И. Сафронов

Бейсик
в задачах и примерах
2-е издание

+150 новых задач

Базовые понятия программирования для начинающих

Линейные, разветвляющиеся и циклические алгоритмы

Графика и графики

Массивы, файлы и подпрограммы

Строки и сортировки

450 задач, примеры и решения

И. Сафронов

Бейсик
в задачах и примерах

2-е издание

Санкт-Петербург
«БХВ-Петербург»
2006

УДК 681.3.06
ББК 32.973.26-018.1Basic
С12

Сафронов И. К.

С12 Бейсик в задачах и примерах. — 2-е изд., перераб. и доп. — СПб.: БХВ-Петербург, 2006. — 320 с.: ил.

ISBN 5-94157-527-0

В сборнике содержатся разработанные автором задачи и примеры для освоения ключевых понятий программирования с использованием языка Бейсик. В занимательной и доступной форме осваиваются виды алгоритмов, переменные, операторы, массивы, подпрограммы. Большое внимание уделяется наиболее популярной среди школьников теме графики. Второе издание книги обусловлено неспадающим читательским интересом, дополнено 150-ю новыми задачами, примерами удивительных и увлекательных игр и программ. Также книга содержит справочник по языку программирования QBasic.

Может использоваться в качестве задачника для учащихся старшей школы.

Для начинающих программистов

УДК 681.3.06
ББК 32.973.26-018.1Basic

Группа подготовки издания:

Главный редактор	*Екатерина Кондукова*
Зам. главного редактора	*Людмила Еремеевская*
Зав. редакцией	*Григорий Добин*
Редактор	*Руслан Абдрахманов*
Компьютерная верстка	*Ольги Сергиенко*
Корректор	*Зинаида Дмитриева*
Дизайн серии	*Игоря Цырульникова*
Оформление обложки	*Елены Беляевой*
Зав. производством	*Николай Тверских*

Лицензия ИД № 02429 от 24.07.00. Подписано в печать 22.12.05.
Формат 60×90 $^{1}/_{16}$. Печать офсетная. Усл. печ. л. 20.
Тираж 5000 экз. Заказ №
"БХВ-Петербург", 194354, Санкт-Петербург, ул. Есенина, 5Б.

Санитарно-эпидемиологическое заключение на продукцию
№ 77.99.02.953.Д.006421.11.04 от 11.11.2004 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

ISBN 5-94157-527-0

Оглавление

Предисловие

О предлагаемой книге

Прошло пять лет с момента выхода в свет первого издания этой книги. Она выдержала несколько допечаток тиража, что свидетельствовало об интересе читателей к излагаемому предмету.

За это время многое изменилось в аппаратной части компьютерной техники, появились новые мощные языки программирования, но, на мой взгляд, не угасает интерес к старому доброму Бейсику.

В новом, исправленном и дополненном издании, я добавил более 220 новых задач и примеров их решения, а также, для фанатов Бейсика, привел много примеров заставок, игр и полезных программ, которые можно либо использовать готовыми, либо (что гораздо ценнее ☺), дополнить, развить и улучшить собственными силами.

Задачи из этой книги могут быть применены для решения на любых других языках программирования, послужат хорошим тренингом для воспитания алгоритмического мышления у начинающих программистов.

Представляемая книга содержит опыт, накопленный автором за время работы преподавателем информатики в школе. Пятнадцать лет назад мы учили школьников работать на программируемых калькуляторах, сегодня — на самой современной вычислительной технике. Но, в любом случае, убеждение, что преподавание основ программирования в школе необходимо, осталось до сих пор, хотя и претерпело какие-то изменения.

Алгоритмизация мышления позволяет человеку выживать в бушующем море информации, формирует системный подход к любым жизненным ситуациям.

Кроме того, за этот пятнадцатилетний период работы в школе автору очень редко приходилось встречать хорошие задачники по программированию, которые были бы насыщены разнообразными примерами. А еще, полагая, что Бейсик сейчас преподается в большинстве своем школьникам и, по опыту зная, насколько им интереснее "живые", не сухие задания, автор попытался вести разговор с читателем на понятном им языке, включая иногда и какие-то сленговые компьютерные словечки.

В данной книге в предисловии вашему вниманию предлагается ряд сведений о языках программирования, в том числе и о языке Бейсик. В первой части вы найдете большой набор авторских и творчески обработанных задач, охватывающих все основные разделы программирования на языке начинающих программистов — Бейсик. Задачи сгруппированы по темам и, в основном, расположены по степени возрастания сложности.

Во второй части вы сможете найти решения некоторых задач с необходимыми пояснениями, а в третьей части и в приложении самые любознательные отыщут много интересных дополнительных сведений об изучаемом языке и представлении информации в компьютере.

Немного истории

Давайте для начала договоримся об определениях, чтобы в дальнейшем говорить на одном языке, поскольку язык алгоритмов должен быть понимаемым ясно и однозначно.

Под *языком программирования* мы будем понимать совокупность средств и правил представления алгоритма в виде, приемлемом для компьютера. Отсюда неискушенный читатель может решить, что компьютер, оказывается, при всей его кажущейся могущественности, не поймет задачи, поставленной ему на простом человеческом языке, будь то русский, английский или даже китайский. Таким образом, существует разделение всех языков программирования на две большие группы — языки высокого и

низкого уровней. Человек считает себя венцом творения (с этим можно согласиться, но можно и поспорить, если внимательней присмотреться к некоторым таким "венцам"), поэтому языком самого высокого уровня считается человеческий язык, и когда компьютер станет его легко понимать, то он вплотную приблизится к человеку. Языком самого низкого уровня считается язык так называемых машинных кодов. Все остальные алгоритмические языки лежат где-то посередине. Например, к языкам низкого уровня принадлежат так называемые языки семейства ассемблеров. Их достоинство в том, что они почти не требуют перевода для компьютера, и он практически сразу выполняет алгоритм. Есть, однако, существенный недостаток — писать программы на таких языках может только очень опытный программист, и получаются они слишком громоздкими. Напротив, языки высокого уровня в достаточно сильной степени приближены к человеческому (чаще к английскому) языку — это и Фортран, и Паскаль, и Си, но выполнение алгоритма компьютером в данном случае несколько тормозится предварительным переводом на язык машинных кодов.

К языкам высокого уровня принадлежит и тот, который рассматривается в данной книге. Он служит своего рода переводчиком между человеком и компьютером, помогая им понять друг друга и совместно добиваться решения поставленных задач. Разработан первый Бейсик в 1964 г. сотрудниками Дартмутского колледжа Дж. Кемени и Т. Курцем. Интересно происхождение названия языка. В прошлом веке один английский миссионер выделил из английского языка триста наиболее употребительных слов, назвал их Basic English и стал обучать туземцев. Опыт оказался весьма успешным, и контакты с аборигенами значительно упростились. Создатели языка Бейсик стремились достигнуть того же эффекта — облегчить понимание между "туземцами" — начинающими программистами, и компьютерами. Аббревиатура BASIC так и расшифровывается — "Beginner's All purpose Symbolic Instruction Code", что в переводе значит "многоцелевой язык символических команд для начинающих".

Идея оказалась удачной, и на десятилетия язык Бейсик стал основным в деле вовлечения в программирование новых и новых

адептов. У автора этой книги есть большое количество выпускников, давно превзошедших своего учителя в деле программирования, успешно работающих у нас и за рубежом, но начинавших с того же Бейсика — тогда еще для ZX-Spectrum. Даже в те времена на несовершенной, зачастую собранной своими руками, технике они писали очень приличные прикладные программы — и для дела, и для игры. Большое достоинство Бейсика, из-за которого его изучение продолжается в школах и поныне, — это возможность создавать диалоговые программы.

Ныне Бейсик вышел за рамки языка для начинающих, и его могучий потомок — Visual Basic позволяет творить на компьютере просто чудеса.

Но вернемся к нашему старому доброму Бейсику. За эти годы было создано несколько его версий — GW-Basic, MSX-Basic, TurboBasic, QuickBasic. Автор за время своей работы в школе программировал на всех этих версиях и практически на всех видах техники, начиная от БК и "Корвета" и заканчивая (на момент написания книги) Pentium 4. И все же в качестве основного для этой книги выбрал Russian QuickBasic. Но я уверен, что искушенный читатель знает, а неискушенный пусть успокоится, все эти версии очень похожи друг на друга базовым набором операторов и конструкций, и, изучив какую-либо одну версию, очень легко перейти на другую. Это примерно как различия русского языка в Санкт-Петербурге и Москве: мы говорим "карточка", а москвичи "проездной", мы говорим "ларек", а они — "палатка", мы произносим "дожди", а они — "дожжи". Можно бесконечно подтрунивать над этими небольшими различиями, но мы прекрасно друг друга понимаем. Итак, давайте говорить на великом и могучем языке высокого уровня — Бейсике.

О языках: интерпретаторы и компиляторы

Как уже было сказано выше, языки высокого уровня — это своего рода посредники в общении между человеком и компьютером. Непосредственно переводом задуманного человеком алго-

ритма с языка программирования на язык машинных кодов занимаются программы-трансляторы.

Трансляторы, в свою очередь, тоже делятся на две большие группы — интерпретаторы и компиляторы.

