режиме мелкого шрифта определяются экспериментально как 11 строк примерно по 40 символов `x`. Большинство шрифтов Windows не моноширинные, т. е. различные символы имеют различную ширину. Поэтому ширину строки в числе символов можно определить только приблизительно для некоторого символа со средней шириной. Если число выводимых строк превышает 9, в окне текста появляется вертикальная полоса прокрутки. Если выводимая строка превысит ширину окна текста, появится горизонтальная полоса прокрутки.

Запуск программы, текст которой подготовлен в окне редактора, осуществляется следующими способами: командой меню **Старт | Старт**; кнопкой **Старт** панели инструментов; клавишей <F5>.

Примечание

Клавиша <F5> используется для запуска приложений во многих реализациях BASIC.

1.3. Особенности BASIC-256 v.0.9.6p для Win32 и v.0.9.6.48 для ALT Linux 5.0.2 Школьный

Программа на языке BASIC-256 представляет собой последовательность строк кода. В каждой строке допускается один оператор или два оператора, разделенные двоеточием. Операторы в рассматриваемых версиях можно писать и строчными, и прописными латинскими буквами. То есть операторы `print x`, `PRINT x` и `Print x` будут выполняться одинаково. Функции `sin(x)`, `SIN(x)` и `Sin(x)` также будут вычисляться одинаково.

Имена переменных могут содержать только латинские буквы и цифры и должны начинаться с буквы. Причем, в именах переменных учитывается регистр: `x` и `X` — это разные переменные. Любые другие символы (символ подчеркивания, знаки препинания, знаки математических операций и другие специальные символы) в именах переменных недопустимы. Исключение составляет знак денежной единицы `$` — он (как во многих других версиях BASIC) ставится в конце имен переменных строкового типа.

В BASIC-256 используются только два типа данных: числа и символьные строки (строковые значения). Например, `z` — числовая переменная, а `z$` — строковая переменная (читается "z строковое"). Строковые значения и строковые константы пишутся в кавычках.

Оператор присваивания (так же, как и во всех версиях BASIC) — знак равенства `=`. Примеры записи операторов присваивания:

- `x = 5.47`
- `y$ = "Привет, BASIC-256"`

Оператор присваивания следует читать как "принять равным" или "присвоить значение". В правой части оператора присваивания можно записывать константу или выражение: `x = x + 5` (икс принять равным икс плюс пять).

Для чисел используется внутреннее представление в форме 32-битных двоичных чисел со знаком с плавающей запятой. В операторах и константах допускается запись чисел только в

естественной форме с десятичной точкой в качестве разделителя целой и дробной части: `2.345`; `-0.0076`.

> ***Внимание!***
>
> Запись чисел в экспоненциальной (полулогарифмической) форме в рассматриваемых версиях BASIC-256 не поддерживается.

Максимально представимые числа по абсолютной величине: `+/-1410065408`. Точность представления результатов при выводе на текстовый экран оператором `print` чисел N < 1 составляет 7 десятичных цифр, включая ноль целых с округлением до 6-го знака после запятой. Так `print 0.0000005` выведет на экран **0**, а `print 0.00000051` выведет **0.000001**. Другими словами, число |N1| <= 0.0000005 считается машинным нулем.

Из сложных структур данных в BASIC-256 определены только одномерные и двумерные массивы: числовые и строковые. Массивы описываются оператором `DIM`:

- `DIM Z(20)` — числовой массив из 20 элементов Z[0] — Z[19];
- `DIM V$(15)` — строковый массив из 15 элементов V$[0] — V$[14].

В общем случае оператор `DIM R(N)` при заданном `N` описывает `N` числовых переменных R[0] — R[N – 1], т. е. отводит для этих переменных необходимую область оперативной памяти.

Оператор `DIM U(N, M)` при заданных `N` и `M` описывает `N*M` числовых переменных U[0,0] — U[N – 1, M – 1]. Такой массив можно записать в виде прямоугольной таблицы из `N` строк по `M` чисел в каждой строке. Элемент U[i,j] при этом будет располагаться в строке i и иметь номер j слева: i = 0 — N – 1, j = 0 — M – 1.

Если размерность массива неизвестна, ее можно определить следующим образом: для одномерного массива `X(N)`: N = X[?]; для двумерного массива `U(N, M)`: N = U[?,], M = U[,?]. Каждый массив должен вводиться отдельным оператором `DIM`.

> ***Внимание!***
>
> В отличие от многих других версий, BASIC-256 не позволяет одним оператором `DIM` описать несколько массивов.

В BASIC-256 допустимы анонимные массивы — последовательности числовых или строковых констант в фигурных скобках, разделенных запятыми. Анонимные массивы можно использовать для присвоения значений обычным массивам:

```
dim Y(4): Y = {10, 20, 30, 40}
```

Приведенная последовательность операторов присвоит элементам массива Y[0] — Y[3] значения, перечисленные в списке анонимного массива: 10, 20, 30, 40. В общем случае, если в списке анонимного массива перечислены N значений, указанная последовательность операторов присвоит их элементам Y[0] — Y[N – 1].

Размерность числового или строкового массива можно переопределить в программе операторами вида `Redim x(n1)`, `Redim y$(n2)`, `Redim x1(n1, m1)`, `Redim y1$(n2, m2)`. При этом, если размерность массива увеличивается, добавленные элементы в числовом массиве инициализируются нулями, а в строковом массиве — пустыми строками. Если новая размерность массива меньше заданной ранее, "лишние" элементы теряются.

ГЛАВА 2

Основные операторы BASIC-256

2.1. Операторы обработки числовых данных

В отличие от многих других реализаций BASIC, оператор `print` в текущей версии BASIC-256 v.0.9.6p выводит на экран не список вывода, а результаты вычисления или обработки только одного выражения и имеет синтаксис `print expr`. Здесь `expr` — сокращение от expression — произвольное выражение, формирующее результат в числовой или строковой форме для вывода на текстовый экран (в текстовое окно) исполнителя. Следует заметить, что сам оператор `print` преобразует перед выводом на текстовый экран любой результат, в том числе числовой, в текстовую строку. Этот факт можно использовать для принудительного формирования списка вывода в составе `expr` в виде одной текстовой строки, как это делается в различных модификациях Visual Basic. Типовые приемы формирования списка вывода в составе `expr` в виде одной текстовой строки рассмотрены в *разд. 2.2*.

Оператор `input` обеспечивает диалоговый ввод данных в числовую или строковую переменную в следующих формах:

- `input expr, x1` — ввод числа с клавиатуры в переменную `x1`;
- `input expr, x2$` — ввод текстовой строки с клавиатуры в переменную `x2$`.

Здесь `expr` — необязательное строковое выражение, представляющее собой побуждающее сообщение для ввода. Например, первая строка приведенной ниже последовательности:

```
input "Введите целое x1 = ", x1
print x1
```

выведет на текстовый экран сообщение: **Введите целое x1 =**. Пользователь в той же строке вводит с клавиатуры нужное число и нажимает клавишу <Enter>. Переменной `x1` будет присвоено введенное число. Вторая строка выведет присвоенное `x1` значение в окно текста.

Над числовыми данными возможны следующие операции:

- сложение: `Y1 = X1 + X2`;
- вычитание: `Y2 = X1 - X2`;
- умножение: `Y3 = X1*X2`;
- деление: `Y4 = X1/X2`;
- возведение в степень: `Y5 = X1 ^ X2`;
- вычисление целого остатка `Y6` от деления целых чисел `X3` и `X4`: `Y6 = X3 % X4`;
- целочисленное деление `Y7 = X5\X6` — при этом вычисляется целая часть частного `Y7` от деления целых чисел `X5` и `X6`.

Например: команда `print 4.5^(-2.5)` выведет на экран результат вычисления: **0.026081**; команда `print 45 % 7` выведет остаток от деления 45 на 7, т. е. **3**; команда `print 45\7` выведет целую часть частного от деления 45 на 7, т. е. **6**.

Для числовых аргументов могут вычисляться следующие функции:

- `abs(x)` — вычисляет абсолютное значение выражения x, например: abs(–4.5) = 4.5;
- `x1 = ceil(x)` — вычисляет наименьшее целое x1 >= x, например: ceil(4.5) = 5; ceil(–4.5) = –4;
- `x2 = floor(x)` — вычисляет наибольшее целое x2 <= x, например: floor(4.5) = 4; floor(–4.5) = –5;

- `x3 = int(x)` — вычисляет целое число x3 путем отбрасывания дробной части числа x, например: int(4.5) = 4; int(–4.5) = –4;
- `pi`, `PI` — встроенная константа π = 3.141593;
- `cos(x)` вычисляет cos x;
- `sin(x)` вычисляет sin x;
- `tan(x)` вычисляет tg x;

> ***ВНИМАНИЕ!***
>
> Для всех тригонометрических функций аргумент `x` задается в радианах. Аргумент `x1`, заданный в градусах, необходимо преобразовать в радианы с помощью переводного множителя 1° = pi/180, например: `y1 = sin(x1*pi/180)`. Можно также воспользоваться для преобразования аргумента `x1`, заданного в градусах, в радианы специальной функцией `radians(x1)`, например:
>
> `y1 = sin(radians(x1))`.

- функция `y = atan(x)` вычисляет arctg x, причем `y` получится в радианах.

  Если необходимо получить результат `y2` в градусах, следует использовать выражение вида `y2 = atan(x)*180/pi` или воспользоваться для преобразования угла из радианной меры в градусную функцией `degrees`: `y2 = degrees(atan(x))`.

  Аналогично для вычисления математических функций arcsin x, arcos x можно в BASIC-256 пользоваться функциями `asin(x)`, `acos(x)`;
- вычисление корней произвольной степени выполняется через операцию возведения в степень:
  - квадратный корень вычисляется так: `x1 = x^(1/2)`;
  - кубический корень: `x2 = x^(1/3)`;
  - корень произвольной степени n: `x3 = x^(1/n)`;
- функция `y1 = log(x)` вычисляет натуральный (Неперов) логарифм ln x по основанию e = 2.73: log(10) = 2.302585;
- функция `y2 = log10(x)` вычисляет десятичный логарифм lg x: log10(1000) = 3;

- логарифм числа x по произвольному основанию a вида y3 = $\log_a x$ можно вычислить, используя стандартное математическое соотношение $\log_a x = \ln x / \ln a$. В BASIC-256 для этого придется записать одно из выражений: `y3 = log(x)/log(a)` или `y3 = log10(x)/log10(a)`.

> ***Примечание***
>
> Следует заметить, что имеющаяся во всех реализациях BASIC функция `exp(x) = e^x`, обратная натуральному логарифму, в BASIC-256 отсутствует. При необходимости ее использования в программе можно ввести константу `e = 2.718282` и использовать функцию вида `e^x` вместо `exp(x)`.

Оператор `rand` формирует псевдослучайное число x в диапазоне 0 <= x < 1, распределенное по закону равной плотности. В отличие от других реализаций BASIC инициализация генератора случайных чисел (функции `rand`) в BASIC-256 выполняется не аргументом, а автоматически при каждой очередной загрузке. После каждой новой загрузки BASIC-256 будет формироваться новая последовательность случайных чисел от 0 до 0.999999. Если необходимо сформировать случайную последовательность чисел в другом диапазоне для моделирования некоторой последовательности событий, можно воспользоваться правилом: любое выражение, содержащее функцию формирования случайной последовательности `rand`, дает в результате случайную последовательность. Формирование выражений можно пояснить следующими примерами:

- бросание монеты — орлу ставится в соответствие 1, решке — 0. Случайная последовательность равновероятных 1 и 0 формируется с помощью выражения: `x = int(2*rand);`
- бросание кубика — формирование случайной последовательности целых чисел `x1` в диапазоне от 1 до 6 реализуется выражением:

```
x1 = 1 + int(6*rand);
```
(2.1)

- вытаскивание карты из хорошо перемешанной колоды из 36 карт — если поставить в соответствие картам от валета до туза числа от 11 до 14, то без учета масти карт процесс моде-

лируется формированием последовательности случайных целых чисел `x2` в диапазоне от 6 до 14 согласно выражению:

```
x2 = 6 + int(9*rand).
```
(2.2)

В общем случае если некоторая последовательность событий моделируется последовательностью случайных целых чисел `x3` в диапазоне от `N1` до `N2`, то можно использовать выражение вида:

```
x3 = N1 + int((N2 - N1 +1)*rand).
```
(2.3)

Оператор `pause x` — задержка на `x` секунд. Здесь `x` — десятичное число. Так, команда `pause 0.15` задерживает исполнение программы на 0.15 секунды. Этот оператор полезен при формировании подвижных изображений в графическом или текстовом окнах или при отображении результата работы программы автоматически сменяющейся с определенной задержкой последовательностью кадров, т. е. в виде слайд-фильма.

Оператор `Sound` — воспроизводит звук или последовательность звуков заданной частоты `F` и длительности `T` через системный динамик. Частота задается в герцах (Гц), длительность в миллисекундах (мс): 1 мс = 0.001 с. Команда `Sound` имеет следующие форматы:

- `Sound F1, T1` или `Sound(F1, T1)` — воспроизводит звук частоты `F1` Гц длительностью `T1` мс, например, команда `Sound(1000, 5000)` воспроизведет звук частоты 1000 Гц длительностью 5 сек. Паузу можно задать частотой 0 и нужной длительностью;
- `Sound Y`, `Sound(Y)`, `Sound {F1, T1, F2,T2, ... , Fn, Tn}` — аргументом команды может быть массив `Y` из `n` пар чисел `Fi`, `Ti`, в том числе анонимный массив.

> ***Примечание***
>
> При некотором навыке в этой форме можно записывать мелодии (табл. 2.1). Слышимость звуков зависит от полосы пропускания системного динамика. Звуки очень низких и очень высоких звуковых частот (ниже 50 Гц, выше 7–8 кГц) могут не воспроизводиться системным динамиком и будут не слышны.

В табл. 2.1 приведены латинские и русские обозначения нот и соответствующие им частоты в Гц для двух октав. Ноты проме-

Таблица 2.1. Таблица соответствия частот и нот [7]

	A_{Sharp} / B_{Flat}	B	C	C_{Sharp} / D_{Flat}	D	D_{Sharp} / E_{Flat}	E	F	F_{Sharp} / G_{Flat}	G	G_{Sharp} / A_{Flat}
ЛЯ		СИ	ДО		РЕ		МИ	ФА		СОЛЬ	
440	466	493	523	554	587	622	659	698	740	784	
220	233	247	262	277	294	311	330	349	370	392	415
								175	185	196	208

жуточных частот обозначены следующим образом: C_{Sharp} это ДО-диез; D_{Flat} это РЕ-бемоль. Длительность звучания нот определяется темпом исполнения — числом ударов метронома в минуту. Каждый удар — четверть ноты. Темп 120 — это 120 ударов метронома в минуту. Целая нота — 2 сек. Половинная нота — 1 сек. Четверть ноты — 0,5 сек.

Оператор `Volume X` или `Volume(X)` — задает громкость звука, воспроизводимого оператором `Sound`. Уровень громкости задается целым числом `X` от 0 до 10. По умолчанию `X = 5`.

Оператор `System Z$` или `System(Z$)` — позволяет из программы в среде BASIC-256 выполнить системную команду (команду операционной системы), заданную текстовой строкой `Z$`, в окне терминала. Это аналог команды Windows **Пуск | Выполнить**. Применение такой команды требует большой осторожности и высокой квалификации пользователя.

2.2. Операторы обработки строковых данных

Оператор `Asc(x$)` — код первого символа строки `x$` или символа `x$`: Asc(A) = 65; asc(blue) = 98 — код символа b.

Оператор `Chr(N)` — формирует символ по коду `N`: chr(90) = Z; chr(123) = {.

Символы стандарта ASCII с кодами от 32 до 126 приведены в табл. 2.2.

Таблица 2.2. Кодирование символов стандарта ASCII (коды от 32 до 126)

Код	Символ	Код	Символ	Код	Символ	Код	Символ	Код	Символ	Код	Символ	Код	Символ	Код	Символ
32	Пробел	44	,	56	8	68	D	80	P	92	\	104	h	116	t
33	!	45	-	57	9	69	E	81	Q	93	]	105	i	117	u
34	"	46	.	58	:	70	F	82	R	94	^	106	j	118	v
35	#	47	/	59	;	71	G	83	S	95	_	107	k	119	w
36	$	48	0	60	<	72	H	84	T	96	`	108	l	120	x
37	%	49	1	61	=	73	I	85	U	97	a	109	m	121	y
38	&	50	2	62	>	74	J	86	V	98	b	110	n	122	z
39	‘	51	3	63	?	75	K	87	W	99	c	111	o	123	{
40	)	52	4	64	@	76	L	88	X	100	d	112	p	124	|
41	(	53	5	65	A	77	M	89	Y	101	e	113	q	125	}
42	*	54	6	66	B	78	N	90	Z	102	f	114	r	126	~
43	+	55	7	67	C	79	O	91	[	103	g	115	s	127	

Следует обратить внимание на два похожих при печати символа:

- chr(39) = ’ — апостроф;
- chr(96) = ` — символ слабого ударения.

Символ ПУСТО (пустая строка) отображается командой chr(0).

Коды от 1 до 31 таблицы символов ASCII отведены для неотображаемых управляющих символов. Например, команды chr(10), chr(13) при выводе в текстовое окно оператором `print` обеспечивают переход в начало следующей строки.

Знак `+` для строковых переменных и значений выполняется как операция конкатенации (объединения) текстовых строк. Строка программы вида:

```
x$ = x1$ + x2$ + x3$
```
(2.4)

сформирует объединенную текстовую строку `x$`, состоящую из трех составляющих текстовых строк `x1$`, `x2$`, `x3$`, записанных последовательно в порядке их следования в выражении. В выражении вида (2.4) могут использоваться и числовые переменные. Они автоматически будут преобразованы в строковое представление. Выражение вида:

```
y$ = "x1 = " + x1 + " " + "x2 = " + x2,
```
(2.5)

где обрабатываются одновременно числовые и строковые значения, сформирует результирующую текстовую строку `y$`, в которую числовые значения `x1` и `x2` войдут также в форме текстовых строк. Выражение (2.5) позволяет в операторе `print` выводить в одной строке значения нескольких переменных, если объединить их в общую текстовую строку, добавив в нее необходимые разделительные пробелы и знаки препинания. Для примера — последовательность строк программы (листинг 2.1)

Листинг 2.1

```
x1 = 25
x2 = -48
print "x1 = " + x1 + ", " + "x2 = " + x2
```

выведет в текстовое окно следующий результат в виде текстовой строки: **x1 = 25, x2 = –48**.

Функция `y$ = string(x)` — преобразует число `x` в строковое представление `y$` (см. функцию `float`).

Функция `x = Float(y$)` — преобразует строковое представление числа `y$` в число с плавающей запятой. Функция `float` — обратная для функции `string` и наоборот. `Float("137.89") = 137.89`.

Функция `N = Instr(S1$, S2$)` — выявляет вхождение строки `S2$` в строку `S1$`. Если `S2$` входит в `S1$`, функция возвращает номер позиции `N`, с которой начинается строка `S2$` внутри строки `S1$` (позиции в строке нумеруются слева направо, начиная с 1). Если строка `S2$` не содержится в `S1$`, функция возвращает **N = 0**. Так, в примере:

```
N = instr("Что такое крокодил", "крокодил")
```

в результате получится `N = 11`.

Функция `N1 = Length(S$)` — возвращает длину (число символов) `N1` в строке `S$`.

Функция `S2$ = Mid(S1$, n1, n)` — выделяет подстроку `S2$` из строки `S1$`, начиная с позиции `n1` длиной `n` символов. В примере `S2$ = mid("Информатика", 3, 5)` получится `S2$ = "форма"`.

Функция `S3$ = left(S1$, n2)` — выделяет из строки `S1$` подстроку `S3$`, содержащую `n2` символов с левого края, т. е. от начала строки: `left("собака" , 3) = "соб"`.

Функция `S4$ = right(S1$, n3)` — выделяет из строки `S1$` подстроку `S4$`, содержащую `n3` символов с правого края, т. е. от конца строки: `right("собака", 3) = "ака"`.

Функция `Lower(S$)` — преобразует латинские буквы строки `S$` в символы нижнего регистра, т. е. в строчные буквы: `Lower("Bear") = "bear"`. Русские буквы и специальные символы (знаки препинания, знаки математических операций) остаются без изменения.

Функция `Upper(S$)` — преобразует латинские буквы строки `S$` в символы верхнего регистра, т. е. в прописные буквы: `Upper("Leningrad") = "LENINGRAD"`. Русские буквы и специальные символы остаются без изменения.

Оператор `Say S$` — произносит текстовую строку `S$` на английском языке. Например, команда `Say "Do you speak English?"`

произнесет приведенную фразу. При этом используется системный преобразователь текста в речь (TTS, Text To Speech Engine). В Windows по умолчанию для этого служит звуковой интерфейс для прикладных программ SAPI (Sound Application Programming Interface). В Linux для той же цели должна быть установлена одна из библиотек FLite или eSpeak. Русские строки не произносятся, русский текст будет воспроизводиться только в виде строк, записанных латинскими буквами.

Строки комментариев включаются в программу операторами `Rem` или `#`. Интерпретатор BASIC-256 при просмотре программы строки комментариев пропускает без обработки.

Оператор `Font fn, p, w` — устанавливает характеристики шрифта для команд отображения текста в окнах текста (`print`) или графики (`text`). Здесь: `fn` (font name) — имя шрифта в операционной системе, например, Times New Roman, Arial; `p` (point) — размер шрифта в пунктах (1 пункт = 1/72 дюйма, 1|72"); `w` (weight) — интенсивность цвета или "жирность" символов от 1 до 100: Light = 25 — слабый (бледный), Normal = 50 — нормальный, Bold = 75 — полужирный.

2.3. Управляющие операторы BASIC-256

В BASIC-256, как и в большинстве других сред программирования, строки программы обрабатываются и исполняются в порядке их записи. Управляющие операторы могут изменять порядок обработки и исполнения строк программы. Управляющие операторы условно можно разделить на следующие группы: операторы формирования условий, операторы анализа условий, операторы циклов, операторы для работы с подпрограммами.

2.3.1. Операторы формирования условий

Специального представления для логического типа данных вида условий, которые могут иметь два значения "выполнено" и "не выполнено", в BASIC-256, как и в большинстве других версий

BASIC, не предусмотрено. Условие отображается числом: выполненному условию соответствует 1, невыполненному — 0. Это легко проверяется: команда `print (7 > 5)` выводит в текстовое окно **1** (выполненное условие); команда `print (3 > 7)` выводит **0** (невыполненное условие).

Тот факт, что условие отображается числом, во многих случаях упрощает формирование сложных условий — условие можно использовать в арифметических выражениях, в частности, в операциях сложения и умножения. Обычно во всех средах программирования используются 6 простых условий и набор простых логических операций для формирования более сложных условий. Простые условия во всех средах программирования одинаковы: > (больше); >= (не меньше); < (меньше); <= (не больше); = (равно); <> (не равно).

Для формирования сложных условий в BASIC-256 предусмотрены 4 стандартных логических операции: AND (логическое И); OR (логическое ИЛИ); NOT (логическое НЕ); XOR (логическое ИСКЛЮЧАЮЩЕЕ ИЛИ). Переключательные (булевы) функции, реализующие перечисленные операции, приведены в табл. 2.3.

Таблица 2.3. Формирование сложных условий с помощью логических операций

Набор	X1	X2	Z1 = (X1 OR X2)	Z2 = (X1 AND X2)	Z3 = (NOT X1)	Z4 = X1 XOR X2
0	0	0	0	0	1	0
1	0	1	1	0	1	1
2	1	0	1	0	0	1
3	1	1	1	1	0	0

Возможность оперировать с условиями в числовой форме используется во многих математических программах (Excel, MATLAB) и позволяет в ряде случаев упростить выражения для сложных функций и формирование сложных условий. Так операцию AND вполне можно заменить операцией арифметического умножения: X1 AND X2 эквивалентно X1*X2, если X1, X2 = {0, 1}.

Рассмотрим пример формирования аналитического выражения сложной функции с использованием условий. Пусть некоторая функция F(x) задана следующим описанием: $F(x) = x^2$ при $-5 <= x <= 5$; $F(x) = 25$ при $x > 5$ или при $x < 5$. С использованием условий как чисел функцию F можно записать одним аналитическим выражением:

```
F = x^2*(x<=5)*(x>=-5) + 25*(x>5) + 25*(x<-5).
```
(2.6)

Легко проверить, что при любом действительном x функция F будет вычисляться правильно по выражению (2.6).

2.3.2. Операторы анализа условий (операторы ветвления)

Условия анализируются комбинацией операторов `If/Then` (Если/То). Если отобразить вычислительный процесс графически, например, с помощью блок-схемы, в нем можно выделить основную последовательность операторов программы и ответвления, возникающие при анализе условий. С помощью операторов `If/Then` можно организовать одну или две дополнительные ветви вычислительного процесса (ветвления на одно и на два дополнительных направления).

Ветвление на одно направление можно организовать двумя способами, в зависимости от того, один или несколько операторов (строк программы) нужно выполнить в дополнительной ветви процесса. В случае если в дополнительной ветви будет только один оператор, оператор ветвления запишется одной строкой:

`If` *условие* `then` *оператор*. (2.7)

Если в дополнительной ветви необходимо выполнить несколько операторов (строк программы), оператор ветвления запишется иначе.

В обоих случаях по оператору `if` анализируется условие. Если оно выполняется, будут исполняться один или несколько операторов дополнительной ветви после слова `then`. Если условие не выполняется, оператор `if/then` пропускается, и будет выполняться следующий по порядку оператор (строка) основной программы.

Ветвление на два дополнительных направления организуется следующим образом:

```
If условие then
операторы1
else
операторы2
End If
```
(2.8)

По оператору `if` анализируется условие. При его выполнении исполняются операторы первой дополнительной ветви *операторы1* после слова `then`. При невыполнении условия исполняются операторы второй дополнительной ветви *операторы2* после слова `else` (иначе). По завершении исполнения любой из дополнительных ветвей продолжится исполнение основной программы — следующего после оператора `if/then/else` оператора.

Во всех случаях, когда оператор ветвления записывается в несколько строк, он заканчивается ключевыми словами `End If`. Классический школьный пример применения операторов ветвления — вычисление действительных корней x1, x2 квадратного уравнения вида $ax^2 + bx + c = 0$. Коэффициенты a, b, c вводятся с клавиатуры (листинг 2.2).

Листинг 2.2

```
# Решение квадратного уравнения
# Ввод коэффициентов a, b, c
Input "a = ", a
Input "b = ", b
Input "c = ", c
# Вычисление дискриминанта d
d = b^2 - 4*a*c
d1 = (abs(d))^(1/2)
# Вычисление корней и вывод результатов
if d >= 0 then
x1 = (-b + d1)/(2*a)
x2 = (-b -d1)/(2*a)
print "x1 = " + x1 +" " + "x2 = " + x2
```

```
else
print "Уравнение не имеет действительных корней"
end if
```

2.3.3. Операторы управления циклами

Цикл — повторяющееся несколько раз исполнение некоторой последовательности строк программы. Число повторений либо указывается явно в операторе цикла со счетчиком, либо зависит от выполнения некоторых условий в операторах цикла с условиями.

Оператор цикла имеет стандартную структуру и содержит следующие компоненты: заголовок цикла, тело цикла — последовательность повторяющихся в цикле операторов (строк) программы, оператор завершения цикла. Условие завершения цикла проверяется или в заголовке цикла, или в операторе завершения цикла.

Цикл *For/Next*

Цикл `For/Next` — типичный цикл со счетчиком. В общем виде такой цикл записывается следующим образом:

```
For x = expr1 to expr2 [step expr3]
Операторы1                                    (2.9)
Next x
```

В первой строке — заголовке цикла — указываются:

- переменная цикла (здесь `x`), ее начальное и конечное значения задаются выражениями `expr1` и `expr2`;
- необязательный параметр `step`, определяемый выражением `expr3`, задает шаг изменения переменной цикла (по умолчанию `expr3 = 1`). При положительном шаге `expr3 > 0` начальное значение параметра цикла меньше конечного `expr2 > expr1`. При отрицательном шаге `expr3 < 0` должно быть `expr2 < expr2`.

Оператор завершения цикла `next x` после каждого исполнения тела цикла выполняет приращения счетчика `x = x + expr3` и проверяет условие завершения цикла `x > expr2` при `expr3 > 0` или

`x < expr2` при `expr3 < 0`. Если условие завершения цикла выполняется, происходит выход из цикла, иначе повторяется тело цикла.

Чаще всего на практике для переменной цикла x, для expr1, expr2, expr3 используются целочисленные значения, но в BASIC-256 это не обязательно. В качестве примера рассмотрим фрагмент программы для вычисления таблицы функции sin x для x от 0° до 90° с шагом 10° (листинг 2.3). В примере учтено преобразование аргумента `x` из градусной меры в радианную.

Листинг 2.3

```
# Вычисление таблицы функции y = sin(x)
For x = 0 to 90 step 10
y = sin(x*pi/180)
print "x = " + x + " " + " y = " + y
Next x
```

Цикл *While/End While* (цикл ПОКА)

В BASIC-256 этот цикл реализован только как цикл с предусловием:

```
While условие
Операторы                                    (2.10)
End While
```

Условие *незавершения* цикла записано в заголовке цикла и проверяется до исполнения тела цикла. Если условие выполнено, исполняется тело цикла — одна или несколько строк программы, обозначенные в (2.10) как *Операторы*. Цикл повторяется, пока выполнено условие. В частном случае возможно, что тело цикла выполнится 0 раз. Пример, приведенный в листинге 2.3 с использованием цикла `While/End While`, можно записать следующим образом (листинг 2.4).

Листинг 2.4

```
x = 0
While x < 91
y = sin(x*pi/180)
```

```
print "x = " + x + " " + "y = " + y
x = x + 10
End While
```

Цикл *Do/Until* (цикл ДО)

Цикл `Do/Until` в BASIC-256 реализован только как цикл с постусловием. Условие *завершения* цикла пишется в операторе завершения цикла и проверяется после исполнения тела цикла. В этом случае всегда тело цикла выполняется хотя бы один раз. В общем виде цикл `Do/Until` записывается следующим образом:

```
Do
операторы                                    (2.11)
Until условие
```

Тело цикла (группа строк *операторы*) исполняется до выполнения условия завершения цикла.

2.3.4. Операторы управления переходами

BASIC-256 допускает использование в программах безусловных и условных переходов. Для указания места перехода или вызова подпрограммы используются *метки*. Метка указывает начало некоторого фрагмента программы, занимает отдельную строку, формируется по правилам формирования имен (любая комбинация символов, начинающаяся с буквы). Метка в строке обязательно оканчивается двоеточием, но при обращении к метке, например, командами `goto`, `gosub`, двоеточие не пишется. Меткам лучше присваивать содержательные имена, особенно если они задают начала подпрограмм. Если это затруднительно, рекомендуется использовать в качестве меток комбинации символов `L1`, `L2`, ... , `LN`, где буква `L` — от английского Label, метка; целое число `1`, `2`, ... , `N` — порядковый номер метки в данной программе.

Оператор `goto LN` обеспечивает переход к строке с меткой `LN:`. Переход называется *безусловным*, если оператор `goto LN` не используется в составе одного из операторов ветвления. В противном случае, если исполнение или неисполнение перехода к строке с меткой связано с выполнением или невыполнением условия,

переход называется *условным*. В современных технологиях программирования принято писать структурированные, т. е. псевдолинейные программы. В этом случае условные и безусловные переходы рекомендуется заменять безусловными и условными вызовами подпрограмм. В качестве классических примеров использования операторов переходов рассмотрим фрагменты программ для чтения кода нажатой клавиши и для ожидания нажатия некоторой клавиши для продолжения программы.

```
L1: x = key
if x = 0 then goto L1
print x
```
(2.12)

В примере (2.12) оператор `key` читает код ASCII нажатой клавиши, если он существует, и присваивает его переменной `x`. Если ни одна клавиша не нажималась, `x = 0`, и выполняется переход на метку `L1` — тем самым организуется цикл ожидания нажатия клавиши. Если какая-нибудь клавиша нажималась, в текстовом окне печатается ее код. Для отображаемых клавиш печатается ASCII код от 0 до 255. Для управляющих клавиш печатаются большие коды клавиш N(<Клавиша>) — например: N(<BS>) = 16777219; N(<Enter>) = 16777220; N(<F5>) = 16777268; N(<Up>) = 16777235 (здесь <Up> — клавиша "стрелка вверх").

В примерах (2.13) и (2.14) для перехода к следующему кадру (окну) программы необходимо нажать клавишу ПРОБЕЛ в (2.13) или любую клавишу в (2.14). В обоих примерах с помощью условного перехода к метке организуется цикл ожидания нажатия нужной клавиши.

```
Print "Для продолжения нажмите ПРОБЕЛ"
L2:
y$ = chr(key)
if y$ = <> " " then goto L2
```
(2.13)

```
print "Для продолжения нажмите любую клавишу"
L3:
y1$ = chr(key)
if y$ =’’ then goto L3
```
(2.14)

2.3.5. Операторы для обслуживания подпрограмм

Подпрограммы (процедуры) — последовательности операторов (строк) программы, которые можно вызывать одним оператором из любого места основной программы. Правильно построенные структурированные программы обычно содержат большое число подпрограмм. При организации подпрограмм в любой среде программирования необходимо решить три вопроса. Первый из них — расположение подпрограммы по отношению к основной программе: в специальном разделе основной программы, как в Turbo Pascal, Delphi, Free Pascal; после основной программы (большинство версий BASIC, Fortran); перед основной программой; в любом месте вне основной программы (Visual Basic). Второй вопрос — оформление подпрограммы, организация вызова подпрограммы и возврата на продолжение основной программы после завершения подпрограммы. Третий вопрос — способ передачи параметров из основной программы подпрограмме и наоборот, т. е. способ присвоения начальных значений внутренним переменным подпрограммы из основной программы и способ передачи результатов работы подпрограммы переменным основной программы.

В BASIC-256, как в большинстве версий BASIC, подпрограммы пишутся после основной программы. Основная программа заканчивается оператором `End`, по нему прекращается ее исполнение. Первой строкой подпрограммы является метка, играющая роль имени подпрограммы при ее вызове. Подпрограмма вызывается отдельной строкой основной программы вида `Gosub` *`метка`*. Каждая подпрограмма заканчивается оператором `return`, обеспечивающим возврат на продолжение основной программы после завершения работы подпрограммы.

В BASIC-256 нет разделения переменных на локальные и глобальные, нет понятия области действия переменных и их описаний. Все переменные в BASIC-256 имеют одинаковый статус и доступны и в основной программе, и в подпрограммах. Поэтому при разработке программы можно организовать единое пространство переменных. Тем самым проблема передачи параметров

подпрограмме и проблема возвращаемых значений подпрограммой в основную программу снимаются. Допускаются вложенные обращения к подпрограммам, т. е. из одной подпрограммы можно вызывать другую. Число уровней вложения не оговаривается. Оформление подпрограмм можно пояснить следующим примером (листинг 2.5). Для наглядности основную программу от подпрограммы и подпрограммы друг от друга можно отделять пустыми строками.

Листинг 2.5

```
Cls
Input "Введите число x от 1 до 100, x = ", x
If x <= 50 then
Gosub L1
Else
Gosub L2
End If
Print "До свидания"
End

L1:
Print "Привет, Василий"
Return
L2:
Print "Привет, Саша"
return
```

В этом примере с клавиатуры вводится число `x`. Если `x <= 50`, вызывается подпрограмма `L1`, иначе вызывается подпрограмма `L2`. В первом случае в текстовое окно выводятся две строки:

Привет, Василий
До свидания

Во втором случае в окно текста выводятся строки:

Привет, Саша
До свидания

2.4. Операторы для обработки дат и времени

Данные типа дат и времени приходится обрабатывать в программах, выполняющих, например, финансовые и экономические расчеты. В BASIC-256 эти операторы позволяют отображать текущую дату и время по данным системного таймера: `day`, `month`, `year` для отображения текущей даты; `hour`, `minute`, `second` для отображения текущего времени.

Оператор `Year, year()` — возвращает год в формате 4-х цифр.

Оператор `Month, month()` — возвращает системный номер месяца.

> ***ВНИМАНИЕ!***
>
> Месяцы от января до декабря имеют системные номера от 0 до 11, т. е. системные номера месяцев на единицу меньше номеров месяцев, принятых для отображения дат.

Оператор `Day, day()` — возвращает текущую системную дату (число) от 1 до 31.

Текущую дату с системного таймера можно отобразить на экране в русском формате **число.месяц.год**, например, следующим способом:

```
Print "Текущая дата: " + day + "." +
+ (month + 1)+ "." + year.
```
(2.15)

Оператор `Hour, hour()` — возвращает системное время в часах в 24-часовом формате (от 0 до 23).

Оператор `Minute, minute()` — возвращает из системного времени минуты текущего часа от 0 до 59.

Оператор `Second, second()` — возвращает из системного времени число секунд текущей минуты от 0 до 59.

Отобразить на экране текущее время в принятом в России формате **часы.минуты.секунды** можно следующим образом:

```
Print "Текущее время – " + hour +
+ "." + "minute" + "." + second.
```
(2.16)

> ***ПРИМЕЧАНИЕ***
>
> Перечень операторов и функций BASIC-256 в сжатом виде приведен в *приложении 1*.

ГЛАВА 3

Графика в BASIC-256

В любой среде процедурно-ориентированного программирования, в том числе в BASIC-256, для работы с графикой имеется примерно следующий набор операторов: операторы для управления графическим экраном (очистка, установка размеров по горизонтали и вертикали); операторы для формирования некоторых стандартных элементов изображений (графических примитивов: точек, отрезков, окружностей, прямоугольников и т. п.); операторы управления цветом и фоном; операторы отображения текстовых строк на графическом экране. По умолчанию выходное графическое окно (графический экран) имеет размеры 300×300 пикселов. Начало координат графического окна, точка с координатами (0, 0) — его левый верхний угол. Ось абсцисс X направлена горизонтально вправо; ось ординат Y направлена вертикально вниз.

3.1. Операторы управления графическим экраном

Оператор `Clg` (clear graphic screen) — очистка окна графики. Окно окрашивается в цвет фона.

Оператор `Color` *цвет*; `color r, g, b`; `color rgb` — устанавливает цвет точек и линий на графическом экране. Параметр *цвет* может быть задан тремя способами: английским названием цвета — та-

кие названия определены для 18 цветов (табл. 3.1), однобайтовыми кодами интенсивности красного `r` (red), зеленого `g` (green) и синего `b` (blue) цветов (0 <= `r`, `g`, `b` <= 255); `rgb`-кодом цвета, вычисляемым по формуле:

$$rgb = 2^{16}*r + 2^{8}*g + b \qquad (3.1)$$

При этом rgb-код цвета однозначно связан с составляющими интенсивности цвета r, g, b:

$$r = rgb\backslash 2^{16};\ rgb1 = rgb - 2^{16}*r;\ g = rgb1\backslash 2^{8};\ b = rgb1 - 2^{8}*g \qquad (3.2)$$

Таблица 3.1. Характеристики популярных цветов в BASIC-256

Названия цветов: русское, английское	**Интенсивности составляющих цвета r, g, b**
Черный, black	0, 0, 0
Белый, white	255, 255, 255
Красный, red	255, 0, 0
Темно-красный, darkred	128, 0, 0
Зеленый, green	0, 255, 0
Темно-зеленый, darkgreen	0, 128, 0
Синий, blue	0, 0, 255
Темно-синий, darkblue	0, 0, 128
Циан (морская волна), cyan	0, 255, 255
Темный циан, darkcyan	0, 128, 128
Пурпурный, purple	255, 0, 255
Темно-пурпурный, darkpurple	128, 0, 128
Желтый, yellow	255, 255, 0
Темно-желтый, darkyellow	128, 128 ,0
Оранжевый, orange	255, 102, 0
Темно-оранжевый, darkorange	176, 61 ,0
Серый, grey	164, 164 ,164
Темно-серый, darkgrey	128, 128 ,128

Например, все приведенные здесь операторы установки цвета:

```
color red; color 255, 0, 0; color 16711680
```
(3.3)

устанавливают красный цвет линий в окне графики.