Компиляторы сначала переводят всю программу, написанную на алгоритмическом языке, в машинные коды, и после этого очень быстро исполняют ее. Быстрота выполнения — это плюс компиляторов. Но они требуют довольно большой предварительной работы, поскольку мы сможем увидеть результат выполнения программы только после успешной компиляции — перевода, а на этом этапе программа-компилятор обычно требует устранить все синтаксические ошибки. Поэтому невозможность видеть промежуточные результаты — это небольшой минус компиляторов.

К компиляторам принадлежат, например, языки Паскаль, Си, Турбо Бейсик.

Интерпретаторы покомандно переводят алгоритм с языка программирования на язык машинных кодов и тут же исполняют переведенную команду. В случае допущенной ошибки программа-интерпретатор прекращает работу и просит исправить неверную конструкцию. К интерпретаторам относятся как раз в основном языки семейства Бейсик. В том числе и рассматриваемый в этой книге Russian QuickBasic.

Достоинство интерпретаторов — в возможности видеть промежуточные результаты выполнения алгоритма и по ходу дела вносить в исполняемый алгоритм изменения. Недостаток — гораздо более медленная работа по сравнению с компиляторами.

Для большего понимания я бы сравнил процесс трансляции с процессом перевода текста с иностранного языка на русский. Компиляция — это письменный перевод, когда я получаю текст и целиком его перевожу, чтобы затем, уже на иностранном, его довольно быстро изложить. Интерпретация — это синхронный перевод, когда я после каждой произнесенной фразы ее перевожу.

Ну а теперь, если вышеизложенное не вызвало затруднений в понимании, вперед, к сверкающим вершинам Бейсика.

Дружественный интерфейс

Правда, у подножия сияющих вершин ваших будущих программных достижений я вас немножко приторможу с тем, чтобы вы отчетливо понимали, что плодами вашего творчества будут пользоваться и другие люди, зачастую ничего не сведущие в программировании. А поэтому я коротко изложу свои взгляды на мои представления об оформлении программ или о так называемом дружественном интерфейсе. Ну, кто хоть немного знает английский, тот уже может догадаться, что "интер" — это "между", а "фэйс" — это то, что бывает об "тэйбл", т. е. "лицо". Соответственно, интерфейс — "между лицами", в данном случае имеет смысл в значении оформления диалога между пользователем и компьютером в процессе исполнения написанной программистом (вами, уважаемый читатель!) программы.

Кстати, короткое лирическое отступление об английском языке в деле программирования. Это не роскошь, а суровая необходимость. Опять же вспоминаю с улыбкой своих выпускников, особенно тех, кто учил в школе немецкий. Мой программистский сленг обогатился такими терминами, как "филе наме" (file name), "гамовер" (game over), "лине" (line), "цирцле" (circle). Но произношение — это как раз не самое страшное, а вот писать и понимать ограниченный набор (Basic English) англоязычных терминов необходимо.

Итак, несколько моих принципов по оформлению программ:

- "Кашу маслом не испортишь" — как можно больше поясняющих комментариев к вашему алгоритму, и пользователям вашей программы хорошо, и (что немаловажно для учащихся) преподаватель всегда очень радуется. Но самое главное, лично вам будет очень просто редактировать и отлаживать программу. К сожалению, этот принцип зачастую игнорируется начинающими программистами, что снижает как текущие оценки, так и олимпиадные, а кроме того, очень затрудняет понимание сути алгоритма и нахождение возможностей для его улучшения.
- "Fool Proof" — защита от дурака. Ваша программа должна выдерживать натиск неграмотного пользователя (от хакера-то

все равно защиты не найти!) и сопротивляться его попыткам при исполнении программы ввести цифры, где надо было имя, или нажать клавишу <Пробел>, когда была указана клавиша <Enter>.

- Доходчивость. При написании программы не стоит рассчитывать, что работать с ней будут умные люди (хотя и такие встречаются среди пользователей), а потому при исполнении вашего алгоритма программа должна максимально подробно объяснять человеку, что она от него хочет, и какую "пимпочку" в данный момент надо нажать.
- Ну и конечно, "красота спасет мир". Пусть вы делаете даже просто программу проверки знания таблицы умножения, но сделайте это красиво — примените цветовое и шрифтовое оформление, а можно ведь и звуковые эффекты. Не скупитесь на экранные похвалы в адрес пользователя вашей программы — ласковое слово и кошке приятно!

В общем, я попробую в тех случаях, где буду приводить решения каких-то, на мой взгляд, ключевых обобщающих заданий, соблюсти эти принципы. А вам рекомендую брать с них пример.

Основные этапы решения алгоритмических задач на компьютере

Подход многих моих учеников к решению поставленных мной задач по программированию достаточно однообразен и в народе носит название "метода тыка". Получив задание, такие, с позволения сказать, "программисты" садятся к компьютеру и начинают комбинировать операторы языка Бейсик в различных вариантах, пытаясь достичь требуемого результата. Да, иногда получается. Но чаще — нет. Почему? Потому что отсутствует системный подход к задаче, что, во-первых, разбазаривает самый драгоценный человеческий ресурс — время, а во-вторых, не приносит плодов ни в программировании, ни в жизни.

Предлагаю метод, как, на мой взгляд, надо подходить к решению программистских задач. А вы уж решайте сами. "Это вам не ме-

лочь по карманам тырить", — как говорил незабвенный Остап Бендер, — "тут думать надо!".

Итак, этапы решения алгоритмических задач на компьютере.

1. *Постановка задачи*. Один из самых главных этапов. Вы должны добиться от того, кто дает вам задачу (это можете быть и вы сами), ясной и четкой ее постановки. Вы однозначно и вполне определенно должны понять, что будет результатом решения задачи. Каковы исходные данные? Существуют ли ограничения для этих данных? Можно сказать, что точность и четкость в постановке задачи — это половина дела. Напротив, в случае недопонимания каких-то моментов вероятность непроизводительной траты времени и отрицательного результата резко возрастает.
2. Следующий этап — решение вопроса "Как будет реализовываться поставленная задача?". Как достичь требуемых результатов? Каковы способы и методы достижения уясненных на первом этапе целей?
3. После первых двух этапов наступает пора еще одного очень важного момента — этапа *разработки алгоритма решения* поставленной задачи, т. е. структуризация, разбиение задачи на последовательность простых модулей, каждый из которых легко может быть реализован на языке программирования.
4. Очередной этап — непосредственный перевод словесного алгоритма или его блок-схемы на выбранный язык программирования и ввод полученной программы в компьютер.
5. После ввода программы обычно выясняется, что где-то мы допустили просто синтаксические ошибки, где-то недоработали алгоритм, где-то не хватает исходных данных и т. д. Поэтому теперь начинается *отладка программы*, иными словами, устранение ошибок и неточностей, допущенных на предыдущих этапах.
6. После того как программа заработала, необходимо проверить ее на правильность работы, используя набор контрольных данных (в тех случаях, где это возможно). Так, например, если мы написали программу для расчета корней квадратного уравнения по заданным коэффициентам, то можем проверить

работу программы, вводя такие коэффициенты, для которых предварительно были рассчитаны значения корней или их отсутствие. Это так называемый *тестовый этап*.

7. После тестового этапа (если программа его выдержала!) можно применять программу по назначению. Ну и последнее. Страна должна знать своих героев, а потому завершающим этапом работы по решению алгоритмической задачи следует считать *документирование*, т. е. распечатку листинга программы, снабженную необходимыми комментариями автора. С этого момента разработанная программа становится интеллектуальной собственностью программиста, и им начинает гордиться семья и школа.

ЧАСТЬ 1
Язык Бейсик

Прежде всего, очень важное замечание! Под операционными системами Windows 2000 и Windows XP изначально не поддерживается кириллический шрифт при работе с графикой, поэтому вы рискуете увидеть вместо нормальных пунктов меню и комментариев в своей программе следующие "иероглифы":

```
·tŻŹ ąžŻŘ ŷŰĽ źŹžŵąŲŷŨũ Ũ ŽąĽŨŻąŵŘŷńũ űŹũ`·
```

Выходы есть:

1. Найти в Интернете русификатор под именно эти операционные системы.
2. Работать не в Russian QuickBasic, а просто в QBasic.
3. Писать русские слова транслитом (т. е. заменять русские буквы латинскими аналогами, например "привет!" — "privet!".
4. Параллельно с Windows 2000 или XP поставить Windows'98 и наслаждаться Russian QuickBasic!

Главное, понимать, что это никак не отражается на работе программ.

Оболочка Russian QuickBasic

Прежде чем непосредственно приступить к программированию, надо научиться пользоваться средой предлагаемого к изучению языка. А поскольку она на русском языке, то это не составит большого труда.

Итак, нам предстоит узнать:

- где взять русский Бейсик;
- как его запустить и начать работу;
- как вводить текст программы в окне редактора;
- как запускать программы на выполнение;
- как сохранить программу на диске в виде файла и открыть уже существующую для просмотра и редактирования.

Где взять русский Бейсик

Собственно, проблем, я думаю, с этим возникнуть не должно — надо поспрашивать у учителей информатики, у друзей на компакт-дисках может быть, в конце концов обращайтесь к автору книги — не откажу. Рабочая версия занимает около 400 Кбайт.

Запуск русского Бейсика и начало работы

Существует три основных варианта.

- Если у вас на компьютере ничего, кроме MS-DOS, нет, то придется открыть каталог с Бейсиком, а затем в командной строке набрать имя запускающего файла `qbasic` и нажать клавишу <Enter>.
- Если у вас есть операционная оболочка Norton (или Volkov) Commander, то задача упрощается — переходите на панель, где содержится каталог с Бейсиком, открываете ее клавишей <Enter> или двойным щелчком левой кнопкой мыши, затем при помощи стрелок управления курсором находите файл qbasic.exe и запускаете его нажатием клавиши <Enter> или двойным щелчком левой кнопкой мыши.
- Самый модный вариант, если у вас MS Windows: найдите на Рабочем столе ярлык **QuickBasic** и дважды щелкните по нему левой кнопкой мыши.

Во всех трех случаях экран очищается и появляется среда русского Бейсика (рис. 1.1).

Рис. 1.1. Первое окно после запуска русского Бейсика

Далее следуйте инструкции, появившейся на экране. При нажатии клавиши <Enter> вы попадаете в Руководство для начинающих, а, нажав клавишу <Esc> — в окно редактора (рис. 1.2).

Компьютер теперь готов к вводу и редактированию ваших программ.

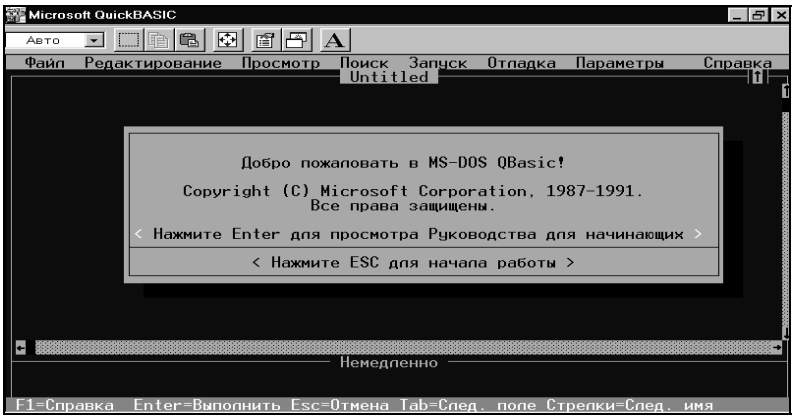


Рис. 1.2. Окно редактора

Вам для работы предоставляется два окна:

- верхнее — окно редактирования;
- нижнее — окно немедленного выполнения.

Переход из одного окна в другое осуществляется клавишей <F6> или щелчком в соответствующем окне левой кнопкой мыши.

В окне немедленного выполнения вы можете сразу видеть результаты работы набранных команд после нажатия клавиши <Enter>.

Как вводить текст программы в окне редактора

В окне редактора мигает курсор — горизонтальная светящаяся черточка, указывающая, где будет вводиться текст программы при наборе его с клавиатуры.

Если вы заметили, что что-то набрали неправильно, есть несколько путей исправить допущенные ошибки.

- *Удаление лишних символов*. Слева от курсора — клавиша <Backspace>. Символ, под которым мигает курсор и справа от курсора — клавиша <Delete>.
- *Вставка недостающих символов*. Курсор установить под тот символ, слева от которого необходимо произвести вставку и набрать нужные символы. Текст при этом раздвинется автоматически.
- *Переход из режима вставки в режим замены*. Режим вставки включается автоматически. В режим замены и обратно мы переходим нажатием клавиши <Insert>. Курсор при этом принимает вид светящегося прямоугольника. Теперь, при вводе какого-либо символа, будет стираться тот символ, который находился прежде на этом месте. Автоматическое раздвижение строки в этом случае не происходит.
- *Переход на следующую строку* осуществляется нажатием клавиши <Enter>. Если вы случайно нажали клавишу <Enter> в середине строки, то вторая половина ее перейдет вниз. Пугаться этого не надо. Нажимайте клавишу <Backspace>.

- *Копирование и перемещение фрагментов* текста программы. Сначала фрагмент нужно выделить. Выделение производится либо мышью при удерживаемой левой кнопке, либо клавишами управления курсором <←> и <→> при нажатой клавише <Shift>. После выделения возможны варианты:
  - удаление фрагмента — клавиша <Delete>;
  - копирование фрагмента в буфер — сочетание клавиш <Ctrl>+<Insert>;
  - вырезка фрагмента в буфер — сочетание клавиш <Shift>+ +<Delete>;
  - вставка фрагмента из буфера в новое место (сколько угодно раз) — сначала курсор нужно поместить в новое место, а затем — сочетание клавиш <Shift>+<Insert>.

Впрочем, все то же самое можно сделать через меню **Редактирование**.

Как Бейсик сообщает об ошибках

Если в программе была допущена ошибка, то после запуска программы на экране появится окно с сообщением об ошибке. У вас есть два варианта действий — выбрать кнопку **Справка**, чтобы получить информацию о допущенной ошибке, либо нажать клавишу <Enter> или <Esc>. Тогда вы вернетесь в окно редактирования, где курсор будет находиться в том месте, где компьютер нашел ошибку.

Запуск программы на выполнение

Как же запустить программу? Когда вы написали программу и хотите посмотреть, а что, собственно, из этого получилось, то надо нажать клавишу <F5>. Программа будет исполнена в случае отсутствия синтаксических ошибок, и тогда вы увидите результаты ее работы и сообщение внизу экрана: "Чтобы продолжить, нажмите любую клавишу...". Эта надпись вызывает иногда смятение в душах неопытных программистов, и они начинают судорожно искать на клавиатуре надпись "Любая клавиша". Надеюсь,

у вас до этого не дойдет. Если же в программе есть ошибки, то вы их исправляете, и у вас вновь два варианта:

- запустить программу с места, где она прервалась — клавиша <F5>;
- запустить программу сначала — сочетание клавиш <Shift>+ +<F5>.

Если вы хотите видеть окно, в котором видны результаты выполнения программы, то нажмите клавишу <F4>.

Основы есть? Если возникают вопросы, то, во-первых, при выборе тех или иных пунктов меню внизу появляется краткая информация о нем на русском языке, а по нажатии клавиши <F1> вы получите более подробные сведения. Во-вторых, в самом меню есть справка и по командам языка Бейсик, и по содержанию. Учитесь пользоваться справочным материалом!

Предупреждение

Если у вас отсутствует мышь, то попадание в меню осуществляется клавишей <Alt>, а работа в нем — клавишами управления курсором <←> и <→> и клавишей <Enter>.

Сохранение и открытие файлов в Бейсике

Написав программу и желая ее сохранить на диске, выберите в меню **Файл** команду **Сохранить** или **Сохранить как**. Откроется диалоговое окно, в котором вы должны выбрать диск и каталог для сохранения, дать имя файлу и нажать кнопку **ОК**. После этого в заголовке окна редактирования появится имя, назначенное файлу.

Предупреждение

Будьте внимательны к тому, *куда* вы сохраняете файл, и *под каким именем*. Иначе могут возникнуть проблемы с поиском вашей работы!

Чтобы открыть уже существующий файл, в меню **Файл** выберите команду **Открыть**. Появится диалоговое окно, в котором надо найти диск и каталог, где записан файл, указать его имя и нажать кнопку **ОК**.

Чтобы начать работу с новым файлом, в меню **Файл** выберите команду **Новый**.

Для выхода из Бейсика в меню **Файл** существует команда **Выход**. Если вы забыли сохранить свою программу, то Бейсик напомнит вам об этом до выхода и предложит сохранить ваш труд.

И только после этого вы выйдете в ту среду, из которой Бейсик был загружен.

Алфавит языка

В любом учебнике иностранного языка вначале дается его алфавит, т. е. набор символов для записи слов, предложений и всевозможных понятий этого языка. У языка Бейсик тоже есть алфавит, который содержит в себе следующие символы:

- Заглавные (или прописные) буквы латинского алфавита: A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P, Q, R, S, T, U, V, W, X, Y, Z. При наборе программы, впрочем, нет нужды следить за тем, чтобы буквы были заглавными. Интерпретатор сам изменит строчные буквы на заглавные.
- Арабские цифры: 1, 2, 3, 4, 5, 6, 7, 8, 9, 0.
- Разделители: , (запятая), ; (точка с запятой), . (точка), : (двоеточие), ‘ (апостроф), " (кавычки), ((открывающая скобка),) (закрывающая скобка), символ <Пробел>.
- Знаки арифметических операций: + (сложение), – (вычитание), * (умножение), / (деление), ^ (возведение в степень).
- Знаки операций отношений: > (больше), < (меньше), = (равно), <> (не равно), >= (больше либо равно), <= (меньше либо равно).

Переменная и что в ней меняется

Компьютер, как и вы, уважаемые читатели, обладает памятью. Она бывает разная. В процессе отладки вашей программы компьютер напрягает эту самую память, размещая в ней исходные данные, обрабатывая их, используя ваш алгоритм, получая результаты и доводя их до вашего сведения — чаще на экран мони-

тора. Я думаю, для вас не секрет, что память эта называется ОЗУ (Оперативное Запоминающее Устройство), или по-английски RAM (Random Access Memory). Это, собственно, одно из основных устройств компьютера, имеющее, правда, ограниченный объем, измеряющийся в мегабайтах.