По умолчанию же устанавливается `color black` или `color 0, 0, 0` — черный цвет линий.

Оператор `Rgb(r, g, b)` — вычисляет rgb-код цвета по интенсивности его составляющих `r` (red — красный), `g` (green — зеленый), `b` (blue — синий) в соответствии с соотношением (3.1); 0 <= r, g, b <= 255.

Оператор `Getcolor`, `getcolor()` — возвращает текущий rgb-код цвета, установленный последним оператором `color`.

Оператор `Graphheight`, `graphheight()` — возвращает текущую высоту окна графики (максимальное значение y).

Оператор `Graphwidth`, `graphwidth()` — возвращает текущую ширину окна графики (максимальное значение x).

Оператор `Graphsize xmax, ymax` — устанавливает новые текущие размеры окна графики по горизонтальной и вертикальной осям X (`xmax`) и Y (`ymax`) и перезапускает графический модуль BASIC-256.

> ***Примечание***
>
> В BASIC-256 предельные размеры графического окна, которые можно задать, `xmax` × `ymax` = 800×600 пикселов.

Оператор `Fastgraphics` — переключает видеокарту в ускоренный графический режим. Ускоренный режим сохраняется до остановки программы или до появления оператора обновления графического экрана `refresh`. Оператор `fastgraphics` обеспечивает значительное ускорение отображения сложных анимаций и устраняет мерцание.

Оператор `Refresh` — обновляет графический экран. При этом будут вновь прорисованы только те элементы, которые создавались после последнего оператора `refresh`. Оператор работает только при включенном ускоренном графическом режиме после оператора `fastgraphics`.

Оператор `Clickb`, `clickb()`(от click button) — если курсор мыши находился в пределах графического окна, оператор возвращает код `b` нажимавшейся последний раз кнопки трехкнопочной мыши: 0 — ни одна кнопка не нажималась; 1 — левая кнопка; 2 — правая кнопка; 4 — средняя кнопка. Если были нажаты одновременно две или три кнопки, возвращается сумма кодов. Код запоминается в специальном буфере (регистре) состояния мыши. Там же запоминаются координаты `x`, `y` курсора мыши в графическом окне в момент нажатия любой кнопки.

Операторы `Clickx`, `clickx()`, `clicky`, `clicky()` — возвращают, соответственно, координаты `x` и `y` курсора мыши в графическом окне в момент последнего нажатия любой кнопки.

Оператор `Clickclear`, `clickclear()` — сбрасывает в 0 буфер кнопок мыши (очищает `b`, `x`, `y`), установленный во время последнего нажатия любой кнопки.

Оператор `Mouseb`, `mouseb()` (от mouse button) — возвращает код нажатой в данный момент кнопки мыши при нахождении указателя мыши в графическом окне: 0 — не нажата ни одна клавиша; 1 — нажата левая кнопка; 2 — нажата правая кнопка; 4 — нажата средняя кнопка; сумму кодов — если одновременно нажаты две или три кнопки.

Операторы `Mousex`, `mousex()`, `mousey`, `mousey()` — читают из буфера мыши и запоминают, соответственно, координаты `x` и `y` указателя мыши в окне графики в момент нажатия любой кнопки мыши.

3.2. Операторы формирования графических примитивов в графическом окне

Оператор `Plot x, y` — ставит на графическом экране точку с координатами (`x`, `y`) текущего цвета, установленного последним оператором `color`.

Оператор `Pixel (x, y)` — возвращает rgb-код цвета точки (`x`, `y`) графического окна.

Оператор `Line x1, y1, x2, y2` — вычерчивает отрезок текущего цвета с координатами концов (`x1`, `y1`)—(`x2`, `y2`).

Оператор `Rect x0, y0, w, h` — вычерчивает закрашенный прямоугольник шириной `w` (width — по оси X), высотой `h` (height — по оси Y), левый верхний угол прямоугольника расположен в точке (`x0`, `y0`). Стороны прямоугольника параллельны осям координат.

Оператор `Poly array, poly {x1, y1, x2, y2, ... , xn, yn}` — вычерчивает по заданному массиву `array` координат `n` вершин (`x1`, `y1`), (`x2`, `y2`), ... , (`xn`, `yn`) закрашенный многоугольник текущего цвета. Например, последовательность операторов, приведенная в листинге 3.1, вычерчивает закрашенный треугольник красного цвета с координатами вершин (50, 50), (200, 50), (100, 250).

Листинг 3.1

```
Clg
Color red
Dim X(6)
X = {50, 50, 200, 50, 100, 250}
Poly X
```

Оператор `Stamp x0, y0, [scale], [angle], array; stamp x0, y0, [scale], [angle], {x1, y1, x2, y2, ..., xn, yn}` — копирует многоугольник, заданный координатами `n` вершин в массиве `array` или в массиве `{x1, y1, ... , xn, yn}`. Координаты многоугольника отсчитываются не от начала координат, а от точки окна графики с координатами (`x0`, `y0`). Размеры многоугольника можно изменить необязательным параметром `scale` (масштаб): `Scale` < 1 уменьшает размеры фигуры, `scale` > 1 увеличивает. Изображение многоугольника можно повернуть на угол `angle` в радианах (тоже необязательный параметр): фигура поворачивается по часовой стрелке при `angle` > 0 и против часовой стрелки при `angle` < 0. По умолчанию, т. е. при отсутствии в записи оператора `stamp` параметров `scale` и `angle`, принимается `scale` = 1, `angle` = 0. В этом случае оператор записывается, например, так:

```
stamp x0, y0, {x1, y1, ..., xn, yn}.
```

Фрагмент программы, приведенной в листинге 3.2, вычерчивает в графическом окне окрашенный в зеленый цвет прямоугольный треугольник с вершиной прямого угла в точке (50,50), длины сторон уменьшает в два раза (`scale` = 0.5), треугольник повернут относительно точки (50,50) по часовой стрелке на угол `pi/3` или на 60 градусов (`angle` = pi/3).

Листинг 3.2

```
Clg
Color green
stamp 50,50,0.5, pi/3, {0,0,0,100,100,100}
```

Оператор `Circle x, y, r` — вычерчивает текущим цветом, установленным по умолчанию или последним оператором `color`, закрашенную окружность радиуса `r` с центром (`x`, `y`). В примере (листинг 3.3) вычерчивается закрашенная окружность красного цвета радиуса `r` = 50 пикселов с центром в точке (100, 100).

Листинг 3.3

```
Color red
Circle 100, 100, 50
```

Оператор `Text x, y, S$` — отображает текстовую строку `S$` в окне графики, начиная с точки с координатами (`x`, `y`). Цвет текста установлен последним оператором `color`. Шрифт установлен последним оператором `font`.

ГЛАВА 4

Операторы для работы с файлами

В BASIC-256 предусмотрен следующий набор операторов для работы с текстовыми файлами: `Close`, `Eof`, `Open`, `Read`, `Readline`, `Reset`, `Write`, `Writeline`, `Exists`, `Size`, `Seek`. Операторы `WAVplay`, `WAVstop` обеспечивают управление асинхронным воспроизведением звуковых файлов типа wav в фоновом режиме.

Оператор `Close` — закрывает открытый файл. Если никакой файл не открыт, никаких действий не выполняет.

Оператор `Eof, eof()` — возвращает двоичный флаг (признак, имеющий значения `true/false` или `1/0`). Он сигнализирует о том, что прочитан символ конца файла `End Of File` (EOF). Чаще всего используется в операторах анализа условий или в циклах с условиями, например, следующего вида: `if not eof then … end if`; `while not eof … end while`; `do … until eof`.

Оператор `Open file$` — открывает для чтения и записи файл, полное обозначение которого задано строковой переменной `file$`. Полное обозначение или спецификация файла — это его обозначение в командных строках операционной системы, включающее путь и собственно обозначение файла. Одновременно может быть открыт только один файл. Если файл не существует на указанном в переменной `file$` пути, оператор `open` создает пустой файл с указанным в переменной `file$` именем и расширением.

Оператор `Exists(path$)` — возвращает двоичный флаг (`true/false` или `1/0`), который указывает, существует или не существует указанный в строковой переменной `path$` путь к файлу.

Оператор `Read, read()` — читает фрагмент очередной строки (token) из открытого в данный момент файла. Ограничителями фрагмента строки могут быть следующие символы в различных комбинациях: пробелы, символы табуляции (tab), символы конца строки (перевод строки, возврат каретки). Оператор `read` используется в конструкциях вида `x = read, y$ = read` в зависимости от типа читаемых из строки данных.

Оператор `Readline, readline()` — читает из открытого в данный момент файла очередную строку целиком. Используется в конструкции вида `x$ = readline`.

> ***Примечание***
>
> В поставленных в школы и в имеющихся на сайте разработчика версиях BASIC-256 оператор `Read` не работает. Рекомендуется использовать оператор `Readline`.

Оператор `Reset` — очищает открытый в настоящее время файл. Все данные, хранившиеся ранее в файле, теряются. Указатель помещается в начало первой строки файла.

Оператор `Write S$` — записывает строку `S$` в конец открытого файла, в текущую строку.

Оператор `Writeline S$` — добавляет строку `S$` в конец файла с новой строки.

Оператор `Seek location` — перемещает указатель, который указывает место чтения или записи в открытом файле, на заданное в переменной `location` число байтов от начала файла.

Оператор `Size, size()` — возвращает размер открытого в данный момент файла в байтах.

Оператор `WAVplay file$` — проигрывает асинхронно в фоновом режиме звуковой файл типа `WAV`, спецификация которого задана строковой переменной `file$`.

Оператор `WAVstop` — прекращает фоновое воспроизведение звукового файла, ранее запущенное оператором `wavplay`.

ГЛАВА 5

Порядок разработки и отладки программ в среде BASIC-256

Порядок разработки программ школьного уровня в среде BASIC-256 практически не отличается от порядка разработки несложных программ в других средах процедурно-ориентированного программирования, используемых в школьных курсах информатики и ИКТ (Turbo Pascal 7.0, QBasic 4.5 и др.) для иллюстрации разделов "Основы алгоритмизации" и "Элементы программирования". Основные особенности этих программ: очень простой интерфейс пользователя, не требующий специального проектирования; большинство демонстрационных примеров, иллюстрирующих стандартные приемы программирования типовых школьных задач, однокадровые, требующие всего одного рабочего экрана; минимальный объем программ, позволяющий рассмотреть на уроке пример, ввести, отладить и продемонстрировать его работу за допустимые по санитарным нормам 15–30 минут работы с компьютером для школьников 8–9 классов. Обычный порядок разработки школьных программных проектов, включая индивидуальные проектные задания, можно записать в следующем виде.

1. Анализ содержания задачи. Выявление базового набора переменных, их типов и содержательное описание.
2. Разработка и запись алгоритма в одной из формализованных форм. Наиболее популярные у учителей информатики формы

записи алгоритмов — содержательная форма и блок-схема. В процессе подробной разработки алгоритма может выявиться необходимость добавления дополнительных вспомогательных переменных, которые явно не следуют из содержания задачи, но необходимы для ее рационального решения. В алгоритме надо предусмотреть ввод данных и необходимые для этого диалоги с пользователем, вывод результатов работы программы и дополнительный вывод для контроля правильности исполнения отдельных этапов программы. После отладки программы эти дополнительные строки можно будет удалить.

3. Кодирование программы в среде программирования, т. е. формальная запись алгоритма через последовательность строк по правилам среды программирования.

4. Подготовка контрольных наборов данных и отладка программы.

5. Сохранение проекта. Проект должен быть оформлен в отдельной папке. В состав проекта необходимо включить файл программы (файлов программы может быть несколько для разных вариантов программы); текстовые файлы контрольных наборов данных для проверки правильности работы программы; текстовые файлы результатов работы программы на контрольных наборах данных.

Отладка программы, если это необходимо, осуществляется в следующем порядке.

1. После выполнения этапа кодирования путем контрольных пусков программы по сообщениям интерпретатора устраняются все синтаксические ошибки. Контрольные пуски выполняются командами **Старт | Старт** или кнопкой **Старт**. Если интерпретатор при просмотре программы обнаруживает ошибку, он останавливает процесс просмотра и выдает сообщение об ошибке с указанием строки текста программы и номера позиции, с которой начинается ошибочная часть в записи строки. Строки программы в окне редактора нумеруются сверху вниз так: 1, 2, 3, ... Аналогично нумеруются позиции в строке: слева направо: 1, 2, 3, ...

Рассмотрим следующий фрагмент программы:

```
cls
x = 10: y = 5
print x mod y
```
(5.1)

При его просмотре интерпретатор выдаст следующее сообщение об ошибке на русском языке: **Ошибка синтаксиса на строке 3 в колонке 9**. В строке 3 в позиции (колонке) 9 записан оператор (функция) `mod`. В BASIC-256 такой оператор недопустим — для вывода на экран остатка от деления целых чисел `x` и `y` вместо `mod` нужно записать символ `%`. Так что строку 3 примера (5.1) нужно переписать следующим образом: `print x % y`. Теперь при повторной интерпретации исправленного фрагмента сообщения об ошибке не будет. Ошибки выявляются и исправляются последовательно по мере просмотра всех строк программы.

2. После выявления и исправления всех синтаксических ошибок начинается собственно отладка программы, т. е. исправление содержательных или семантических ошибок (ошибок в алгоритме или в данных). Их можно выявить только с помощью контрольных наборов данных с заранее просчитанными результатами.
3. Если результаты на контрольных наборах данных не совпадают с полученными результатами, необходимо выявить, на каком этапе выполнения программы возникает ошибка. Это можно сделать несколькими способами. Во всех вариантах необходимо выбрать переменные для наблюдения, значения которых могут указать на возможную ошибку.

Вариант 1

Предусмотрим отладочный вывод переменных наблюдения на текстовый экран. После отладки программы строки отладочного вывода можно удалить или превратить в комментарии, поставив перед каждой из них символ `#` или оператор `Rem`. Рассмотрим пример.

```
# Primer_5_2
cls
x = 5: y = 10
for i = 1 to 5
x = x + 2: y = y - 1
next i
print "x = " + x + " y = " + y
```
(5.2)

Пусть вы предполагаете, что в теле цикла во фрагменте (5.2) есть ошибка в алгоритме или в данных, и что-то работает неправильно. Естественно, в качестве переменных наблюдения выбрать изменяющиеся в цикле переменные `x` и `y`. Для проверки правильности вычислений тела цикла можно в конце тела цикла добавить строку вывода текущих значений `x` и `y`. Фрагмент при этом примет следующий вид:

```
# Primer_5_3
cls
x = 5: y = 10
for i = 1 to 5
x = x + 2: y = y - 1
print "x = " + x + " y = " + y
next i
print "x = " + x + " y = " + y
```
(5.3)

После исполнения фрагмента (5.3) на текстовый экран будет выведен следующий результат:

x = 7 y = 9

x = 9 y = 8

x = 11 y = 7

x = 13 y = 6

x = 15 y = 5

x = 15 y = 5

Первые 5 строк выведенных результатов показывают результаты вычислений в теле цикла. По ним можно судить о правильности вычислений и правильности данного фрагмента алгоритма. Если программа зависнет, можно для выхода из нее воспользоваться

кнопкой **Стоп** панели инструментов или командой меню **Старт | Стоп**.

Вариант 2

При сложном и непрозрачном алгоритме вычислений можно использовать режим отладки с пошаговым (по одной строке) на каждое нажатие кнопки или на каждую подачу команды исполнением. Для этого перед началом исполнения программы включается режим отладки одной из следующих команд: командой меню **Старт | Отладка** или кнопкой **Отладка** панели инструментов. В режиме отладки каждый шаг (оператор) программы исполняется по нажатию кнопки **Шаг** панели инструментов или подачей команды меню **Старт | Шаг**. В окне редактора в режиме отладки исполняемая в данный момент строка программы выделяется более темным цветом. Выйти из режима отладки можно по нажатию кнопки **Стоп** или по команде меню **Старт | Стоп**.

Вариант 3

Можно вместо организации отладочного вывода переменных наблюдения в окне (на экране) текстового вывода наблюдать изменение всех переменных задачи (только в режиме отладки) в специальном окне просмотра переменных. Последнее вызывается на экран и убирается с экрана командой меню **Просмотр | Окно просмотра переменных**. При пошаговом исполнении программы в окне просмотра переменных отражаются все изменения значений переменных программы. Это позволяет выявить ошибку или неточность в алгоритме или в самой программе.

Вариант 4

В BASIC-256 не предусмотрена возможность задания остановов в контрольных точках программы для ее отладки. Контрольные точки выбираются после строк, завершающих основные этапы алгоритма задачи, например, после завершения циклических фрагментов и после завершения других важных фрагментов про-

граммы. В каждой контрольной точке можно вывести на текстовый экран переменные слежения, по значениям которых удается судить о правильности выполнения данного этапа алгоритма. Большую программу следует отлаживать по отдельным этапам.

Временные контрольные точки можно создать искусственно следующим образом — в процессе отладки каждого этапа после строки, завершающей отлаживаемый этап, добавить строки контрольного вывода переменных слежения и поставить оператор `End`. По окончании отладки этапа, т. е. после внесения всех необходимых изменений и проверки правильности работы фрагмента, реализующего данный этап, все дополнительные строки программы, добавленные для отладки этапа, включая строку с оператором `End`, следует удалить или превратить в комментарии. После чего добавить строки отладочного вывода и оператор `End` в конец следующего отлаживаемого фрагмента программы и продолжить процесс отладки. И продолжать так до завершения отладки всей программы.

Вместо добавления временной строки с оператором `End` в конец каждого отлаживаемого этапа можно поставить такую строку с оператором `End`, но помеченную меткой, например, `Mend: End` в конец всей программы. В этом случае в конце каждого отлаживаемого этапа можно будет добавлять временную строку `goto Mend`. По завершении процесса отладки очередного этапа эту строку следует удалять или превращать в комментарий.

Отлаженную программу необходимо сохранить в заранее подготовленной папке проекта. Сделать это можно стандартной для большинства операционных систем, включая Windows и Linux, командой вида **Файл | Сохранить как**, указав путь и имя файла. В диалоговом окне команды **Файл | Сохранить как** путь сохранения указывается в поле **Папка**. В Windows по умолчанию сохранение предполагается в папке **C:\Program files\BASIC256**. При необходимости удаляется путь по умолчанию и вводится нужный путь. Имя сохраняемого файла указывается в поле **Имя файла**. Тип файла указывается в поле **Тип файла**. По умолчанию в Windows файлы BASIC-256 сохраняются в виде текста про-

граммы с расширением kbs, например, Primer1.kbs. Другой вариант сохранения в виде скомпилированного байт-кода (в виде внутреннего кода BASIC-256, в котором каждый оператор кодируется специальным байтом) осуществляется командой **Старт | Сохранить байт-код** с указанием пути и имени файла. В этом случае файл сохраняется с расширением kbc — в нашем примере Primer1.kbc. Файлы формата kbc исполняются быстрее, чем файлы формата kbs.

ГЛАВА 6

Примеры программ для иллюстрации типовых школьных задач

6.1. Вывод в окно текста

Автор при изучении любой среды программирования в качестве первого знакомства школьников с примером простой программы использовал задачу вывода на экран небольшого фрагмента текста. Важно, чтобы учащиеся не вводили готовый текст, а придумывали его сами. Примерами таких текстов могут быть: краткая автобиография учащегося, которая может ему понадобиться при поступлении в профессиональное учебное заведение (ПТУ, колледж, вуз), или, например, заявление о приеме в спортивную секцию или в учебное заведение. Допускается использовать и другие полезные примеры текстовых документов, которые учащиеся могут придумать сами. Пусть это будет текст в 5–6 предложений. Примерный шаблон автобиографии учащимся можно продиктовать.

Задание может выглядеть следующим образом. Написать программу вывода на текстовый экран текста: "Меня зовут Николай Васильевич Антонов. Я родился 23 ноября 1995 г. в г. Рязани. В настоящее время я — учащийся 9-го класса средней школы № 47 г. Рязани. Проживаю по адресу: ул. Великанова, д. 14, кв. 47, г. Рязань, 390044. Домашний телефон 34-57-32. Мобильный телефон 8-910-567-94-47. Мать работает в школе № 47 учителем

биологии. Отец работает на Приборном заводе начальником цеха."

Листинг 6.1

```
# Pimer_6_1
cls
print "Меня зовут Николай Васильевич Антонов."
Print "Я родился 23 ноября 1995 г. в г. Рязани."
Print "В настоящее время я – учащийся 9-го класса средней
школы № 47 г. Рязани."
Print "Проживаю по адресу: ул. Великанова, д. 14, кв. 47,
г. Рязань, 390044."
Print "Домашний телефон 34-57-32."
Print "Мобильный телефон 8-910-567-94-47."
Print "Мать работает в школе № 47 г. Рязани."
Print "Отец работает на Приборном заводе начальником цеха."
```

Если учащиеся не будут успевать набрать этот текст (листинг 6.1) за отведенное на уроке время, его можно сократить на 2–3 предложения. Естественно, каждый учащийся пишет программу с выводом текста своей автобиографии.

Учащиеся, выполняя этот проект, приобретают следующие знания, умения и навыки: они получают представление о правилах ввода текста программы, знакомятся с интерфейсом программы, в первый раз видят, что набранная ими программа при запуске выводит текст их автобиографии на экран. Это производит впечатление доступности освоения программирования как вида деятельности.

6.2. Формирование подвижных объектов

Задачи формирования подвижных объектов всегда вызывают у учащихся неизменный интерес. В следующих двух примерах подвижным объектом будет символьная строка на текстовом экране.

До рассмотрения этих задач учащиеся должны изучить:

- ❒ цикл `For ... Next`;
- ❒ функции, определяющие текущие ширину и высоту графического экрана: `graphwidth`, `graphheight`;
- ❒ получить представление о типах шрифтов, ширине и высоте символов при выводе текста на экран.

Перед вводом текста программы следует подать команду установки размера шрифта для вывода текста на текстовый экран: **Просмотр | Размер шрифта | Мелкий**. При этом условии ширина видимой строки текста на текстовом экране составит 39 пробелов + три символа **xxx**, которые и будут двигаться по экрану. Значение ширины строки в символах определено экспериментально. Для других символов она будет другой. Фиксирована лишь ширина текстового и графического экранов в пикселах: по умолчанию `graphwidth` = 300 пикселов; максимальное значение `graphwidth` = 800 пикселов.

Пример 6.2.1

Рассмотрим вначале более простую задачу — движение комбинации символов **xxx** в первой строке экрана. По умолчанию при каждом запуске программы текстовый экран очищается, текстовый курсор ставится в первую позицию первой строки.

Общая идея формирования подвижного изображения любого объекта заключается в следующем:

1. На экран выводится изображение объекта в некотором текущем положении.
2. Далее дается временная задержка на T секунд.
3. Затем объект стирается и выводится на экран уже в новом положении.

От величины T будет зависеть скорость движения объекта по экрану V — чем больше T, тем меньше скорость V. В программе T будет задаваться оператором `Pause T` в секундах. В примерах принято T = 0.2 сек. Скорость V при этом получается в числе символьных позиций в строке в секунду.

В BASIC-256 отсутствует специальный оператор позиционирования текста на текстовом экране. Для реализации этой функции приходится применять искусственный прием. Вначале выводимая строка имеет вид `x$ = "xxx"`. Перед каждым следующим выводом в цикле к строке `x$` слева добавляется пробел. За счет этого в выводимой строке `x$` подстрока `"xxx"` смещается каждый раз на одну позицию вправо относительно начала строки. Алгоритм решения задачи подробно пояснен в комментариях к программе (листинг 6.2.1).

Листинг 6.2.1

```
# Primer_6_2_1
# Присвоение переменной x$ (выводимой строке) начального
значения
x$ = "xxx"
# Цикл, задающий движение в первой строке экрана
for k = 1 to 39
cls
# Вывод строки, отображающей подстроку "xxx" строки x$
в текущем положении
print x$
# Задержка на 0.2 сек
pause 0.2
# Смещение подстроки "xxx" в строке x$ на один пробел вправо
x$ = " " + x$
next k
```

Пример 6.2.2

Теперь можно рассмотреть задачу целиком с указанием конкретного номера строки N, в которой будет происходить движение комбинации символов **xxx**. Номер строки N будет интерактивно вводиться по запросу программы. Чтобы попасть из строки 1 в начало строки N, следует N – 1 раз подать команду `print` с пустым списком вывода. Это необходимо делать отдельным внутренним циклом перед каждым выводом выходной строки `x$`. Подробнее алгоритм поясняется в комментариях к программе (листинг 6.2.2).

Листинг 6.2.2

```
# Программа Primer_6_2_2
# Ввод номера строки N
Input "Введите номер строки N = ", N
x$ = "xxx"
# Основной цикл, задающий движение в N-й строке экрана
for k = 1 to 39
cls
# Переход из строки 1 в строку N
for k1 = 1 to (N-1)
print
next k1
# Вывод строки, отображающей подстроку "xxx" строки x$ в
текущем положении
print x$
# Задержка на 0.2 сек
pause 0.2
# Смещение подстроки "xxx" в строке x$ на один пробел вправо
x$ = " " + x$
next k
```

При формировании индивидуальных заданий, выдаваемых учащимся для самостоятельной или зачетной работы, можно разнообразить задания, например, следующими способами.

- Задавать в качестве движущейся комбинации символов имя учащегося, символ его года, название его знака зодиака, год рождения. В качестве номера строки можно задавать номер месяца или число из даты рождения.

  Тогда вместо диалогового ввода номера строки в программе будет строка, например, вида `N = 10`, если учащийся родился в октябре, или `N = 17`, если учащийся родился 17-го числа.

- Можно изменять и ширину графического экрана в пикселах. При этом значения лучше брать из стандартного ряда: 360, 480, 640, 800.

  В программе для этого в качестве первой строки нужно записать оператор, задающий характеристики графического экрана: `Graphsize W, H`. Здесь `W` — ширина графического экрана,

`H` — его высота в пикселах. Так, строка `Graphsize 480, 360` установит графический экран W×H = 480×360 пикселов. При этом ширина текстового экрана тоже станет равной 480 пикселам, а длина текстовой строки увеличится примерно до 64 символов.

6.3. Вычисление таблицы функции

При рассмотрении этой темы ставятся следующие цели:

- ❒ показать учащимся практический полезный пример программы для решения вычислительной задачи;
- ❒ освоить правила применения цикла `For ... Next`;
- ❒ показать на практике на простом примере все этапы проектирования программы;
- ❒ показать на примере порядок включения комментариев в программу;
- ❒ показать пример оформления результатов работы программы.

Пример решения задачи для простой и хорошо известной учащимся функции y = sin x рассматривается на уроке. На одном из следующих занятий дается зачетная работа. Каждый учащийся получает индивидуальное задание в форме проекта. Задание содержит: математическую запись функции, интересующий интервал изменения переменной x, рекомендуемое число значений функции n (обычно n = 10 – 20). При выводе результатов требуется предусмотреть вывод на экран следующей информации: класс, фамилия и имя учащегося, номер задания, заголовок таблицы функции, заголовок для значений аргумента и значений функции, собственно таблицу значений.

Порядок выполнения зачетной работы

Учащийся прорабатывает проект дома и представляет в тетради для зачетных работ следующую информацию: задание, анализ задачи, описание алгоритма, текст программы. На специально отведенном для реализации проекта уроке учащийся отлаживает

программу и сохраняет проект в личной папке. В качестве имени файла он должен записать свою фамилию латинскими буквами с добавлением порядкового номера работы, например: Vasilyev1.kbs. На пользовательском диске создается папка Projects, в ней создаются папки классов, в папке каждого класса создаются личные папки учащихся. Все свои проекты каждый учащийся сохраняет в личной папке в течение всего учебного года. Номер задания каждого проекта обычно выбирается по номеру учащегося в списке в классном журнале.

Пример 6.3.1

Рассчитать и вывести на текстовый экран таблицу функции y = sin x в диапазоне изменения x от 0 до 90° через 10°. Фамилия учащегося и номер задания здесь выбираются условными за пределами списка в журнале. В данном примере мы выберем номер задания 40, фамилия учащегося — Васильев, класс — 9А.

Анализ задачи

Кроме переменной x других переменных не понадобится. Необходимо учесть необходимость преобразования аргумента x из градусов в радианы.

Алгоритм решения задачи

Алгоритм решения задачи можно записать следующим образом:

Начало

1. Подготовка экрана консоли
2. Вывод строки заголовка задания
3. Вывод строки заголовка таблицы
4. Вывод строки заголовка данных в форме: **x градусов**, 5 пробелов, **y = sin x**
5. Цикл по x от 0 до 90 с шагом 10 [
6. `y = sin(x*п/180)`
7. Вывод строки таблицы вида: **x**, 8 пробелов, **y**]

Конец

Примечание

Здесь и далее в записи алгоритма открывающая и закрывающая квадратные скобки обозначают начало и конец цикла. Роль оператора `Начало` в программе будет играть оператор комментария с названием программы. Специального оператора, задающего имя программы, в BASIC-256 не предусмотрено.

Листинг 6.3.1

```
# Primer_6_3_1
Cls
Print "Класс 9А,    Задание № 40,   Васильев Александр"
Print "Таблица функции y = sin x"
Print "x градусов" + "     " + y
For x = 0 to 90 step 10
y = sin(radians(x))
print x + "        " + y
next x
end
```

Примечание

Оператор `end`, играющий роль оператора `Конец` в алгоритме, в данном примере необязателен.

6.4. Поиск данных в массивах по заданным условиям

При изучении данной темы учащиеся:

- приобретают понятие о массивах и способах присвоения им значений;
- получают опыт в применении операторов ветвления и оформлении содержательных условий в виде математических зависимостей;
- осваивают навыки решения часто встречающихся в практике задач поиска информации.

По этой теме рассматриваются два примера задач на поиск информации в числовых массивах и в массивах строковых данных.

Затем учащимся дается зачетная работа на началах, изложенных в разд. 6.3.

Пример 6.4.1

Задан числовой массив из N чисел. Значение N и числа массива вводятся с клавиатуры. Найти максимальное и минимальное числа `xmax` и `xmin`.

Анализ задачи

Для решения задачи кроме переменных, введенных в задаче, понадобятся также следующие переменные: числовой массив x(N) с переменными x(0) — x(N–1)и i — номер текущего элемента массива в процессе его просмотра.

Алгоритм поиска максимального `xmax` и минимального `xmin` чисел построен следующим образом. Перед началом просмотра `xmax` и `xmin` присваивается в качестве текущих значений значение первого элемента массива x[0] (максимальное и минимальное значения в списке из одного элемента). Для каждого из следующих элементов массива x[1] — x[N–1] в цикле осуществляется стандартный шаг поиска максимального и минимального элементов — если на очередном шаге `i` для текущего значения `xmax` выполняется условие `xmax < x[i]`, то осуществляется замена `xmax = x[i]`. Аналогично для текущего значения `xmin` при выполнении условия `xmin > x[i]` производится замена `xmin = x[i]`.

Алгоритм решения задачи

Алгоритм решения задачи можно записать следующим образом:

Начало

1. Подготовка текстового экрана
2. Ввод числа элементов массива N
3. Описание массива x(N)
4. Цикл ввода элементов массива x[i] для i = 0, 1, ..., N–1
5. Присвоение начальных значений переменным `xmax` и `xmin`: `xmax = x[0]`; `xmin = x[0]`. Текущие значения максимального и

минимального элементов принимаются равными нулевому элементу массива `x[0]`

6. Цикл просмотра и поиска максимального и минимального элементов в массиве: для i от 1 до N–1 исполнять [
7. Если `xmax < x[i]`, то `xmax = x[i]`
8. Если `xmin > x[i]`, то `xmin = x[i]`]
9. Вывод результатов поиска `xmax` и `xmin`

Конец

Листинг 6.4.1

```
# Primer_6_4_1
Cls
Print "Класс 9А,    Задание № 40,   Васильев Александр"
# Ввод N
Input "Введите N = ", N
# Ввод элементов массива x[0] – x[N-1]
For i = 0 to N-1
Print "Для i = " + i
Input "Введите x[i] = ", x[i]
Next i
xmax = x[0]: xmin = x[0]
# Цикл просмотра и поиска
For i = 1 to N-1
If xmax < x[i] then xmax = x[i]
If xmin > x[i] then xmin = x[i]
Next i
# Вывод результатов поиска
Cls
Print "Результаты поиска:"
Print "xmax = " + xmax + "  xmin = " + xmin
end
```

При анализе программы рекомендуется обратить внимание учащихся на следующие блоки программы и приемы программирования:

- на расстановку комментариев в тексте программы;
- на организацию диалога при вводе N и элементов массива x[i];

- на оформление вывода результатов в окне вывода текста;
- на то, что оператор `end` в этом примере необязателен, потому что после последней строки с оператором `end` других операторов нет.

Следует также пояснить учащимся, что условия поиска в задачах с числовыми массивами могут быть разнообразными: вывести все целые числа из массива и подсчитать их количество, вывести все отрицательные числа из массива и подсчитать их количество, вывести все числа, делящиеся на 3, и подсчитать их количество.

Для некоторых примеров есть смысл показать формирование условий. Так, при выделении целых чисел — целая часть числа равна самому числу: `int(x[i]) = x[i]`. При выделении чисел, делящихся на 3, — остаток от деления числа на 3 равен нулю: `x[i] % 3 = 0`.

Пример 6.4.2

Задан массив из 20 слов на русском языке — например, в виде анонимного массива названий животных вида {медведь, слон, заяц, бык, тигр, барс, ягуар, куница, мышь, крыса, лев, кошка, рысь, горностай, соболь, белка, морж, тюлень, дельфин, волк}. Вывести на экран все слова, начинающиеся на заданную букву, и подсчитать их количество. Интересующая буква вводится с клавиатуры.

Анализ задачи

Для решения задачи понадобятся следующие переменные: строковый массив слов Y$(20) с переменными Y$[0] — Y$[19]; номер (индекс) элемента I (причем, I меняется от 0 до 19); для подсчета количества слов понадобится счетчик слов K (перед началом цикла просмотра элементов массива следует принять K = 0, при обнаружении очередного слова на заданную букву K следует увеличивать на 1). Заданную букву, введенную с клавиатуры, следует запомнить в переменной X$. Условие поиска (первая буква очередного слова совпадает с заданной) следует записать в виде: `mid(Y$[I], 1, 1) = X$`.

Алгоритм решения задачи

Алгоритм решения задачи можно записать следующим образом:

Начало

1. Подготовка экрана
2. Ввод буквы в переменную X$
3. Формирование массива Y$(20)
4. K = 0
5. Цикл просмотра и поиска: для I от 0 до 19 исполнять [
6. Если `mid(Y$[I], 1, 1) = X$`, то [
7. Вывод Y$[I] на текстовый экран
8. K = K + 1]]
9. Вывод числа выбранных слов K

Конец

Листинг 6.4.2

```
# Primer_6_4_2
Cls
Input "Введите первую букву слова X$ = ", X$
Dim Y$(20)
Y$ = {медведь, слон, заяц, бык, тигр, барс, ягуар, куница,
мышь, крыса, лев, кошка, рысь, горностай, соболь, белка, морж,
тюлень, дельфин, волк}
K = 0
Print "Отобраны следующие слова на букву " + X$
For I = 0 to 19
If mid(Y$[I], 1, 1) = X$ then
    Print Y$[I]
    K = K + 1
End if
Next I
Print
Print "Количество слов на выбранную букву " + X$ + "= " + K
End
```

При анализе программы следует обратить внимание учащихся на то, что условия поиска в массивах строковых данных можно сформулировать очень многими способами:

- поиск слов, у которых задана вторая, третья или последняя буква;
- поиск слов заданной длины;
- поиск самых коротких или самых длинных слов с указанием их длины и количества.

Рассмотрим некоторые примеры записи условий поиска.

При поиске слов с заданной последней буквой X$ в массиве Y$ для каждого слова Y$[I] сначала определяется его длина L, затем буква с номером L сравнивается с заданной X$:

```
L = length(Y$[I])
If  mid(Y$[I], L, 1) = X$ then print Y$[I]
```

При поиске слов максимальной длины в программе необходимы два цикла просмотра слов в массиве Y$. Первый цикл нужен для определения максимальной длины слова в массиве Lmax. Сначала выбирается начальное значение `Lmax = length(Y$[0])`. Далее в цикле для каждого I сравнивается длина L очередного слова Y$[I] с Lmax:

```
L = length(Y$[I])
If L > Lmax then Lmax = L
```

Во втором цикле из массива слов выбираются слова максимальной длины Lmax:

```
If length(Y$[I]) = Lmax then print Y$[I]
```

6.5. Задачи сортировки

Сортировка — упорядочение объектов или данных по некоторым признакам. Числовые данные можно сортировать в порядке возрастания или убывания. Строковые данные сортируются в алфавитном порядке или в порядке, обратном алфавитному. Строки латинского алфавита легко сортировать по ASCII-кодам букв. Прописные латинские буквы A–Z имеют коды от 65 до 90. Поря-

док букв в алфавите соответствует возрастанию кодов. Аналогичным свойством обладают строчные латинские буквы a–z с кодами от 97 до 122. В русском языке не для всех букв их порядок в алфавите соответствует возрастанию их кодов, поэтому для русских строк задача сортировки решается сложнее, и ее строгое решение выходит за рамки школьного курса.

Далее будут рассмотрены два примера решения задач сортировки. Известно много различных алгоритмов сортировки, отличающихся скоростью исполнения (необходимым числом операций сравнений и перестановок) и требуемым объемом оперативной памяти. В школьных курсах обычно ограничиваются простым алгоритмом сортировки методом "пузырька". Для числового массива X(N) при сортировке чисел в порядке возрастания он реализуется как последовательность циклов, каждый из которых в свою очередь представляет цикл из операций сравнений и перестановок. Элементарная операция сравнения и перестановки представляет собой следующую последовательность действий. На шаге I сравниваются элементы массива X[I] и X[I+1]. Если X[I] > X[I+1], X[I] и X[I+1] в массиве необходимо поменять местами (переставить): X[I] ← X[I+1]. То есть сравниваться друг с другом на шаге 1 будут X[0] и X[1], на шаге 2 — X[1] и X[2], на шаге (N–1), соответственно, X[N–2] и X[N–1]. В первом цикле сравнений и перестановок наибольшее из чисел "опустится" в последний элемент массива X[N–1] и займет свое место, которое оно должно занять по окончании сортировки. Во втором цикле сравнений и перестановок можно ограничиться (N–2) шагами. В результате второе по величине число попадет в X[N–2]. В худшем случае может понадобиться (N–2) таких циклов, если в исходном массиве числа были расположены в порядке убывания. Поэтому в худшем случае число операций сравнений и перестановок Q при сортировке методом "пузырька" составит Q = (N–1)*(N–2). Для сокращения числа циклов просмотра можно ввести специальный признак P — число перестановок элементов массива X, выполненное в текущем цикле просмотра. Если в очередном цикле просмотра окажется, что P <= 1, это значит, что выполнена последняя перестановка, сортировка завершена, и числа в массиве расположены в нужном порядке.