Если вы пишете достаточно сложную и уже нелинейную программу, то наверняка потребуются *переменные*, т. е. такие области этой самой оперативной памяти, которые имеют имя, данное нами, и значения, которые могут меняться. Имя переменной в ходе выполнения программы постоянно, а значение может меняться многократно. Этот процесс можно сравнить со сдаваемой на лето дачей. Дача — это участок (область памяти компьютера), имеющий уникальный и неповторимый адрес, по которому его можно найти (имя переменной) и который не меняется, и каждое лето на дачу приезжают новые жильцы (значения переменной).

Каковы правила на этот счет в Бейсик? Так как Russian QuickBasic — это язык, рассчитанный на использование из-под DOS (Disk Operation System, дисковая операционная система), то существуют ограничения на имена переменных:

- имя переменной должно состоять не более чем из сорока символов;
- в качестве символов можно использовать только латинские буквы, цифры;
- имя переменной не может начинаться с цифры;
- категорически запрещены в именах файлов символы точки, запятой, звездочки, вопросительного знака, пробела.

Примеры правильных имен переменных:

- X, Y, Z, IVAN;
- IVAN3, S1, T234, LOVE7, R6N8F43;
- NM, MAX, GAVGAV.

Примеры неправильных имен переменных:

- Г56 (использована русская буква);
- ИВАН (использована кириллица);

- YOU+ME (использован недопустимый символ "+");
- 23DROVA (имя переменной начинается с цифры).

Переменные различаются по типу хранимой в них информации. Два наиболее крупных типа — числовой (для хранения различных чисел) и строковый (для хранения символов и строк). Во втором случае к имени переменной добавляется обязательный символ $ (на клавиатуре — там же, где цифра 4, при нажатой клавише <Shift>), например, X$ или QUIKE3$.

Арифметика в Бейсике

Прежде чем двигаться дальше *("Как трудно двигаться дальше..." — из песни Бориса Гребенщикова)*, необходимо напомнить, что в те далекие времена, когда только зарождались алгоритмические языки, а словосочетание "персональный компьютер" вызывало у тех, кто его слышал, сомнения в здравомыслии его произносившего, так вот, в те самые времена считалось, что компьютер (от англ. *compute* — вычислять), т. е. "вычислитель" только и предназначен для того, чтобы считать в тысячи, нет — в миллионы раз быстрее человека. По сути, это действительно так, и если вы немного представляете себе физику происходящего в компьютере, то все, что вы ни делаете за компьютером — сочиняете стихи или музыку, рисуете картинки, играете, общаетесь в Сети — все внутри этого "вычислителя" сводится к цифровой двоичной форме и к действиям, элементарным арифметическим действиям над этими самыми числами.

Отсюда вывод — если хочешь быть программистом, надо дружить с математикой. Начнем?

Итак, компьютер умеет вычислять элементарные арифметические выражения. Но для того, чтобы он смог это сделать, мы должны представить это самое выражение в понятном ему виде, а именно:

- в отличие от арифметики, выражение должно быть записано в *одну строку* безо всяких числителей и знаменателей;
- для записи арифметических действий допустимо использовать только перечисленные ниже знаки:

- + (сложение, слева от клавиши <Backspace> или на малой цифровой клавиатуре "серый плюс");
- – (вычитание, то же, что дефис, или на малой цифровой клавиатуре "серый минус");
- * (умножение, там же, где цифра 8 на основной клавиатуре при нажатой клавише <Shift> или на малой цифровой клавиатуре "серая звездочка");
- / (деление, на разных клавиатурах бывает в разных местах или на малой цифровой клавиатуре "серый слэш");
- ^ (возведение в степень, при выбранном латинском шрифте там же, где цифра 6 на основной клавиатуре при нажатой клавише <Shift>);
- () (скобки, там же, где цифры 9 и 0 на основной клавиатуре при нажатой клавише <Shift>);

- ❒ недопустим пропуск знака умножения между коэффициентом и переменной, как это возможно в алгебре (например, нельзя писать 2x, а надо 2*X, или нельзя 5d, а надо 5*D);
- ❒ дробная часть отделяется от целой *точкой*, а не *запятой* (нельзя писать 3,14, а надо 3.14);
- ❒ допустимо опускать в записи десятичной дроби *ноль*, стоящий перед точкой (вместо 0.123 можно .123).

Чтобы компьютер вычислил выражение правильно, необходимо помнить о приоритете выполнения действий. Тут все как в элементарной математике:

- ❒ сначала выполняются действия в скобках (в Бейсике скобки используются только круглые, в сложных выражениях они могут быть и двойные, и тройные, и т. д.);
- ❒ далее вычисляются функции, если они есть;
- ❒ затем выполняется возведение в степень;
- ❒ потом умножение и деление;
- ❒ в последнюю очередь — сложение и вычитание.

Действия одинаковой очередности выполняются слева направо.

Приведем ряд примеров перевода арифметических выражений в пригодный для Бейсика вид.

- ❒ В арифметике $\frac{1}{X}$ в Бейсике `1/X`
- ❒ В арифметике $2x^2$ в Бейсике `2*X^2`
- ❒ В арифметике $\frac{3+4}{8-6}$ в Бейсике `(3+4)/(8-6)`

Замечание

В последнем случае хочу обратить внимание на обязательность скобок как в числителе, так и в знаменателе. Потому что при записи `3+4/8-6` компьютер сначала произведет деление 4 на 8, затем прибавит 3 и вычтет 6. В ответе получится –2,5 вместо правильных 3,5. Будьте внимательны!

Чтобы потренироваться, вот вам ряд заданий на запись арифметических выражений в виде, пригодном для вычислений на Бейсике.

1. $\frac{1}{5}+\frac{1}{6}+\frac{1}{7}+\frac{1}{8}+\frac{1}{9}$

2. $2^2+3^2+4^2+5^2+6^2$

3. $\frac{1+2}{3+4}:\frac{5+6}{7+8}$

4. $\frac{2^3+3^2+4^3}{9^3+5\times6\times7}$

5. $10\times\sqrt{25+11}$

Подсказка: корень квадратный (впрочем, как и любой другой) всегда можно представить в виде степени. А вот как, узнайте на уроках математики! Да про скобки не забудьте!

6. $\cfrac{1}{2+\cfrac{3}{4+\cfrac{5}{6+\frac{7}{8}}}}$

Теперь наоборот. Следующие выражения Бейсика запишите в обычной арифметической форме:

7. `(5^3*3-4)^.5`

8. `44/4*5+10-2^3/.3`

9. `5+2/6-7+3^2`

10. `5+2/6-7+3^2`

11. `(4*(5-4^2)^2`

12. `3+8/4-7*3^2`

Ну и чтобы закончить с арифметикой, расскажу еще о двух действиях, которые вы навряд ли использовали в школе, а в программировании они могут оказаться весьма полезными. Эти действия применимы только к целым числам!

Первое из них — *деление нацело*. Для этого действия используется знак `\` (так называемый обратный слэш). Компьютер в этом случае делит числа как обычно, но в качестве результата деления представляет только целую часть, отбрасывая дробную. Например:

- `41\4=10`
- `25\9=2` и т. д.

И второе — нахождение *целого остатка от деления*. Для этого действия нет специального знака и оно выполняется при помощи оператора `MOD`. Оператор `MOD` действует как и предыдущий, но в качестве результата представляет целочисленный остаток от деления. Например:

- `41 MOD 4 = 1`
- `25 MOD 9 = 7`

Действия деления нацело и нахождения целочисленного остатка выполняются до обычных умножения и деления.

Если это вам понятно, давайте вычислим несколько выражений.

13. `20\6`

14. `20 MOD 6`

15. `34\4`

16. `34 MOD 4`

17. `2\5`

18. `2 MOD 5`

19. `4*7\3 MOD 6/3`

20. `24 MOD 5\3`

Оператор присваивания

Представьте, что вы разработали алгоритм, продумали, какие в нем будут участвовать переменные, придумали им имена и... что же дальше? Как же сообщить компьютеру их значения? Как менять эти значения? Итак, мы знакомимся с первым оператором языка Бейсик — *оператором присваивания*. Знайте, что имя переменной после присваивания будет служить для компьютера своего рода ссылкой на адрес в памяти, где значение этой самой переменной разместилось.

В старых версиях Бейсика оператор присваивания всегда начинался со слова `LET`. Сейчас этого нет, и оператор присваивания выглядит просто как обычное математическое равенство. Но это вовсе не так!

Предупреждение

В левой части оператора присваивания может находиться только имя той переменной, в которую будет заноситься новое значение и больше ничего! В правой части оператора присваивания может находиться: конкретное значение, присваиваемое переменной (в зависимости от типа переменной — число или символ, или строка); арифметическое или алгебраическое выражение, содержащее как конкретные числа, так и имена других переменных, уже имеющих значения; имя другой переменной, уже имеющей значение.

Обе части оператора присваивания соединяются знаком равенства.

Тип переменных в левой и правой частях оператора присваивания должен совпадать!

Когда интерпретатор языка Бейсик встречает в программе оператор присваивания, то он обрабатывает его следующим образом: производятся все необходимые вычисления и операции в правой

части оператора, находится конкретное значение (числовое или символьное), и это значение заносится в переменную, имя которой указано в левой части.

Предупреждение

При этом старое значение переменной, имя которой указано в левой части оператора присваивания, безвозвратно теряется. Значения переменных, участвующих в правой части, не изменяются.

Если вы в своей программе указываете впервые какую-либо переменную, не присвоив ей перед этим никакого значения, то интерпретатор по умолчанию считает ее значение равным нулю.

Синтаксис оператора присваивания

Произносится оператор присваивания следующим образом: "Присвоить Зет значение Икс минус три Игрек квадрат в скобках плюс семь" (рис. 1.3). Ни в коем случае нельзя говорить: "Зет равно..." и далее по тексту.

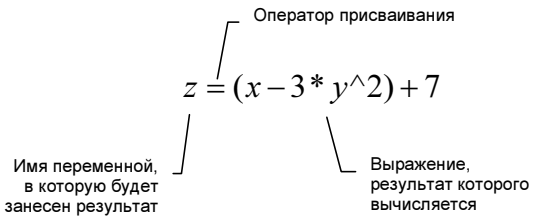


Рис. 1.3. Пример использования оператора присваивания

Рассмотрим работу приведенного выше оператора присваивания на примере фрагмента маленькой программы:

```
X=15
Y=2
Z=(X-3*Y^2)+7
```

Вопрос. Чему будет равен Z после выполнения всех операторов присваивания?

Решение. Интерпретатор делает следующее:

- присваивает переменной X значение 15;
- присваивает переменной Y значение 2;
- вычисляет значение правой части третьего оператора присваивания `(15-3*2^2)+7`. Получается 10;
- присваивает переменной Z значение 10.

Ответ. После выполнения всех операторов присваивания значение Z будет равно 10.

Вопрос. А если бы первые два оператора присваивания отсутствовали, то чему было бы равно значение Z?

Ответ. В этом случае интерпретатор принял бы значения X и Y по умолчанию равными 0, вычислил бы с этими нулями значение правой части и оказалось бы, что Z в этом случае равно 7.

Пока все было похоже на традиционную математику. Но есть один небольшой нюанс. Давайте рассмотрим такой фрагмент программы:

```
X=2
X=X+1
```

Если читать эту запись как математическую, то первый оператор совершенно обычен: "Икс равен двум". Но второй сразу заставляет бунтовать испорченную холодной логикой голову. Так ведь не может быть! Как это "Икс равен Икс плюс один"? Но что противоречит здравому смыслу в математике, легко поддается пониманию в программировании. Ведь это не математическое равенство, а рассматриваемый нами оператор присваивания. Разберем его работу.

Итак, вопрос. Чему будет равно значение X после выполнения двух операторов присваивания?

Решение. Сначала переменной X присваивается значение 2. Затем интерпретатор обращается к правой части второго оператора присваивания, находит в памяти значение X. Оно равно 2. Прибавляет к нему 1. Получается 3. Пересылает получившийся результат в правую часть, в ту же самую переменную X. Старое значение ее, равное 2, стирается, записывается новое — 3.

Ответ. После выполнения двух операторов присваивания значение переменной X будет равно 3.

Теперь упражнения. Если следующие операторы Бейсика написаны правильно, то напишите слово "верно". Если нет, то укажите на ошибку. Все операторы рассматривайте по отдельности, т. к. они не являются частью общей программы.

21. `3+6=Z`

22. `W=5+8`

23. `5+8`

24. `S+R=76`

25. `V=(X+Y+Z)/3`

26. `F+F-5`

27. `H=H*2`

28. `X=X*S`

29. `D=E=1`

30. `A+B=C-D`

31. Записать операторы, которые переменной S присваивают среднее арифметическое чисел A, B и C.

32. Записать операторы, которые переменной S присваивают расстояние между точками с координатами X1, Y1 и X2, Y2.

33. Записать операторы, которые переменной S присваивают длину гипотенузы C по катетам A и B.

34. Записать операторы, которые переменной S присваивают площадь треугольника со сторонами X, Y, Z.

35. Записать оператор присваивания, который меняет знак у значения переменной R.

36. Чему будут равны значения переменных X и Y после выполнения операторов?

```
X=3
Y=6
X=Y
Y=X
```

37. Поменять местами значения переменных X и Y.

38. Присвоить переменной W сумму цифр трехзначного числа K.

39. Присвоить переменной S значение длины светового года в километрах. Световой год — расстояние, которое свет проходит за астрономический год, т. е. 365 суток. Скорость света принять равной 300 000 км/с.

40. Присвоить переменной V значение объема кирпича с размерами A — длина, B — ширина, C — высота.

41. Записать в форме оператора присваивания формулу перевода долларов США в рубли (курс по состоянию на июль 2000 г. — 28 руб. за $1).

Выводим результаты

Пока мы вводили в компьютер исходные данные, используя оператор присваивания. Другие способы ввода будут рассмотрены далее.

Сейчас же хочется рассмотреть более важную на первых порах составляющую Бейсика — вывод полученных результатов на экран монитора, а также вывод всякого рода текстовых сообщений. То есть мы знакомимся с могущественным оператором `PRINT`.

Этот оператор столь многообразен в своих применениях, что мы позволим себе посвятить ему несколько страниц, чтобы читатель увидел и понял все его возможности и особенности.

Первая возможность оператора `PRINT` заключается в том, что он предписывает компьютеру вычислить арифметическое или алгебраическое выражение и вывести результат на экран. Таким образом, наши первые программы на Бейсике будут состоять всего из одного-двух операторов. Например:

```
PRINT 2*2
```

После запуска такой мощной программы компьютер напряжется, подумает и высветит на экране ответ — 4. После оператора `PRINT` вы можете писать с целью получения результатов все те (или другие) выражения, с которыми мы будем упражняться в следующих разделах книги.

Например:

```
PRINT(3+4)/(8-6)
```

В данном случае ответом будет 3,5.

Замечание

Возможны ошибки при использовании оператора PRINT для вычисления арифметических или алгебраических выражений. Если при наборе допущена неточность в написании операторов языка Бейсик, появится сообщение "Синтаксическая ошибка".

Примеры распространенных ошибок:

- `23-11+7` не дана команда PRINT.
- `RRINT 2+2` неверно написано слово PRINT.
- `PRINT(4+7)/3+2)` не хватает открывающей скобки в знаменателе.
- `PRINT 23*6/2=` в выражении использован лишний символ <=>.

Курсор обычно указывает то место в программе, где допущена ошибка, поэтому ничего не бойтесь, внимательно изучите свое выражение, внесите необходимые исправления и запустите программу снова.

Замечание

Сообщение об ошибке "Деление на ноль" последует, если при вычислении выражения с использованием операции деления обнаружилось деление на ноль.

Например, в знаменателе получается ноль:

```
PRINT(4+8)/(3*2-6).
```

Замечание

Еще одно сообщение вы можете увидеть при работе с вычислениями. Переполнение последует, если в результате вычислений получается число, не размещающееся в памяти компьютера (довольно часто это бывает при работе с возведением в степень).

Например:

```
PRINT 10^100
```

Теперь, прежде чем мы перейдем к упражнениям с оператором `PRINT`, хотелось бы облегчить ваш труд по набору. Дело в том, что вместо пяти символов слова `PRINT` вы можете ставить просто знак `?`. Интерпретатор Бейсика чудесным образом после запуска программы сам заменит все знаки вопроса на оператор `PRINT`, т. е. вместо

```
PRINT 2*2
```

можно смело писать

```
? 2*2
```

И второе, если ваше выражение настолько длинно, что не помещается в экранной строке, то позвольте компьютеру самому перенести часть выражения на следующую строку. Ни в коем случае не делайте этого при помощи клавиши <Enter>.

Вспомнив, что корень квадратный (как, впрочем, и любой другой степени) из числа можно представить в виде возведения в степень, вычислите и с помощью оператора `PRINT` выведите на экран результаты следующих выражений.

42. Предскажите, какой результат будет получен при вычислении значения выражения `9^1/2`. Проверьте свое предположение на компьютере.

43. $\sqrt{5}$

44. $\sqrt{12^2+5^2}$

45. $\sqrt{\sqrt[3]{2+\sqrt[4]{3}}}$

46. $\dfrac{\sqrt[4]{6}}{(5-\sqrt{8})^3}$

47. $81^{-0.25}$

48. $\sqrt{3+\sqrt{3+\sqrt{3+\sqrt{3}}}}$

Потренировались в вычислении арифметических выражений? Поехали дальше.

Оператор PRINT позволяет, как мы уже говорили, не только вычислять арифметические выражения, но и выводить на экран надписи, что предоставляет программисту оформлять тот самый дружественный интерфейс. Ведь если после запуска программы я вижу на экране число, то я не всегда могу понять, к чему оно относится, результат ли это вычислений, или количество лет, оставшихся мне в жизни. Поэтому, чтобы получить законченный программный продукт, мы будем использовать вторую уникальную возможность оператора PRINT.

Итак, если мы хотим вывести на экран сообщение, то необходимо прежде всего дать компьютеру команду PRINT (или знак вопроса, что то же самое), после которой указать сообщение, взятое в кавычки.

Предупреждение

Если после оператора PRINT выражение не заключить в кавычки, компьютер будет пытаться вычислять (если только это не строковые переменные), а то, что взято в кавычки, выводить на экран без изменений, в том виде, в каком это записано в кавычках.

Например, напишем, казалось бы, два совершенно одинаковых оператора и проследим их действие:

```
PRINT 2*2
PRINT "2*2"
```

После запуска программы в первом случае мы получим на экране ответ — цифру 4, а во втором просто выражение 2*2, т. к. при этом оно было взято в кавычки.