Пример 6.5.1

Задан числовой массив X из N элементов. Значение N и числа X[0] — X[N–1] вводятся с клавиатуры. Выполнить сортировку чисел в порядке возрастания.

Анализ задачи

Следует ввести в рассмотрение два числовых массива: X(N) — исходный массив и Y(N) — отсортированный массив. N — количество чисел в массивах. То есть Y[0] должен принять наименьшее из чисел, а в Y[N–1] должно попасть наибольшее из чисел. I — номер текущего шага сравнений и перестановок в очередном цикле просмотра. K — номер цикла просмотра (начинается с 1). В цикле просмотра K номер шага I изменяется от 1 до (N–K). Внешний цикл сортировки следует организовать как цикл с условием вида "Пока P > 0 исполнять...". Перед его началом переменной P можно присвоить любое положительное значение, чтобы основной цикл сортировки начал исполняться. В дальнейшем в каждом цикле просмотра массива Y$ переменная P вычисляется автоматически как число перестановок, выполненных в текущем цикле просмотра. Перед началом каждого цикла просмотра переменной P присваивается 0. Сортировка завершена, если в очередном цикле просмотра не выполнялось ни одной перестановки, т. е. после выхода из цикла просмотра остается P = 0. Z — буферная переменная для выполнения перестановок. Если Y[I–1] > Y[I], то выполняется перестановка, т. е. Y[I–1] и Y[I] меняются местами, и P увеличивается на 1: [Z = Y[I]; Y[I] = Y[I–1]; Y[I–1] = Z; P = P + 1]. Внешние квадратные скобки здесь задают начало и конец блока операторов, исполняемых как один оператор, в том числе и начало и конец цикла в операторах цикла. Это обозначение используется, например, в различных версиях Logo и в некоторых версиях школьного алгоритмического языка.

Алгоритм решения задачи

Алгоритм решения задачи можно записать следующим образом:

Начало

1. Подготовка экрана
2. Ввод N

3. Описание массивов X(N), Y(N)
4. Цикл ввода массива X(N)
5. Цикл пересылки X(N) → Y(N)
6. P = 5; K = 1
7. Внешний цикл сортировки, формирует циклы просмотров:
8. Пока P > 0 исполнять [
9. P = 0
10. Внутренний цикл просмотра, сравнений и перестановок:
11. Для I от 1 до (N–K) исполнять [
12. Если Y[I–1] > Y[I], то [Z = Y[I]; Y[I] = Y[I–1]; Y[I–1] = Z; P = P + 1]]
13. K = K + 1]
14. Вывод заголовков массивов X и Y для вывода результатов
15. Цикл вывода исходного массива X и отсортированного массива Y

Конец

В конце записи 11 этапа алгоритма стоят три закрывающие квадратные скобки: первая — конец блока 11, выполняющего анализ очередной пары чисел массива Y, их перестановку при необходимости и увеличение на 1 общего числа перестановок P в данном цикле просмотра; вторая скобка — конец очередного внутреннего цикла просмотра; третья скобка — конец внешнего цикла сортировки, управляющего циклами просмотров. Чтобы обеспечить наглядность для учащихся вывода результатов сортировки, на шагах 12 и 13 предусмотрены вывод заголовков и самих массивов X и Y. Учащиеся имеют возможность визуально сравнить расположение чисел в исходном массиве X и в отсортированном массиве Y.

Листинг 6.5.1

```
# Primer_6_5_1
Cls
# Ввод N
Input "Введите N = ", N
```

```
Dim X(N): dim Y(N)
# Ввод массива X и формирование начального значения массива Y
For I = 0 to N-1
Input "Для I = " + I + " X[I] = ", X[I]
Y[I] = X[I]
Next I
P = 5: K = 1
# Основной цикл сортировки
While P > 0
P = 0
# Внутренний цикл просмотра и перестановок
For I = 1 to (N-K)
    If  Y[I-1] > Y[I] then
    Z = Y[I-1]
    Y[I-1] = Y[I]
    Y[I] = Z
    P = P +1
    End If
Next I
K = K + 1
End While
# Вывод исходного массива X и отсортированного массива Y
cls
Print "Массив X" + "     " + "Массив Y"
For I = 0 to (N-1)
X[I] + "                    " + Y[I]
Next I
End
```

В программе по сравнению с алгоритмом объединены в один цикл ввод с клавиатуры чисел в массив X и пересылка массива X в массив Y. Все остальные пояснения приведены в комментариях к программе. В заголовке для выводимой информации между словами "Массив X" и "Массив Y" вставлены 5 пробелов. Выводимые в цикле числа из массива X и массива Y разделены 20 пробелами.

Пример 6.5.2

Задан массив из 15 слов, записанных прописными латинскими буквами: {VOLK, SLON, BEGEMOT, NOSOROG, MYSH, KRYSA, MEDVED, RYS, KUNITSA, ZAYATS, LISA, AIST, NERPA, MORG, AKULA}. Отсортировать слова в массиве в алфавитном порядке.

Анализ задачи

В задачах сортировки строковых данных имеет смысл понятие глубины сортировки. Дело в том, что каждая группа слов, начинающихся на одну и ту же букву, могут внутри группы дополнительно сортироваться с учетом второй, третьей и последующих букв. Глубина сортировки — число учитываемых букв. В нашем примере ограничимся сортировкой с глубиной 1, т. е. сортировка будет осуществляться с учетом только первых букв слов из массива.

В отличие от примера 6.5.1, здесь нам понадобятся строковые массивы X$(15) — исходный массив слов, Y$(15) — рабочий массив для сортировки и строковая буферная переменная Z$ для перестановки слов. Все остальные переменные I, P, K будут иметь тот же самый смысл, что и в примере 6.5.1: I — текущий номер элемента в массиве (I = 0–14). В данном примере N (число элементов в массиве) — фиксировано (N = 15). Поэтому ввод N и цикл ввода для массива X$ не требуется. Цикл пересылки слов из X$ в Y$ остается. K — номер цикла просмотра массива Y$ (сначала K = 1, т. е. первому циклу присвоен номер 1, а по завершении каждого цикла просмотра K увеличивается на 1). P — число перестановок элементов массива Y$, выполненных в текущем цикле просмотра. Перед началом каждого цикла просмотра принимается P = 0. Перед началом внешнего цикла сортировки `While P > 0` переменной P можно присвоить любое значение, удовлетворяющее условию P > 0, чтобы внешний цикл сортировки начал исполняться. Далее признак P в каждом цикле просмотра будет вычисляться автоматически как число перестановок элементов. Признаком завершения сортировки явится событие, когда в очередном цикле просмотра окажется P = 0.

ASCII-коды прописных латинских букв возрастают от 65 до 90 в направлении от A до Z. Это позволяет сравнивать в процессе просмотра слов в массиве Y$ не сами слова, а ASCII-коды их первых букв с помощью функции `ASC`: если `Asc(Y$[I-1]) > Asc(Y$[I])`, то следует в массиве Y$[I–1] и Y$[I] поменять местами, т. е. выполнить очередную перестановку и увеличить признак P на 1. Алгоритм решения задачи будет полностью соответствовать алгоритму примера 6.5.1 с учетом оговоренных изменений. Текст программы приведен в листинге 6.5.2.

Листинг 6.5.2

```
# Primer_6_5_2
Cls
Dim X(15): dim Y(15)
# Формирование массива X и начального значения массива Y
X$ = {VOLK, SLON, BEGEMOT, NOSOROG, MYSH, KRYSA, MEDVED, RYS,
KUNITSA, ZAYATS, LISA, AIST, NERPA, MORG, AKULA}
For I = 0 to 14
Y$[I] = X$[I]
Next I
P = 5: K = 1
# Внешний цикл сортировки
While P > 0
P = 0
# Внутренний цикл просмотра и перестановок
For I = 1 to 14
    If  Asc(Y$[I-1]) > Asc(Y$[I] then
    Z$ = Y$[I-1]
    Y$[I-1] = Y$[I]
    Y$[I] = Z$
    P = P +1
    End If
Next I
K = K + 1
End While
# Вывод исходного массива X$; и отсортированного массива Y$
cls
Print "Массив X$" + "      " + "Массив Y$"
```

```
For I = 0 to 14
X$[I] + "                    " + Y$[I]
Next I
End
```

Совокупность объектов для сортировки может быть задана двумя и более массивами данных. Каждый объект может характеризоваться именем и какими-то числовыми или строковыми характеристиками. Например, работник характеризуется фамилией и годом рождения. Река — названием, протяженностью (длиной) в километрах, максимальной шириной и глубиной. Озеро может характеризоваться названием, максимальной протяженностью с севера на юг и с запада на восток, площадью "зеркала", максимальной глубиной, запасами воды. Страна — названием, названием континента или части света, численностью населения, площадью, размерами валового национального продукта, национальным и религиозным составом. Поскольку данные типа записей в BASIC-256 не определены, для представления данных о каждом таком объекте понадобится два или более массивов. Столько же массивов понадобится для представления отсортированных по некоторому признаку данных. Далее рассматривается пример решения задачи сортировки для объектов, описываемых двумя характеристиками.

Пример 6.5.3

Задана таблица с данными о населении стран Европы и США (табл. 6.1). Отсортировать данные в таблице (Страна — Население) в порядке возрастания численности населения.

Анализ задачи

В таблице приведены данные о населении 21 страны. Для представления исходной таблицы нужны 2 массива: строковый массив X1$(21) для названий стран и числовой X2(21) для численности населения. Аналогично для выполнения сортировки и размещения отсортированных данных понадобятся массивы Y1$(21) для названий стран после сортировки и Y2(21) для численности населения после сортировки. Для организации обменов понадо-

Таблица 6.1

Страна	Население, тыс. чел.	Страна	Население, тыс. чел.
Россия	132 000	Норвегия	4620
США	309 469	Польша	38 138
Германия	81 758	Чехия	10 468
Великобритания	61 113	Словакия	5380
Франция	63 000	Румыния	22 200
Испания	44 109	Сербия	7366
Португалия	10 560	Хорватия	5000
Нидерланды	10 581	Италия	60 231
Швеция	9000	Швейцария	7509
Дания	5512	Бельгия	10 585
Финляндия	5545		

бятся буферные переменные Z1$ для перестановки названий стран и Z2 для перестановки данных о численности населения. Остальные переменные I, P, K имеют тот же смысл, что в примере 6.5.2: I — номер элемента в массиве. K — номер цикла просмотра данных о численности в массиве Y1$. P — число выполненных перестановок в очередном цикле просмотра. В исходных массивах данные должны размещаться таким образом, что если X1$[I] — название I-й страны, то X1[I] — численность населения той же страны. Аналогично должны быть связаны данные в отсортированных массивах: Y1$[I] и Y2[I].

Перестановки данных должны быть организованы следующим образом. Проверяется условие Y2[I–1] > Y2[I]. Если оно вы выполнено, выполняются перестановки двух пар переменных: Y1$[I–1] и Y1$[I], Y2[I–1] и Y2[I]. После этого число перестановок в текущем цикле просмотра P увеличивается на 1. В алгоритме это можно записать в виде следующей строки: Если Y2[I–1] > > Y2[I], то [Z1$ = Y1$[I–1]; Y1$[I–1] = Y1$[I]; Y1$[I] = Z1$; Z2 = Y2[I–1]; Y2[I–1] = Y2[I]; Y2[I] = Z2; P = P + 1]. Занесение данных из таблицы в массивы X1$, X2 можно сделать с помощью

анонимных массивов. Пересылку массивов X1$ в Y1$ и X2 в Y2 можно организовать в общем цикле. Весь остальной алгоритм и соответствующая ему программа не будут отличаться от примера 6.5.2.

Листинг 6.5.3

```
# Primer_6_5_3
Cls
# Описание массивов данных
Dim X1$(21): dim Y1$(21)
Dim X2(21): dim Y2(21)
# Формирование массивов X1$, X2  и начальных значений массивов
Y1$, Y2
X1$ = {Россия, США, Германия, Великобритания, Франция,
Испания, Португалия, Нидерланды, Швеция, Дания, Финляндия,
Норвегия, Польша, Чехия, Словакия, Румыния, Сербия, Хорватия,
Италия, Швейцария, Бельгия}
X2 = {132000, 309469, 81758, 61113, 63000, 44109, 10560,
10581, 9000, 5512, 5545, 4620, 38138, 10468, 5380, 22200,
7366, 5000, 60231, 7509}
For I = 0 to 20
Y1$[I] = X1$[I]
Y2[I] = X2[I]
Next I
P = 5: K = 1
# Внешний цикл сортировки
While P > 0
P = 0
# Внутренний цикл просмотра и перестановок
For I = 1 to 20
    If  Y2[I-1]) > Y2[I] then
    Z1$ = Y1$[I-1]
    Y1$[I-1] = Y1$[I]: Y1$[I] = Z1$
    Z2 = Y2[I-1]: Y2[I-1] = Y2[I]
    Y2[I] = Z2:  P = P +1
    End If
Next I
K = K + 1
End While
```

```
# Вывод отсортированных массивов Y1$, Y2
cls
Print "Страна" + "          " + "Население"
For I = 0 to 20
Y1$[I] + "           " + Y2[I]
Next I
End
```

Для наглядности вывода, чтобы выводимые данные оказывались под нужными заголовками, между заголовками "Страна" и "Население" вставлено 9 пробелов, а между данными Y1$[I] и Y2[I] — 10 пробелов.

6.6. Формирование подвижных изображений на текстовом экране

Изображение на текстовом экране формируется из каких-нибудь символов. Для создания эффекта движения используется следующий прием. Пусть некоторый объект (комбинация символов) перемещается горизонтально в строке N с позиции 1 до позиции M. Устанавливается курсор в некоторое текущее положение: строка N, позиция x. Выводится изображение объекта. Дается временная задержка на T. Величина T определяет видимую скорость движения по экрану — чем больше T, тем медленнее движение. После этого изображение стирается. Позиция в строке увеличивается на 1 и формируется изображение в новом положении. Процесс повторяется от позиции 1 до позиции M. Практически тот же принцип заложен при формировании подвижного изображения на телевизионном экране.

Пример 6.6.1

Комбинация символов ******* движется горизонтально слева направо в строке N от начала строки до заданной позиции M. Значение N вводится с клавиатуры. M выбирается в диапазоне 60–100.

Анализ задачи

В BASIC-256 нет оператора, устанавливающего курсор в заданную позицию на экране. Эту функцию приходится формировать искусственно. После команды очистки экрана `cls` курсор устанавливается в первую позицию первой строки текстового экрана. Чтобы его установить в позицию 1 строки N, нужно (N–1) раз подать команду `print`. Заданную позицию в строке можно сформировать с помощью вспомогательной строковой переменной X$. Перед началом цикла движения X$ присваивается значение пустой строки: `X$ = ""`. Нужное изображение в текущей позиции формируется как `X$ + "***"`. Изображение объекта в текущем положении стирается оператором очистки экрана `cls`. Задержка на время T формируется оператором `pause T` или пустым циклом `For... Next`, повторенным нужное число раз. Для формирования изображения в следующей позиции к строке X$ добавляется пробел: `X$ = X$ + " "`. Цикл формирования изображения повторяется M раз. Номер позиции I в строке N будет соответствовать номеру текущего цикла формирования изображения. Пусть в позиции I строки N размещается первый слева символ объекта.

Алгоритм решения задачи

Алгоритм решения задачи можно записать следующим образом:

Начало

1. Подготовка экрана
2. Ввод N
3. Основной цикл движения объекта: для I от 1 до 80 исполнять [
4. Цикл установки курсора в заданную строку: для K от 1 до (N–1) исполнять [`print`]
5. Формирование начальной позиции в строке X$: X$ = ""
6. Формирование изображения объекта "***" в позиции I строки N: вывод X$ + "***"
7. Задержка на T = 0.1 сек.
8. Приращение позиции в строке: X$ = X$ + " "
9. Очистка экрана]

Конец

Листинг 6.6.1

```
# Primer_6_6_1
Cls
Input "Введите номер строки N = ", N
X$ = ""
# Основной цикл движения по экрану
For I = 1 to 80
# Цикл установки в строку N
For K = 1 to (N-1)
Print
Next K
Print X$ + "***"
Pause 0.1
X$ = X$ + " "
Cls
Next I
End
```

6.7. Ввод/вывод данных посредством текстовых файлов

Текстовые файлы — единственный вид файлов, с помощью которых возможен ввод/вывод данных в BASIC-256. В версии 0.9.6р возможны лишь чтение и запись строковых данных. Операторы вида `X = read`, `X$ = read` в этой версии не работают. Чтение возможно только оператором вида `X$ = readline` строки текста из файла целиком. Если строка содержит числа, разделенные пробелами, их строковое представление придется выделять из строки X$ принудительно в отдельную переменную Y$ соответствующим фрагментом программы пользователя и преобразовывать в число оператором вида `X = float(Y$)`. Запись возможна операторами вида: `write X1$` в текущую строку; `writeline X$` в новую строку. Если необходимо последовательно записывать числа из массива X(N) в одну текстовую строку, разделительные пробелы следует вставлять принудительно после каждого числа, например, последовательностью операторов следующего вида:

```
For I = 1 to N
Write X[I] + " "
Next I
```

Если в строку записывается только одно число Z, пробелы не нужны: `writeline string(Z)`. Если в строку записывается последовательность чисел Блокнотом или программно для последующего программного чтения, пробел после последнего числа в строке обязателен. Для доступа к файлу рекомендуется в специальной переменной F$ сформировать спецификацию файла, т. е. полное обозначение в командной строке с указанием пути. Далее его следует открыть для записи и чтения оператором `open F$`. Если необходимо читать или дописывать данные в существующий файл, этого достаточно. Если нужно файл очистить от старой информации, используется оператор `reset`. Если путь указан правильно, а файл не существует, оператор `open F$` создает пустой файл. После завершения работы с файлом его следует закрыть оператором `close`, чтобы не потерять информацию при завершении работы программы. В BASIC-256 одновременно может быть открыт только один файл. Подробнее все это поясняется в приведенных далее примерах программ.

Пример 6.7.1

Листинг 6.7.1

```
#Пример 6.7.1. Чтение/Запись текстовых строк посредством файла
Cls
# Открытие или создание файла Result.txt, его инициализация и
запись двух строк
F2$ = "C:\OldDisc\Kursy\FreeSW\Programming\BASIC256\Z1-
BASIC256\Result.txt"
Open F2$
Reset
Writeline "Привет"
Writeline "До свидания"
Close
```

```
# Открытие файла, построчное чтение записанных строк и вывод
на текстовый экран
open F2$
While not Eof
X$ = readline
print X$
end while
close
end
```

Программа (листинг 6.7.1) оператором `open` создает и/или открывает файл Result.txt с полным обозначением F2$. Если файл существовал, оператор `Reset` его очищает и устанавливает указатель в начало файла (в первую позицию первой строки).

Далее в него операторами `writeline` пишутся две строки, и файл закрывается оператором `close`. Записанный файл в Windows можно просмотреть Блокнотом. Далее этот же файл вновь открывается, построчно читается в переменную X$ и выводится на текстовый экран до исчерпания файла, т. е. до появления символа конца файла `Eof`. В данном примере будут выведены на экран те же две записанные строки: **Привет** и **До свидания**.

По окончании работы с файлом его следует закрыть, чтобы не потерять записанные данные. Если открытый существующий файл не инициализировать оператором `reset`, новые строки при записи будут добавляться к существующим. Приведенный пример иллюстрирует принципиальные особенности записи и чтения текстовых данных посредством текстовых файлов.

Пример 6.7.2

Из файла Test1X.txt, где X — номер файла от 1 до 5 (вводится с клавиатуры), читается и автоматически суммируется строка чисел. Сумма записывается в файл Result1X.txt. Числа в файле Test1X.txt разделены произвольным числом пробелов. После последнего числа в строку перед записью в файл обязательно добавляется пробел, иначе чтение будет неверным.

Анализ задачи

Рассматриваемая задача представляет собой пример обработки числовых данных, выделяемых из текстовой строки. F1$ — строка для обозначения пути к входному и выходному файлам. X — номер для файлов теста и результата. F3$ — спецификация входного файла. F4$ — спецификация выходного файла. X$ запоминает текстовую строку с последовательностью чисел, читаемую из файла Test1X.txt. Z — накапливаемая сумма чисел. I — номер анализируемого символа строки X$. N — длина строки. Y$ — переменная для выделения строкового представления очередного числа из строки X$. Перед выделением каждого числа переменной Y$ присваивается значение пустой строки.

Алгоритм решения задачи

Алгоритм решения задачи можно записать следующим образом:

Начало

1. Подготовка экрана
2. Формирование обозначений входного и выходного файлов
3. Чтение строки чисел X$ из файла Test1X.txt
4. Определение длины строки N
5. Инициализация переменных I и Z: I = 1; Z = 0
6. Пока I < N исполнять [
7. Пропуск пробелов перед очередным числом подпрограммой `Probel`
8. Выделение строкового представления очередного числа Y$ из строки X$ подпрограммой `Number`
9. Преобразование строкового представления числа Y$ в число и добавление к сумме Z]
10. Вывод Z для контроля
11. Формирование выходной строки Z$ из числа Z
12. Запись строки Z$ в файл Result1X.txt

Конец

`Probel`. Вспомогательный алгоритм

Начало

1. # Пропускает пробелы в строке перед очередным числом в строке X$
2. Пока I-й символ строки X$(I) равен пробелу исполнять [
3. Если I < N, то I = I + 1]

Конец

`Number`. Вспомогательный алгоритм

Начало

1. Пока I-й символ строки X$(I) не пробел, исполнять [
2. Y$ = Y$ + X$[I]
3. Если I < N, то I = I + 1]

Конец

Листинг 6.7.2

```
# Primer_6_7_2
Cls
# Формирование спецификаций входного и выходного файлов
F1$ = "C:\OldDisc\Kursy\FreeSW\Programming\BASIC256\Z1-
BASIC256\"
Input "Введите номер файла X от 1 до 5: X = ", X
F3$ = F1$ + "Test1" + X +".txt"
F4$ = F1$ + "Result1" + X + ".txt"
# Чтение строки чисел из файла Test1X.txt в переменную X$
Open F3$
X$ = Readline
# Определение длины строки N, печать X$ и N
Print X$
N = Length(X$)
Print N
# Выделение чисел из строки X$ и их суммирование в переменной Z
I = 1: Z = 0
While I < N
Gosub Probel
```

```
Y$ = ""
Gosub Number
Z = Z + Float(Y$)
End While
Print Z
Close
# Формирование выходной строки из числа Z и запись в файл
Result1X.txt
Z1$ = Z +" "
Open F4$
Reset
Writeline Z1$
Close
End
Probel:
While Mid(X$,I,1) = " "
If I < N Then I = I + 1
End While
Return

Number:
While Mid(X$,I,1) <> " "
Y$ = Y$ + Mid(X$,I,1)
If I < N Then I = I + 1
End While
Return
```

ГЛАВА 7

Примеры решения графических задач в BASIC-256

Графические операторы BASIC-256 подробно рассмотрены в *главе 3*. Далее приводится ряд примеров, иллюстрирующих графические возможности этой среды программирования.

Пример 7.1

Вывести на графический экран ломаную линию из N отрезков со случайными координатами точек. Отрезки вычерчиваются случайными цветами. N вводится с клавиатуры в диапазоне 10 <= N <= 100. Использовать размер графического экрана по умолчанию и определить его.

Анализ задачи

Для вычерчивания ломаной из N отрезков необходимо сформировать координаты (N+1) точек. Координаты X1, Y1 начала первого отрезка формируются отдельно. Координаты концов отрезков X2, Y2 формируются в цикле. После вычерчивания очередного отрезка координаты точки его конца приравниваются координатам начала следующего отрезка.

Алгоритм решения задачи

Начало

1. Подготовка экрана
2. Ввод N
3. Определение характеристик графического экрана: высоты H и ширины W
4. Формирование координат начальной точки X1, Y1
5. Цикл по I от 1 до N [
6. Формирование случайных координат I-й точки: `X2 = Int(W*rand); Y2 = Int(H*rand)`
7. Формирование случайного цвета. Интенсивность образующих цветов выбирается случайным образом в диапазоне от 50 до 200: `R = 50 + Int(151*rand); G = 50 + Int(151*rand); B = 50 + Int(151*rand)`
8. Вычертить отрезок (X1, Y1) — (X2, Y2)
9. X1 = X2; Y1 = Y2]

Конец

Листинг 7.1

```
# Primer_7_1
Clg
Input "Введите число отрезков N от 32 до 100 N = ", N
# Определение характеристик графического экрана
H = graphheight: W = graphwidth
# Формирование координат и цвета точек. Вывод на графический
экран
X1 = Int(W*rand): Y1 = Int(H*rand)
For I = 1 to N
X2 = Int(W*rand): Y2 = Int(H*rnd)
R = 50 + Int(151*rand)
G = 50 + Int(151*rand)
B = 50 + Int(151*rand)
Color R, G, B
line X1, Y1, X2, Y2
X1 = X2: Y1 = Y2
Next I
```

Пример 7.2

На графическом экране размером 640×480 вычертить фигуру, состоящую из N (N <= 15) вложенных друг в друга правильных K-угольников (K <= 40), каждый из которых окрашен случайным цветом CL. Все многоугольники вписаны в окружности с равномерно изменяющимися радиусами R от R1 до R1/3 с общим центром в центре экрана O(320, 240). Параметры N, K, R1 вводятся с клавиатуры. Окружности на фигуре не отображаются.

Анализ задачи

Начинать вычерчивание следует с многоугольника наибольшего размера и вычерчивать их последовательно в порядке убывания радиуса описанной окружности R — только в этом случае будут видны все N многоугольников. Для решения задачи, кроме указанных в условии переменных N, K, R, понадобятся следующие дополнительные переменные: DR — шаг изменения радиуса описанной окружности: DR = (R – R/3)/(N – 1). X1(K), Y1(K) — массивы абсцисс и ординат вершин текущего многоугольника. P(M), M = 2*K — общий массив координат вершин текущего многоугольника с номером I, с радиусом описанной окружности R = R1 – DR*(I – 1); I = 1, 2, ..., N. Для I = 1 R = R1; для I = N R = R1/3.

Если условно обозначить вершины многоугольника и их координаты как A0(X1[0], Y1[0]), A1(X1[1], Y1[1]), ..., AK–1(X1[K–1], Y1[K–1]), то общий массив координат P запишется в следующем виде: P = {X1[0], Y1[0], X1[1], Y1[1], ..., X1[K–1], Y1[K–1]}. Связь между координатами вершин AJ(X1[J], Y1[J]), J = 0, 1, 2, ..., K–1 и элементами массива P получится из последовательности: P[0] = X1[0], P[1] = Y1[0]; P[2] = X1[1], P[3] = Y1[1]; ... P[2*J] = = X1[J], P[2*J+1] = Y1[J]; ... P[2*K–2] = X1[K–1], P[2*K–1] = = Y1[K–1].

Положение начальных вершин многоугольников можно выбрать любым. Для определенности примем, что все начальные вершины располагаются на горизонтальном луче, исходящем из центра экрана параллельно оси X и с тем же направлением. Напомним,

что началом координат графического экрана является его левый верхний угол. Ось X при этом направлена горизонтально вправо, ось Y — вертикально вниз. Координаты вершин многоугольников при этом будут описываться через его радиус описанной окружности R следующими соотношениями. Начальные вершины A0: X1[0] = 320 + R; Y1[0] = 240. Произвольные вершины AJ: X1[J] = 320 + R*cos(2*PI*J/K), Y1[J] = 240 + R*sin(2*PI*J/K); J = 0, 1, ..., K–1.

Алгоритм и программа будут строиться по конструкции типа "цикл в цикле". Внешний цикл строится по номеру многоугольника I, изменяющемуся от 1 до N. Внутренний цикл формируется по номеру вершины многоугольника J, который изменяется от 0 до K–1.

Алгоритм решения задачи

Начало

1. Подготовка текстового и графического экранов (установка нужных режимов, очистка)
2. Ввод числа многоугольников N, числа вершин многоугольника K, радиуса описанной окружности первого многоугольника R1
3. Формирование описаний массивов X1(K), Y1(K), P(2*K)
4. Вычисление шага изменения радиуса описанной окружности DR:

   ```
   DR = 2*R1/(3*(N-1))
   ```

5. Внешний цикл по номеру многоугольника I от 1 до N[
6. Вычисление радиуса описанной окружности R для I-го многоугольника:

   ```
   R = R1 - (I-1)*DR
   ```

7. Вычисление и установка случайного цвета I-го многоугольника по составляющим r, g, b: `r = 50 + Int(151*rand); g = 50 + Int(151*rand); b = 50 + Int(151*rand)`
8. Внутренний цикл по номеру вершины J от 0 до K–1[

9. Вычисление координат вершины AJ:

   `X1[J] = 320 + R*cos(2*PI*J/K);`

   `Y1[J] = 240 + R*sin(2*PI*J/K)`

10. Формирование общего массива координат вершин

    `P:  P[2*J] = X1[J];`

11. `P[2*J+1] = Y1[J].` Конец внутреннего цикла по J:]

12. Вычерчивание I-го многоугольника по массиву координат вершин P. Конец внешнего цикла по I:]

Конец

Листинг 7.2

```
# Primer_7_2
# Подготовка графического и текстового  экранов
Graphsize 640, 480
Cls
Clg
# Ввод N, K, R
Input "Введите число многоугольников от 2 до 15 N = ", N
Input "Введите число вершин многоугольника от 3 до 30 K = ", K
Input "Введите радиус описанной окружности внешнего_
многоугольника от 100 до 240 R1 = ", R1
Dim X1(K): Dim Y1(K)
Dim P(2*K)
DR = 2*R1/(3*(N-1))
# Внешний цикл по номеру многоугольника
For I = 1 To N
   # Вычисление радиуса описанной окружности и цвета I-го
многоугольника
   R = R1 - (I-1)*DR
   r = 50 + Int(151*rand):  g = 50 + Int(151*rand)
   b = 50 + Int(151*rand): Color r, g, b
   # Внутренний цикл по номеру вершины J
   For J  = 0 To (K-1)
      # Вычисление координат вершины AJ и занесение их в массив P
      X1[J] = 320 + R*cos(2*PI*J/K): Y1[J] = 240 +
                                              R*sin(2*PI*J/K)
```

```
      P[2*J] = X1[J]:   P[2*J+1] = Y1[J]
   Next J
# Вычерчивание I-го многоугольника
Poly P
Next I
End
```

Пример 7.3

Построение правильного незакрашенного многоугольника по следующим данным, вводимым с клавиатуры: число вершин: N (от 3 до 50); радиус описанной окружности: R (от 100 до 300 пикселов). Цвет сторон многоугольника C формируется случайным образом в виде целого числа RGB.

Анализ задачи

Многоугольник удобно расположить на экране так, чтобы центр описанной окружности совпадал с центром экрана размером 800×600 пикселов — точкой O(400, 300). Для решения задачи, кроме указанных в условии, понадобятся следующие переменные: массивы координат вершин X1(N+1), Y1(N+1); I — номер текущей точки AI, причем I = 0, 1, ..., N. При этом вершины A0 и AN совпадают. Координаты вершин AI(X[I], Y[I]) вычисляются по соотношениям, аналогичным соотношениям в *примере 7.2*: `X[I] = 400 + R*cos(2*PI*I/N); Y[I} = 300 + R*sin(2*PI*I/N);` I = 0, 1, 2, ..., N.

Случайный RGB-код цвета сформируем в соответствии с соотношением (3.1) в следующей форме: `RGB = 256*(256*(50 + Int(200*rand) + (50 + Int(200*rand)) + 50 + Int(200*rand)`. Интенсивности всех трех составляющих цвета r, g и b формируются одинаково: `r = 50 + Int(200*rand); g = 50 + Int(200*rand); b = 50 + Int(200*rand).`

Алгоритм решения задачи

Начало

1. Подготовка текстового и графического экранов
2. Ввод числа сторон N и радиуса описанной окружности R
3. Описание массивов координат точек X1(N+1), Y1(N+1)
4. Формирование случайного цвета линий RGB:
   `r = 50 + Int(200*rand); g = 50 + Int(200*rand);`
   `b = 50 + Int(200*rand); RGB = 256*(256*r + g) + b`
5. Цикл формирования координат вершин AI и вычерчивания сторон многоугольника:
6. для I от 0 до N
7. `[X1[I] = 400 + R*cos(2*PI*I/N);`
   `Y1[I] = 300 + R*sin(2*PI*I/N);`
8. Если I >= 1, то вычертить отрезок по точкам (A(I–1) – AI)]

Конец

Листинг 7.3

```
# Primer_7_3
# Подготовка текстового и графического экранов
Graphsize 800, 600
Cls
Clg
# Ввод N, R
Input "Введите число сторон многоугольника от 3 до 50 N = ", N
Input "Введите радиус описанной окружности от 100
                                                  до 300 R = ", R
Dim X1(N+1): Dim Y1(N+1)
# Формирование случайного цвета CRGB
r = 50 + Int(200*rand):  g = 50 + Int(200*rand)
b = 50 + Int(200*rand):  RGB = 256*(256*r + g) + b
color CRGB
# Цикл формирования массивов координат вершин X1, Y1
и вычерчивания сторон многоугольника
For I = 0 to N
```

```
X1I] = 400 + R*cos(2*PI*I/N):  Y1I] = 300 + R*sin(2*PI*I/N)
If I >= 1 Then Line X1[I-1], Y1[I-1], X1[I], Y1[I]
Next I
End
```

Пример 7.4

Вертикальный отрезок длины L летит горизонтально с левого края графического экрана на правый. Длина отрезка L, его цвет C, высота полета над осью X и скорость движения V выбираются произвольно при программировании задачи.

Анализ задачи

Идея формирования подвижного изображения на графическом экране не отличается от аналогичной идеи формирования движения комбинации символов на текстовом экране в *разд. 6.2*. Формируется изображение отрезка цвета C в некотором текущем положении по координате X. Задается временная задержка T. Далее изображение отрезка стирается или путем очистки графического экрана, или путем прочерчивания того же отрезка белым цветом. Дается приращение координате X, и отрезок вычерчивается в новом положении.

Выберем следующие параметры для уточнения задачи. Режим графического экрана примем по умолчанию 300×300 пикселов. L = 100. H = 100. C = 0, 0, 200 (синий цвет отрезка). T = 0.2 сек. Увеличение T замедляет движение по экрану, уменьшение ускоряет. Для наглядности при программировании этой задачи рекомендуется нарисовать схему с изображением графического экрана и отрезка в текущем положении. Во всех задачах на формирование подвижных объектов задержку T следует подбирать с учетом производительности конкретной модели компьютера. В примерах 7.4–7.8 задержка подобрана так, чтобы движение объекта наглядно отображалось на экране, исходя из следующих характеристик компьютера: процессор двухъядерный Intel Core 2 Duo E6750 с тактовой частотой 2,66 ГГц; ОЗУ 4 Гбайт. Для более медленных компьютеров, возможно, значения задержек T придется увеличивать по сравнению с приведенными в примерах.

Алгоритм решения задачи

Начало

1. Подготовка графического и текстового экранов
2. Выбор значений для переменных L, H, T и цвета C
3. Цикл движения отрезка по координате X от 0 до 300
4. [Вычертить отрезок по координатам точек (X, H) — (X, H+L).
5. Временная задержка на T секунд.
6. Стереть отрезок.]

Конец

Листинг 7.4

```
# Primer_7_4
# Подготовка текстового и графического экранов.
Graphsize 300, 300
Cls
Clg
# Выбор значений L, H, T, C
L = 100: H = 100
T = 0.15: C = 0, 0, 200
# Цикл горизонтального движения отрезка
For X = 0 To 300
Line X, H, X, H+L
Pause T
Clg
Next X
End
```

Пример 7.5

Вертикальный отрезок длиной L движется по графическому экрану слева направо параллельно прямой с уравнением Y = K*X. Начальное положение отрезка: (0, H) — (0, H+L). Параметры L, H, задержка T, задающая скорость движения, и коэффициент наклона K выбираются произвольно при программировании задачи.

Анализ задачи

Математическая модель задачи практически аналогична рассмотренной модели задачи *примера 7.4* о горизонтальном движении отрезка AB с начальными координатами концов A(0, H), B(0, H + L). Изменяется только зависимость для координат YA и YB концов отрезка: YA = H + K*X; YB = H + L + K*X при X изменяющемся от 0 до 300.

Коэффициент `K = tg(phi)`, где `phi` — угол наклона направления движения (прямой Y = K*X) к оси X. Если учесть, что ось Y графического экрана направлена вниз, то при K > 0 отрезок AB в "полете" будет снижаться (координаты YA, YB будут увеличиваться при увеличении X). При K < 0 отрезок на экране движется вверх, т. е. YA, YB уменьшаются при увеличении X. Других отличий в описании алгоритма нет. Поэтому сразу перейдем к разработке программы.

Листинг 7.5

```
# Primer_7_5
# Подготовка экрана
Graphsize 300, 300
Cls
Clg
# Установка значений параметров L, H, T, K и цвета отрезка
(синий)
L = 100: H = 100
T = 0.05: Color 0, 0, 200
K = -0.3
# Цикл движения отрезка
For X = 0 To 300
Line X, H + K*X, X, H+L +K*X
Pause T
Clg
Next X
End
```

Пример 7.6

Равнобедренный прямоугольный треугольник ABC со сторонами AB = 80 (основание), AC = BC = 50 (боковые стороны) летит по графическому экрану слева направо параллельно прямой Y = K*X. Графический экран имеет размеры 640×480. В начальном положении AB лежит на оси Y: XA = XB = 0; YA = 200, YB = 280. Треугольник движется вершиной C в сторону оси X. Коэффициент K (–0.5 <= K <= 0.5) и задержка T <= T <= 0.2 сек.) вводятся с клавиатуры.

Анализ задачи

Для правильного понимания задачи рекомендуется на рисунке показать начальное положение треугольника ABC. Математическая модель движения каждой точки треугольника в данном примере не отличается от модели движения в *примере 7.5*. Высота CD, опущенная из вершины C на основание AB, вычисляется из решения прямоугольного треугольника ACD: CD = (502 – 402)1/2 = = 30. Отсюда определяются начальные координаты точки C: XC = 30, YC = 240. Теперь в цикле движения при изменении X от 0 до 610 координаты всех точек треугольника ABC описываются следующими соотношениями: XA = XB = X; XC = X + 30; YA = 200 + K*X; YB = 280 + K*X; YC = 240 + K*X.

Поскольку основной алгоритм формирования движения объекта в этом примере не отличается от аналогичных алгоритмов в *примерах 7.4* и *7.5*, можно сразу перейти к рассмотрению программы.