Ну что ж, попробуем теперь оформить более красиво программу о вычислении площади квадрата по длине его стороны, равной 5 м.

```
CLS     ' это команда очистки экрана, для тех, кто не помнит
? "Площадь квадрата со стороной, равной 5 м составляет"
? 5^2
```

На экране мы увидим в левом верхнем углу надпись "Площадь квадрата со стороной, равной 5 м составляет", а под ней число 25. Уже лучше, не так ли? Я уже понимаю, что означает число 25.

Но пока все еще не очень красиво. Хотелось бы, чтобы сообщение и ответ были бы в одной строке, и чтобы единица измерения площади хоть как-то присутствовала. Как это ни удивительно, но оператор PRINT может и это!

Он позволяет выполнять одновременно несколько действий, указанных после него в одной строке. Разделителем действий может служить точка с запятой или запятая.

Рассмотрим пример.

```
CLS
? "Площадь квадрата со стороной 5 м равна ";5^2;" кв. м"
```

После запуска программы мы увидим в левом верхнем углу чистого экрана надпись: "Площадь квадрата со стороной 5 м равна 25 кв. м".

То есть в операторе PRINT было указано три действия:

- вывести на экран надпись "Площадь квадрата со стороной 5 м";
- вычислить 5 в квадрате;
- вывести на экран надпись "кв. м".

Так как в качестве разделителя использовалась точка с запятой, то все три составные части выведены друг за другом, без пробелов (исключение составляет числовой результат, перед которым компьютер всегда вставляет дополнительный пробел для возможного знака минус в случае получения отрицательного результата).

Вот это уже законченный результат!

Теперь надо бы разобраться в разделителях и постараться их не путать.

Разделитель *точка с запятой* в последовательности действий оператора PRINT позволяет выводить текстовые сообщения и результаты вычислений в одной экранной строке подряд.

В компьютере экранная строка подразумевается поделенной на зоны размером в 14 символов, и разделитель *запятая* в последовательности действий оператора PRINT позволяет выводить текстовые сообщения и результаты вычислений тоже в одной экран-

ной строке, но каждое отдельное действие выводится в следующую зону печати. Этот формат служит главным образом для оформления таблиц с ровными колонками. Например, очередная программа:

```
? "Фамилия", "Телефон"
?
? "Иванов", "212-85-06"
? "Петров", "712-23-45"
? "Сидоров", "100-09-11"
```

после запуска выведет на экран следующую таблицу:

```
Фамилия         Телефон

Иванов          212-85-06
Петров          712-23-45
Сидоров         100-09-11
```

Обратите внимание на вторую строку программы, в которой стоит оператор `PRINT` без данных. Такая запись позволяет получить *пустую строку*. Это удобно, например, для отделения заголовка таблицы от данных.

Кроме того, если вы исполните команду

```
? "Печать"
```

то компьютер выведет на экран слово "Печать" и переведет курсор на следующую строку, где и будут выводиться результаты последующих действий программы.

Если же вы исполните команду

```
? "Печать";     ' команду завершает точка с запятой
```

то курсор останется в этой же строке после слова "Печать". Это так называемая *печать без перевода строки*.

49. Выясните на компьютере, как отличаются при выводе на печать следующие строки:

```
? "X";"Y";"Z"
? "X","Y","Z"
? "X","Y";"Z"
```

50. С помощью одного оператора `PRINT` вычислите пять арифметических выражений: `5+2`, `5-2`, `5*2`, `5:2`, 5^2. В качестве разделителя используйте сначала точку с запятой, а потом запятую.

51. Вычислите и напечатайте с подсказкой на чистом экране:

- `Сумма=312,66+79,44`
- `Корень из двух=`$\sqrt{2}$
- `Результат равен=2*3*4*5*6`

52. Напечатайте на чистом экране слова "Печать", "через", "строку" одно под другим через строку.

53. Предскажите действие команды

```
?, "Эксперимент"
```

Проверьте свой прогноз.

Замечание

В целях более рационального использования экранного места при написании программы иногда допустимо в одной строке писать несколько операторов языка Бейсик. В таких случаях в качестве разделителя операторов применяется *двоеточие*.

Например:

```
CLS:?"Экран чист"
```

Стандартные функции Бейсика

Пока мы не забыли элементарную арифметику в Бейсике и вывод результатов (надеюсь, у вас это получается, причем в законченной, красивой форме с подсказками), необходимо ознакомиться со стандартными функциями языка Бейсик, чтобы в полной мере ощутить его вычислительные способности.

Бейсик позволяет работать со стандартными алгебраическими функциями, перечисленными в табл. 1.1.

Предупреждение

Функция `SQR` применима только к положительным аргументам.

Таблица 1.1. Стандартные алгебраические функции

Функция	Описание	Пример	Результат
ABS	Возвращает абсолютную величину (модуль) аргумента	? ABS(-2.56)	2,56
SQR	Возвращает квадратный корень из аргумента	? SQR(9)	3
SIN	Возвращает значение синуса аргумента (аргумент указывается в радианах)	? SIN(3)	0,14112
COS	Возвращает значение косинуса аргумента (аргумент указывается в радианах)	? COS(3)	0,98992599
TAN	Возвращает значение тангенса аргумента (аргумент указывается в радианах)	? TAN(3)	–0,1425465
EXP	Находит значение показательной функции e^x, где x — аргумент, а e — основание натурального логарифма, равное ≈2,718...	? EXP(1)	2,718282
LOG	Возвращает логарифм по основанию e	? LOG(3)	1,09861229
INT	Возвращает целую часть аргумента (от англ. *INTeger* — целый)	? INT(30,1) ? INT(30,9)	30 30
SGN	Возвращает 1, если аргумент больше нуля; 0, если аргумент равен нулю; –1, если аргумент меньше нуля (от англ. *SiGN* — знак)	? SGN(3) ? SGN(0) ? SGN(-3)	1 0 –1

Любознательный читатель сразу заметит, а где же милый его сердцу котангенс. Нет! Но ведь его просто получить из тангенса, не правда ли? Самые распространенные ошибки начинающих при написании стандартных функций (хотя вы все равно их допустите!):

- пишут ABC вместо ABS для модуля;
- пишут TG вместо TAN для тангенса.

Главное правило при использовании стандартных функций — брать аргумент в скобки:

- нельзя `SIN3`, а надо `SIN(3)`;
- нельзя `SQR9`, а надо `SQR(9)`.

Стандартные функции могут входить в арифметические выражения. В качестве аргументов также можно использовать арифметические выражения.

Примеры правильной записи стандартных функций представлены в табл. 1.2.

Таблица 1.2. Примеры записи стандартных функций

Алгебра	Бейсик
$\sin 3 + \cos 6 + \mathrm{tg}\,4$	`SIN(3)+COS(6)+TAN(4)`
$\sqrt{\lvert\sin 45^\circ - \cos^2 30^\circ\rvert}$	`SQR(ABS(SIN(45)-COS(30)^2))`

Перед упражнениями расскажем про *радианную меру углов*, потому что не все, к сожалению, любят тригонометрию. Итак,

1 радиан$=\dfrac{180^\circ}{\pi}$, отсюда 1 градус$=\dfrac{\pi}{180^\circ}$, следовательно, $\sin 30^\circ$ в Бейсике будет представлено как `SIN(30*3.14/180)`, $\pi \approx 3{,}14$.

А вот теперь можно поупражняться.

Запишите на Бейсике следующие выражения:

54. $\sqrt{1+z^3}$

55. $\lvert x+by\rvert$

56. $\cos^2 x^3$

57. 2^{2+x}

58. $\sqrt[3]{2+x}$

59. $\sqrt[7]{x^7 + 7^x}$

60. $\dfrac{a + \sin^3 b^2}{\cos 25 + |\text{ctg}\ \ 60|}$

Представьте в общепринятой математической форме:

61. `(-R+SQR(S^2-6*A*B))/(3*A)`

62. `X/Y*(D-F)+(X+Y)/D/F`

63. `X1+TAN(F2-V3)/3*ABS(X2-LOG(4)*Y3)/EXP(-2)`

Выводим данные в заданном месте экрана

Экран компьютера в текстовом режиме представляет собой условную сетку из столбцов и строк. В стандартном режиме таких столбцов 80 (т. е. в строке может разместиться не более 80 символов), а строк 30.

В Бейсике есть оператор, который позволяет управлять выводом данных на экран, что существенно повышает восприятие программ пользователем. Текст или результаты вычислений всегда выводятся на экран начиная с той позиции, в которой на момент вывода находится курсор. Изменить эту позицию поможет оператор `LOCATE`.

Правильно записывается он так:

```
LOCATE Y,X
```

где `X` — номер столбца экрана, а `Y` — номер строки ($0 \le X \le 80$ и $0 \le Y \le 30$).

В операторе `LOCATE` используются две величины, которые мы будем называть *операндами*. В этом операторе они должны быть целыми.

Например, команды

```
CLS:LOCATE 13,40:?"S"
```

позволяют последовательно:

- ❒ очистить экран и перевести курсор в позицию X=0, Y=0;
- ❒ перевести курсор в позицию X=40, Y=13;
- ❒ напечатать в центре чистого экрана букву S.

Выполним ряд упражнений с использованием оператора `LOCATE`.

64. Наберите и запустите программу, которая выведет в центре чистого экрана ваши имя и фамилию. Обеспечьте симметричное расположение надписи относительно сторон экрана.

65. Укажите последовательные положения курсора. Предскажите, как будет расположен текст на экране после исполнения следующих маленьких программ:

- `LOCATE 20,7:CLS:?"НЕ ТУДА"`
- `LOCATE 12,10:?"А ГДЕ ЖЕ ТЕКСТ?":CLS`
- `CLS:?"ЕЩЕ РАЗ НЕ ТУДА": LOCATE 0,0`

66. С помощью нескольких команд напечатайте слова "Раз!" и "Два!" в разных местах экрана.

67. Напишите программу, которая вычислит и напечатает в центре чистого экрана значение выражения: $2^3 + 3^3 + 4^3 + 5^3$.

Над ним в качестве заголовка напечатайте надпись "Сумма кубов".

68. В центре чистого экрана выведите слово "СЕРЕДИНА", окруженное рамкой из звездочек, как показано ниже:

```
* * * * * * * * * * *
* С Е Р Е Д И Н А *
* * * * * * * * * * *
```

69. Напечатайте свое имя по диагонали, начиная от верхнего левого угла экрана.

Итак, мы научились вычислять и красиво выводить полученные результаты на экран. А теперь мы научимся вводить эти самые исходные данные.

Вводим данные

Чем хорош Бейсик? Тем, что позволяет писать программы, весьма дружественно настроенные к пользователю, запрашивающие у него свойственные только ему параметры и решающие поставленную задачу для вполне конкретных данных вполне конкретного человека или любого другого объекта.

По негласным законам программирования хорошая программа не должна зависеть от исходной информации и обязана решать поставленную задачу для произвольных данных. Например, если это задача о расчете среднего роста учеников вашего класса, то программа должна быть написана таким образом, чтобы при введении данных о росте всех учеников результатом был бы средний рост вашего класса, а при введении данных о росте только мальчиков — средний рост мальчиков.

Это позволяет делать оператор `INPUT`.

Оператор *INPUT*

Оператор `INPUT` обеспечивает запрос данных, необходимых для выполнения программы, непосредственно у пользователя, которые тот вводит прямо с клавиатуры. Это дает возможность вести диалог с пользователем и решать задачу, исходя из его конкретных запросов.

Работает оператор `INPUT` следующим образом. В том месте программы, где вы хотите запросить у пользователя какие-то данные, вы должны написать `INPUT`, а после него через пробел — переменную или несколько переменных, в которые будут занесены вводимые данные. Например:

```
CLS:?"Каков Ваш рост в см?"
INPUT R
?"Ваш рост превышает 1 м на ";R-100;"см"
```

Эта элементарная программа работает так: оператор `CLS` очищает экран, оператор `PRINT` выводит на экран надпись: "Каков Ваш рост в см?". Далее вступает в действие оператор `INPUT`. Когда программа встречает этот оператор, она приостанавливает свое

действие, выводит на экран знак вопроса, после которого мигает курсор. Таким образом программа показывает, что она ожидает от пользователя ввода данных с клавиатуры. Неискушенный пользователь в этот момент обычно пугается, что программа зависла и начинает звать на помощь. Тот же, кто внимательно прочитает эту книгу, будет знать, что делать, а именно ответить на заданный вопрос, в данном случае — набрать на клавиатуре свой рост и нажать клавишу <Enter>. После таких абсолютно правильных действий пользователя программа заносит введенное число в память в переменную с именем R, где оно и будет храниться, после чего переходит к исполнению следующей за `INPUT` команды. В рассматриваемой программе она определяет вывод на экран результатов — надписи "Ваш рост превышает 1 м на ", затем компьютер вычислит разность между значением переменной R и 100 см, выведет ее на экран и добавит единицу измерения "см".

Очень запоминается следующий пример диалоговой программы, которая запрашивает у пользователя имя, а затем здоровается с ним:

```
CLS:?"Как вас зовут?"
INPUT NAME$  Rem Знак $ указывает на то, что ожидается ввод
             Rem не числовой, а символьной информации
? "Здравствуйте"; NAME$
```

Эта программа очищает экран (оператор `CLS`), затем выводит сообщение "Как вас зовут?", следом вступает в дело оператор `INPUT`. При этом программа приостанавливает свое действие и ждет от человека ввода его имени, например Ваня, которое запоминается компьютером в переменной `NAME$`. После этого на экране появится сообщение "Здравствуйте, Ваня!". Ваня обычно радуется, как маленький, даже если это уже взрослый человек.

Для тех, кто в дальнейшем планирует изучать язык Паскаль, пожалуй, стоит запомнить именно такую конструкцию оператора `INPUT`: сначала вывод сообщения оператором `PRINT`, а затем запрос данных. Но Бейсик в этом смысле облегчает работу и позволяет объединить два действия в одном. Тогда обе вышеприведенные программы обретут иной вид.

Про средний рост:

```
CLS:INPUT "Каков Ваш рост в см";R
?"Ваш рост превышает 1 м на";R-100;"см"
```

Про "Здравствуйте, Ваня!"

```
CLS:INPUT "Как вас зовут";NAME$
? "Здравствуйте"; NAME$
```

То есть в операторе INPUT сначала можно разместить подсказку (но только одну), а затем после точки с запятой указать имя запрашиваемой переменной, или, если их несколько, перечислить через запятую.

Кроме того, обратите внимание, что в модифицированных вариантах программ в вопросах опущены знаки вопросов, т. к. оператор INPUT при работе сам выставляет знак вопроса.

Рассмотрим примеры.

Далее приведена программа, запрашивающая длины катетов, а затем вычисляющая длину гипотенузы прямоугольного треугольника.

```
CLS:INPUT "Введите длины двух катетов";A,B
C=SQR(A^2+B^2)
?"Длина гипотенузы =";C
```

Когда оператор INPUT требует не одно, а несколько значений, то после запуска программы их надо вводить в том порядке, в котором они запрашиваются, а по окончании нажать клавишу <Enter>.

Если вы случайно ввели меньше или больше данных, чем было необходимо, и нажали клавишу <Enter>, то появится сообщение "Ввод сначала". Это означает, что вы спокойно, без паники должны осуществить ввод данных заново.

Перед упражнениями приведем еще один пример программы с оператором INPUT:

```
CLS: INPUT "Как вас зовут";NAME$
? "Здравствуйте,"; NAME$
INPUT "Введите год вашего рождения, а затем текущий
год";YEAR1,YEAR2
```

```
R=YEAR2-YEAR1
?"Вам около";R;"лет"
```

Запустим программу и проследим ее работу (выделенные полужирным символы должны набираться пользователем в процессе работы программы).

```
Rem Нажмите <Shift>+<F5>
Как вас зовут? Ваня   Rem После ввода нажмите клавишу <Enter>
Здравствуйте, Ваня
Введите год вашего рождения, а затем текущий год? 1986, 2000
Rem После ввода нажмите клавишу <Enter>
Вам около 14 лет
```

Несколько маленьких упражнений.

70. Напишите программу, запрашивающую ваш год рождения, год рождения вашей мамы и печатающую, во сколько лет мама вас родила.

71. Напишите дружественную по отношению к пользователю программу. Пусть она обращается к вам по имени, а затем предложит последовательно ввести прилагательное, существительное, наречие и глагол, чтобы в результате вывести на экран несложную фразу с использованием введенных слов.

72. Напишите программу, запрашивающую три стороны треугольника A, B, C и вычисляющую его площадь по формуле Герона:

$$S = \sqrt{p(p-a)(p-b)(p-c)},$$

где $p = (a+b+c)/2$ — полупериметр.

73. Напишите программу, запрашивающую высоту дома h (в метрах), ускорение свободного падения g, и вычисляющую время падения кирпича t (в секундах) с крыши этого дома по формуле:

$$t = \sqrt{\frac{2h}{g}}$$

74. Запросите у пользователя валютный курс на сегодняшний день, затем — имеющуюся у него рублевую сумму и рассчитайте, сколько долларов и сколько евро он может купить на эти деньги.

75. Дискета 3,5" вмещает 1,44 Мбайт. Рукопись содержит X страниц текста. На каждой странице — Y строк по Z символов в каждой. Сколько дискет потребуется для записи рукописи? X, Y и Z запрашиваются.

76. Документ содержит текст из X строк по Y символов в каждой и точечную черно-белую фотографию 10×15 см. Каждый квадратный сантиметр содержит 600 точек, любая точка описывается 4-мя битами. Каков общий информационный объем документа в килобайтах?

77. Игра "Zavr In The Sky" требует для установки на жесткий диск **А** мегабайт свободного места. На жестком диске сейчас **Б** мегабайт свободного места. Сколько флэш-карт по **В** мегабайт понадобится, чтобы освободить недостающее пространство? (Данные А, Б и В запрашиваются).

78. Жесткий диск пуст и имеет объем свободного пространства **Г** гигабайт.

а) Сколько книг, каждая из которых состоит из **Д** страниц, на каждой странице **Е** строк, в каждой строке **Ж** символов, можно записать на такой жесткий диск?

б) Если учесть, что каждая такая книга 3 см толщиной, то какой высоты в метрах будет стопка, если все их сложить друг на друга?