Листинг 7.6

```
# Primer_7_6
# Подготовка текстового и графического экранов
Graphsize 640, 480
Cls
Clg
# Ввод параметров T и K и установка красного цвета
треугольника
Input "Введите K от -0.5 до 0.5, K = ", K
```

```
Input "Введите T от 0.01 до 0.2, T = ", T
Color 200, 0, 0
# Цикл движения фигуры
For X = 0 To 310
Line X, 200 + K*X, X, 280 + K*X
Line X, 200 + K*X, X + 30, 240 + K*X
Line X, 280 + K*X, X + 30, 240 + K*X
Pause T
Clg
Next X
End
```

Пример 7.7

Отрезок AB длиной L (50 <= L <= 200) вращается вокруг центра графического экрана 400×400. Цвет отрезка, скорость вращения (задержка T, 0.0001 <= T <= 0.1) и число полных оборотов P (1 <= P <= 10) задаются произвольно.

Анализ задачи

Построим математическую модель задачи. Считаем, что вращение идет в положительном направлении (по часовой стрелке от оси X к оси Y). Начальное положение отрезка: A(200, 200) — центр экрана и центр вращения; B(200 + L, 200) — луч AB направлен параллельно оси X и в ее же направлении. Текущий угол поворота F отрезка AB относительно оси X имеет пределы 0 <= F <= P*2*PI. Шаг поворота DF выберем произвольно 50 или $\pi/36$ (шаг влияет на скорость вращения и его плавность). При вращении координаты точки B изменяются по следующему закону: XB = 200 + L*cos F; YB = 200 + L*sin F. Точка A (центр вращения) — имеет постоянные координаты: A(200, 200). Напомним, что в описании алгоритма открывающая [и закрывающая] квадратные скобки являются операторами начала и конца цикла или блока операторов и в данном примере ограничивают тело цикла.

Алгоритм решения задачи

Начало

1. Подготовка текстового и графического экранов
2. Установка значений параметров задачи: L, P, DF, T и цвета отрезка
3. Цикл вращения по F от 0 до 2*π*P с шагом DF
4. [Вычертить отрезок AB в текущем положении:
   (200, 200) – (200 + L*cos F, 200 + L*sin F)
5. Задержка на T сек.
6. Стереть отрезок AB]

Конец

Листинг 7.7

```
# Primer_7_7
# Подготовка текстового и графического экранов
Graphsize 400, 400
Cls
Clg
# Установка параметров задачи L, P, DF, T и зеленого цвета
отрезка
L = 100:  P = 5
DF = PI/36:  T = 0.005
Color 0, 200, 0
# Цикл вращения отрезка AB
For F = 0 To 2*PI*P step DF
Line 200, 200, 200 + L*cos(F), 200 + L*sin(F)
Pause T
Clg
Next F
End
```

Пример 7.8

Равносторонний треугольник ABC со стороной L (50 <= L <= 250) вращается вокруг вершины A, помещенной в центр экрана размером 500×500. Параметры задачи: L, число полных оборотов треугольника P, шаг DF изменения угла поворота F треугольника относительно оси X и задержка T, определяющие скорость вращения, а также цвет треугольника — задаются произвольно. Начальное положение треугольника ABC: основание AB расположено параллельно оси X; вершина B — справа от центра вращения; вершина C — вниз.

Анализ задачи

Математическая модель задачи для данного примера не имеет существенных отличий от модели *примера 7.7*. Для лучшего понимания рекомендуется нарисовать треугольник, повернутый на угол F. Угол поворота F удобно отсчитывать для основания AB относительно оси X в положительном направлении — в сторону оси Y (по часовой стрелке). Начальные координаты вершин треугольника: A(250, 250) — центр вращения и центр экрана; B(250+L, 250); C(250+L/2, 250+L*31/2). Начальные координаты вершины C вычисляются из решения треугольника ABC относительно высоты CD, опущенной из вершины C на основание AB. При повороте треугольника на угол F координаты вершин описываются следующими соотношениями.

- `XA = YA = 250` — центр вращения;
- `XB = 250 + L*cos(F);`
- `YB = 250 + L*sin(F);`
- `XC = 250 + L*cos(F+PI/3);`
- `YC = 250 + L*sin(F+PI/3).`

При вычислении координат вершины C учтено, что сторона AC повернута относительно стороны AB на 60°, т. е. на $\pi/3$, в положительном направлении. Алгоритм для данного примера аналогичен алгоритму *примера 7.7*. Перейдем сразу к рассмотрению программы.

Листинг 7.8

```
# Primer_7_8
# Подготовка текстового и графического экранов
Graphsize 500, 500
Cls
Clg
# Установка параметров задачи L, P, DF, T и красного цвета
треугольника
L = 150: P = 10
DF = PI/72:  T = 0.001
Color 200, 0, 0
# Цикл вращения треугольника
For F = 0 To 2*PI*P Step DF
XA = 250: YA = 250
XB = 250 + L*cos(F): YB = 250 + L*sin(F)
XC = 250 + L*cos(F+PI/3): YC = 250 + L*sin(F+PI/3)
Line XA, YA, XB, YB
Line XA,YA, XC, YC
Line XB, YB, XC, YC
Pause T
Clg
Next F
End
```

Пример 7.9

Построить графики следующих функций:

- $Ym = Xm^2$ для $10 <= Xm <= 10$;
- $Ym = \sin(Xm)$ для $-\pi <= Xm <= \pi$.

Здесь Xm, Ym — математические переменные (переменные для записи исходной математической зависимости).

Анализ задачи

Разработка программы построения графика функции на графическом экране — это единственный пример задачи моделирования в школьном курсе информатики, где необходимо явно реализо-

вать все этапы моделирования. Автор обычно совмещал изложение этого материала с изложением раздела Моделирование. В задаче необходимо выделить следующие этапы.

1. Выбор рационального расположения осей координат графика X1, Y1 на графическом экране. Оси координат графического экрана, как и в *примерах 7.5–7.8*, обозначаются X и Y).
2. Рациональный выбор масштабов Mx, My для отображения графика.
3. Вычисление моделирующей зависимости вида Y = F(X) в координатах графического экрана и пределов изменения переменной X.

Этапы 1 и 2 связаны между собой. Поэтому возможен вариант выполнения этих этапов по методу двухшагового последовательного приближения: примерные расчеты обоих этапов на первом шаге и уточнение расчетов на втором шаге. Необходимость всех этих расчетов связана с тем, что исходная функция для отображения в виде графика задана в математических координатах Xm, Ym, а все графические операторы (`Plot`, `Line`, `Circle`, `Poly` и др.) работают только в координатах графического экрана X, Y. Кроме того, необходимо рационально использовать площадь графического экрана для расположения графика — при правильном выборе размещения осей координат графика и масштабов отображения функции график должен занимать не менее 2/3 площади экрана.

Оси графического экрана (начало координат — левый верхний угол, ось X — влево, ось Y — вниз) неудобны для восприятия графиков. В математике принято ось абсцисс направлять влево, а ось ординат — вверх. При этом формально можно в осях X, Y отображать графики только в первом квадранте. При знакопеременных пределах изменения значений аргумента Xm и функции Ym "в лоб" без преобразования координат такие графики отобразить нельзя.

Кроме того, представьте себе зависимость Ym = sin(Xm) при 0 <= Xm <= π. Формально ее можно отобразить в координатах графического экрана X, Y. Но при отображении в таком виде максимальное значение синуса будет равно 1 пикселу. Другими

словами, без правильного масштабирования график будет просто не виден.

Итак, для построения графиков понадобятся следующие три системы координат:

- Xm, Ym — координаты (переменные), в которых в общем виде описана исходная математическая зависимость:

$$Ym = Fm(Xm); \qquad (7.9.1)$$

- X1, Y1 — система масштабированных координат, в которой будет отображаться график в виде зависимости Y1 = F1(X1);
- X, Y — система координат графического экрана, в которой работают графические операторы.

Для координат X1, Y1 начало — точка на графическом экране O1(X0, Y0). Направление X1, Y1 — классическое для математики: Y1 — вверх, X1 — вправо. Координаты X1, Y1 связаны с математическими координатами Xm, Ym масштабами Mx, My:

$$Mx = X1/Xm,\ My = Y1/Ym. \qquad (7.9.2)$$

Размерность Mx, My — в пикселах на математическую единицу.

Задача анализа — перейти от исходной (моделируемой) математической зависимости Ym = Fm(Xm) к зависимости в координатах графического экрана вида Y = F(X).

$$X1 = X - X0;\ Y1 = Y0 - Y. \qquad (7.9.3)$$

Связь координат графика X1, Y1 с координатами графического экрана определяется выражением (7.9.3) и поясняется рис. 7.1.

Получим моделирующую зависимость Y = F(X) в общем виде. Для этого вначале в исходной зависимости (7.9.1) нужно заменить Xm, Ym на X1, Y1 в соответствии с соотношениями (7.9.2) и решить получившееся уравнение относительно Y1:

$$Y1 = My*Fm(X1/Mx). \qquad (7.9.4)$$

Далее в (7.9.4) следует заменить X1, Y1 на графические координаты X, Y в соответствии с выражениями (7.9.3) и решить полученное уравнение относительно Y:

$$Y = Y0 - My*Fm((X - X0)/Mx). \qquad (7.9.5)$$

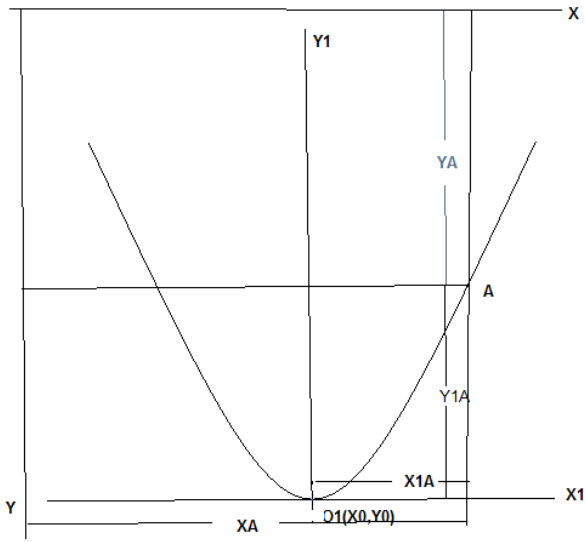

Рис. 7.1. Связь координат точки A на графике в системах X–Y и X1–Y1: XA = X0 + X1A; YA = Y0 – Y1A

Для получения конкретного вида моделирующей зависимости (7.9.5) из моделируемой зависимости (7.9.1) необходимо выбрать положение осей графика X1, Y1 на графическом экране (найти точку O1(X0, Y0)) и выбрать масштабы Mx, My.

Для дальнейших расчетов понадобятся следующие величины:

- Xmax, Ymax, Xmin, Ymin — максимальные и минимальные значения координат графического экрана. Например, если установлен режим графического экрана 640×480, то 0 <= X <= <= 639, 0 <= Y <= 479, т. е. Xmax = 639, Ymax = 479. При этом минимальные значения координат Xmin = Ymin = 0. Надо отметить, что при выходе значений координат графического экрана за допустимые пределы при выполнении графических операторов прерывания работы программы и сообщений об ошибке в BASIC-256 не возникает. Просто часть изображения объекта, выходящая за пределы экрана, не будет видна;
- Xmmax, Xmmin, Ymmax, Ymmin — максимальное и минимальное значения математических координат Xm и Ym.

Введем в рассмотрение диапазоны значений переменных. Диапазоны изменения графических координат X, Y и диапазоны изменения математических величин Xm, Ym опишутся следующими выражениями:

$$DX = |Xmax - Xmin|, \quad DY = |Ymax - Ymin|;$$
$$DXm = |Xmmax - Xmmin|, \; DYm = |Ymmax - Ymmin|. \tag{7.9.6}$$

Координаты точки O1(X0, Y0) — начала координат для системы координат X1, Y1, связанной с графиком, в системе координат экрана формально можно вычислить следующим образом:

$$X0 = Int(Xmax*|Xmmin|/DXm);$$
$$Y0 = Int(Ymax*|Ymmax|/DYm). \tag{7.9.7}$$

Но часто рациональнее будет воспользоваться следующими соображениями. Если пределы по Xm заданы симметрично или почти симметрично относительно 0, как в нашем примере (–10 <= Xm <= 10), целесообразно ось Y1 графика провести вертикально вверх по центру графического экрана. То есть для выбранного режима экрана X0 = 320. Если функция Fm(Xm) в заданном диапазоне Xm не меняет знака (конкретно в нашем примере Ym = X2, Fm(Xm) >= 0), целесообразно ось координат графика X1 провести слева направо на высоте 10–20 пикселов от нижнего края графического экрана (Ymax = 479), поэтому примем Y0 = 460.

Если бы пределы изменения значений Fm(Xm) были симметричны относительно Ym = 0, как это имеет место для функции вида Ym = sin(Xm), когда в пределах $-\pi <= Xm <= \pi$ значения синуса лежат в пределах –1 <= Ym <= 1, то ось X1 целесообразно было бы провести слева направо через центр выбранного окна графики: Y0 = 240.

Выбор масштабов рассмотрим формально для произвольной зависимости (7.9.1), а затем интерпретируем соотношения для заданной в примере функции $Ym = Xm^2$. В общем случае с учетом соотношений (7.9.6) целесообразно начальные приближения для Mx, My вычислять следующим образом:

$$Mx = DX/DXm; \quad My = DY/DYm. \tag{7.9.8}$$

Полученные из (7.9.8) значения Mx, My для удобства пересчета математических координат Xm, Ym в координаты масштабированного графика X1, Y1 и наоборот следует округлить до ближайших меньших значений чисел из ряда: 0.001, 0.002, 0.005, 0.01, 0.02, 0.05, 0.1, 0.2, 0.5, 1, 2, ..., 9, 10, 20, ..., 90, 100, 200, 300, ..., 900, 1000, ...

Чтобы выяснить, не возникает ли при выбранных Mx, My выхода графика за пределы выбранного графического экрана, следует проверить выполнение следующих соотношений:

$$\begin{aligned} &X0 + Mx*Xmmax <= Xmax;\ X0 >= Mx*|Xmmin|; \\ &Y0 + My*|Ymmin| <= Ymax;\ Y0 >= My*|Ymmax|. \end{aligned} \qquad (7.9.9)$$

Если все соотношения (7.9.9) выполнены, график не выходит за пределы графического экрана, и X0, Y0, Mx, My выбраны правильно. Можно в соответствии с соотношением (7.9.5) рассчитать моделирующую зависимость для последующего использования в алгоритме и программе. Если какие-то из соотношений (7.9.9) нарушаются, можно или изменить характеристики графического экрана, помня о том, что его предельные размеры для BASIC-256 равны 800×600, или увеличить Mx, My, выбрав для них следующие рекомендованные значения.

Вернемся к исходному примеру. Для зависимости $Ym = Xm^2$ в пределах –10 <= Xm <= 10 на экране 640×480 выбраны: X0 = 320, Y0 = 460. Рассчитаем Mx и My.

Xmmin = –10; Xmmax = 10. По соотношениям (7.9.6) получим:

DXm = 10 – (–10) = 20; DYm = 100 – 0 = 100; DX = 639; DY = 479.

Приближенные значения для масштабов Mx, My получим из выражений (7.9.8):

Mx = DX/DXm = 639/20 = 31.95; My = DY/DYm = 479/100 = 4.79.

После округления до ближайших меньших рекомендованных значений принимаем: Mx = 30; My = 4.

По соотношениям (7.9.9) проверяем, помещается ли график на графическом экране. Получим: для графика 20 <= X <= 620, 20 <= Y <= 420; сравним для графического экрана 0 <= X <= 639;

0 <= Y <= 479. Видно, что график не выходит за пределы графического экрана. Это значит, что X0, Y0, Mx, My выбраны удачно, можно в соответствии с выражением (7.9.5) рассчитать моделирующую зависимость:

$$Y = 460 - 4*(X - 320)/30)^2 = 460 - (X - 30)^2/225; \quad 20 <= X <= 620. \tag{7.9.10}$$

Итак, моделируемая зависимость примера 7.9 имеет вид:

$$Ym = Xm^2 \text{ для } -10 <= X <= 10.$$

Моделирующая зависимость в координатах графического экрана получилась в форме:

$$Y = 460 - (X - 30)^2/225 \text{ для точек графика } 20 <= X <= 620.$$

Последние данные — пределы изменения X для графика — нужны для организации в программе цикла формирования точек графика. В рассмотренном примере Xm имеет знакопеременные пределы, а Ym > = 0.

Рассмотрим для наглядности еще один пример расчета моделирующей зависимости для построения графика упоминавшейся в рассуждениях функции Ym = sin(Xm) в диапазоне $-\pi <= X <= \pi$ при тех же характеристиках окна графики: 640×480, что и в предыдущем примере. Отличие от рассмотренного примера только в одном: Xm и Ym — знакопеременные.

Поскольку заданная зависимость — нечетная функция с симметричными пределами значений по Xm и Ym, целесообразно выбрать начало координат для осей X1, Y1 масштабированного графика — точку O1(X0, Y0) — в центре графического экрана: X0 = 320, Y0 = 240. Поскольку $-\pi <= Xm <= \pi$, –1 <= Ym <= 1, то: $DXm = 2\pi = 6.28$; DYm = 2.

Рассчитываем масштабы Mx, My: Mx = DX/DXm = 639/6.28 = = 101.75; My = DY/DYm = 479/2 = 239.5. Округляем Mx, My до ближайших рекомендованных (меньших) значений: Mx = 100; My = 200. Проверяем пределы изменения значений графика в координатах графического экрана: (X0 + Mx*Xmin) = 320 – 100* *3.14 = 6; (X0 + Mx*Xmax) = 320 + 100*3.14 = 634; (Y0 – – MY*Ymin) = 240 + 200*1 = 440; (Y0 – MY*Ymax) = 240 – – 100*1 = 40.

Для точек графика в координатах графического экрана X, Y справедливы соотношения: 6 <= X <= 634; 40 <= Y <= 440. То есть при выбранных X0, Y0, Mx, My все точки графика окажутся в пределах графического экрана 640×480. Моделирующую зависимость получим из (7.9.5) в следующем виде:

$$Y = 240 + 200*\sin((X - 320)/100) \quad \text{для } 6 <= X <= 634. \tag{7.9.11}$$

Итак: моделируемая зависимость: Ym = sin(Xm) для –3.14 <= X <= 3.14; моделирующая зависимость: Y = 240 + 200*sin(0.01* *(X – 320)) для 6 <= X <= 634.

Для построения графика $Ym = Xm^2$ в пределах –10 <= Xm <= 10 по зависимости (7.9.10) приведем алгоритм и программу. На графике отобразим оси координат графического экрана X, Y и графика X1, Y1.

Алгоритм решения задачи

Начало

1. Подготовка текстового и графического экранов
2. Построение осей координат (X, Y) и (X1,Y1) с обозначениями
3. Цикл по X от 20 до 620
4. [$Y = 460 – (X – 30)^2/225$
5. Вывести точку (X, Y)]
6. Вывести на график обозначения X, Y, X1, Y1

Конец

Листинг 7.9.1 (построение графика $Ym = Xm^2$)

```
# Primer_7_9_1
# Подготовка текстового и графического экранов.
Graphsize 640, 480
Cls
Clg
# Оси X, Y (синие)
Color 0, 0, 200
```

```
Line 0, 0, 639, 0
Line 0, 0, 0, 479
# Оси X1, Y1 (красные)
Color 200, 0, 0
Line 0, 460, 639, 460
Line 240, 0, 240, 479
# Цикл построения графика
For X = 20 To 620
Y = 460 - (X - 30)^2/225
Plot X, Y
Next X
# Вывод обозначений осей и графика
Color 0, 0, 0
Font "Arial",10, 75
Text 5, 465, "Y"
Text 630, 5, "X"
Text 325, 5, "Y1"
Text 630, 445, "X1"
Text 120, 40, "График Ym = Xm^2"
End
```

Листинг 7.9.2 (построение графика Ym = sin(Xm))

```
# Primer_7_9_2
# Подготовка графического и текстового экранов
Graphsize 640, 480
Cls
Clg
# Вычерчивание осей X, Y (синие)
Color 0, 0, 200
Line 0, 0, 639, 0
Line 0, 0, 0, 479
# Вычерчивание осей X1, Y1 (красные)
Color 200, 0, 0
Line 0, 240, 639, 240
Line 320, 0, 320, 479
# Цикл построения графика Ym = sin(Xm)
For X = 6 To 634
Y = 240 + 200*sin(0.01*(X-320))
```

```
Plot X, Y
Next X
# Вывод обозначений осей и графика (черные)
Color 0, 0, 0
Font "Arial",10, 75
Text 5, 465, "Y"
Text 625, 5, "X"
Text 325, 5, "Y1"
Text 625, 225, "X1"
Text 120, 10, "График Ym = sin(Xm)"
End
```

Для этого примера ранее выполнены все расчеты. Алгоритм не имеет принципиальных отличий от алгоритма *примера 7.9.1*. Все комментарии содержатся в листинге программы.

ГЛАВА 8

Обработка строковых данных

С операторами обработки строковых данных мы познакомились в *разд. 2.2*. Рассмотрим некоторые примеры типовых школьных задач, связанных с обработкой строк.

Пример 8.1

С клавиатуры вводятся произвольная текстовая строка S$ длиной L не более 256 символов и отдельный символ T$. Подсчитать число вхождений символа T$ в строку S$. Если T$ — буква латинского алфавита, то подсчитать число вхождений этой буквы в строчном и прописном видах.

Анализ задачи

Согласно табл. 2.2 коды NL прописных латинских букв от A до Z лежат в пределах от 65 до 90. Коды аналогичных строчных букв на 32 больше. То есть NL("Z") = 90, а NL("z") = 122. Это соотношение справедливо для всех 26 латинских букв в кодировке ASCII. Рассматриваемый пример представляет собой стандартную задачу поиска с усложненным условием. В задаче следует ввести специальные переменные для условий поиска: W1 = 1, если T$ — прописная латинская буква; W2 = 1, если T$ — строчная латинская буква. Условие W1 выполнено, если 65 <= NL(T$) <= 90. Условие W2 выполнено, если 97 <= NL(T$) <= 122.

Используя тот факт, что в BASIC-256 выполненное условие трактуется как 1, а невыполненное как 0, можно сформировать для вычисления W1 и W2 следующие выражения: NL = ASC(T$); W1 = (NL >= 65)*(NL <= 90); W2 = (NL >= 97)*(NL <= 122). Сравнение I-го символа строки S$ с T$ можно организовать с помощью вспомогательных переменных V, V1, V2, V3: V = 1, если I-й символ строки S$[I] удовлетворяет условиям поиска, и V = 0 в остальных случаях.

В содержательной форме условия для V1–V3 формулируются следующим образом: если T$ не является латинской буквой, то V1 = 1 при совпадении очередного символа строки с заданным (S$[I] = T$); если T$ является строчной латинской буквой, то V2 = 1, если очередной символ строки совпадает с заданным или с тем же строчным символом с кодом на 32 меньше (S$[I] = T$) или (S$[I] = CHR(NL – 32)); если T$ является прописной латинской буквой, то V3 = 1 при совпадении очередного символа строки с заданным или с тем же строчным символом, но с кодом на 32 больше ((S$[I] = T$) или (S$[I] = CHR(NL + 32)). Теперь эти условия для вычисления V можно записать в виде общего аналитического выражения через составляющие условия W1, W2, V1–V3:

$$\begin{gathered} V1 = (NOT\ (W1 + W2))*(S\$[I] = T\$); \\ V2 = W1*((S\$[I] = T\$) + S\$[I] = CHR(NL + 32)); \\ V3 = W2*((S\$[I] = T\$) + (S\$[I] = CHR(NL - 32)); \\ V = V1 + V2 + V3. \end{gathered} \qquad (8.1)$$

Алгоритм решения задачи

Начало

1. Подготовка текстового экрана
2. Ввод строки S$
3. Вычисление длины строки N
4. Ввод символа для поиска T$
5. Установка в 0 счетчика символов CNT

6. Формирование условий принадлежности символа T$ множеству латинских букв W1, W2
7. Цикл просмотра и анализа символов строки S$ по номеру символа I от 1 до N
8. [Вычисление условий V1 – V3, V
9. CNT = CNT + V]
10. Вывод строки S$ и счетчика символов CNT

Конец

Листинг 8.1

```
# Primer_8_1
# Подготовка экрана
Cls
Input "Введите строку S$ = ", S$
N = Length(S$)
Input "Введите символ для поиска T$ = ", T$
NL = ASC(T$): CNT = 0
# Формирование условий W1 и W2
W1 = (NL >= 65)*(NL <= 90):  W2 = (NL >= 97)*(NL <= 122)
# Цикл просмотра и анализа символов строки S$
For I = 1 To N
T1$ = Mid(S$, I, 1)
V1 = (NOT(W1 + W2))*(T1$ = T$)
V2 = W1*((T1$ = T$) + (T1$ = CHR(NL + 32))
V3 = W2*((T1$ = T$) + (T1$ = CHR(NL – 32))
V = V1 + V2 + V3
CNT = CNT + V
Next I
Print "S$ = " + S$
Print
Print "Число символов " + T$ + "  CNT = " + CNT
End
```

Пример 8.2

С клавиатуры вводится текстовая строка S$ длиной L <= 256 символов. Слова в строке разделены одним или несколькими пробелами. Подсчитать число слов в строке. Словом считать любую комбинацию символов, ограниченную с двух сторон пробелами или пробелом и началом или концом строки.

Анализ задачи

Для построения математической модели задачи необходимы следующие переменные: текстовая строка S$; длина строки L; номер очередного символа строки I (1 <= I <= L); счетчик слов N; условие: очередной символ строки — пробел (`Mid(S$, I, 1) = " "`); условие: очередной символ строки — не пробел (`Mid(S$, I, 1) <> " "`); условие: конец строки не достигнут (I < N).

Программу удобно организовать в виде структуры, состоящей из основной программы `Stroka` и двух подпрограмм: `Probel` и `Slovo`. Программа `Stroka` реализует общий цикл просмотра строки и вызывает подпрограммы. Подпрограмма `Probel` просматривает пробелы между словами. Подпрограмма `Slovo` выполняет просмотр слов в строке и их подсчет.

Алгоритм решения задачи

Алгоритм *Stroka*

Начало

1. Подготовка экрана
2. Инициализация счетчика символов I и счетчика слов N: I = 1; N = 0
3. Ввод текстовой строки S$
4. Вычисление длины строки L
5. Цикл просмотра строки: пока I < L, выполнять
6. [Пропуск пробелов перед словом с помощью алгоритма `Probel`

7. Просмотр символов слова и подсчет слов с помощью алгоритма `Slovo`]
8. Вывод счетчика слов N

Конец

Алгоритм *Probel*

Начало

1. Пока I-й символ пробел и не достигнут конец строки, выполнять
2. [I = I + 1]

Конец

Алгоритм *Slovo*

1. Пока I-й символ не пробел и не достигнут конец строки, выполнять
2. [I = I + 1]
3. N = N + 1

Конец

Листинг 8.2

```
# Primer_8_2
# Stroka
# Подготовка экрана
Graphsize 300, 300
Cls
# Инициализация I, N
I = 1: N =0
# Ввод строки
Input "Введите строку S$ = ", S$
L = Length(S$)
# Цикл просмотра строки
While I < L
Gosub Probel
Gosub Slovo
End While
```

```
Print
Print "Число слов в строке N = " + N
End

Probel:
While (Mid(S$, I, 1) = " ")*(I < L) = 1
I = I + 1
End While
Return

Slovo:
While (Mid(S$,I,1) <> " ")*(I < L) = 1
I = I + 1
End While
N = N + 1
Return
```

Рассмотренный *пример 8.2* кроме своего основного назначения — иллюстрации решения простой задачи обработки текстовой строки — показывает вариант использования и оформления вспомогательных алгоритмов в описании алгоритма и подпрограмм в тексте программы.

ГЛАВА 9

Решение заданий ЕГЭ средствами BASIC-256

В течение последних пяти лет в заданиях части С ЕГЭ по информатике для 11-го класса непосредственно с программированием связаны задания С1, С2, С4. Рассмотрим в качестве примеров задания части С из демо-версии тестов ЕГЭ по информатике 2011 г. [6] — тематика этих заданий не меняется в течение последних 3 лет, меняются только конкретные условия задач.

Задачи С1 посвящены проверки условий попадания произвольной точки, координаты которой вводятся с клавиатуры, в область, ограниченную несколькими кривыми, заданными аналитическими выражениями. Задачи С2 связаны с поиском в массиве элемента (или элементов), удовлетворяющего двум или более условиям, заданным в содержательной форме. Задачи С4 ориентированы на обработку данных, представленных в виде последовательности текстовых строк с дополнительными требованиями по оптимизации программы, например, по затратам оперативной памяти и времени решения. Фактически для этого при решении задачи требуется рационально выбрать структуры данных. Кроме того, придется учесть, что приводимые в условиях задач примеры фрагментов программ на языке BASIC, несмотря на обратные утверждения разработчиков заданий, ориентированы на синтаксис языков семейства QBasic, Quick Basic. Синтаксис BASIC-256 имеет некоторые отличия, которые придется учитывать при написании соответствующих фрагментов программ.

Задача C1-2011D

Требовалось написать программу, при выполнении которой с клавиатуры вводятся координаты точки на плоскости — действительные числа X, Y — и определяется принадлежность этой точки области, ограниченной кривыми Y = X, Y = –X, Y = X*X – 2, включая границы (рис. 9.1). Программист торопился и написал программу неправильно (листинг 9.C1.1).

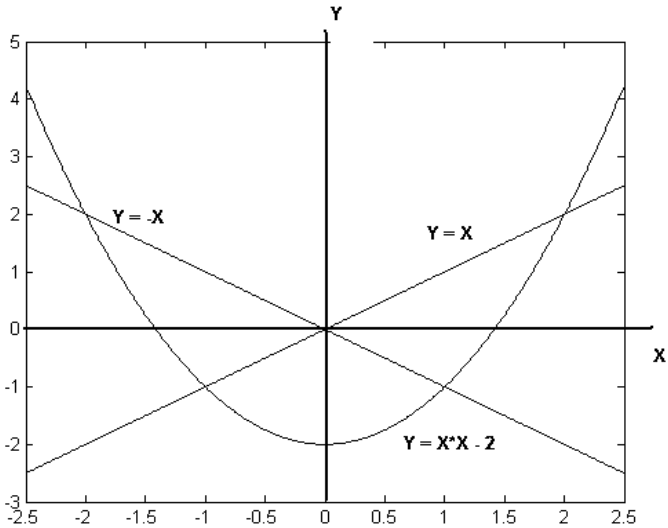

Рис. 9.1. Область, ограниченная кривыми Y = X, Y = –X, Y = X*X – 2

Листинг 9.C1.1. Исходная программа с ошибками

```
Input X: Input Y
If  Y <= X Then
If  Y <= -X Then
If  Y >= X*X - 2 Then
Print "принадлежит"
Else
Print "не принадлежит"
End If
```

```
End If
End If
End
```

Последовательно выполните следующее:

1. Приведите пример таких чисел X, Y, при которых программа неправильно решает поставленную задачу.
2. Укажите, как нужно доработать программу, чтобы не было случаев ее неправильной работы. (Это можно сделать несколькими способами, поэтому укажите любой правильный способ доработки программы.)

Решение

1. Три условия, заданные в программе, при одновременном выполнении (Y <= X, Y <= –X, Y >= X*X – 2) описывают только одну из трех частей области, описанной кривыми, заданными в условии задачи (самую нижнюю). Любая точка, попадающая в две другие части той же области, заданные одновременным выполнением следующих комбинаций условий:

   (–2 <= X <= 0, Y <= –X, Y >= X, Y <= X*X – 2) и

   (0 <= X <= 2, Y <= X, Y >= –X, Y >= X*X – 2)

   не удовлетворяет условиям программы, но удовлетворяет условиям задачи. Примеры точек, попадающих в эти области, — точки на оси X: (–1, 0), (1, 0). Программа их определит как не принадлежащие к заданной области, в то время как они ей принадлежат.
2. Условия принадлежности точки заданной области можно записать, например, следующим образом: |X| <= 2, Y <= |X|, Y >= X*X – 2. При этом доработанная программа примет следующий вид (листинг 9.С1.2).

Листинг 9.С1.2. Доработанная программа

```
Cls
Input X: Input Y
If  Abs(X) <= 2 Then
```

```
If  Y <= Abs(X) Then
If  Y >= X*X - 2 Then
Print "принадлежит"
Else
Print "не принадлежит"
End If
End If
End If
End
```

Задача C2-2011D

Дан целочисленный массив из 30 элементов. Элементы массива могут принимать значения от 0 до 1000. Опишите на русском языке или на одном из языков программирования алгоритм, который позволяет подсчитать и вывести среднее арифметическое элементов массива, имеющих нечетное значение. Гарантируется, что в исходном массиве хотя бы один элемент имеет нечетное значение. Исходные данные объявлены так, как показано в листинге 9.C2.1. Запрещается использовать переменные, не описанные в листинге 9.C2.1, но разрешается не использовать часть из них.

Листинг 9.C2.1. Описание массива

```
N = 30
Dim A(N) As Integer
Dim I, X, Y As Integer
For I = 1 To N
Input A(I)
Next I
```

Решение

Из описания данных в листинге 9.C2.1 для имеющихся переменных выберем следующее назначение: I — индекс элемента в массиве A; X — переменная для накопления суммы нечетных элементов массива A с начальным значением X = 0; Y — число нечетных элементов массива A с начальным значением Y = 0.

Идея алгоритма

Алгоритм предусматривает последовательный просмотр элементов массива с проверкой условия нечетности, например, вида (A(I) % 2 = 1). При обнаружении нечетного элемента A(I) он подсчитывается в переменной Y и добавляется к сумме нечетных элементов X: Y = Y +1, X = X + A(I). Потом вычисляется среднее арифметическое в той же переменной X: X = X/Y.

Фрагмент программы, реализующий описанный алгоритм, с учетом отсутствия в BASIC-256 явного описания целочисленных данных, примет следующий вид (листинг 9.C2.2).

Листинг 9.C2.2. Вычисление среднего арифметического нечетных элементов массива

```
Cls
N = 30
Dim A(N)
X = 0: Y = 0
For I = 0 To 29
Print "Введите целое A(I) для I = " + I
Input A(I)
If A(I)  % 2 = 1 Then
Y = Y + 1
X = X +A(I)
End If
Next I
X = X/Y
Print "Среднее арифметическое нечетных элементов X = " + X
End
```

Задача C4-2011D

На вход программе подается набор символов, заканчивающийся точкой (в программе на языке BASIC символы можно вводить по одному в строке, пока не будет введена точка, или считывать данные из файла). Напишите эффективную, в том числе и по используемой памяти (укажите используемую версию языка программирования, например, Borland Pascal 7.0), которая сначала

будет определять, есть ли в этом наборе символы, соответствующие десятичным цифрам. Если такие символы есть, то можно ли переставить их так, чтобы полученное число было симметричным (читалось одинаково как слева направо, так и справа налево). Ведущих нулей в числе быть не должно, исключение — число 0, запись которого содержит ровно один ноль. Если требуемое число составить невозможно, то программа должна вывести слово "NO". А если возможно, то в первой строке следует вывести слово "YES", а во второй — искомое симметричное число. Если таких чисел несколько, программа должна выводить максимальное из них. Например, пусть на вход подаются символы: `Do not 911 to 09 do`. В данном случае программа должна вывести:

YES
91019

Решение

Для решения задачи понадобятся следующие переменные. X$ — переменная для ввода входной символьной строки. Y(10) — массив из 10 элементов для хранения количества каждой из цифр, обнаруженных в строке X$: Y(0) — число обнаруженных нулей; Y(1) — число обнаруженных единиц; ... Y(9) — число обнаруженных девяток. N1 — общее число найденных в строке X$ цифр. I — индекс элемента в массиве Y.

Формирование симметричного числа возможно при выполнении следующих условий. Число цифр в строке N1 > 0: при четном N1 все Y(I) четные; при нечетном N1 допустимо одно нечетное Y(I); если все цифры — нули, их число может быть любым.

Алгоритм

Начало

1. Подготовка экрана
2. Ввод строки с клавиатуры
3. Инициализация массива Y нулями. Анализ символов строки и выявление цифр. Запись цифр в массив Y. Подсчет числа найденных цифр в N1

4. Анализ массива цифр на возможность формирования симметричного числа: проверка наличия цифр; проверка четности числа цифр. Вывод "NO" в случае невозможности.
5. Формирование симметричного числа максимальной величины и вывод результата.
6. Вывод результата.

Конец

Листинг 9.С4.1

```
Cls
Input "Введите строку с клавиатуры", X$
Rem Блок 3
Rem Инициализация массива Y нулями
For I = 0 To 9
Y(I) = 0
Next I
N1 = 0
Rem Анализ N-го символа строки X$ на соответствие цифре от 0_
до 9
Rem Подсчет общего количества найденных цифр N1
N = 1
While  Mid(X$, N, 1) <> "."
For I = 0 To 9
If Mid(X$, N, 1) = String(I) Then
Y(I) = Y(I) + 1
N1 = N1 + 1
End If
Next I
N = N + 1
End While
Rem Блок 4
Rem P1 = 1 – признак отсутствия цифр в строке.
Rem P2 = 0 при N1>0 – признак того, что все Y(I) –
 четные, I = 0, 1,..., 9
Rem P2 = 1 – только одно число из Y(I) – нечетное
Rem P2 > 1 – два или более из Y(I) – нечетные
I1 – номер последнего  нечетного Y(I)
P1 = 1: P2 = 0
```

```
R$ = ""
For I = 0 To 9
P1 = P1*(Y(I) = 0)
P2 = P2 + (Y(I) mod 2)
IF (Y(I) mod 2 = 1) Then I1 = I
Next I
Rem Проверка невозможности формирования симметричного числа
IF ((P1 = 1) OR (P2 > 1)) Then R1$ = "NO" Else R1$ = "YES"
Rem Блок 5
Rem Проверка на 0
If ((N1 > 0)*(N1 = Y(0)) = 1) Then R$ = "0"
Rem Формирование симметричного числа максимальной величины
If P2 <= 1 Then
Rem Формирование старшей части числа
For I = 9 To 0 Step -1
K = Y(I)\2
If ((K > 0)*((Y(I) mod 2) = 0) = 1 Then
   For K1 = 1 To K
   R$ = R$ + String(I)
   Next K1
End If
Next I
Rem Формирование средней части числа
     If P2 = 1 Then
     For K1 = 1 To Y(I1)
     R$ = R$ + String(I1)
     Next K1
     End If
Rem Формирование младшей части числа
For I = 0 To 9
K = Y(I)\2
If ((K > 0)*((Y(I) mod 2) = 0) = 1 Then
   For K1 = 1 To K
   R$ = R$ + String(I)
   Next K1
End If
Next I
End If
End
```

ГЛАВА 10

Задачник по программированию

Изучение раздела "Элементы программирования" ряд авторов учебников по информатике и ИКТ рекомендуют вывести из базовых курсов в профильные. Остается надеяться, что приведенный здесь материал будет полезен учителям и учащимся, независимо от того, в какой части курса его изучать.

Материал главы условно можно разделить на две части:

- ❒ освоение основных операторов языка, типов данных и базовых алгоритмических конструкций;
- ❒ изучение способов решения типовых школьных задач (поиск данных, сортировка данных, формирование подвижных и неподвижных изображений в текстовом и графическом режимах экрана, построение графиков математических функций).

Учителя могут использовать материалы каждой темы для проведения зачетных работ и проектов по индивидуальным заданиям. Учащимся материал будет полезен для тренировки и подготовки к зачетам и экзаменам.

При ссылках на задачи главы будут использоваться обозначения вида 10.1.1-3: задание 10.1.1, задача 3.

10.1. Освоение операторов языка и типов данных BASIC-256

Каждое задание *раздела 10.1* содержит 4 задачи и рассчитано на два академических часа. *Задача 1* имеет целью проверку знаний

типов данных и правил записи выражений в BASIC-256. При ее решении рекомендуется для каждой ошибочной записи числа указать вид ошибки, например: недопустимый символ в записи числа; неподдерживаемая форма записи числа. *Задача 2* ориентирована на запись программы вычисления сложного выражения. Исходное выражение записано в обычной математической форме, как в задачнике по алгебре. Проверяется знание правил записи математических выражений в BASIC-256 и умение правильно записать последовательность строк программы. Ввод и отладка программы, получение конкретного числового результата не предполагается. *Задача 3* ориентирована на правила записи выражений для формирования случайных чисел с помощью генератора случайных чисел с получением результата. *Задача 4* ориентирована на операции с символьными строками с получением результата. Все задачи содержательные.

Рекомендуется по этим заданиям проводить не одну, а две зачетные работы по одному часу. В первую из них включить задачи 1 и 2. Во вторую — задачи 3 и 4. Далее рассмотрены примеры решения типового задания *раздела 10.1* (задачи 10.1.П-1–10.1.П-4).

Задача 10.1.П-1

Укажите допустимые данные и выражения из следующего списка: 2.7812; π; ctg 45°; tg 0.51; 3PI/4; ctg PI/4; SQR 25; 25^(1/2); SQR("Петрова"); MID$(25346,3,3). Для недопустимых данных и функций укажите, как их правильно задать.

Решение

Решение первых задач каждого задания удобно представить в виде таблицы следующего вида (табл. 10.1).

Таблица 10.1. Решение задачи 10.1.П-1

Исходное выражение	Верно/Неверно	Правильная запись или комментарий
2.7812	Верно	Действительное число
π	Неверно	Константа pi или PI
Ctg 45°	Неверно	1/tan(45*pi/180) или 1/tan(radians(45))

Таблица 10.1 *(окончание)*

Исходное выражение	Верно/Неверно	Правильная запись или комментарий
Tg 0.51	Неверно	Tan(0.51)
3PI/4	Неверно	3*PI/4
Ctg PI/4	Неверно	1/tan(pi/4)
SQR 25	Неверно	Такой числовой функции нет
25^(1/2)	Верно	$\sqrt{25} = 5$
SQR("Петрова")	Неверно	Такой функции для обработки строк нет
MID$(25346,3,3)	Неверно	MID("25346", 3, 3) = "346"

Задача 10.1.П-2

Вычислить выражение при X = 2.5, Y = 1.2:

$$Z = \frac{\operatorname{tg}^2 \dfrac{X}{10} + \sqrt[3]{\sin^2(Y/10)}}{\dfrac{X^2 + Y^3}{\sin 57^\circ + \operatorname{tg} 0.5 + 3.47}}.$$

Решение

Исходное выражение записано в математической форме. Необходимо записать последовательность операторов для вычисления выражения в соответствии с правилами записи выражений в алгоритмическом языке. Общий алгоритм решения таких задач запишется в следующем виде.

1. Подготовка экрана.
2. Присвоение значений переменным X, Y.
3. Выбор и вычисление вспомогательных переменных: Z1 — выражение для числителя; Z2 и Z3 — выражения для знаменателей.
4. Вычисление Z.

Соответствующий алгоритму фрагмент программы приводится в листинге 10.1.П-2.

Листинг 10.1.П-2

```
Cls
X = 2.5: Y = 1.2
Z1 = tan(X/10)^2 + sin(Y/10)^(2/3)
Z2 = X^2 + Y^2
Z3 = sin(57*PI/180) + tan(0.5) + 3.47
Z = Z1/(Z2*Z3)
```

Пример записи выражения с условиями рассмотрен в разд. 2.3.1, выражение (2.6).

Задача 10.1.П-3

В подъезде имеются квартиры с номерами от 235 до 270. Подошедший гость забыл номер квартиры своих друзей — 261. Он делает наугад 5 звонков через домофон в квартиры со случайными номерами. Позвонит ли он в нужную квартиру?

Решение

Необходимо сформировать номер квартиры N как случайное целое число в диапазоне 235–270 и вывести первые 5 чисел. В соответствии с соотношением (2.3) случайное целое число в диапазоне от N1 до N2 формируется с помощью выражения:

```
N = N1 + Int((N2 – N1 +1)*rand).
```

В задаче: `N1 = 235`; `N2 = 270`; `N = 235 + Int(36*rand)`. Первые 5 случайных номеров квартир по этой модели: 257, 246, 259, 262, 265. Нужный номер квартиры — 261 — в этот список не попал. Фрагмент программы для формирования этого списка имеет следующий вид (листинг 10.1.П-3).

Листинг 10.1.П-3

```
cls
for i = 1 to 5
N = 235 + Int(36*rand)
print N
next i
```

Задача 10.1.П-4

Клиент электронной почты Дягилев Василий Дмитриевич решил выбрать себе в качестве ника (псевдонима) для адреса комбинацию символов, содержащую последовательность из сочетаний первых двух символов своих фамилии, имени и отчества в записи латинскими буквами. Записать выражение для формирования ника в виде переменной `Nick$`.

Решение

Введем в рассмотрение переменные `X$ = "Dyagilev"`, `Y$ = "Vasily"`, `Z$ = "Dmitrievich"`. Тогда необходимый ник можно сформировать, например, с помощью следующего выражения: `Nick$ = Left(X$, 2) + Left(Y$, 2) + Left(Z$, 2)`. Тот же результат **DyVaDm** можно получить с использованием функции `Mid`:

```
Nick$ = Mid(X$, 1, 2) + Mid(Y$, 1, 2) + Mid(Z$, 1, 2)
```

Задание 10.1.1

1. В следующих примерах найдите ошибки в записи числовых данных: +35; –47#; 3.71E–; 0.11E2; 28E–1; 727,379; E–2 3.71; 2.71E-4#; –1.37E–57; 2.128E150; .00003578. Для правильно записанных чисел напишите `Верно`. Для ошибочных записей укажите `Неверно` и поясните почему.
2. Вычислить выражение: $Z = X^2 + Y^2$ для $X >= 2Y$; $Z = 2XY$ для $X < 2Y$ при $X = 6$, $Y = 3$.
3. Стороны прямоугольника A и B задаются целыми случайными числами следующего вида: $10 <= A <= 20$; $20 <= B <= 30$. Вычислить площадь прямоугольника S.
4. Вычислить выражение `MID(X$,9,5) + " в " + MID(X$,23,6)` для `X$ = "Средняя школа No 53 г. Рязани"`.

Задание 10.1.2

1. Можно ли в BASIC-256 точно задать следующие числа в естественной форме: 357; $–3.91*10^5$; 291.35947; –8900.23475;

112000000; 0.27311869; 2^{24}; 10^{7}; $0.734*10^{-5}$; $-4.37*10^{-6}$; $-16*2^{16}$?

Дать полный ответ: `Можно` или `Нельзя`. В случае `Нельзя` указать, почему, и в каком виде числа будут восприняты при вводе.

2. Вычислить следующее выражение при X = 10, Y = 3:

$$Z = \frac{\sin 37° + \cos 73° + \mathrm{tg}23°}{X^2 - 2XY^2 + Y^4}.$$

3. Вычислить площадь треугольника S со случайными целыми значениями основания B и высоты H, причем: 40<=B<=80; 30<=H<=60.

4. Вычислить выражение:

$$R = \frac{ASC("Смирнова") + ASC("Юдина") + ASC("Енгалычев")}{\sin 45° + \cos 60° + \mathrm{ctg}23°}.$$

Задание 10.1.3

1. Можно ли преобразовать в целые числа следующие величины: 271E3; –3.1E–3; –9.5E5; 2.11E3; 5.36E4; 6.311E–10? Если можно, укажите, в каком виде число воспримет BASIC-256. Если нельзя, запишите число в естественной форме и укажите, почему.

2. Вычислить выражение при X = 1, Y = 2:

$$Z = \frac{X^2 + 2X\sin 2X\cos 2X - 3X\mathrm{tg}X}{Y^4 - 3X^2Y^2 + Y^4}.$$

3. Два ученика A и B по очереди бросают кубик с числом очков от 1 до 6. Кто из них наберет больше очков, если они трижды бросают кубик по очереди (бросок A, бросок B)?

4. Вычислить выражение вида `CHR(N1) + CHR(N2) + CHR(N3) + CHR(N4) + CHR(N5)`, где N1–N5 — последовательность случайных кодов в диапазоне от 65 до 90.

Задание 10.1.4

1. Запишите код своего почтового отделения, номера своего домашнего и мобильного телефонов, домашние и мобильные телефоны двух своих друзей как числа в естественной и экспоненциальной формах. Воспримет ли эти данные BASIC-256 как числа? Дать полный ответ.
2. Вычислить выражение:

$$Z = \frac{(\mathrm{INT}(X^2) + \mathrm{INT}(Y^2))^2}{\cos\pi\frac{X}{Y} + \frac{Y}{X}} \text{ для } X = 2.5,\ Y = 1.5.$$

3. Вы набираете "наудачу" два номера телефона в виде случайной последовательности из 6 цифр каждый. Какие номера Вы наберете?
4. Вычислить выражение вида: `"Десятые классы – " + CHR$(240) + CHR(200) + CHR(220).`

Задание 10.1.5

1. Запишите номер Вашего дома, квартиры, домашнего и мобильного телефонов во всех известных Вам формах записи чисел.
2. Вычислить выражение $Z = \dfrac{\mathrm{tg}\frac{\pi}{2}X + \mathrm{tg}\frac{\pi}{2}Y}{X^2 + Y^2}$ для $X = 0.5,\ Y = 0.8$.
3. В колоде 36 карт — от 6 до туза. Картам от валета до туза присвоены числа очков от 11 до 14. Двое играющих А и Б по очереди три раза берут по одной карте. Сколько очков наберет каждый из них.
4. Вычислить выражение `X$ + Y$ + Z$`, если `X$, Y$, Z$` — Ваши имя, отчество и фамилия.

Задание 10.1.6

1. Запишите во всех формах представления чисел коды букв Вашего имени.

2. Вычислить выражение целых X:

   $Z = X^3 + 3X^2 - 4X + 7$ для четных X;

   $Z = X^2 + 7X + 3$ для нечетных X.

3. Старый номер автомобиля содержит 4 цифры. Сравните результаты формирования этого номера двумя способами: а) как случайного целого числа от 1 до 9999; б) как последовательности из 4 случайных цифр от 0 до 9.

4. Запишите Вашу фамилию через операцию склеивания символов, сформированных по кодам букв.

Задание 10.1.7

1. Запишите во всех известных Вам формах представления чисел номер Вашей школы, класс и номера домашнего мобильного телефонов. Буква в обозначении класса трактуется как цифра в соответствии с алфавитом: класс 10Б имеет номер 102.

2. Вычислить выражение

$$Z = \frac{\sqrt[3]{(X+Y)^2 + (X-2Y)^2}}{\sin 23^\circ + \mathrm{tg}23^\circ} \text{ для } X = 10,\ Y = 3.$$

3. В группе 25 учащихся. Преподаватель вызывает их отвечать случайным образом по номеру в списке в журнале. Если планируется вызвать пять учащихся, кого вызовут четвертым?

4. Что получится в результате исполнения следующей последовательности команд:

```
X$="Internet": MID(X$,1,3) + "ra" + MID(X$,6,3)
```

Задание 10.1.8

1. Запишите сегодняшнюю дату (число, месяц и год) во всех известных Вам формах представления чисел.

2. Вычислить выражение для X = 3.5, Y = 2.5:

$$Z = \frac{X^{3/2} + Y^{2/3}}{\sin\frac{X}{Y} + \mathrm{tg}\frac{Y}{X}}.$$

3. В группе 20 учащихся. Преподаватель вызывает их отвечать случайным образом по номеру в списке с конца. Если планируется вызвать четверых учащихся, будет ли вызван учащийся с номером 19?
4. Присвойте переменной `Y$` какую-нибудь фамилию из учащихся Вашей группы. Как определить длину фамилии и код ее третьего символа, если фамилия может быть произвольной?

Задание 10.1.9

1. Запишите размер Вашего головного убора, обуви и плаща во всех известных Вам формах записи чисел.
2. Вычислить выражение при X = 3.12, Y = 1.27:

$$Z = \frac{\sin X^{3/2} + \cos Y^{2/3}}{\sqrt[4]{XY^2 + X^2Y}}.$$

3. Сформировать последовательность из 6 символов со случайными кодами в диапазоне от 224 до 253 и присвоить полученную строку переменной `X$`.
4. Присвойте переменным `X$`, `Y$`, `Z$` свои фамилию, имя и отчество. Каков смысл и что получится в результате исполнения следующей команды: `(LEN(X$) – LEN(Y$))/LEN(Z$)`?

Задание 10.1.10

1. Укажите, какие из перечисленных операций допустимы? Для допустимых операций укажите смысл. Для недопустимых по Вашему мнению операций укажите причины.

```
X$ = 5;
Z = X$ + Y$;
Y = 4.75E-12;
Y = -5.746;
V$ = Смирнова;
print MID$("Татьяна", 5,3);
Z = X\Y.
```

2. Вычислить выражение при X = 4.51, Y = 2.73:

$$Z = \frac{\sin\sqrt[3]{X^2 + Y^2}}{(2X + 3Y - 4)(4X - Y)^2}.$$

3. Социологи проводят опрос жителей Вашего дома. Они случайным образом выбирают 5 номеров квартир из всего множества. Будет ли выбрана для опроса Ваша квартира?

4. Присвойте переменным `X$`, `Y$` и `Z$` соответственно Ваши фамилию, имя и отчество. Придумайте операцию, которая выделит и выведет на экран Ваши инициалы, разделенные точками. Например, для учащегося Енгалычева Павла Степановича на экран следует вывести Е.П.С.

Задание 10.1.11

1. Допустимы ли операции вида:

```
Z = "Аверьянов"+5 + MID$("Ефимова",4,4) + ".";
Y$ = MID$("Яковлев",5,3) + ", T" + MID$("Заигров",3,3)?
```

   Для допустимых операций приведите результат, для недопустимых укажите причину.

2. Вычислить выражение при X = 1.63, Y = –0.56:

$$Z = \frac{\sqrt[3]{(5X + 3Y)^5 - (6X - 1.5Y - 1)^3}}{\sin 23^\circ + \cos 31^\circ - \mathrm{tg}1}.$$

3. По некоторой улице идет прохожий. Он заходит в дома со случайными номерами на четной стороне улицы и спрашивает интересующих его людей. Будем считать, что на улице 42 дома, и идет он с начала улицы. В какие первые пять домов он зайдет?

4. Переменной `X$` присваивается произвольное слово из 4 букв. Запишите операцию, которая выводит на экран слово, сформированное из исходного путем перестановки букв в обратном порядке.

Задание 10.1.12

1. Допустимы ли следующие операции?

```
Z1 = SQR("Татьяна Феоктистова" + 4.576 – X);
Z2 = 1.576 + X + 46# + 1@.392\56%;
Z3$ = MID$("Николай Сличенко",8,7) + MID$(" Олег
                                        Газманов",1,5).
```

2. Для допустимых операций укажите результат, для недопустимых — причину.
3. Вычислить выражение при X = 1.21, Y = 2.12:

$$Z = \frac{\text{Lehgth("1234")} + \sin 17^\circ + \left(\frac{X}{Y}\right)^2}{(23X - 6Y + Y^2)\text{tg}^2 57^\circ}.$$

4. Вы решили нанести неожиданные визиты в следующем месяце своим знакомым. Числа месяца Вы решили выбрать случайным образом. В какие числа Вы нанесете первые три визита?
5. Присвойте переменной `X$` строку, содержащую Ваши фамилию, имя и отчество. Запишите операцию, которая преобразует `X$` так, чтобы получилась строка, содержащая имя, отчество и фамилию.

Задание 10.1.13

1. Поясните, каким будет результат исполнения следующей последовательности команд: `X$ = "Надежда Васильевна Ефимова"; print; print MID(X$,20,7): print MID(X$,1,7).`
2. Вычислить выражение при X = 1.71, Y = 0.57:

$$Z = \frac{\sin^2(X + Y) + \cos^2(X - Y)}{\sqrt[3]{X^2 + Y^2 - 3XY}}.$$

3. Представитель некоей партии во время предвыборной кампании заносит листовки в дома по случайным зависимостям следующим образом. Сначала он случайным образом выбирает сторону улицы: четную или нечетную. Затем дом на выбран-

ной стороне улицы — по порядку от начала улицы. И затем — квартиру в доме. Считаем условно, что на улице 60 домов и все они по 90 квартир. По какому адресу зайдет агитатор в первый раз?

4. Пусть `X1$ = "Алла Пирогова", X2$ = "Маша Васина", X3$ = "Людмила Зинченко", X4$ = "Кристина Ормондт"`. Каким должен быть результат операции `LENGTH(X1$ + X2$ +X3$ + X4$)`?

Задание 10.1.14

1. Можно ли в BASIC-256 выполнить следующие операции?

```
X = 5.74\41.32; Y$ = "Коробецкая Надежда Андреевна";
Z$ = "56.78";Y = 43.7 % 3.6;
X1 = 23# + 56% + 67@.
```

2. Для допустимых операций приведите результат, для недопустимых — причины.
3. Вычислить выражение при X = 1.11, Y = 0.53:

$$Z = \frac{\operatorname{tg}\dfrac{X-Y}{X+Y} + \sin^2 2X + \cos^2 Y}{\dfrac{X+Y}{X-Y} + \operatorname{tg}45^\circ}.$$

4. Для шифровки сообщения случайным образом выбирается страница шифровальной книги (от 3 до 250), строка на странице (от 1 до 43) и символ в строке (от 1 до 64). Укажите, какими будут координаты первых двух заменяющих символов.
5. Каким будет результат следующей последовательности операций? `X$ = "Татьяна"; Y$ = "Светлана"; Z$ = "Михаил"; G$ = MID(X$ + Y$ + Z$,8,8) + "+" + MID(X$+Y$+Z$,16,6) + "="`

Задание 10.1.15

1. Для допустимых из приведенных операций укажите результаты, для недопустимых — причины:

```
print X$ + Y$ – Z$;
print (LEN(X$) + LEN(Y$) – LEN(Z$));
```

```
Z$ = MID(X$,3,3) + MID(Y$,4,2);
print (LEN(X$) + 57 – X1)\Y1.
```

2. Вычислить выражение при X = 10, Y = 3.5:

$$Z = \frac{\sqrt{(X+Y)^3 - (X-2Y)^2}}{\sin 57° + \mathrm{tg}0.5 + 3.47}.$$

3. Социологи проводят телефонный опрос жителей микрорайона по случайно выбранным шестизначным номерам телефонов, начинающихся с цифр 35. Какими будут первые 5 телефонов? Позвонят ли они Вам в числе первых пяти?

4. Пусть `X$ = "ЗАРИТОВСКИЙ"`. Придумайте операцию, позволяющую преобразовать X$ так, чтобы получилась фамилия СМИРНОВА. Недостающие буквы можно добавлять в виде подстрок.

Задание 10.1.16

1. Укажите допустимые числовые данные и выражения из следующего списка. Для недопустимых данных и функций укажите, как их правильно задать:

3.14159;	ctg PI/4;
π;	SQR 25;
SIN 46°;	25^(1/2);
tg 0.75;	SQR("Петрова");
3PI/4;	MID$(25346,3,3).

2. Вычислить выражение при X = 2, Y = 1.25:

$$Z = \frac{\sin 3\dfrac{\pi X}{Y} + \mathrm{ctg}\dfrac{\pi Y}{2X}}{\sqrt[3]{X^3 + Y^3 - 3X^2Y + 4Y}}.$$

3. Социологи при опросе формируют случайный 6-значный телефонный номер для АТС 34 в виде последовательности из 4 случайных чисел от 0 до 9. Какими будут первые три номера?

4. Присвойте переменным X$, Y$ и Z$ фамилию, имя и отчество одного из Ваших преподавателей. Каким будет результат выполнения операции вида:

```
Print MID$(X$,1,1) + MID$(Y$,1,1) + MID$(Z$,1,1)?
```

Задание 10.1.17

1. Пусть переменные X, Y, Z — целого типа. Укажите, какие из перечисленных операций допустимы и как их правильно записать? Для недопустимых укажите причины или тип результата:

```
Z = X2 + Y2 – 3XY +70;
print: print: print X\25;
Y MOD 5;
print 150/25;
56*43;
(X – Y + 2XY)/(X + Y)2; (X + Y) MOD 7.
```

2. Вычислить выражение при X = 0.5, Y = 0.87:

$$Z = \frac{\sqrt{\left(\sin\frac{\pi X}{2} + \cos\frac{\pi X}{2}\right)^{3/2} + \mathrm{tg}^2 35^\circ}}{(X+Y)^{2/3} + 5X + 4Y + 3}.$$

3. Вы играете в кости с приятелем (бросаете 2 кубика с числом очков от 1 до 6). Ваш бросок — первый. Приятель — второй. Кто из вас выиграет после партии из 3 бросков?

4. Укажите результат следующей последовательности действий:

```
X$ = "Вячеслав Николаевич Любимов";
print MID(X$,21,7) + " " + MID(X$,1,8) + " " + MID(X$,10,10).
```

Задание 10.1.18

1. Укажите верные и неверные типы данных и функции из приведенного ниже списка. Для верных укажите тип, для неверных — причину и правильную запись:

1.234E+03;	1999;
24&$47.97E–03;	"2003";
"Школа 53";	"####.###";
Класс 11-Г;	1.762E+40;
12.759E–05;	234931.

2. Вычислить выражение при X = 10, Y = 3:

$$Z = \frac{\sqrt[3]{(X+Y)^2 + (10Y - 2X)^{3/2}}}{\sin 23^\circ + \mathrm{tg}43^\circ + \mathrm{ctg}1}.$$

3. Два игрока A и B играют в "очко" обычной колодой карт. Вначале вытаскивает 3 карты игрок A, затем 3 карты игрок B. Кто выиграет?

4. Присвойте переменным `X$`, `Y$`, `Z$` фамилию, имя и отчество одного из Ваших учителей и укажите результат исполнения последовательности команд вида:

```
cls; print; print; print X$ + " " +Y$ + " " + Z$.
```

Задание 10.1.19

1. Укажите верные и неверные типы данных и функции из приведенного ниже списка. Для верных укажите тип, для неверных — причину и правильную запись:

"@@@@@.@@@";	–267@32;
*^$#@@@**;	"368.32.27.185";
–123.567;	Nord@262.170.0.49;
314.47E–03;	4.7398E–35;
@145.367;	"43.9478".

2. Вычислить выражение при X = 7, Y = 2:

   $Z = X^2 + Y^2 + 3XY$ для X <= 1.57Y;

   $Z = 2XY - 3Y - 2X$ для X > 1.57Y.

3. Сторона равностороннего треугольника A задается целым случайным числом следующего вида: 20 <= A <= 30. Вычислить площадь треугольника S.

4. Пусть `X$` — строка, содержащая Ваши фамилию, имя и отчество. Вычислить выражение вида:

```
MID(X$,5,5) + " в " + MID(X$,10,4) + MID(X$,15,5).
```

Задание 10.1.20

1. Укажите верные и неверные типы данных и функции из приведенного ниже списка. Для верных укажите тип, для неверных — причину и правильную запись:

πX/2;	1.54E04;
sin 23°;	ctg 78°;
tg 1.2;	–49.67@#9;
sin(X)*cos(2*X);	175.0;
$Y = X^2 + 4X + 12$;	1.752*10^3.

2. Вычислить выражение при X = 1, Y = 0.7:

$$Z = \frac{X^2 tgX + 2X \sin 3X \cos \pi X / 4 - 3XtgX}{Y^4 - 3X^2 + 2Y^2 + Y^4}.$$

3. Два ученика A и B играют в кости — по очереди бросают два кубика с числом очков от 1 до 6 каждый. Кто из них наберет больше очков, если они трижды бросают кости по очереди (бросок A, бросок B).

4. Вычислить выражение вида:

```
CHR(N1) + CHR(N2) + CHR(N3) + CHR(N4) + CHR(N5),
```

где N1–N5 — последовательность случайных кодов в диапазоне от 97 до 122.

Задание 10.1.21

1. Запишите номера домов, квартир, телефонов своего и двоих Ваших одноклассников во всех известных Вам формах записи чисел.

2. Вычислить выражение при X = 0.3, Y = 0.4:

$$Z = \frac{\operatorname{ctg}\frac{\pi}{2}X + \operatorname{ctg}\frac{\pi}{2}Y}{X^2Y + XY^2}.$$

3. В колоде 36 карт от 6 до туза. Картам от валета до туза присвоены числа очков от 11 до 14. Двое играющих A и B берут по три карты подряд каждый. Сколько очков наберет каждый из них.

4. Вычислить выражение `Z$ = Left(X$ + Y$ + Z$, 10)`, если `X$`, `Y$`, `Z$` — Ваш адрес: улица, номер дома и квартиры.

Задание 10.1.22

1. Запишите во всех известных Вам формах представления чисел даты рождения свою и своих родителей: число, месяц и год.

2. Вычислить выражение: $Z = X^4 + 2X^3 - 5X^2 + 7X - 10$ для целых X; $Z = X^3 + 7X^2 - 8X + 12$ для дробных и смешанных X. Проверьте при X = 5 и при X = 3.25.

3. ГАИ проверяет автомобили на трассе, формируя номер как последовательность трех случайных чисел от 0 до 9. Какие первые три автомобиля проверит ГАИ.

4. Запишите Ваше имя через операцию склеивания символов, т. е. букв имени.

Задание 10.1.23

1. Запишите во всех известных формах представления чисел свои год, число и месяц рождения, номер дома, номер квартиры и номера домашнего и мобильного телефонов.

2. Вычислить выражение для X = 7, Y = 2.56:

$$Z = \frac{\sqrt{(2X+3Y)^3 + 12Xtg40^\circ}}{\sin\frac{\pi X}{6} + ctg40^\circ}.$$

3. В группе 25 учащихся. Преподаватель вызывает их отвечать по случайному номеру с конца списка. При этом по случайному номеру 1 вызывается 25-й в списке. Кого вызовут в числе первых пяти?

4. Что получится в результате исполнения следующей последовательности команд?

```
X$="World Wide Web";
print: print MID(X$,1,1) + "W" + MID(X$,7,1).
```

Задание 10.1.24

1. Укажите, какие из перечисленных операций допустимы? Для недопустимых по Вашему мнению операций укажите причины и правильную запись.

```
X$ = "25";
Z$ = X + Y - Z;
Y = "4.213E10";
Y% = 57;
V$ = @Vassya@;
Z% = X%\Y%;
print MID(X$, 5,3), где X$ — Ваша фамилия;
X# = Y$/Z;
Z = Intranet;
Y$ = "Ryazan".
```

2. Вычислить выражение для X = 3, Y = 1.5:

$$Z = \frac{\sin X^{3/5} + tgY^{2/7} - \cos 15^\circ}{(2XY + 3Y^2 - 4X)XY^3}.$$

3. Социологи проводят опрос жителей Вашего дома. Они случайным образом выбирают 5 номеров квартир из всего множества. В какие квартиры они пойдут?

4. Присвойте переменной `X$` фамилию, имя и отчество любого преподавателя. Придумайте операцию, которая выделит и выведет на экран его инициалы, разделенные точками. Например, для преподавателя Игоря Валерьевича Каклюгина на экран следует вывести И.В.К.

Задание 10.1.25

1. Допустимы ли операции вида:

```
Z$ = "Домогаров" + "5" + MID("Ефимова",4,4) + ".";
Y$ = MID$("Яковлев",5,3) + ", T" + MID$("Заигров",3,3);
Z = (X +Y$ – 7)/(23 + XY – #)?
```

   Для допустимых операций приведите результат, для недопустимых укажите причину.

2. Вычислить выражение при X = 2, Y = 1.5:

$$Z = \frac{\sqrt[5]{(3+X+4Y)^{2/3} - (6-X+2Y)^2}}{\sin\frac{2\pi X}{10} + \cos 21^\circ - \mathrm{ctg}0.5}.$$

3. По улице идет агитатор. Он заходит в дома со случайными номерами на нечетной стороне улицы в квартиры со случайными четными номерами. Будем считать, что на улице 50 домов и в каждом доме 100 квартир. По каким первым пяти адресам он зайдет?

4. Переменной `X$` присваивается произвольное слово из 4 букв. Запишите операцию, которая выводит на экран строку, сформированную из исходного слова путем вставки пробелов между всеми буквами. Например, из слова СЛОН должна получиться строка С Л О Н.

Задание 10.1.26

1. Поясните, каким будет результат исполнения следующей последовательности команд:

```
X$ = "Beresneva Olga Gordeevna";

print: print MID(X$,11,4);

print MID(X$,16,9).
```

2. Вычислить выражение для X = 1.2, Y = 0.4:

$$Z = \frac{\mathrm{tg}^2(X+2Y)+\mathrm{ctg}^2(X-2Y)}{\sqrt[4]{X^3+6XY-2Y^2+5.74}}.$$

3. Представитель одной из партий во время предвыборной кампании заносит листовки по случайно выбранным адресам следующим образом. Сначала он случайным образом выбирает дом, затем — квартиру в доме. Считаем условно, что на улице 60 домов и все они по 90 квартир. По каким первым трем адресам зайдет агитатор?

4. Пусть `X1$ = "Алла Рогачева"`, `X2$ = "Маша Ракитина"`, `X3$ = "Людмила Турченкова"`, `X4$ = "Кристина Ормондт"`. Допустима ли следующая операция? Если допустима, то каков ее результат?

```
Y = INT((LENGTH(X1$) + LENGTH(X2$) + LENGTH(X3$) +
LENGTH(X4$))^(1/2))
```

Задание 10.1.27

1. Укажите допустимые данные и функции из следующего списка. Для недопустимых данных и функций укажите причины и как их правильно задать, если это возможно.

1.25@3E–2;	SQRT 100;
2π; cos 46°;	25^(1/3);
ctg 0.75;	(Кондаков)^(1/2);
3PI/4;	MID("25346",3,3).
tg PIX/4;	

2. Вычислить выражение при X = 1.75, Y = –1.23:

$$Z = \frac{\operatorname{tg}^2\frac{\pi X}{3} + \cos^3\frac{\pi Y}{5}}{\sqrt[3]{X^{3/2} + Y^{2/3} - 2XY}}.$$

3. Инспектор управления образования случайным образом выбирает по одному из каждой параллели классов с 8-го по 11-й для проведения контрольных работ. В школе 5 одиннадцатых, по 4 десятых и девятых, 3 восьмых класса. Классы в каждой параллели в школе, как обычно, обозначены буквами А, Б, В, ... То есть 11-е классы обозначены как 11А — 11Д. Какие классы выберет инспектор в каждой из параллелей?

4. Присвойте переменным `X$`, `Y$` и `Z$` названия Вашего города, района, улицы. Каким будет результат вычисления выражения:

```
R$ = Left(X$, 3) + " " + Right(Y$, 3) + " " + Mid(Z$, 2,1)?
```

Задание 10.1.28

1. Запишите во всех известных Вам формах записи чисел номера автобусов, троллейбусов, трамваев и маршрутных такси, останавливающихся на ближайшей к Вашему дому остановке транспорта.

2. Вычислить выражение при Y = 0.75, X = 1.2:

$$Z = \frac{\operatorname{tg}\frac{2\pi X}{7} + \cos\frac{3\pi Y}{2X}}{\sqrt[5]{X^5 + Y^5 - 3X^3Y^2 + 5XY}}.$$

3. Социолог при опросе формирует случайный 6-значный телефонный номер для АТС 36 в виде последовательности из двух случайных целых чисел от 00 до 99. По каким трем первым номерам он позвонит?

4. Присвойте переменным `X$`, `Y$` и `Z$` фамилию, имя и отчество одного из Ваших родителей. Каким будет результат выполнения операции вида:

```
print MID(X$,1,1) +"." + MID(Y$,1,1) + "." + MID(Z$,1,1) +"."?
```

Задание 10.1.29

1. Для следующих данных и выражений указать: для правильных по Вашему мнению — результат; для неправильных записей — вид ошибки и способ правильной записи.

   X = "Васильев";

   Y = 25E3;

   X1$ = "Internet";

   X2$ = "5643";

   X3$ = 56 + " " + "Ivanov";

   X7 = –1.783;

   $Z = X^2 + Y^{2/3}$;

   Z1 = "34 + Str".

2. Вычислить выражение при Y = 0.85, X = 0.56:

$$Z = \frac{\sin^2 \frac{\pi X}{3} + \cos^3 \frac{\pi Y}{5}}{(X+Y)^{2/3} + (X-Y)^{3/2}}.$$

3. Три игрока I1, I2, I3 вытаскивают из колоды на 36 карт по очереди по одной карте. Картам от валета до туза без учета масти присвоены числа от 11 до 14. Кто из игроков выиграет после того, как каждый из них возьмет по 3 карты?

4. Присвойте переменным `X$`, `Y$`, `Z$` названия городов Вашей области. Вычислите выражение вида:

   `R$ = Left(X$, 3) + " " + Mid(Y$, 2, 2) + " " + Right(Z$, 3) + ".".`

Задание 10.1.30

1. Укажите допустимые данные и выражения из следующего списка. Для недопустимых данных и выражений укажите причины и как их правильно задать, если это возможно.

   X = Moskva;

   Y$ = 5.76;

X1 = sin 37°;

X2$ = 57.843 + " " + String(87.976) + ".";

X3 = 57.624 + Float("235.14") – Int(–43.076);

$X4 = \sin^2(2X + \pi/6) + tg^{2/3}(X — 2\pi/7)$.

2. Вычислить выражение при X = 0.9, Y = 2.5:

$$Z = \frac{X^{2/3} + \left(\frac{Y}{X}\right)^{3/2}}{\sqrt{\sin^2 Y + ctg^3(Y / X) + 5.78}}.$$

3. Три игрока в кости I1, I2, I3 по очереди бросают по два кубика с числом очков от 1 до 6. Кто выиграет после 4 циклов игры?

4. Присвойте переменным `X$`, `Y$`, `Z$` названия трех крупных рек России. Вычислите выражение вида:

```
Right(X$, 3) + " " + Left(YS, 2) + " " + Mid(Z$, 2, 2) + "."
```

10.2. Циклы с параметром

Во всех заданиях символ или комбинация символов движутся по определенным маршрутам. Маршрут задается описаниями начальной и конечной позиций первого слева символа движущейся строки вида (N1, M1) – (N1, M2) – (N2, M2) – (N2, M3). Здесь N1, N2 — номера строк, M1, M3 — номера позиций в строке. Строки нумеруются сверху вниз, начиная с 1. Позиции нумеруются слева направо, начиная с 1.

Число символьных позиций в строке зависит от типа и размера шрифта. Их можно изменять командой BASIC-256 вида `Font fn, p, w` *(см. разд. 2.2)*, где: `fn` — имя шрифта; `p` — высота шрифта в пунктах от 8 до 72; `w` — интенсивность шрифта от 1 до 100. Рекомендуется перед решением этих задач установить размер шрифта командой меню BASIC-256 **Просмотр | Размер шрифта | Мелкий**. Примеры разработки программ этого типа рассмотрены в *разд. 8.2*.

10.2.1. Символ "$" движется по маршруту: (1, 1) – (1, 60) – (3, 60) – (3, 1).

10.2.2. Символ "$" движется по маршруту (3, 1) – (3, 60) – (1, 60) – (1, 1).

10.2.3. Гусеница вида "$$$" ползет по маршруту (5, 1) – (5, 55) – (2, 55) – (2, 1).

10.2.4. Комбинация "***" движется по экрану по маршруту: (20, 50) – (20, 1) – (1, 1) – (1, 50).

10.2.5. Символ "*" движется по маршруту (1, 1) – (1, 45) – (12, 45) – (12, 1).

10.2.6. Ваше имя в виде комбинации символов движется по маршруту (5, 1) – (5, 40) – (2, 40) – (2, 1).

10.2.7. Название Вашей улицы в виде комбинации символов движется по маршруту: (2, 1) – (2, 40) – (10, 40) – (10, 1).

10.2.8. Ваша фамилия в виде символьной комбинации движется по маршруту: (1, 1) – (1, 42) – (6, 42) – (6, 1).

10.2.9. Название Вашего населенного пункта в виде комбинации символов движется со скоростью 5 позиций/сек по маршруту (3, 1) – (3, 30) – 13, 30) – (13, 1).

10.2.10. Комбинация "###" движется по маршруту: (15, 46) – (15, 1) – (5, 1) – (5, 46).

10.2.11. Первые три буквы Вашего имени в виде комбинации символов движутся по маршруту: (3,1) – (3, 47) – (9, 47) – (9, 1).

10.2.12. Первая буква Вашей фамилии движется по маршруту: (7, 1) – (7, 42) – (12, 42) – (12, 1).

10.2.13. Символ "M" движется по маршруту: (17, 45) – (17, 1) – (7, 1) – (7, 45).

10.2.14. Первые 4 буквы Вашего имени движутся по маршруту: (16, 1) – (16, 40) – (4, 40) – (4, 1).

10.2.15. Ваше имя латинскими буквами движется по маршруту: (8, 40) – (8, 1) – (2, 1) – (2, 40).

10.2.16. Слово "Vid" движется по маршруту: (2, 1) – (2, 40) – (13, 40) – (13, 1).

10.2.17. Слово "Victory" движется по маршруту: (3, 35) – (3, 1) – (8, 1) – (8, 35).

10.2.18. Слово "Tiger" движется по маршруту: (1, 36) – (1, 1) – (14, 1) – (14, 36).

10.2.19. Слово "Lion" движется по маршруту: (17, 1) – (17, 37) – (6, 37) – (6, 1).

10.2.20. Ваше имя латинскими буквами движется со скоростью 6 позиций/сек по маршруту: (13, 35) – (13, 1) – (2, 1) – (2, 35).

10.2.21. Ваша фамилия латинскими буквами движется со скоростью 5 позиций/сек по маршруту: (15, 1) – (15, 32) – (4, 32) – (4, 1).

10.2.22. Ваше имя латинскими буквами движется со скоростью 6 позиций/сек по маршруту: (13, 35) – (13, 1) – (2, 1) – (2, 35).

10.2.23. Слово "Info" движется со скоростью 8 позиций/сек по маршруту: (13, 4) – (13, 40) – (5, 40) – (5, 1).

10.2.24. Слово "лев" движется со скоростью 9 позиций/сек по маршруту: (3, 5) – (3, 40) – (15, 40) – (15, 1).

10.2.25. Слово "Time" движется со скоростью 10 позиций/сек по маршруту: (10, 2) – (10, 38) – (15, 38) – (15, 4).

10.2.26. Слово "CAT" движется со скоростью 5 позиций/сек по маршруту: (1, 2) – (1, 39) – (12, 39) – (12, 3).

10.2.27. Первые 3 буквы Вашей фамилии в виде символьной комбинации движутся со скоростью 8 позиций/сек по маршруту: (2, 3) – (2, 40) – (11, 40) – (11, 5).

10.2.28. Ваши инициалы (первые буквы фамилии, имени, отчества) движутся со скоростью 4 позиции/сек по маршруту: (20, 3) – (20, 39) – (7, 39) – (7, 2).

10.2.29. Слово "June" движется со скоростью 7 позиций/сек по маршруту: (16, 37) – (16, 1) – (6, 1) – (6, 38).

10.2.30. Первые 4 буквы Вашего отчества движутся со скоростью 6 позиций/сек по маршруту: (13, 40) – (13, 1) – (2, 1) – (2, 38).

10.3. Задачи поиска

Программы поиска входят в состав многих системных и офисных программ: операционных систем, текстовых редакторов. Программы поиска удобны для иллюстрации применения в программах операторов ветвления. Каждое задание представляет собой небольшой проект, который включает ввод некоторого массива данных и поиск в этом массиве данных, удовлетворяющих заданным условиям.

Учащиеся при выполнении проектов учатся записывать условия поиска в содержательной задаче для чисел или символьных строк (найти в массиве чисел все числа, делящиеся на 3; найти в массиве названий стран все страны, названия которых оканчиваются на букву Я и т. п.). Во всех последующих заданиях 10 <= N <= 20. Примеры решения задач поиска рассмотрены в *разд. 6.4*.

10.3.1. В массиве из N чисел (N и числа вводятся с клавиатуры и представляют собой сторону квадрата) выделить те, для которых площадь S(I) > 100, и подсчитать их количество.

10.3.2. В массиве из N чисел (N и числа вводятся с клавиатуры и представляют собой сторону равностороннего треугольника) выделить те, для которых площадь треугольника S(I) >= 40, и подсчитать их количество.

10.3.3. В массиве из N чисел выделить все четные и нечетные числа, подсчитать количество тех и других. Числа задаются анонимным массивом, который формируется самостоятельно.

10.3.4. В массиве из N чисел, где каждое число — сторона равнобедренного прямоугольного треугольника, выделить те, для которых площадь S <= 30, и подсчитать их количество. N и числа вводятся с клавиатуры.

10.3.5. В массиве из N слов (N и слова вводятся с клавиатуры) выделить все слова на букву "А" без учета регистра и подсчитать их количество.

10.3.6. В массиве из N слов (N и слова вводятся с клавиатуры) выделить все слова, оканчивающиеся на букву "а", и подсчитать их количество.

10.3.7. В массиве из 10–20 слов выделить те, у которых вторая буква "о" (русская), и подсчитать их количество. Слова заданы анонимным массивом.

10.3.8. В массиве из 10–20 слов выделить все слова, у которых предпоследняя буква "е", и подсчитать их количество. Слова заданы анонимным массивом.

10.3.9. В массиве из 10–20 слов выделить все слова, у которых третья буква "р" (русская), и подсчитать их количество. Слова заданы анонимным массивом.

10.3.10. В массиве из 10–20 слов (иностранные названия животных и птиц) выделить все слова на букву `X$`, вводимую с клавиатуры, и подсчитать их количество. Пример массива: {wolf, bear, cow, mouse, rat, dog, cat, hare, crocodile, rabbit, tiger, lion, elephant, hen, leopard, fox, cock, eagle, squirrel, pig}.

10.3.11. В массиве из 10–20 чисел выделить те, которые делятся на 3, 5 или 7, и подсчитать их количество. Слова заданы анонимным массивом.

10.3.12. В массиве из N чисел (N и числа вводятся с клавиатуры и представляют собой радиусы окружностей) выделить те, для которых площадь S >= 20, и подсчитать их количество.

10.3.13. В массиве из N действительных чисел (N и числа вводятся с клавиатуры) выделить натуральные числа и подсчитать их количество.

10.3.14. В массиве из 10–20 чисел (вводятся с клавиатуры и представляют собой стороны правильного треугольника) выделить те, для которых периметр кратен 5 или 10.

10.3.15. В массиве из 10–20 слов (вводятся с клавиатуры) выделить все слова, состоящие из трех букв, и подсчитать их количество.

10.3.16. В списке из 10 учащихся вашего класса выделите номера и фамилии учащихся на заданную букву. Буква и фамилии вводятся с клавиатуры. Номера — в порядке ввода, начиная с 1.

10.3.17. В списке из 12 имен учащихся Вашего класса подсчитайте, сколько из них заданных. Например, вводится имя "Екатери-

на" или "Катя" — программа подсчитывает, сколько в списке Екатерин или Кать.

10.3.18. В списке из 10 чисел выделить числа в диапазоне от X1 до X2 и подсчитать их количество. X1 и X2 вводятся с клавиатуры.

10.3.19. В списке из 10 фамилий Ваших учителей выделить все фамилии на заданную букву и подсчитать их количество. Буква и фамилии вводятся с клавиатуры.

10.3.20. С клавиатуры вводится произвольная фамилия учащегося или учащейся вашего класса. Программа подсчитывает количество заданных букв (буква также вводится) в этой фамилии.

10.3.21. В списке из 10 городов (населенных пунктов) Вашей области выделить названия на заданную букву. Буква и названия вводятся с клавиатуры.

10.3.22. С клавиатуры вводятся 10 произвольных фамилий учащихся Вашего класса. Подсчитать, сколько из них оканчиваются на буквы "а" или "в".

10.3.23. С клавиатуры вводятся 10 фамилий учащихся Вашего класса. Программа определяет самую короткую. Если их несколько — выводит все.

10.3.24. В списке из 10 фамилий учащихся Вашего класса выделить самую длинную, самую короткую и определить их длины. Фамилии вводятся с клавиатуры.

10.3.25. В списке из 10 фамилий учащихся вашего класса выделить фамилии с четным и нечетным количеством букв и подсчитать их количества.

10.3.26. Вводятся с клавиатуры 10 фамилий учащихся Вашего класса. Выделить все фамилии, в которых встречается буква "а", и подсчитать их количество.

10.3.27. В массиве из 12 названий европейских столиц выделить те, которые начинаются на заданную первую букву. Буква и названия вводятся с клавиатуры.

10.3.28. С клавиатуры вводится произвольная фраза длиной до 50 символов. Программа подсчитывает число слов в этой фразе.

Слова разделены строго одним пробелом. В начале и в конце фразы пробелов нет.

10.3.29. В массиве из 10 названий произвольных городов России выделить такие, у которых длина названия не превышает N букв. Названия и N вводятся с клавиатуры.

10.3.30. В массиве из 10 произвольных фамилий учащихся Вашего класса (фамилии вводятся с клавиатуры) выделить такие, которые оканчиваются на две заданные буквы (например, ин, ов, ко). Требуемое окончание `X$` вводится с клавиатуры.

10.3.31. В массиве из 10 произвольных названий населенных пунктов Вашего региона выделить такие, которые оканчиваются на заданную букву, и подсчитать их количество. Буква `X$` и массив вводятся с клавиатуры.

10.3.32. В массиве из 10 произвольных названий стран Европы выделить такие, которые содержат в названии заданную букву. Названия стран и буква вводятся с клавиатуры.

10.3.33. В массиве из данных о 10 странах Европы (данные включают название страны и ее население) выделить такие, население которых от 10 до 20 млн чел. Названия стран и цифры населения вводятся в соответствующие массивы из списков.

10.3.34. В массиве из данных о 10 странах Европы (названия и площади в тыс. кв. км) выделить такие, у которых площади лежат в пределах от 10 до 30 тыс. кв. км. Названия и площади вводятся из списков.

10.3.35. В массиве из 10 названий районов своего региона (названия вводятся из списка) вывести названия районов наибольшей и наименьшей длины.

10.4. Задачи сортировки

Программы сортировки входят в состав всех операционных систем и большинства офисных программ. Понятие о сортировке, ее видах и алгоритмах дается в теоретической части. Подробно рассматривается какой-нибудь один алгоритм, например, сортировка методом "пузырька". В заданиях рассматриваются двухкомпо-

нентные данные. Сортировка выполняется по одному из компонентов. В отсортированных данных выводятся оба компонента. Неявно дается понятие о сортировке записей в базах данных.

Задание выдается заранее и выполняется как индивидуальный проект. В домашней части проекта учащиеся подбирают из литературы или из Интернета нужные данные, составляют таблицы данных, алгоритмы и программы. Весь этот материал включается в описание проекта. В классной части выполняют ввод, отладку и запись программ на жесткий диск в папку Sortirovka внутри папки своего класса. В качестве имен файлов, как обычно, рекомендуется использовать до 8 символов фамилии латинскими буквами. Примеры решения задач сортировки рассмотрены в *разд. 6.5*.

10.4.1. В массиве из данных о 10 странах Европы (названия и численность населения в тыс. чел.) выполнить сортировку данных в порядке возрастания населения. Данные в массивы вводятся с клавиатуры.

10.4.2. В массиве из данных о 10 странах Европы (названия и площади в тыс. кв. км вводятся из анонимных массивов) выполнить сортировку названий стран в порядке убывания населения.

10.4.3. В массиве из данных о 10 городах России (названия и население задаются в виде анонимных массивов) выполнить сортировку названий городов в порядке возрастания населения.

10.4.4. В массиве из данных о 10 населенных пунктах своего региона (названия и население вводятся с клавиатуры) выполнить сортировку названий в порядке убывания населения.

10.4.5. В массиве из данных о 10 районах своего региона (данные включают названия районов и их население, вводятся из анонимных массивов) выделить три района с наибольшим населением (вывести названия районов и их население).

10.4.6. В массиве из данных о 10 районах своего региона (названия районов и площади в кв. км задаются в виде анонимных массивов) вывести данные о трех районах наименьшей площади.

10.4.7. В массиве из данных о 10 районных центрах своего региона (названия и даты основания задаются в виде анонимных массивов) вывести данные о трех старейших районных центрах.

10.4.8. В списке 10 учащихся Вашего класса (фамилии и имена) выполнить сортировку по фамилиям в алфавитном порядке. Выводить фамилии и имена. Форму ввода данных (с клавиатуры или в виде анонимных массивов) выбрать самостоятельно.

10.4.9. В списке 10 учащихся Вашего класса (фамилии и имена) выполнить сортировку по именам в алфавитном порядке. Выводить фамилии и имена. Данные вводятся с клавиатуры.

10.4.10. В списке 10 учащихся Вашего класса (фамилии и имена) выполнить сортировку по фамилиям в обратном алфавитном порядке. Выводить имена и фамилии. Данные заданы в форме анонимных массивов.

10.4.11. В списке 10 учащихся Вашего класса (фамилии и имена) выполнить сортировку по именам в обратном алфавитном порядке. Выводить фамилии и имена. Данные вводятся с клавиатуры.

10.4.12. В списке 10 учителей Вашего класса (фамилии, имена и отчества) выполнить сортировку по фамилиям в алфавитном порядке. Выводить фамилии, имена и отчества. Данные задать в виде анонимных массивов.

10.4.13. В списке 10 учителей Вашего класса (фамилии, имена и отчества) выполнить сортировку по фамилиям в обратном алфавитном порядке. Выводить имена, отчества и фамилии. Форма ввода данных — из анонимных массивов.

10.4.14. В списке 10 учителей Вашего класса (фамилии, имена и отчества) выполнить сортировку по именам в алфавитном порядке. Выводить фамилии, имена и отчества. Данные задаются в виде анонимных массивов.

10.4.15. В списке 10 учителей Вашего класса (фамилии, имена и отчества) выполнить сортировку по именам в обратном алфавитном порядке. Данные задаются в виде анонимных массивов. Выводить имена, отчества и фамилии.

10.4.16. Данные о 10 металлах содержат названия и плотности в г/см куб. Выполнить сортировку по плотности в порядке возрастания. Вывести названия металлов и плотности. Данные задаются в виде анонимных массивов.

10.4.17. Данные о 10 металлах содержат названия и плотности в г/см куб. Выполнить сортировку по названиям в алфавитном порядке. Вывести названия металлов и плотности. Данные вводятся с клавиатуры.

10.4.18. Данные о 10 газах содержат названия и плотности в г/см куб. Выполнить сортировку по плотности в порядке убывания. Вывести названия газов и плотности. Данные задаются в виде анонимных массивов.

10.4.19. Данные о 10 газах содержат названия и плотности в г/см куб. Выполнить сортировку по названиям в обратном алфавитном порядке. Вывести названия газов и плотности. Данные вводятся с клавиатуры.

10.4.20. Данные о 10 Ваших учебниках содержат фамилии авторов и названия. Выполнить сортировку по названиям в алфавитном порядке. Вывести авторов и названия. Данные вводятся с клавиатуры.

10.4.21. Данные о 10 Ваших учебниках содержат фамилии авторов и названия. Выполнить сортировку по фамилиям авторов в алфавитном порядке. Вывести авторов и названия. Данные заданы анонимными массивами.

10.4.22. Данные о 10 Ваших учебниках содержат названия и число страниц. Выполнить сортировку по названиям в обратном алфавитном порядке. Вывести названия и число страниц. Данные вводятся с клавиатуры.

10.4.23. Данные о 10 Ваших учебниках содержат фамилии авторов и число страниц. Выполнить сортировку по числу страниц в порядке убывания. Вывести авторов и число страниц. Данные задаются как анонимные массивы.

10.4.24. Данные о 10 школах Вашего города содержат номер (название) школы и численность учащихся. Выполнить сортировку по номерам (названиям) школ в порядке убывания. Вывести номера (названия) школ и численность учащихся. Данные задаются как анонимные массивы.

10.4.25. Данные о 10 школах Вашего города содержат номер (название) школы и численность учащихся. Выполнить сортировку

по численности учащихся в порядке возрастания. Вывести номера (названия) школ и численность учащихся. Данные вводятся с клавиатуры.

10.4.26. Данные о реках России содержат название и длину в км. Выполнить сортировку данных по названиям в алфавитном порядке. Вывести названия и длины. Данные задаются как анонимные массивы.

10.4.27. Данные о реках России содержат название и длину в километрах. Выполнить сортировку данных по длинам рек в порядке убывания. Вывести названия и длины. Данные вводятся с клавиатуры.

10.4.28. Данные о 10 морях России содержат названия и глубины. В названиях морей при вводе данных слово `море` опускать, например: Каспийское, Черное, Балтийское и т. п. Выполнить сортировку данных по возрастанию глубин. Вывести названия морей и глубины. Данные задаются анонимными массивами.

10.4.29. Данные о 10 морях России содержат данные о названиях и глубинах. В названиях морей при вводе данных слово `море` опускать, например: Каспийское, Черное, Балтийское и т. п. Выполнить сортировку данных по названиям морей в алфавитном порядке. Вывести названия морей и глубины. Данные вводятся с клавиатуры.

10.4.30. Данные о 10 горных вершинах мира содержат названия и высоты в метрах. Слова `гора` или `пик` в названиях не использовать, например: Эльбрус, Казбек, Джомолунгма, Победы, Ленина и т. п. Выполнить сортировку данных по высотам в порядке убывания. Данные задаются в виде анонимных массивов.

10.4.31. Данные о 10 горных вершинах мира содержат названия и высоты в метрах. Слова `гора` или `пик` в названиях не использовать, например: Эльбрус, Казбек, Джомолунгма, Победы, Ленина и т. п. Выполнить сортировку данных по названиям в алфавитном порядке. Данные вводятся с клавиатуры.

10.5. Моделирование математических, экономических и физических зависимостей

Теоретические и мировоззренческие вопросы моделирования, такие как классификация моделей, понятие об информационном моделировании и способах построения компьютерных моделей, рассматриваются в теоретической части курса. В качестве примера построения моделей рассматриваются задачи вычисления таблиц и построения графиков функций. Желательно согласовать этот материал по времени с изучением раздела "Анализ элементарных функций" в математике.

Каждое из приведенных далее заданий ориентировано на выполнение проекта, состоящего из двух связанных частей. Первая часть — разработка программы расчета таблицы заданной функции с выводом оформленной таблицы в текстовое окно. Вторая часть — разработка программы построения графика с выводом оформленного графика в графическое окно. Обе части представляют собой самостоятельные разделы проекта с отдельными описаниями, которые оцениваются и защищаются независимо.

Задание на проект выдается заранее. Оно содержит математическую запись функции и пределы изменения аргумента X. Предварительно подробно рассматриваются все этапы выполнения проекта с подробными примерами:

- расчет таблицы функции (с помощью программы);
- построение графика функции на бумаге в математических координатах;
- определение максимального и минимального значений функции;
- расчет масштабов по осям X и Y — Mx и My соответственно — с учетом характеристик графического окна;
- выбор размеров графического экрана и положения осей координат графика;

- ❒ получение моделирующей зависимости в координатах графического экрана;
- ❒ разработка алгоритма и моделирующей программы.

Программы практически все однотипны. Отличаются друг от друга несколькими строками текстов. Разница — в пределах изменения аргумента, величине шага по оси аргумента для расчета точек графика и в виде функции. По опыту автора наибольшую сложность для учащихся представляет расчет моделирующей зависимости.

В описании первого раздела проекта должны быть приведены:

- ❒ таблица функции;
- ❒ алгоритм и программа для ее расчета;
- ❒ максимальное и минимальное значения функции, необходимые для расчета масштабов Mx и My во втором разделе проекта;
- ❒ примерный график функции, построенный по таблице (он понадобится для контроля работы программы, разработанной во втором разделе проекта).

Описание второго раздела проекта должно содержать:

- ❒ расчет масштабов и положения осей на графическом экране на основании данных, полученных при выполнении первого раздела проекта;
- ❒ рассчитанную моделирующую зависимость;
- ❒ алгоритм и программу построения графика.

В классе для обоих разделов проекта выполняются: ввод программ, их отладка и запись на диск в папки с именами файлов до 8 символов фамилии латинскими буквами. Файлы программ первого и второго разделов проекта сохраняются, соответственно, в папках Tabl и Graphic внутри папки класса.

Примеры разработки программ для расчета таблиц функций и построения их графиков приведены, соответственно, в *разд. 6.3* и *главе 7 (примеры 7.1, 7.2)*. Ответы на задачи *разд. 10.5* приведены в *приложении 2*.

10.5.1. Разработать программы для расчета таблицы и построения графика функции Y = tg(PI*X/2) для 0 <= X <= 0.8.

10.5.2. Разработать программы для расчета таблицы и построения графика функции Y = ctg(PI*X/2) для 0.2 <= X <= 1.

10.5.3. Разработать программы для расчета таблицы и построения графика функции Y = sin(PI*X/2) для –1 <= X <= 1.

10.5.4. Разработать программы для расчета таблицы и построения графика функции Y = cos(PI*X/2) для 0 <= X <= 2.

10.5.5. Разработать программы для расчета таблицы и построения графика функции Y = X^(1/2) для 0 <= X <= 10.

10.5.6. Разработать программы для расчета таблицы и построения графика функции Y = X^(3/2) для 0 <= X <= 5.

10.5.7. Разработать программы для расчета таблицы и построения графика функции Y = X^(2/3) для 0 <= X <= 10.

10.5.8. Разработать программы для расчета таблицы и построения графика функции Y = X^3 для 0 <= X <= 10.

10.5.9. Разработать программы для расчета таблицы и построения графика функции Y = 3^X для –5 <= X <= 5.

10.5.10. Разработать программы для расчета таблицы и построения графика функции Y = X^(1/3) для –50 <= X <= 50.

10.5.11. Разработать программы для расчета таблицы и построения графика функции Y = LG X для 0.2 <= X <= 100.

10.5.12. Разработать программы для расчета таблицы и построения графика функции Y = tg(PI*X) для –0.45 <= X <= 0.45.

10.5.13. Разработать программы для расчета таблицы и построения графика функции Y = tg(PI*X/2) для –0.8 <= X <= 0.8.

10.5.14. Разработать программы для расчета таблицы и построения графика функции Y = 0.5*(3^X + 3^(–X)) для –3 <= X <= 3.

10.5.15. Разработать программы для расчета таблицы и построения графика функции Y = 0.5*(3^X – 3^(–X)) для –4 <= X <= 4.

10.5.16. Разработать программы для расчета таблицы и построения графика функции Y = X^(1/4) для 0 <= X <= 10000.

10.5.17. Разработать программы для расчета таблицы и построения графика функции Y = X^(5/3) для 0 <= X <= 30.

10.5.18. Разработать программы для расчета таблицы и построения графика функции Y = X^(3/5) для 0 <= X <= 100.

10.5.19. Разработать программы для расчета таблицы и построения графика функции Y = X^(5/4) для 0 <= X <= 50.

10.5.20. Разработать программы для расчета таблицы и построения графика функции Y = X^(4/5) для 0 <= X <= 100.

10.5.21. Разработать программы для расчета таблицы и построения графика функции Y = (2^X – 2^(–X))/(2^X + 2^(–X)) для –5 <= X <= 5.

10.5.22. Разработать программы для расчета таблицы и построения графика функции Y = (2^X + 2^(–X))/(2^X – 2^(–X)) для 0.2 <= X <= 5.

10.5.23. Разработать программы для расчета таблицы и построения графика функции Y = 1/X для 0.05 <= X <= 10.

10.5.24. Разработать программу для построения графика функции Y = 1/X^(1/2) для 0.01 <= X <= 100.

10.5.25. Разработать программы для расчета таблицы и построения графика функции Y = 1/(X^2) для 0.2 <= X <= 5.

10.5.26. Разработать программы для расчета таблицы и построения графика функции Y = 2.5*X^2 – 5.7*X + 23.5 для –5 <= X <= 5.

10.5.27. Разработать программы для расчета таблицы и построения графика функции Y = X^3 – 6*X^2 + 10*X – 20 для –10 <= X <= 10.

10.5.28. Разработать программы для расчета таблицы и построения графика функции Y = 1/(X^2 + 6*X + 9) для –2.8 <= X <= 5.

10.5.29. Разработать программу для построения графика функции Y = 10/X^(1/3) для 0.2 <= X <= 8.

10.5.30. Разработать программы для расчета таблицы и построения графика функции Y = tg X/2 для –3 <= X <= 3.

10.5.31. Разработать программы для расчета таблицы и построения графика функции Y = Y0*(1 + X/100)^P. Здесь: Y — сумма

вклада через P лет; Y0 — сумма начального вклада; X — банковский процент. Принять Y0 = 1000 руб., P = 5. Процент X меняется от 0.5 до 20.

10.5.32. Разработать программы для расчета таблицы и построения графика функции Y = Y0*(1 + X/100)^P. Здесь: Y — сумма вклада через P лет; Y0 — сумма начального вклада; X — банковский процент. Принять Y0 = 1000 руб., X = 20. Срок P меняется от 1 месяца до 20 лет.

10.5.33. Разработать программы для расчета таблицы и построения графика функции P = LN(Y/Y0)/LN(1 + X/100). Здесь: Y — сумма вклада через P лет; Y0 — сумма начального вклада; X — банковский процент. Построить график P = P(X) — зависимости срока P накопления заданной суммы Y от банковского процента X при Y0 = 1000 руб., Y = 1500 руб., для X от 1 до 40.

10.5.34. Разработать программы для расчета таблицы и построения графика функции P = LN(Y/Y0)/LN(1 + X/100). Здесь: Y — сумма вклада через P лет; Y0 — сумма начального вклада; X — банковский процент. Построить график P = P(Y) — зависимости срока P накопления заданной суммы Y от величины Y при Y0 = 1000 руб., X = 20, для Y от 1100 до 3000 руб.

10.5.35. Разработать программы для расчета таблицы и построения графика функции Y = X0/X. Здесь: Y — относительный уровень инвестиционного спроса (спроса на банковские кредиты); X — банковский процент по кредитам; X0 — значение банковского процента X, при котором спрос на инвестиции принят за 1. Принять: X0 = 40; X изменяется от 10 до 100.

10.5.36. Разработать программы для расчета таблицы и построения графика функции Y = A^(X–X0) – 1. Здесь: Y — предложение сбережений (в процентах от дохода); X — относительный доход по отношению к минимальной зарплате; X0 — значение относительного дохода X, при котором сбережения отсутствуют. A — эмпирический коэффициент. Принять: A = 1.1, X0 = 10, X изменяется от 10 до 100.

10.5.37. Разработать программы для расчета таблицы и построения графика функции Y = Y0/X. Здесь: Y — относительная вели-

чина спроса (объема продаж) на товар от относительной цены X; Y0 — константа. Принять Y0 = 3, X от 0.3 до 20.

10.5.38. Разработать программы для расчета таблицы и построения графика функции Y = Y0*X^(3/2). Здесь: Y — относительная величина спроса (объема продаж) на некоторые продовольственные товары (хлеб, картофель, спиртное) от относительной цены X в условиях инфляции; Y0 — константа. Принять Y0 = 1.3. X от 1 до 20.

10.5.39. Бомба сброшена с самолета на высоте H0 и летит вниз. Зависимость текущей высоты бомбы над землей от времени T имеет вид: $H = H0 - GT^2/2$. Считать, что бомба точно попадет в цель на земле. Разработать программы для расчета таблицы и построения графика функции H = H(T) от момента сброса (T = 0) до попадания в цель (T = T0). Принять: H0 = 10 000 м; $G = 9.8$ м/сек^2 — ускорение свободного падения.

Указание

Вначале следует определить время падения бомбы на землю T0 из уравнения: $H0 - G*T0^2/2 = 0$.

10.5.40. Автомобиль двигался равномерно прямолинейно со скоростью V0 = 72 км/час. Вблизи светофора водитель начал торможение с замедлением A = 2 м/сек^2. При этом путь при торможении автомобиля S зависит от времени T в соответствии с функцией $S = V0*T - A*T^2/2$. Разработать программу для расчета таблицы и построения графика функции S = S(T) на интервале 0 <= T <= T0. Здесь T = 0 — момент начала торможения. T0 — время полной остановки автомобиля, определяется из уравнения V0 = A*T0.

Заключение

При написании этой книги автор не ставил себе целью дать полное описание операторов языка BASIC-256. Самое полное описание языка изложено в монографии Д. Ренью [7].

Поскольку BASIC-256 не является языком профессионального программирования, а используется только как учебный язык для ознакомления с элементарными основами программирования на примерах решения типовых школьных задач и разработки школьных проектов, описание языка дано с той степенью обстоятельности, которая необходима для указанных целей. Подробнее, чем в имеющихся руководствах, рассмотрены следующие вопросы. Ввод/вывод числовых и строковых данных посредством текстовых файлов — эта тема используется в настоящее время на олимпиадах по информатике всех уровней от школьного до международного и, как правило, вообще не рассматривается в существующих изданиях. Построение графиков функций — очень благодарная тема для школьных проектов по программированию, однако она также часто остается в стороне в традиционных изданиях.

Полезен по мнению автора учителям и учащимся будет Задачник по программированию *(глава 10)*. Его целью было дать возможность учителям и учащимся выполнять зачетные работы и проекты строго по индивидуальным заданиям. Кроме того, по сложившейся в настоящее время традиции, в книге рассмотрено решение заданий ЕГЭ средствами BASIC-256 *(глава 9)*.

Автор заранее благодарен за любые замечания и пожелания по улучшению содержания книги.

ПРИЛОЖЕНИЕ 1

Справочник по BASIC-256

Числовые и строковые выражения в строках программы вычисляются с учетом приоритетов, указанных в табл. П1.1. Операции с одинаковым приоритетом выполняются в порядке их записи в выражении слева направо.

Таблица П1.1. Приоритет операторов BASIC-256

Приоритет исполнения	Операторы	Описание
1	( )	Выражения в круглых скобках вычисляются в первую очередь
2	^ , функции	Возведение в степень и функции вычисляются во вторую очередь
3	-	Унарный минус (минус в начале выражения)
4	*, /	Умножение, деление
5	%, \	Вычисление остатка и частного от деления целых чисел
6	+, -	Сложение чисел, конкатенация (объединение) строк, вычитание чисел
7	<, <=, >, >=, =, <>	Сравнение чисел и символьных строк. Строки сравниваются по кодам символов. Используются в записи условий в операторах ветвления и в операторах циклов с условиями
8	NOT	Логическое НЕ
9	AND	Логическое И
10	OR	Логическое ИЛИ
11	XOR	Логическое ИСКЛЮЧАЮЩЕЕ ИЛИ

Операторы и функции BASIC-256 для облегчения поиска сгруппированы в табл. П1.2 в алфавитном порядке в три тематические группы: операторы ввода/вывода данных в окне текста; операторы обработки числовых, строковых и графических данных; операторы управления. К последним отнесены операторы, изменяющие порядок исполнения программы: ветвления, переходы, циклы, подпрограммы.

Таблица П1.2. Операторы и функции BASIC-256

Оператор или функция	Описание
Ввод/вывод данных в окне текста	

`Input Y$, X`
`Input Y$, X$`
	При исполнении оператора `Input` в окно текста выводится диалоговое сообщение `Y$`. В ответ вводится число `X` или строка `X$` и нажимается клавиша <Enter>. Примеры организации диалога при вводе:
	`Input "Введите целое X от 1 до 100, X = ", X`
	`Input "Введите Ваше имя X$ = ", X$`

`Print X` `Print Y$`	Вывод числа `X` или строки `YS` в окно текстового вывода
Операторы и функции обработки числовых, строковых и графических данных	
`X^(1/2)`	Вычисление квадратного корня из `X`
`X^(1/n)`	Вычисление корня степени `n` из `X`
`Abs(X)`	Абсолютное значение выражения `X`
`Acos(X)`	Arccos `X`, –1 <= X <= 1 — функция арккосинус
`X = Asc(Y$)`	Возвращает код первого символа `X` строки `Y$`, 0 <= `X` <= 255; см. `Chr(X)`
`Asin(X)`	Arcsin `X`, –1 <= `X` <= 1 — функция арксинус
`Atan(X)`	Arctg `X` — функция арктангенс
`X1 = Ceil(X)`	Возвращает наименьшее целое число `X1`, не меньшее `X`; см. `Int(X)`, `Floor(X)`
`Y$ = Chr(X)`	Возвращает символ `Y$` по коду `X`, 0 <= `X` <= 255; см. `Asc(Y$)`

Таблица П1.2 *(продолжение)*

Оператор или функция	**Описание**
`Circle x, y, r`	Вычерчивает в окне графики текущим цветом окружность радиуса `r` с центром в точке (`x`, `y`). Текущий цвет установлен последним оператором `Color`
`X = Clickb`	Возвращает код нажатия кнопок мыши: X = 0 — не было нажатий после последнего сброса кода; X = 1 — левая кнопка; X = 2 — правая кнопка; X = 4 — средняя кнопка; при нажатии разных кнопок — сумма кодов. Для каждой кнопки фиксируется одно нажатие. См. `Clickclear`
`Clickclear` `Clickclear()`	Сбрасывает в 0 переменные состояния мыши `Clickb`, `Clickx`, `Clicky`; после этого можно выявить факт очередного нажатия одной из кнопок мыши и положение указателя мыши при этом в окне графики
`Clickx, Clicky`	Переменные `Clickx`, `Clicky` запоминают координаты `x`, `y` указателя мыши в окне графики при нажатии любой клавиши; см. `Clickclear`
`Clg`	Очищает окно графики
`Cls`	Очищает окно текста
`Close`	Закрывает открытый файл. Открыт может быть только один файл. Если открытого файла нет, пропускается

`Color cname`
`Color r, g, b`
`Color RGB`
	Оператор `Color` устанавливает цвет графических примитивов в окне графики и цвет текста в окне текста. Цвет можно задать: именем `cname` (см. табл. 3.1); кодами интенсивностей трех составляющих цветов: красного — `r`, зеленого — `g`, синего — `b` (0 <= `r`, `g`, `b` <= 255); RGB-кодом цвета: RGB = r*2^16 + g*2^8 + b. Примеры:
	`Color red`
	`Color 255, 0, 0`
	`Color 16711680`
	Все три оператора задают красный цвет линий и точек в окне графики и цвет текста в окне текста

`Cos(X)`	`Cos X` — функция косинус
`Day`	Возвращает текущую системную дату (число текущего месяца) от 1 до 31
`X1 = Degrees(X)`	Возвращает угол `X1`, преобразованный из градусов `X` в радианы. См. `Radians`

Таблица П1.2 *(продолжение)*

Оператор или функция	Описание
`Dim X(N)` `Dim Y$(N)`	`N` — целое число. Операторы резервируют память для одномерных массивов из `N` числовых переменных X[0] — X[N–1] или из `N` строковых переменных Y$[0] — Y$[N–1]

`Dim X(N1,N2)`
`Dim Y$(N1, N2)`
	`N1`, `N2` — целые числа. Определяет двумерные массивы числовых или строковых переменных размерности `N1*N2`:
	X(0,0) — X(N1–1,N2–1) или Y$(0,0) — Y$(N1–1,N2–1)
	Анонимные массивы — списки числовых или строковых констант, разделенных запятыми, в фигурных скобках:
	`Dim X(4)`
	`X = {10, 20, 30, 40}`
	Элементы массива `X` примут следующие значения: X[0] = 10; X[1] = 20; X[2] = 20; X[3] = 40

`Eof`	`Eof` — End Of File — двоичный признак конца файла. Используется при чтении данных из файла: Eof = 1, конец файла достигнут; Eof = 0, конец файла не достигнут
`Exists(F$)`	Возвращает двоичный флаг существования файла, спецификация которого задана строкой F$. Если файл существует, `Exists(F$)` = 1; если файл не существует, `Exists(F$)` = 0
`FastGraphics`	Оператор включает ускоренный графический режим до остановки программы. Режим `Fastgraphics` означает, что режим отображения в окне графики не меняется до подачи команды `Refresh`. Режим можно использовать для значительного ускорения отображения сложных анимаций и исключения мерцания. Рекомендуется при формировании анимации все графические операторы выполнять в подпрограммах и использовать единственную команду `Refresh` после выполнения всех графических операторов
`Z1 = Float(Z$)`	Преобразует строковое представление действительного числа `Z$` в число `Z1`; см. `String(Z1)`
`X1 = Floor(X)`	Возвращает наибольшее целое число X1 <= X; см. `Ceil(X)`, `Int(X)`
`Font FN$, P, W`	Устанавливает параметры шрифта для команды `Text`: `FN$` — имя шрифта, например, "Times New Roman" или "Arial"; P — высота шрифтовой строки в пунктах вместе с межстрочным промежутком, 1 пункт = 1/72 дюйма; `W` — интенсивность шрифта, число от 1 до 100: бледный шрифт Light: P = 25, нормальная интенсивность Normal: P = 50, полужирный шрифт Bold: P = 75. См. `Text`

Таблица П1.2 *(продолжение)*

Оператор или функция	Описание
`Graphheight`	`Ym = Graphheight` — возвращает текущую высоту окна графики `Ym`, максимальное значение координаты `Y`
`Graphwidth`	`Xm = Graphwidth` — возвращает текущую ширину графического окна `Xm`, максимальное значение координаты `X`
`Graphsize Xm,Ym`	Устанавливает размеры окна графики по горизонтали `Xm` и по вертикали `Ym` (максимальные значения по координатам `X` и `Y` в пикселах). Максимальные значения для Xm × Ym = 800×600. По умолчанию Xm × Ym = 300×300
`Hour`	Возвращает значение текущего часа системного времени в формате от 0 до 23
`Instr(Y1$, Y2$)`	Функция проверяет вхождение строки `Y2$` как подстроки в строку `Y1$`. Если строка `Y2$` является подстрокой строки `Y1$`, то возвращается номер начального символа строки `Y2$` в строке `Y1$`. Символы строки `Y1$` нумеруются слева направо, начиная с 1. В случае невхождения `Y2$` в `Y1$` возвращается 0
`X1 = Int(X)`	Функция возвращает целое число `X1`, получаемое из `X` путем отбрасывания дробной части; см. `Ceil(X), Floor(X)`
`X = Key`	Функция `Key` возвращает целое число `X`, соответствующее коду нажатой на клавиатуре клавиши. Если нажата символьная клавиша, возвращается ее однобайтовый ASCII-код в диапазоне 0 <= X <= 255 (см. функции `Asc(X), Chr(X$)`). Управляющие клавиши отображаются двухбайтовыми кодами: 256 <= X <= 65535
`N = Length(X$)`	Функция `Length` возвращает длину `N` строки `X$` в числе символов
`Line`	`Line X1,Y1,X2,Y2` вычерчивает в окне графики текущим цветом отрезок с координатами точек (X1, Y1) — (X2, Y2)
`S$ = Lower(Y$)`	Строка `S$` формируется из строки `Y$` заменой всех латинских букв на символы нижнего регистра (на строчные латинские буквы). Русские буквы и специальные символы остаются без изменения. См. `Upper(Y$)`
`S$ = Left(Y$, N2)`	Функция выделяет подстроку `S$` из строки `Y$` с начала строки (слева) длиной N2 символов. См. `Mid(Y$, N1, N2), Right(Y$, N2)`
`Log(X)`	Ln X — логарифм натуральный
`Log10(X)`	Lg X — логарифм десятичный

Таблица П1.2 *(продолжение)*

Оператор или функция	Описание
`Mid(Y$, N1, N2)`	`S$ = Mid(Y$, N1, N2)` — функция выделяет подстроку `S$` в строке `Y$`, начиная с символа `N1` длиной `N2` символов. См. `Left(Y$, N2)`, `Right(Y$, N2)`
`Minute`	Функция `Minute` возвращает составляющую системного времени в минутах от 0 до 59. См. `Hour`, `Second`
`Month`	Функция `Month` возвращает составляющую системной даты — номер месяца от 0 (январь) до 11 (декабрь). Для вывода правильной даты следует выводить `(Month + 1)`. См. `Day`, `Month`, `Year`

`Mouseb`
`Mousex`
`Mousey`
	`Mouseb` возвращает код нажатой кнопки мыши: 0 — не было нажатий; 1 — левая кнопка; 2 — правая кнопка; 4 — средняя кнопка; сумма кодов — при нажатии разных кнопок. Для каждой кнопки запоминается одно нажатие.
	`Mousex`, `Mousey` возвращают текущие или последние координаты `x` и `y` курсора мыши в окне графики. См. `Clickb`, `Clickx`, `Clicky`

`Open FN$`	Открывает файл с обозначением в строке `FN$` для чтения и записи. Путь к файлу в строке `FN$` можно задавать в абсолютной и в относительной форме. Одновременно можно открыть только один файл. См. `Close`
`Pause T`	Задает задержку исполнения программы на `T` секунд. `Pause 0.15` задает задержку на 0.15 секунды. Оператор применяется, например, в программах формирования подвижных изображений для задания скорости движения объекта в окнах текста или графики
`PI, pi`	Константа π = 3.141593
`Plot X,Y`	Оператор ставит точку с координатами (X, Y) в окне графики текущим цветом, заданным последним оператором `Color`. См. `Line`, `Circle`

`Poly M`	Числовой массив `M` содержит последовательно `N` пар чисел, представляющих собой координаты вершин N-угольника в окне графики. Оператор `Poly M` строит в окне графики окрашенный текущим цветом N-угольник. Массив `M` можно задать как анонимный. Например, следующий фрагмент вычертит в окне графики зеленый закрашенный треугольник с вершинами в точках (100, 100), (200, 200), (250, 100). См. `Stamp`.
	`M = {100, 100, 200, 200, 250, 100}`
	`Color Green`
	`Poly M`

Таблица П1.2 *(продолжение)*

Оператор или функция	Описание
`X = Radians(X1)`	Преобразует угол из радианной меры X1 в градусы X. См. `Degrees`

`Rand`	Оператор `X = Rand` формирует случайное число X в диапазоне 0 <= X < 1 с распределением по закону равной плотности. Инициализация генератора случайных чисел происходит только при загрузке BASIC-256 и действует в течение всего сеанса работы. Специального оператора инициализации нет. Следующий фрагмент формирует последовательность из 10 целых случайных чисел в диапазоне от 20 до 30 и выводит их в окно текста.
	`For I = 1 To 10`
	`X = 20 + Int(11*Rand): Print X`
	`Next I`

`Readline`	`X$ = Readline` — в переменную `X$` читается строка из текстового файла
`Rect`	`Rect X, Y, W, H` вычерчивает в окне графики закрашенный текущим цветом прямоугольник с верхним левым углом в точке (`X, Y`) шириной `W` (в направлении оси `X`) и высотой `H` (в направлении оси `Y`)
`Redim`	Переопределение в программе числового или строкового массива на другой размер. `Redim X(N1), Redim Y$(N1)` — переопределение числового и строкового массивов на размер `N1`. Пусть прежний размер массивов был N. Если N1 > N, то новые элементы числового массива инициализируются нулями, а у строкового массива пустыми строками. Если N1 < N, то "лишние" элементы обоих массивов теряются
`Refresh`	Оператор `Refresh` работает только в режиме `FastGraphics` и выполняет обновление окна графики. На экране остаются только те графические объекты, которые были сформированы после предыдущего оператора `Refresh`
`Rem` `#`	Операторы `Rem` и `#` задают строки комментариев в программе. Эти строки интерпретатором BASIC-256 не обрабатываются
`Reset`	Оператор `Reset` очищает открытый файл. Все данные, хранившиеся в этом файле, теряются
`Right(Y$, N2)`	`S$ = Right(Y$, N2)` — функция выделяет подстроку `S$` из строки `Y$` длиной `N2` символов с конца `Y$` (справа). См. `Left(Y$,N2), Mid(Y$,N1, N2)`

Таблица П1.2 *(продолжение)*

Оператор или функция	Описание
`Say X$`	Произносит через системный динамик текстовую строку `X$`, записанную латинскими буквами. Текст на английском языке произносится правильно. Русский текст следует для озвучивания писать латинскими буквами
`Second`	Функция `Second` возвращает составляющую системного времени секунды текущей минуты в формате от 0 до 59. См. `Hour`, `Minute`
`Seek X`	Функция `Seek X` перемещает указатель в открытом файле на `X` байтов от начала файла. Указатель задает в файле место чтения или записи
`Sin(X)`	`Sin X` — функция синус; `X` — действительный аргумент
`Size`	`X = Size` возвращает размер в байтах `X` текущего открытого файла
`Sound F, T`	Оператор `Sound F, T` воспроизводит звук частоты `F` длительности `T`. Частота `F` задается в герцах, длительность `T` — в миллисекундах. 1 миллисекунда = 0.001 секунды
`Stamp`	Оператор `Stamp` вычерчивает в окне графики окрашенный многоугольник, заданный массивом координат `Z`. Координаты в массиве `Z` заданы относительно опорной точки (`X, Y`) с масштабным коэффициентом `M`. Многоугольник может быть повернут по часовой стрелке относительно опорной точки на угол `F` в радианах. По умолчанию: масштабный коэффициент M = 1; угол поворота F = 0. M < 1 уменьшает размеры фигуры, M > 1 увеличивает. F > 0 обеспечивает поворот фигуры относительно опорной точки по часовой стрелке,; соответственно F < 0 поворачивает фигуру против часовой стрелки. Если многоугольник имеет N сторон, массив Z должен содержать N пар чисел следующего вида: `Z = {X1,Y1, X2,Y2,…,XN,YN}`. Возможны следующие варианты записи оператора `Stamp`: `Stamp X, Y, M, F, Z` — с полным набором параметров (с масштабированием коэффициентом M и поворотом на угол F); `Stamp X, Y, Z` — отображение многоугольника без масштабирования и поворота; `Stamp X, Y, M, Z` — отображение многоугольника с масштабированием без поворота.

Таблица П1.2 *(продолжение)*

Оператор или функция	Описание

	В следующем примере в окне графики будет отображаться красный треугольник, уменьшенный вдвое по отношению к размерам, заданным относительными координатами, и повернутый относительно опорной точки (150,100) на π/4 против часовой стрелки. См. `Poly`.
	`Cls`
	`Clg`
	`Dim Z(6)`
	`Z = {0, 20, - 100, 150, 100, 150}`
	`Stamp 150, 100, 0.5, -PI/4, Z`

`String(X)`	`Y$ = String(X)` преобразует действительное число `X` в строковое представление `Y$`. См. `Float`
`Tan(X)`	`Tg X` — функция тангенс, аргумент X в радианах. См. `Sin`, `Cos`
`Text X, Y, S$`	Оператор `Text X, Y, S$` выводит в окно графики текст из строки `S$` текущим цветом и шрифтом, начиная с точки (`X,Y`) как левого верхнего угла текстовой строки. См. `Color`, `Font`
`S$ = Upper(Y$)`	Функция преобразует все латинские буквы в строке `Y$` в символы верхнего регистра (в прописные или заглавные буквы). Русские буквы и другие символы не изменяются. См. `Lower(Y$)`
`Wavplay` `Wavstop`	Оператор `Wavplay FN$` запускает на исполнение в фоновом режиме звуковой файл формата wav с обозначением в строке `FN$`. Оператор `Wavstop` прекращает его воспроизведение
`Writeline Y$`	Записывает строку `Y$` вместе с символами перехода на новую строку в конец открытого файла
`Year`	Возвращает составляющую системной даты текущий год в виде 4 цифр. См. `Month`, `Day`
Управляющие операторы	
`Do/Until`	`Do` *`Операторы`* `Until` *`условие`* Цикл с постусловием, или цикл ДО. Условие завершения цикла *`условие`* проверяется после исполнения тела цикла (группы строк *`Операторы`*). Выход из цикла выполняется при выполнении условия завершения. Тело цикла всегда исполняется хотя бы один раз

Таблица П1.2 *(продолжение)*

Оператор или функция	Описание

For/Next	`For X = X0 To X1 Step DX`
	Операторы
	`Next X`
	
	Оператор цикла. В заголовке цикла указаны переменная цикла `X`, ее начальное и конечное значения `X0` и `X1`, шаг изменения `DX`. `X0`, `X1`, `DX` — целые или действительные числа. При целых `X1 > X0` и `DX = 1` `Step DX` можно не писать.
	
	Условие завершения цикла (`X > X1`) при `X1 > X0` и `DX > 0` — проверяется оператором `Next X` после формирования следующего значения переменной `X` по правилу: `X = X + DX`. При невыполнении условия завершения тело цикла (группа строк *Операторы*) исполняется очередной раз; при выполнении условия осуществляется выход из цикла. Следующий фрагмент выводит в окно текста 11 значений кубических корней чисел от 0.5 до 5 с шагом 0.5:
	`For X = 0.5 To 5 Step 0.5`
	`Print X^(1/3)`
	`Next X`

Goto L1	Безусловный переход по метке `L1`. Метка пишется на отдельной строке программы с двоеточием — `L1:`. При написании структурированных программ применять оператор `Goto` не рекомендуется. Лучше необходимый фрагмент алгоритма реализовать как подпрограмму

Gosub/Return	`Gosub L1`
	…
	`End`
	`L1:`
	Операторы
	`Return`
	
	`Gosub L1` вызывает подпрограмму, начинающуюся строкой с меткой `L1:`. Все подпрограммы размещаются после строки с оператором `End` основной программы. Каждая подпрограмма начинается со строки метки с двоеточием и заканчивается оператором возврата в основную программу `Return`. Возврат происходит на строку основной программы, следующую за строкой с оператором вызова этой подпрограммы `Gosub L1`

Таблица П1.2 *(окончание)*

Оператор или функция	**Описание**

`If/Then/Else`	Операторы ветвления на одно и два независимых направления.
	Ветвление на одно независимое направление:
	`If` *условие* `Then`
	Операторы1
	`End If`
	Если *условие* выполнено, выполняется последовательность строк *Операторы1*. При невыполнении условия выполняется следующая по порядку после оператора ветвления строка программы.
	Ветвление на два независимых направления:
	`If` *условие* `then`
	Операторы1
	`Else`
	Операторы2
	`End If`
	При выполнении условия выполняется последовательность строк *Операторы1*, иначе *Операторы2*. По завершении оператора ветвления выполняется следующий по порядку оператор программы после оператора ветвления

`While/End While`	Цикл с предусловием или цикл ПОКА строится следующим образом: `While` *условие* *Операторы* `End While` *Условие* проверяется перед исполнением тела цикла (группы строк *Операторы*). Исполнение тела цикла будет повторяться, пока условие выполнено. При нарушении условия происходит выход из цикла на следующие после `End While` оператор или строку программы

ПРИЛОЖЕНИЕ 2

Ответы на задачи раздела 10.5

Большинство задач *главы 10*, кроме заданий *разд. 10.1*, ориентированного на освоение основных операторов и правил записи выражений в среде BASIC-256, имеют характер творческих заданий, ориентированных на проектный метод выполнения. То есть они требуют от учащегося некоторой творческой деятельности, какой-то собственной самостоятельной работы при подготовке данных. Примеры разработки типовых программ для решения задач из *разд. 10.2–10.4* очень подробно рассмотрены в *главе 6*.

По опыту автора наибольшую трудность для учащихся могут представлять задания *разд. 10.5*, ориентированные на графическое моделирование математических, физических и экономических зависимостей.

В *разд. 10.5* приведено 40 задач. Из них задачи 10.5.1–10.5.32 ориентированы на анализ чисто математических функций. Задачи 10.5.31–10.5.38 связаны с построением математических моделей простых экономических задач. В задачах 10.5.39, 10.5.40 предлагается исследовать физические процессы равноускоренного и равнозамедленного движения.

Ответы автор приводит только для этого раздела в виде листингов программ построения графиков моделирующих зависимостей и самих графиков функций. Комментарии не приводятся. Все расчеты, связанные с анализом задач, определением масштабов и моделирующих зависимостей, учащиеся должны выполнять самостоятельно (подробно рассмотренные примеры графического моделирования математических зависимостей приведены в *при-*

мере 7.9. Поэтому автор ограничился при формировании листингов программ в большинстве задач окном графики с размерами по умолчанию 300×300 пикселов). Листинги программ первой части заданий для расчета таблиц функций также не приводятся.

В ответах для каждой из задач приведены листинги программы и построенный с помощью этой программы график функции. Путем качественного сравнения вида графиков учащиеся могут судить о правильности и рациональности рассчитанной ими математической модели задачи. Как уже отмечалось, в большинстве задач с целью уменьшения объема автор использовал размеры окна графики, используемые по умолчанию при загрузке BASIC-256: 300×300 пикселов, но для некоторых задач удобнее оказалось выбрать графическое окно 320×320 пикселов. Учащиеся при решении задач не ограничены объемом графических объектов и с целью более наглядного представления графиков исследуемых функций могут выбирать графические окна размерами и 800×600, и 640×480, и любые другие более удобные варианты.

> ***Примечание***
>
> В поставленных в школы версиях BASIC-256 предельно допустимый размер окна графики 800×600 пикселов.

Для задач *разд. 10.5* в тексте приложения приняты следующие обозначения:

- задача 10.5.1-1 — первая часть проекта для задачи 10.5.1, связанная с разработкой программы расчета таблицы функции;
- задача 10.5.1-2 — вторая часть того же проекта, связанная с расчетом и разработкой программы для построения графической модели зависимости (графика функции).

Задача 10.5.1-2. Y = tg(π*X/2) для 0 <= X <= 0.8.

Листинг для задачи 10.5.1-2

```
cls
clg
for y = 0 to 300
plot 20,y
next y
for x = 0 to 300
plot x, 280
next x
for x = 20 to 260
y = 280 – 80*tan(PI*(x-20)/600)
plot(x,y)
next x
```

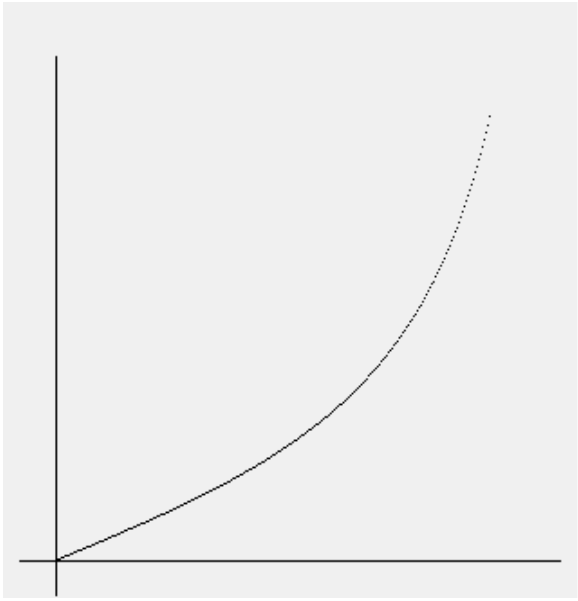

Рис. П2.1. График функции для задачи 10.5.1-2

Задача 10.5.2-2. Y = ctg(π*X/2) для 0.2 <= X <= 1.

Листинг для задачи 10.5.2-2

```
graphsize 360, 300
cls
clg
for y = 0 to 300
plot 20,y
next y
for x = 0 to 360
plot x, 280
next x
for x = 80 to 320
y = 280 – 80/tan(PI*(x-20)/600)
plot(x,y)
next x
```

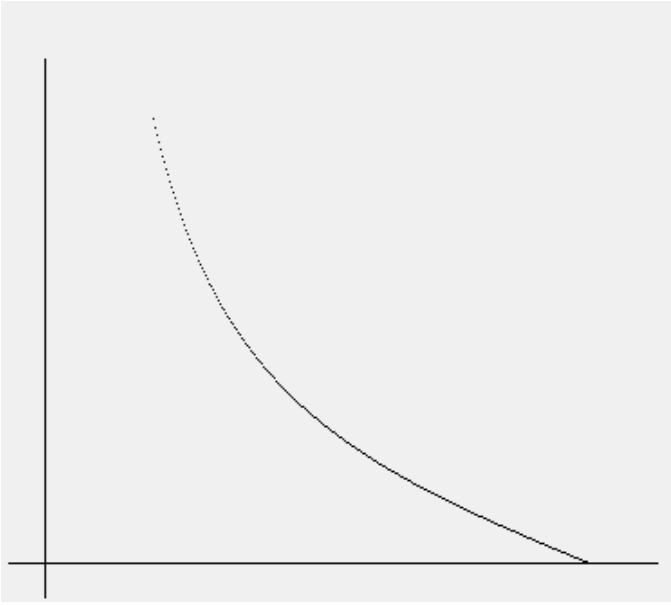

Рис. П2.2. График функции для задачи 10.5.2-2

Задача 10.5.3-2. Y = sin(π*X/2) для 0 <= X <= 2.

Листинг для задачи 10.5.3-2

```
cls
clg
for y = 0 to 300
plot 150,y
next y
for x = 0 to 300
plot x, 150
next x
for x = 50 to 250
y = 150 – 100*sin(PI*(x-150)/200)
plot(x,y)
next x
```

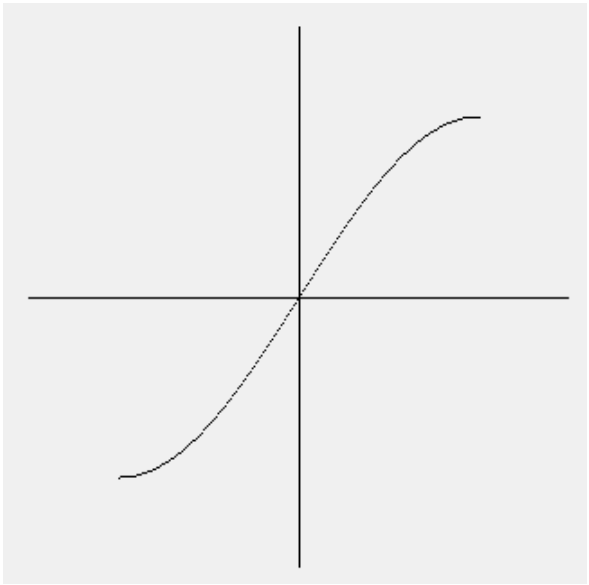

Рис. П2.3. График функции для задачи 10.5.3-2

Задача 10.5.4-2. Y = cos(PI*X/2) для 0 <= X <= 2.

Листинг для задачи 10.5.4-2

```
cls
clg
for y = 0 to 300
plot 50,y
next y
for x = 0 to 300
plot x, 150
next x
for x = 50 to 250
y = 150 — 100*cos(PI*(x-50)/200)
plot(x,y)
next x
```

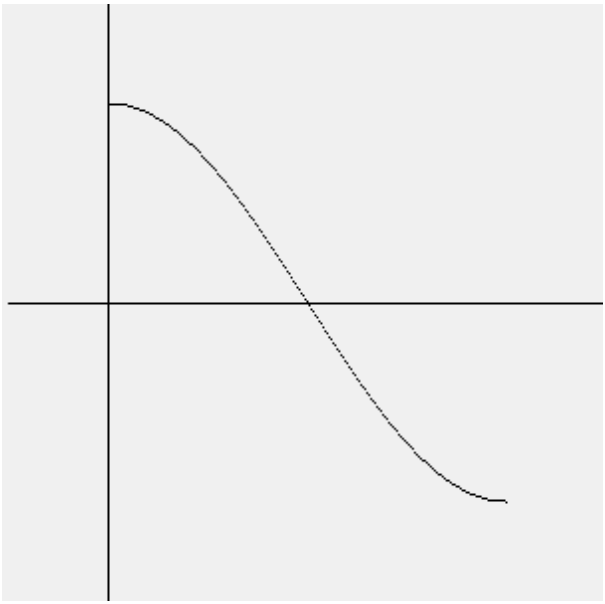

Рис П2.4. График функции для задачи 10.5.4-2

Задача 10.5.5-2. Y = X^(1/2) для 0 <= X <= 10.

Листинг для задачи 10.5.5-2

```
cls
clg
for y = 0 to 300
plot 50,y
next y
for x = 0 to 300
plot x, 280
next x
for x = 50 to 250
y = 280 — 80*((x — 50)/20)^(1/2)
plot(x,y)
next x
```

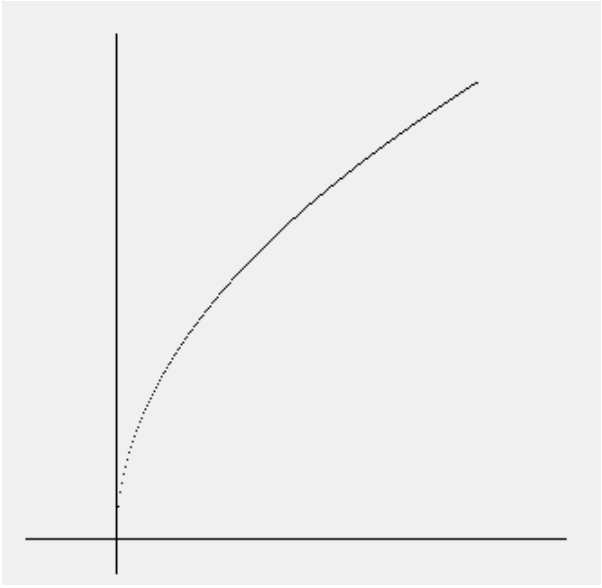

Рис. П2.5. График функции для задачи 10.5.5-2

Задача 10.5.6-2. Y = X^(3/2) для 0 <= X <= 5.

Листинг для задачи 10.5.6-2

```
cls
clg
for y = 0 to 300
plot 20,y
next y
for x = 0 to 300
plot x, 280
next x
for x = 20 to 270
y = 280 — 20*((x — 20)/50)^(3/2)
plot(x,y)
next x
```

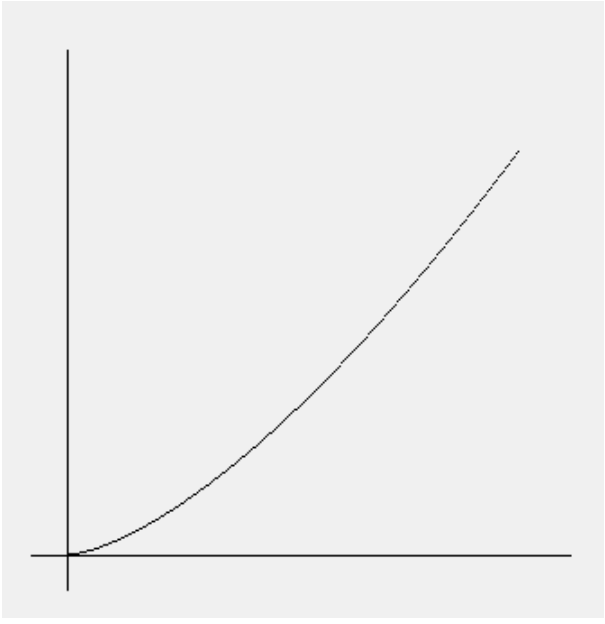

Рис. П2.6. График функции для задачи 10.5.6-2

Задача 10.5.7-2. Y = X^(2/3) для 0 <= X <= 10.

Листинг для задачи 10.5.7-2

```
cls
clg
for x = 0 to 300
plot x,280
next x
for y = 0 to 300
plot 20, y
next y
for x = 20 to 270
y = 280 — 50*((x-20)/25)^(2/3)
plot x,y
next x
```

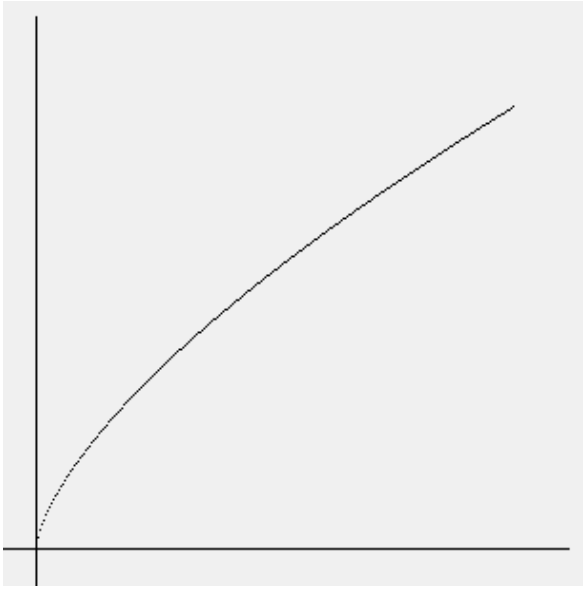

Рис. П2.7. График функции для задачи 10.5.7-2

Задача 10.5.8-2. Y = X^3 для 0 <= X <= 10.

Листинг для задачи 10.5.8-2

```
cls
clg
for x = 0 to 300
plot x,280
next x
for y = 0 to 300
plot 20, y
next y
for x = 20 to 270
y = 280 — 0.25*((x-20)/25)^3
plot x,y
next x
```

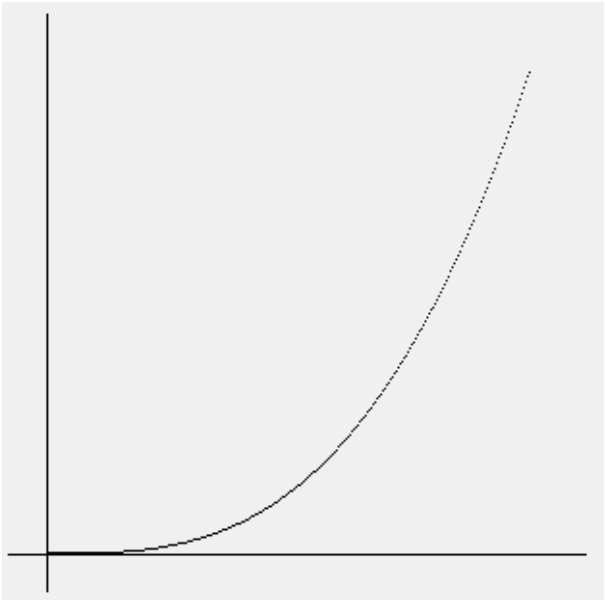

Рис. П2.8. График функции для задачи 10.5.8-2

Задача 10.5.9-2. Y = 3^X для -5 <= X <= 5.

Листинг для задачи 10.5.9-2

```
cls
clg
for x = 0 to 300
plot x,280
next x
for y = 0 to 300
plot 150, y
next y
for x = 0 to 300
y = 280 — 3^((x — 150)/30)
plot x,y
next x
```

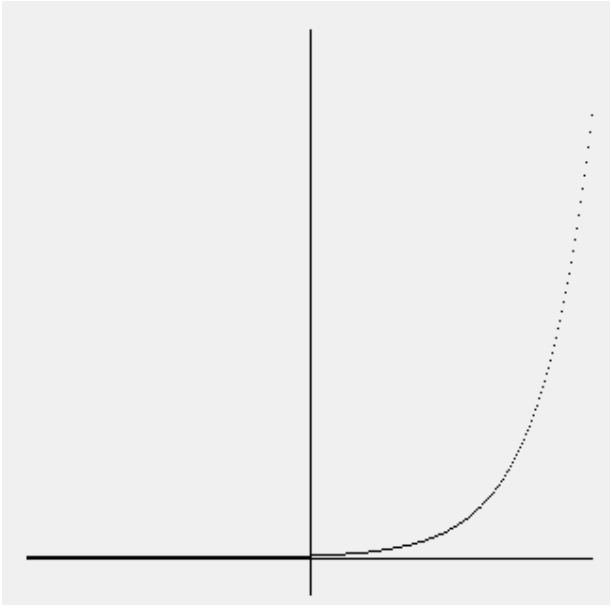

Рис. П2.9. График для задачи 10.5.9-2

Задача 10.5.10-2. Y = X^(1/3) для -50 <= X <= 50.

Листинг для задачи 10.5.10-2

```
cls
clg
for x = 0 to 300
plot x,150
next x
for y = 0 to 300
plot 150, y
next y
for x = 0 to 300
if x < 150 then
y = 150 + 40*((150 — x)/3)^(1/3)
else
y = 150 — 40*((x — 150)/3)^(1/3)
end if
plot x,y
next x
```

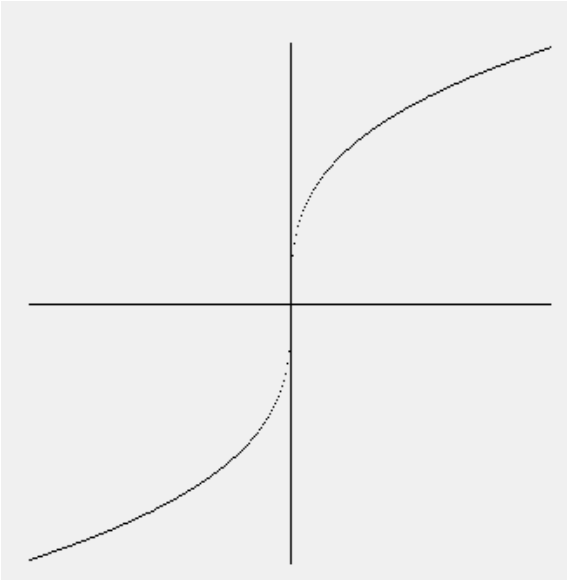

Рис. П2.10. График для задачи 10.5.10-2

Задача 10.5.11-2. Y = LG X для 0.2 <= X <= 100.

Листинг для задачи 10.5.11-2

```
cls
clg
for x = 0 to 300
plot x,210
next x
for y = 0 to 300
plot 20, y
next y
for x = 20.5 to 270
y = 210 — 100*log10((x — 20)/2.5)
plot x,y
next x
```

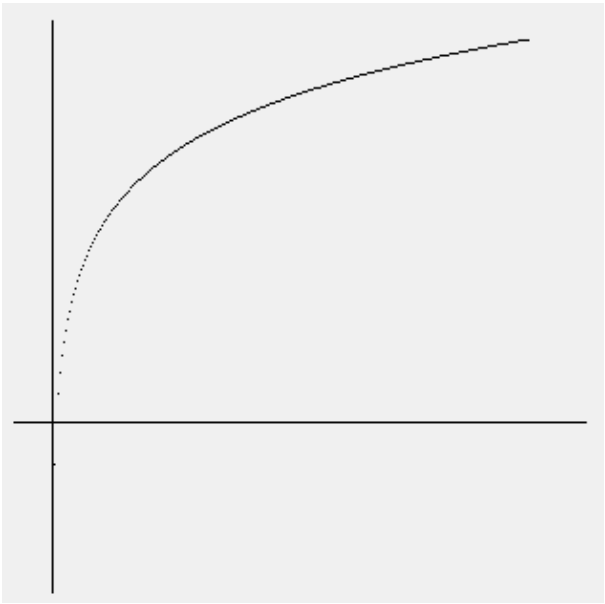

Рис. П2.11. График для задачи 10.5.11-2

Задача 10.5.12-2. Y = tg(π*X) для –0.45 <= X <= 0.45.

Листинг для задачи 10.5.12-2

```
cls
clg
for x = 0 to 300
plot x,150
next x
for y = 0 to 300
plot 150, y
next y
for x = 15 to 285
y = 150 – 20*tan(PI*(x-150)/300)
plot x,y
next x
```

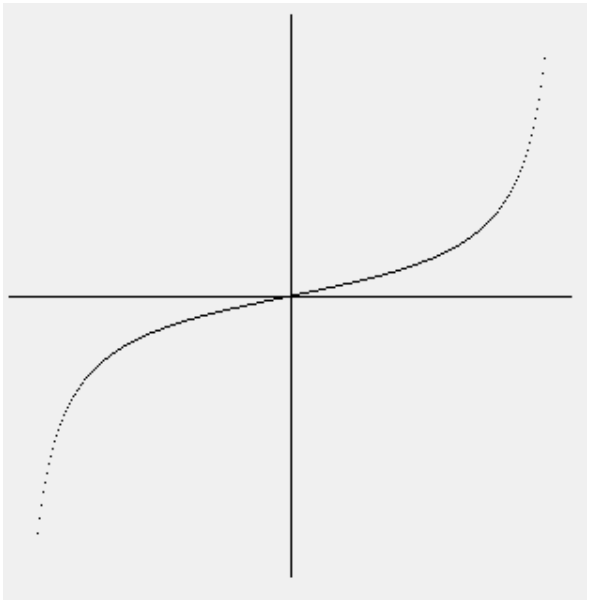

Рис. П2.12. График для задачи 10.5.12-2

Задача 10.5.13-2. Y = tg(PI*X/2) для –0.8 <= X <= 0.8.

Листинг для задачи 10.5.13-2

```
graphsize 320, 320
cls
clg
for x = 0 to 320
plot x,160
next x
for y = 0 to 320
plot 160, y
next y
for x = 32 to 288
y = 160 — 50*tan(PI*(x-160)/320)
plot x,y
next x
```

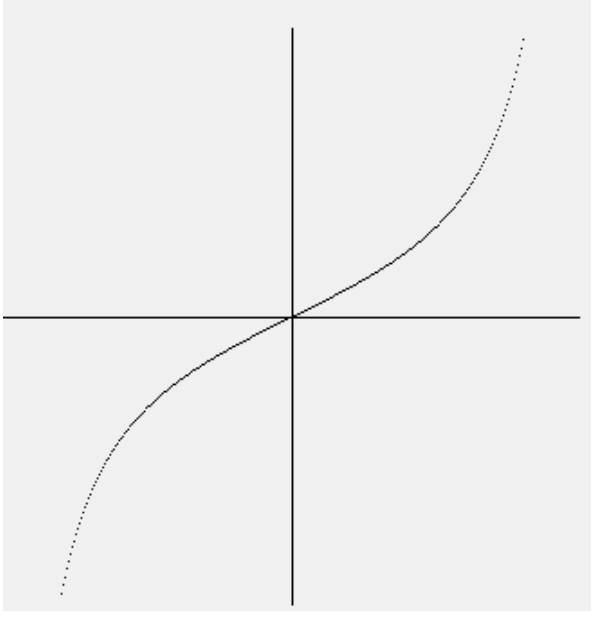

Рис. П2.13. График для задачи 10.5.13-2

Задача 10.5.14-2. Y = 0.5*(3^X + 3^(–X)) **для** –3 <= X <= 3.

Листинг для задачи 10.5.14-2

```
cls
clg
for x = 0 to 300
plot x,290
next x
for y = 0 to 300
plot 150, y
next y
for x = 0 to 300
y = 290 — 20*(3^((x — 150)/50)+3^((150 — x)/50))
plot x,y
next x
```

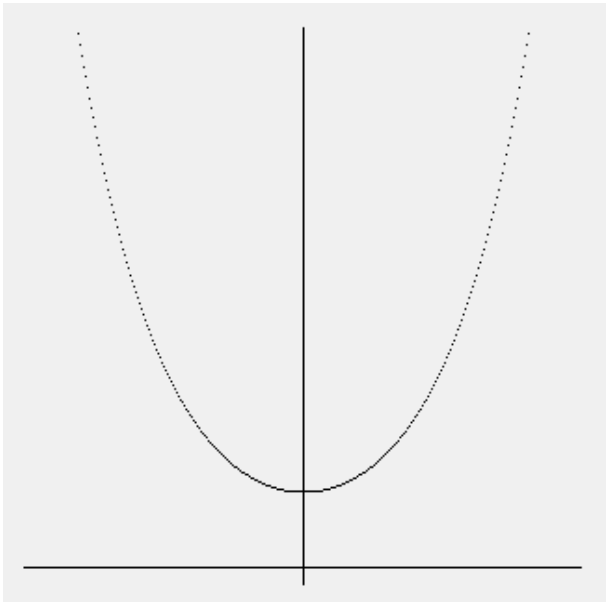

Рис. П2.14. График для задачи 10.5.14-2

Задача 10.5.15-2. Y = 0.5*(3^X – 3^(–X)) для –4 <= X <= 4.

Листинг для задачи 10.5.15-2

```
Cls
Clg
For x = 0 To 300
Plot x, 150
Next x
For y = 0 To 300
Plot 150, y
Next y
For x = 30 To 270
y = 150 — 1.5*(3^((x-150)/30)-3^((150-x)/30))
plot x, y
Next x
```

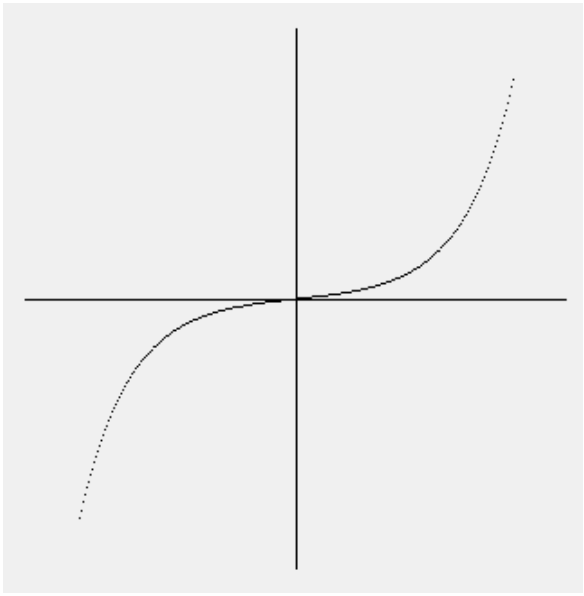

Рис. П2.15. График для задачи 10.5.15-2

Задача 10.5.16-2. Y = X^(1/4) для 0 <= X <= 10000.

Листинг для задачи 10.5.16-2

```
Cls
Clg
For x = 0 To 300
Plot x, 280
Next x
For y = 0 To 300
Plot 20, y
Next y
For x = 20 To 270
y = 280 — 25*((x-20)*40)^(1/4)
plot x, y
Next x
```

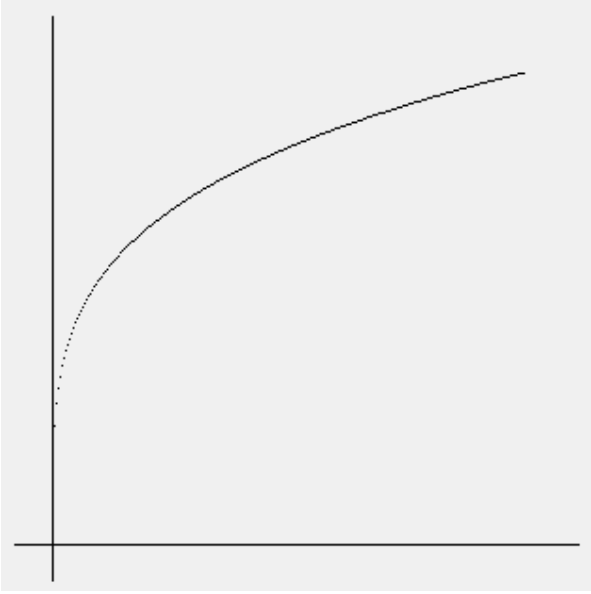

Рис. П2.16. График для задачи 10.5.16-2

Задача 10.5.17-2. Y = X^(5/3) для 0 <= X <= 30.

Листинг для задачи 10.5.17-2

```
Graphsize 320,320
Cls
Clg
For x = 0 To 320
Plot x, 310
Next x
For y = 0 To 320
Plot 10, y
Next y
For x = 10 To 310
y = 310 — ((x-10)/10)^(5/3)
plot x, y
Next x
```

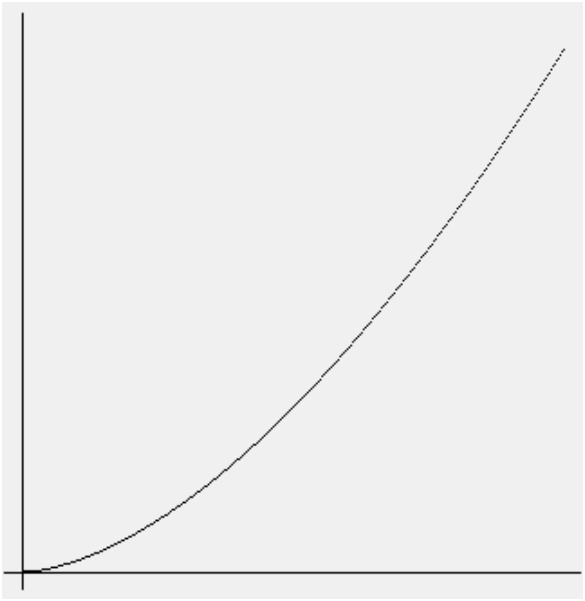

Рис. П2.17. График для задачи 10.5.17-2

Задача 10.5.18-2. Y = X^(3/5) для 0 <= X <= 100.

Листинг для задачи 10.5.18-2

```
Cls
Clg
For x = 0 To 300
Plot x, 270
Next x
For y = 0 To 300
Plot 20, y
Next y
For x = 20 To 270
y = 270 — 16*((x-20)/2.5)^(3/5)
plot x, y
Next x
```

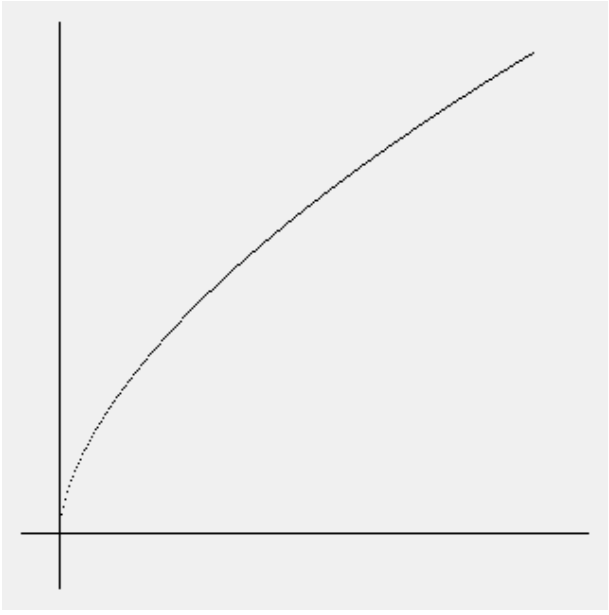

Рис. П2.18. График для задачи 10.5.18-2

Задача 10.5.19-2. Y = X^(5/4) для 0 <= X <= 50.

Листинг для задачи 10.5.19-2

```
Cls
Clg
For x = 0 To 300
Plot x, 280
Next x
For y = 0 To 300
Plot 20, y
Next y
For x = 20 To 270
y = 280-2*((x-20)/5)^(5/4)
plot x, y
Next x
```

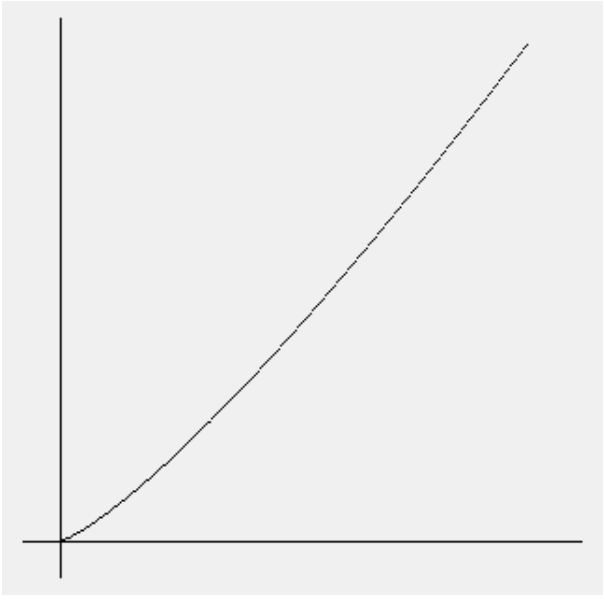

Рис. П2.19. График для задачи 10.5.19-2

Задача 10.5.20-2. Y = X^(4/5) для 0 <= X <= 100.

Листинг для задачи 10.5.20-2

```
Cls
Clg
For x = 0 To 300
Plot x, 290
Next x
For y = 0 To 300
Plot 20, y
Next y
For x = 20 To 270
y = 290-7*((x-20)/2.5)^(4/5)
plot x, y
Next x
```

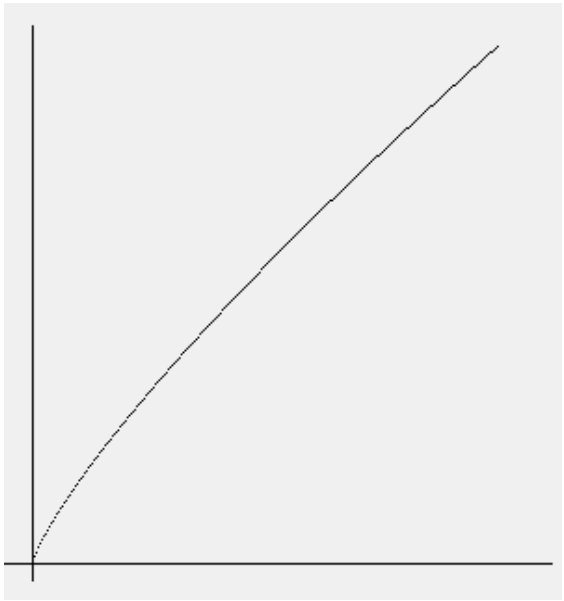

Рис. П2.20. График для задачи 10.5.20-2

Задача 10.5.21-2. Y = (2^X – 2^(–X))/(2^X + 2^(–X))
для –5 <= X <= 5.

Листинг для задачи 10.5.21-2

```
Cls
Clg
For x = 0 To 300
Plot x, 150
Next x
For y = 0 To 300
Plot 150, y
Next y
For x = 0 To 300
y1 = 2^((x-150)/30) — 2^((150-x)/30)
y2 = 2^((x-150)/30) + 2^((150-x)/30)
y = 150 — 150*y1/y2
plot x, y
Next x
```

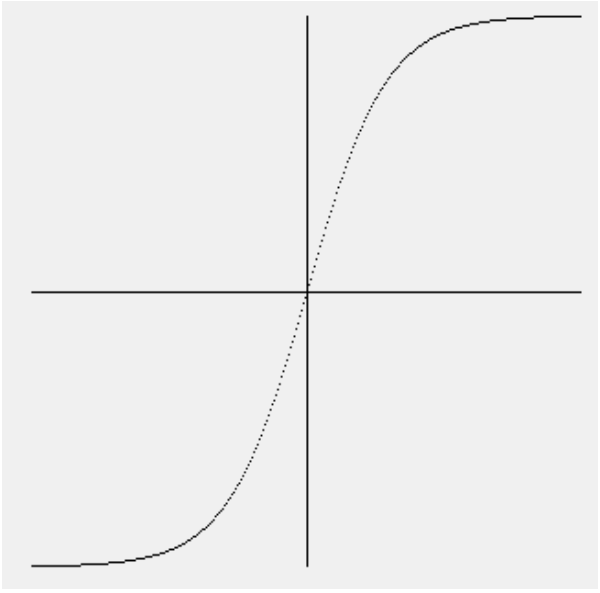

Рис. П2.21. График для задачи 10.5.21-2

Задача 10.5.22-2. Y = (2^X + 2^(–X))/(2^X – 2^(–X)) для 0.2 <= X <= 5.

Листинг для задачи 10.5.22-2

```
Cls
Clg
For x = 0 To 300
Plot x, 295
Next x
For y = 0 To 300
Plot 20, y
Next y
For x = 30 To 270
y1 = 2^((x-20)/50) — 2^((20-x)/50)
y2 = 2^((x-150)/30) + 2^((150-x)/30)
y = 295 — 40*y2/y1
plot x, y
Next x
```

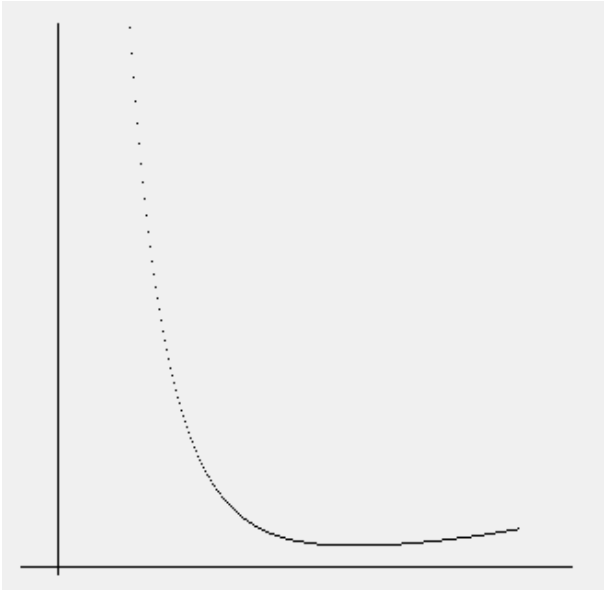

Рис. П2.22. График для задачи 10.5.22-2

Задача 10.5.23-2. Y = 1/X для 0.05 <= X <= 10.

Листинг для задачи 10.5.23-2

```
Cls
Clg
For x = 0 To 300
Plot x, 280
Next x
For y = 0 To 300
Plot 20, y
Next y
For x = 21 To 220
y = 280 — 200/(x-20)
plot x, y
Next x
```

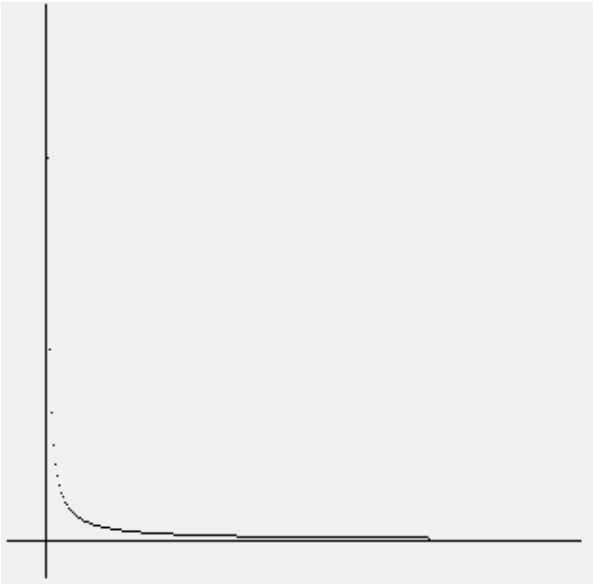

Рис. П2.23. График для задачи 10.5.23-2

Задача 10.5.24-2. Y = 1/X^(1/2) для 0.01 <= X <= 100.

Листинг для задачи 10.5.24-2

```
Cls
Clg
For x = 0 To 300
Plot x, 280
Next x
For y = 0 To 300
Plot 20, y
Next y
For x = 20.02 To 220
y = 280 — 20/((x-20)/2)^(1/2)
plot x, y
Next x
```

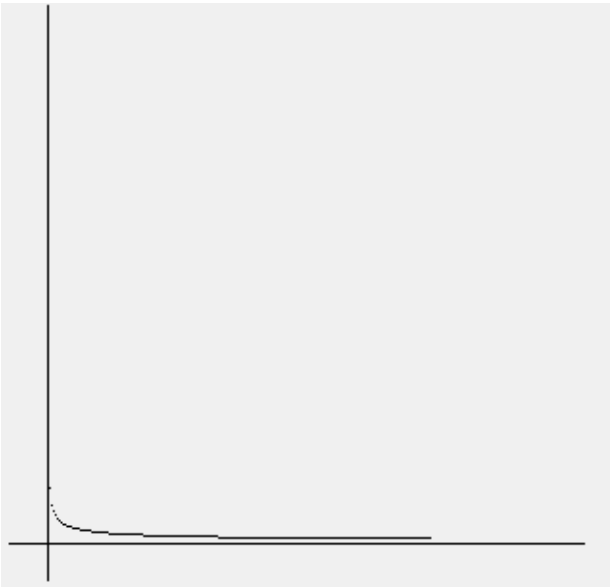

Рис. П2.24. График для задачи 10.5.24-2

Задача 10.5.25-2. Y = 1/(X^2) для 0.2 <= X <= 5.

Листинг для задачи 10.5.25-2

```
Cls
Clg
For x = 0 To 300
Plot x, 280
Next x
For y = 0 To 300
Plot 20, y
Next y
For x = 30 To 270
y = 280 — 25000/(x — 20)^2
plot x, y
Next x
```

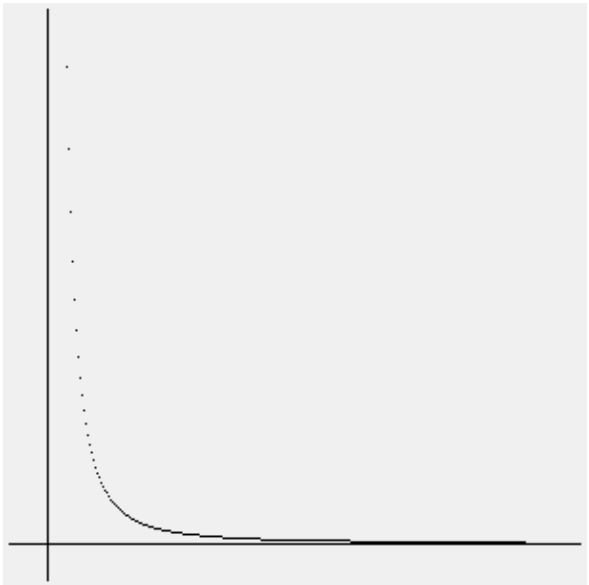

Рис. П2.25. График для задачи 10.5.25-2

Задача 10.5.26-2. Y = 2.5*X^2 – 5.7*X + 23.5 для –5 <= X <= 5.

Листинг для задачи 10.5.26-2

```
Cls
Clg
For x = 0 To 300
Plot x, 270
Next x
For y = 0 To 300
Plot 150, y
Next y
For x = 0 To 300
x1 = (x-150)/30
y = 270 — 5*x1*x1 + 11.4*x1 — 47
plot x, y
Next x
```

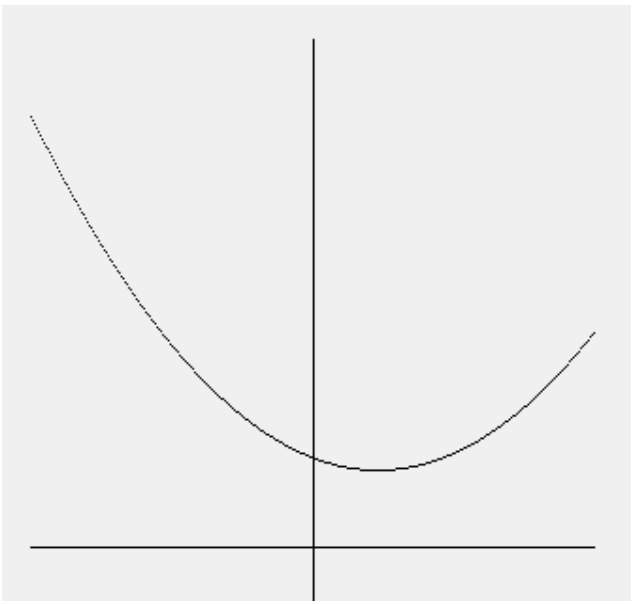

Рис. П2.26. График для задачи 10.5.26-2

Задача 10.5.27-2. Y = X^3 – 6*X^2 + 10*X – 20
для –10 <= X <= 10.

Листинг для задачи 10.5.27-2

```
Cls
Clg
For x = 0 To 300
Plot x, 90
Next x
For y = 0 To 300
Plot 150, y
Next y
For x = 0 To 300
x1 = (x — 150)/15
y = 90 — 0.1*(x1*x1*x1 — 6*x1*x1 + 10*x1 — 20)
plot x, y
Next x
```

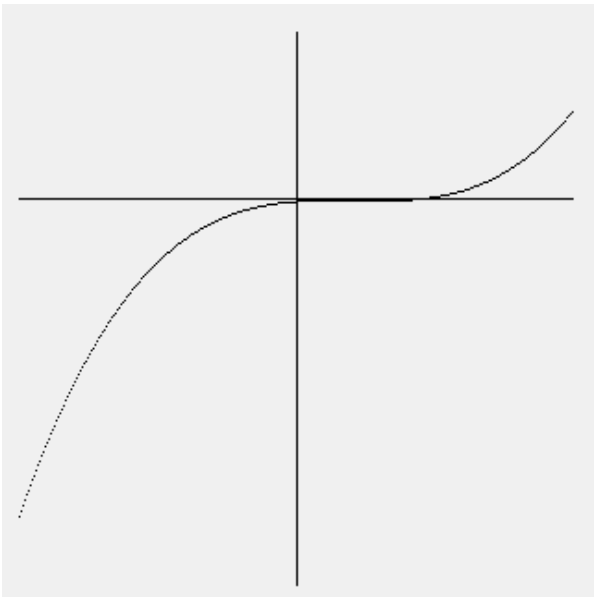

Рис. П2.27. График для задачи 10.5.27-2

Задача 10.5.28-2. Y = 1/(X^2 + 6*X + 9) для –2.8 <= X <= 5.

Листинг для задачи 10.5.28-2

```
Cls
Clg
For x = 0 To 300
Plot x, 280
Next x
For y = 0 To 300
Plot 120, y
Next y
For x = 36 To 270
x1 = (x — 120)/30
y = 280 — 10/(x1*x1+6*x1+9)
Plot x,y
Next x
```

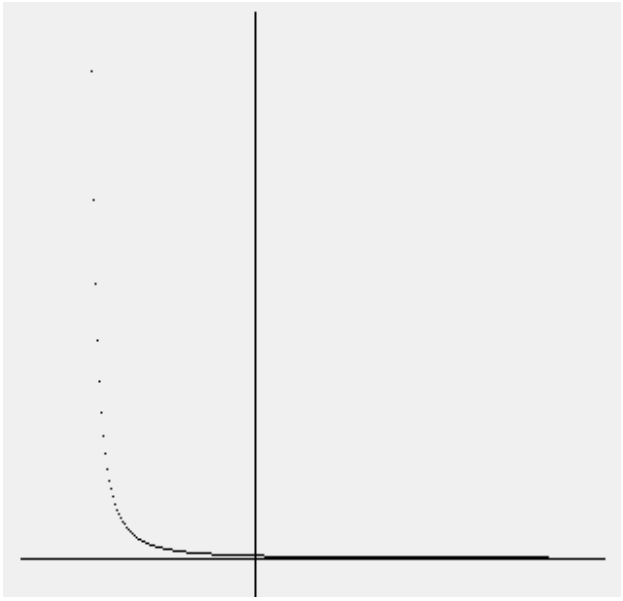

Рис. П2.28. График для задачи 10.5.28-2

Задача 10.5.29-2. Y = 10/X^(1/3) для 0.2 <= X <= 8.

Листинг для задачи 10.5.29-2

```
Cls
Clg
For x = 0 To 300
Plot x, 280
Next x
For y = 0 To 300
Plot 20, y
Next y
For x = 26 To 260
y = 280 — 100/((x — 20)/30)^(1/3)
Plot x,y
Next x
```

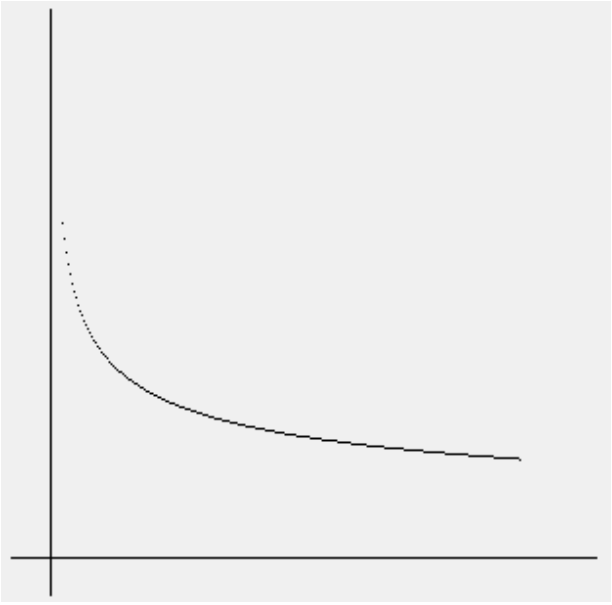

Рис. П2.29. График для задачи 10.5.29-2

Задача 10.5.30-2. Y = tg X/2 для –3 <= X <= 3.

Листинг для задачи 10.5.30-2

```
Cls
Clg
For x = 0 To 300
Plot x, 150
Next x
For y = 0 To 300
Plot 150, y
Next y
For x = 0 To 300
y = 150 — 10*tan((x — 150)/100)
Plot x,y
Next x
```

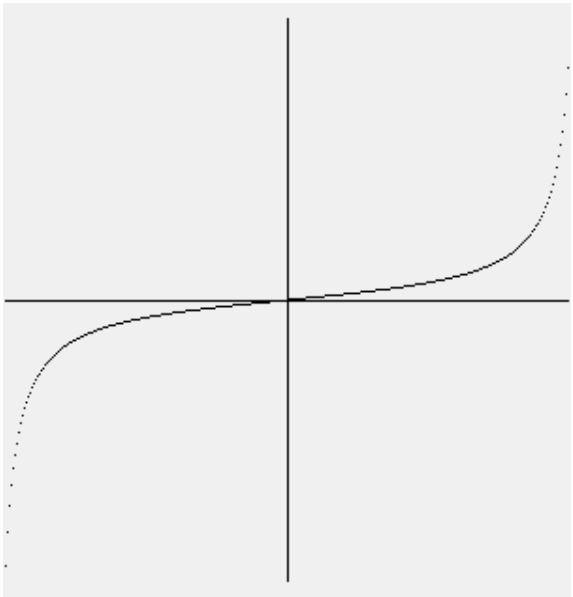

Рис. П2.30. График для задачи 10.5.30-2

Задача 10.5.31-2. Y = 1000*(1 + X/100)^5.

Листинг для задачи 10.5.31-2

```
Cls
Clg
For x = 0 To 300
Plot x, 280
Next x
For y = 0 To 300
Plot 20, y
Next y
For x = 25 To 220
y = 280 — 100*(1 + (x — 20)/1000)^5
Plot x,y
Next x
```

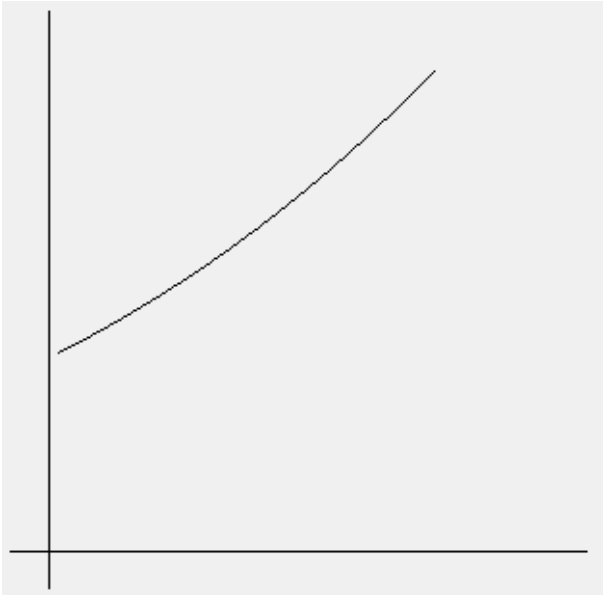

Рис. П2.31. График для задачи 10.5.31-2

Задача 10.5.32-2. Y = 1000*1.2^P. 1/12 <= P <= 20.

Листинг для задачи 10.5.32-2

```
Cls
Clg
For p = 0 To 300
Plot p, 270
Next p
For y = 0 To 300
Plot 20, y
Next y
For p = 21 To 260
y = 270 — 5*1.2^((p — 20)/12)
Plot p,y
Next p
```

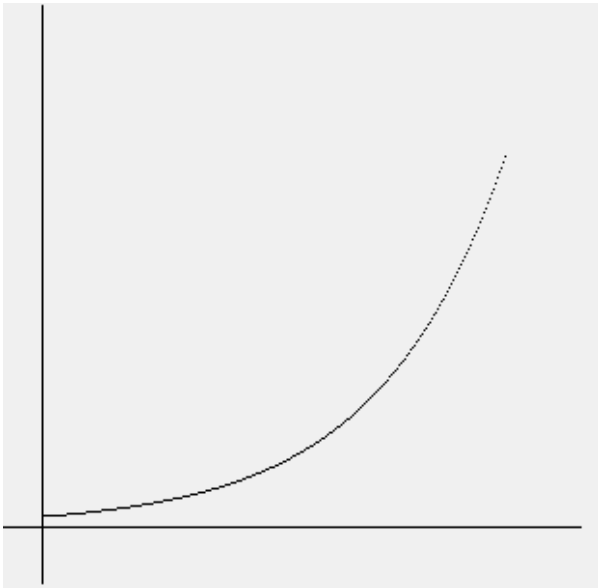

Рис. П2.32. График для задачи 10.5.32-2

Задача 10.5.33-2. P = Ln 1.5/Ln(1 + X/100) для 1 <= X <= 40.

Листинг для задачи 10.5.33-2

```
Cls
Clg
For x = 0 To 300
Plot x, 290
Next x
For p = 0 To 300
Plot 20, p
Next p
For x = 27 To 300
p = 7*Log(1.5)/Log(1 + (x — 20)/700)
Plot x, p
Next x
```

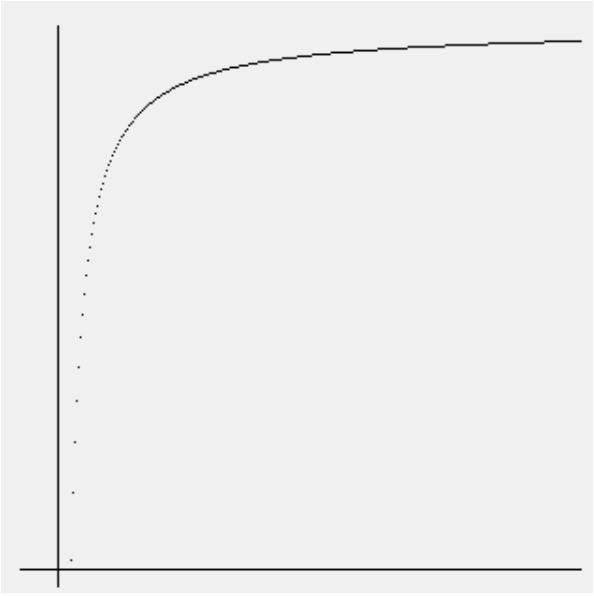

Рис. П2.33. График для задачи 10.5.33-2

Задача 10.5.34-2. P = Ln(Y/1000)/Ln(1.2) для 1100 <= 3000.

Листинг для задачи 10.5.34-2

```
Cls
Clg
For y = 0 To 300
Plot y, 280
Next y
For p = 0 To 300
Plot 0, p
Next p
For y = 110 To 300
p = 280 — 40*Log(y/100)/Log(1.2)
plot y, p
Next y
```

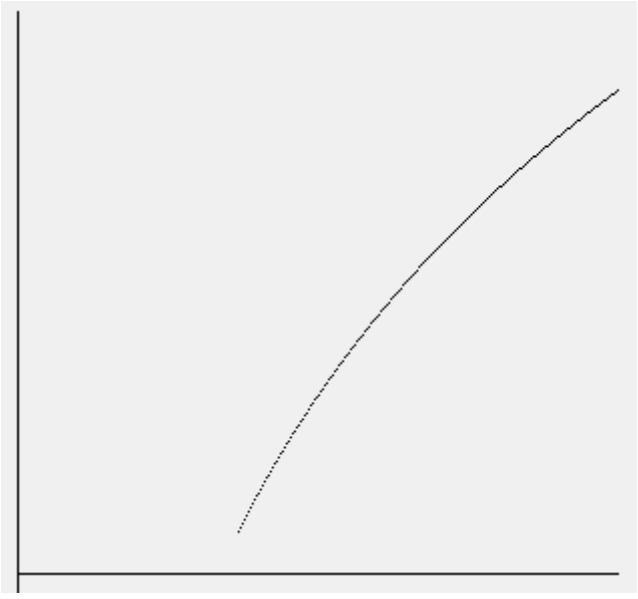

Рис. П2.34. График для задачи 10.5.34-2

Задача 10.5.35-2. Y = 40/X для X = 10 – 100 (зависимость относительного спроса на кредиты Y от банковского процента X).

Листинг для задачи 10.5.35-2

```
Cls
Clg
For x = 0 To 300
Plot x, 280
Next x
For y = 0 To 300
Plot 0, y
Next y
For x = 30 To 300
y = 280 — 7200/x
plot x, y
Next x
```

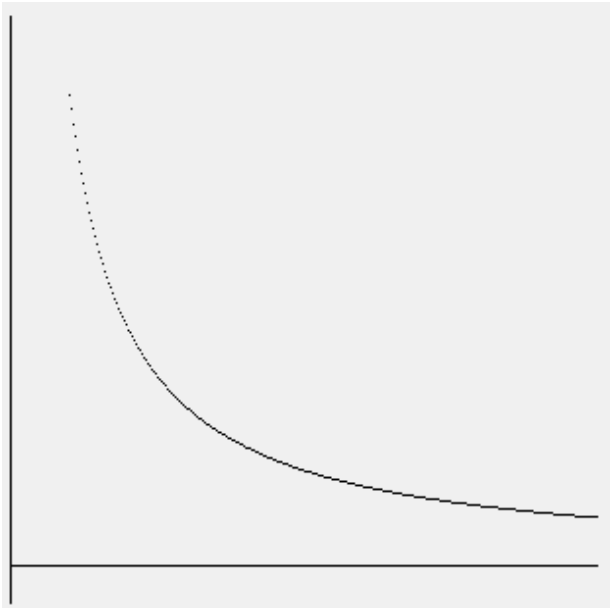

Рис. П2.35. График для задачи 10.5.35-2

Задача 10.5.36-2. Y = 1.1^(X-10) – 1 для 10 <= X <= 100 — зависимость предложения сбережений Y в процентах от дохода от относительного дохода X по отношению к минимальной зарплате).

Листинг для задачи 10.5.36-2

```
Cls
Clg
For x = 0 To 300
Plot x, 290
Next x
For y = 0 To 300
Plot 0, y
Next y
For x = 30 To 300
y = 290 — 0.05*(1.1^(x/3 — 10) — 1)
plot x, y
Next x
```

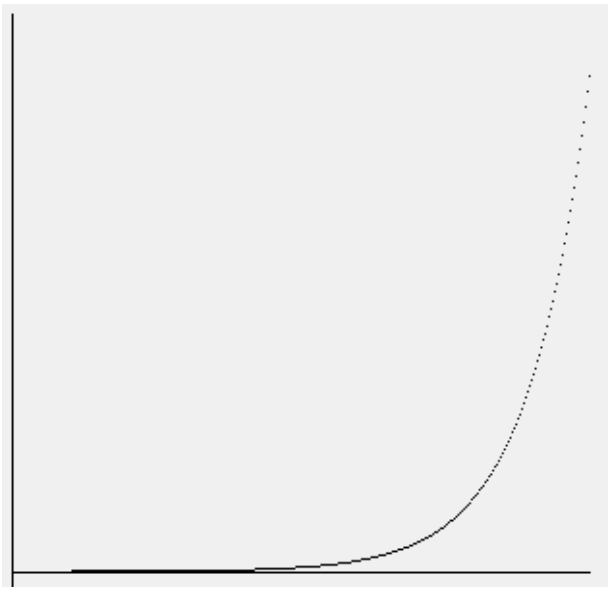

Рис. П2.36. График для задачи 10.5.36-2

Задача 10.5.37-2. Y = 3/X для 0.3 <= X <= 20 — зависимость относительного спроса на товар Y от его относительной цены X.

Листинг для задачи 10.5.37-2

```
Cls
Clg
For x = 0 To 300
Plot x, 280
Next x
For y = 0 To 300
Plot 0, y
Next y
For x = 4.5 To 300
y = 280 — 1125/x
plot x, y
Next x
```

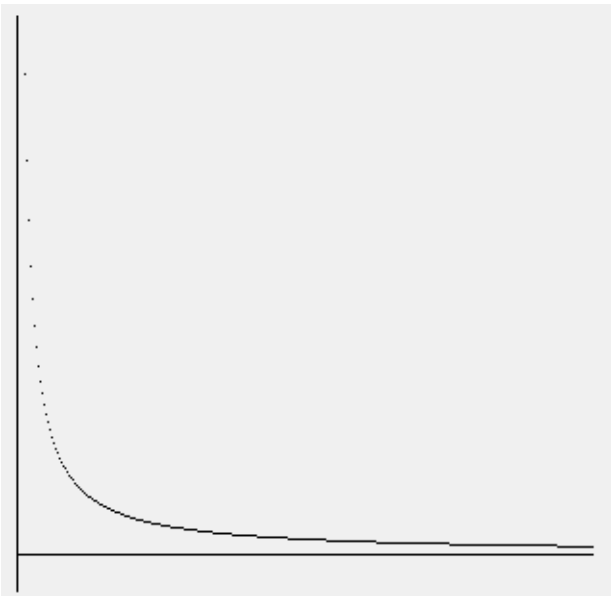

Рис. П2.37. График для задачи 10.5.37-2

Задача 10.5.38-2. Y = 1.3*X^(3/2) для 1 <= X <= 20 — зависимость от относительного спроса Y на продовольственные товары в условиях инфляции от относительной цены X.

Листинг для задачи 10.5.38-2

```
Cls
Clg
For x = 0 To 300
Plot x, 280
Next x
For y = 0 To 300
Plot 0, y
Next y
For x = 15 To 300
y = 280 — 2.6*(x/15)^(3/2)
plot x, y
Next x
```

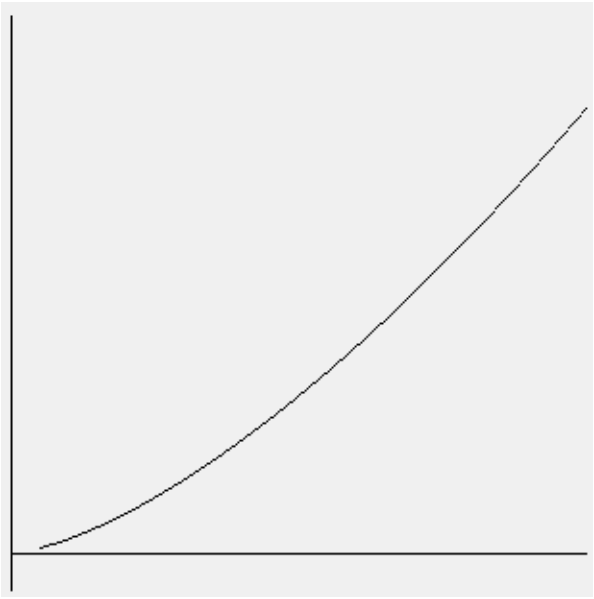

Рис. П2.38. График для задачи 10.5.38-2

Задача 10.5.39-2. H = 10000 – 4.9*T^2 0 <= T <= 45.2 — зависимость высоты бомбы H в метрах от времени T в секундах от момента сброса с самолета (T = 0) до момента попадания в цель (H = 0).

Листинг для задачи 10.5.39-2

```
Cls
Clg
For t = 0 To 300
Plot t, 280
Next t
For h = 0 To 300
Plot 20, h
Next h
For t = 20 To 246
h = 280 — (10000 — 4.9*(t-20)*(t-20)/25)/40
plot t, h
Next t
```

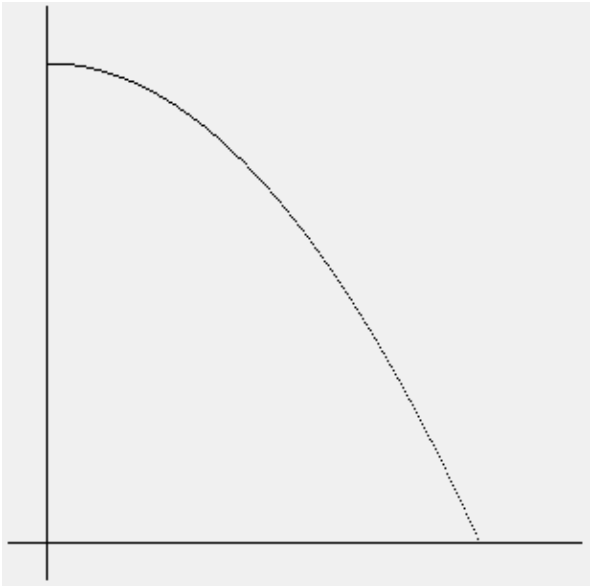

Рис. П2.39. График для задачи 10.5.39-2

Задача 10.5.40-2. S = 20*T – T^2 для 0 <= T <= 10 — зависимость пути S автомобиля в метрах от времени в секундах от момента начала торможения (T = 0) до полной остановки (скорость V = 0).

Листинг задачи 10.5.40-2

```
Graphsize 320, 320
Cls
Clg
For t = 0 To 320
Plot t, 300
Next t
For s = 0 To 320
Plot 20, s
Next s
For t = 20 To 320
s = 300 — 2*(t-20) + (t-20)*(t-20)/300
plot t, s
Next t
```

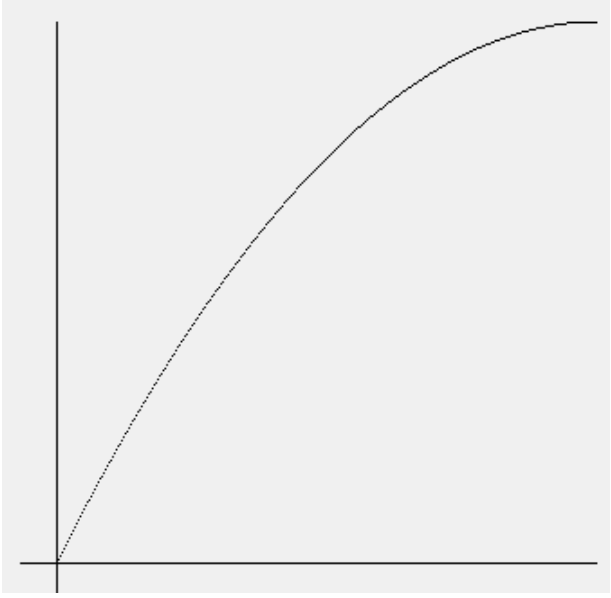

Рис. П2.40. График для задачи 10.5.40-2

Литература

1. **http://kidbasic.sourceforge.net** — сайт группы разработчиков BASIC-256.
2. **http://sourceforge.net/projects/kidbasic/files/basic256/BASIC256-0.9.6p_Win32_Install.exe/download** — загрузка русской версии дистрибутива BASIC-256 v.0.9.6p (12,1 Мбайт).
3. **http://kidbasic.sourceforge.net/en/reference.html** — английская версия руководства по BASIC-256 v.0.9.5 — BASIC-256 Reference 0.9.5.
4. **http://kidbasic.sourceforge.net/de/reference.html** — версия руководства по BASIC-256 v.0.9.5 на немецком языке.
5. RU-BASIC-256 Reference 0.9.6p — встроенный справочник BASIC-256 v.0.9.6p. Вызывается по клавише <F1> после установки программы в ОС Windows XP/Vista/7.
6. **http://ege.edu.ru** — основной сайт информационной поддержки ЕГЭ.
7. **http://www.basicbook.org** — James M. Reneau. So You Want To Learn To Program? Free eBook Edition.

С. Г. Никитенко

Свободное программное обеспечение. BASIC-256 для школы

Группа подготовки издания:

Главный редактор	*Екатерина Кондукова*
Зам. главного редактора	*Людмила Еремеевская*
Зав. редакцией	*Григорий Добин*
Компьютерная верстка	*Ольги Сергиенко*
Корректор	*Зинаида Дмитриева*
Дизайн серии	*Инны Тачиной*
Оформление обложки	*Елены Беляевой*
Зав. производством	*Николай Тверских*

Лицензия ИД № 02429 от 24.07.00. Подписано в печать 31.05.11.
Формат 60×90 1/16. Печать офсетная. Усл. печ. л. 14.
Тираж 1500 экз. Заказ №
"БХВ-Петербург", 190005, Санкт-Петербург, Измайловский пр., 29.

Санитарно-эпидемиологическое заключение на продукцию
№ 77.99.60.953.Д.005770.05.09 от 26.05.2009 г. выдано Федеральной службой по надзору в сфере защиты прав потребителей и благополучия человека.

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