79. Скорость передачи данных по локальной сети — **С** миллионов бит в секунду. Ученик качал игру **Т** минут.

а) Сколько это гигабайт и сколько дискет по 1,5 мегабайта можно заполнить таким объемом информации?

б) Сколько денег (в рублях) придется заплатить ученику за трафик, если первый 1 Гбайт не оплачивается, а все, что сверх его — по **У** копеек за 1 Мбайт.

80. Запросите у пользователя длину ребра куба. Найти площадь грани, площадь полной поверхности и объем этого куба.

Замечание

Помните, что в самой программе вы не указываете ни конкретных имен, ни конкретных слов, а лишь имена переменных, в которых они будут храниться после ввода с клавиатуры.

Операторы *DATA* и *READ*

Существует другой способ ввода данных. Он применяется в том случае, если эти данные заранее известны. Тогда мы организуем в программе некое подобие склада (с помощью оператора DATA), а потом при помощи оператора READ последовательно считываем эти данные в запрашиваемые переменные, в которых они хранятся и обрабатываются. Удобство этого способа в том, что не приходится каждый раз при запуске программы вводить исходные данные (особенно когда их много), а также в простоте изменения исходных данных.

Приведем пример программы.

```
DATA 13, 17, 19
READ A
? A
READ A
? A
READ A
? A
```

На "складе" DATA хранятся три числа — 13, 17 и 19. Первый оператор READ читает со "склада" число 13 и записывает его в переменную A, а затем оператор PRINT выводит ее значение на экран. Потом считывается второе число — 17, вместо старого значения 13 в переменную A записывается новое — 17, которое тоже выводится на экран. Аналогичные действия происходят и в третий раз с последним числом со "склада".

Предупреждение

В программе можно использовать несколько операторов DATA, причем они могут быть расположены в любых местах программы. Все они рассматриваются как единый "склад". А данные считываются все равно последовательно сверху вниз и слева направо.

Еще пример, только теперь считываться будут сразу две переменные, что никак не скажется на исполнении программы.

```
DATA 1, 2, 4
READ A, B
DATA 8
? A, B
READ A, B
? A, B
READ A, B
? A, B
DATA 16, 32
```

При выполнении оператора DATA может возникнуть сообщение об ошибке типа "Нет данных". Подобная ошибка возникает при попытке чтения уже закончившихся данных. Если данных во всех операторах DATA шесть, то и считать их можно только шесть раз.

Можно восстановить все данные, хранящиеся в DATA, командой RESTORE.

А теперь маленькие задания.

81. Напишите программу, рисующую три закрашенных прямоугольника с заданными координатами диагоналей и цветами.

82. Задайте в операторах DATA четыре строки любого стихотворения и напишите программу, выводящую их на экран.

Еще ряд заданий на операторы DATA и READ мы выполним, когда научимся пользоваться оператором цикла.

Алгоритмы

Все, что бы мы ни делали, будь то на компьютере, или в жизни, чаще всего преследует какую-либо цель. И не всегда эта цель достигается. Но если мы как следует сформулируем для себя желаемый результат, потом продумаем четкий план его достижения, то, по крайней мере, на компьютере цель будет достигнута (в жизни, к сожалению, не все так однозначно).

Так вот, четкая, ясная и однозначная последовательность действий, приводящая к достижению результата, называется *алгоритмом*.

Примеры алгоритмов из жизни — это и правила перехода улицы, рецепт пирога, инструкция по изготовлению ядерной бомбы в домашних условиях и т. д.

Сразу попробуйте подумать и ответить, что из перечисленного ниже является алгоритмом, а что нет:

- правила перехода улицы;
- политическая карта мира;
- телефонный справочник;
- рецепт пирога;
- инструкция по пользованию видеомагнитофоном.

Виды алгоритмов

Алгоритмы бывают трех основных видов, которые являются базовыми при написании программ.

Первый тип — *линейный алгоритм*. В нем все действия выполняются в строгом порядке, последовательно, одно за другим. Типичный жизненный пример такого алгоритма — рецепт пирога.

Второй тип — *разветвляющийся алгоритм*. Здесь те или иные действия выполняются в зависимости от выполнения или невыполнения некоего условия. Пример из жизни — правило перехода улицы по светофору. Если горит красный — стоим, если горит зеленый — идем.

Наконец, третий тип — *циклический алгоритм*. Он содержит повторяющиеся действия с какой-либо изменяющейся величиной, так называемым параметром. По циклическому алгоритму можно колоть дрова. Берем полено, ставим на попá, колем топором, берем второе полено и т. д., пока поленья не закончатся, и эта работа нам не надоест.

Линейный алгоритм

Для написания программ с линейным алгоритмом мы уже готовы. Нам известно, как очистить экран и поприветствовать пользователя по имени, запросить у него необходимые данные и ре-

шить вычислительные задачи. Попробуем поупражняться. Следите все же за интерфейсом своих программ!

В следующих заданиях вам предлагается написать программы, производящие требуемые вычисления и выдающие на экран результаты.

83. Расстояние до ближайшей к Земле звезды Альфа Центавра 4,3 световых года. Скорость света принять 300 000 км/с. Скорость земного звездолета 100 км/с. За сколько лет звездолет долетит до звезды?

84. Вычислить количество прожитых составителем программы дней. Учесть, что в високосном году 366 дней.

85. Известна теория биоритмов. С момента рождения жизнь человека подчиняется трем синусоидальным биоритмам. Физический цикл — 23 дня, эмоциональный — 28 дней и интеллектуальный — 33 дня. Первая половина каждого цикла — положительная, вторая — отрицательная. При переходе от положительной к отрицательной фазе в каждом цикле есть так называемый критический день. В физическом цикле — 12-й день, в эмоциональном — 15-й день, в интеллектуальном — 17-й день. Когда два и более цикла находятся в отрицательной фазе, или в критических днях, то в такие дни повышена вероятность физических недомоганий и травм, эмоциональных срывов и ссор, замедленности интеллектуальных процессов и реакции. В Японии, например, в крупных фирмах для каждого работника составлен график и в "плохие" дни их не допускают до работы, дают отдыхать, потому что оплата больничного или брак в работе обойдутся фирме дороже. Теперь к делу. Опираясь на результат предыдущего задания и используя операцию нахождения целочисленного остатка, рассчитайте, каков для вас сегодняшний день по всем трем циклам.

86. Запросите у пользователя валютный курс на сегодняшний день, затем имеющуюся у него рублевую сумму и рассчитайте, сколько долларов он может купить.

87. Есть притча о шахматах, где выигравший запросил у могущественного правителя, чтобы ему был выплачен выигрыш зер-

ном пшеницы по следующим правилам: на первую клетку шахматной доски положить одно зерно, на вторую — два зерна, на третью — четыре, на четвертую — восемь и т. д., иными словами, на каждую последующую в два раза больше зерен, чем на предыдущую. Сколько же зерен должен был бы получить выигравший? (Замечание: на каком-то этапе выполнения этого задания возможно переполнение памяти. Подумайте, как обойти это препятствие.)

88. Кстати, для оценки предыдущего результата еще одно задание. В России ежегодно собирают около 90 млн тонн зерновых. Масса одного зерна около 5 грамм. Сколько зерен в таком урожае, и сколько лет пришлось бы расплачиваться России по условиям предыдущего задания?

89. Продав квартиру, вы получили $52 000 и положили их в банк. Банк начисляет 1% в первый месяц, а каждый следующий — тоже 1%, но уже с получившейся суммы. Сколько денег будет в банке на вашем счету через год? (Нестабильностью нашей экономической системы пренебрегаем.)

90. Допустим, вы получили наследство $1 000 000 и хотите красиво пожить. После долгих раздумий вы решаете, что будете жить на $800 в месяц. На сколько лет вам хватит наследства?

91. Пушка стреляет под углом 30° к линии горизонта. Масса снаряда 30 кг, начальная скорость 500 м/с. Какова будет дальность полета снаряда? (Формулу вспомните из курса физики.)

Нагляднее всего работа с линейными алгоритмами осуществляется в графике, когда мы сразу видим результат своей деятельности. Но для этого нам надо изучить несколько новых операторов.

Графика в Бейсике

Иногда при изучении этой темы приходится сталкиваться с непониманием учащихся. "А зачем так сложно, если в любом графическом редакторе можно сделать то же самое?" Взгляд автора здесь таков. Да, в графическом редакторе можно нарисовать практически что угодно. Но, зная приемы программирования,

создаются быстро такие элементы, на которые в графическом редакторе уйдет очень много времени и не будет возможности многократного повторения и внесения быстрых изменений в рисунок.

Прежде всего, необходимо сказать несколько слов о включении графических режимов. При входе в оболочку Бейсика по умолчанию включается текстовый режим, в котором можно производить вычисления и выводить результаты на экран. Но если мы хотим пользоваться графическими возможностями языка, то должны объяснить это компьютеру посредством включения графического режима командой `SCREEN 12`. Таких режимов несколько, но автор рекомендует пользоваться двенадцатым режимом, потому что он обладает наибольшей разрешающей способностью и позволяет получать качественные графические объекты. После включения графического режима мы можем давать компьютеру команды рисования графических примитивов.

Замечание

Графический режим включается только один раз.

В режиме `SCREEN 12` экран представляет собой координатную сетку с началом в левом верхнем углу, вправо от которого увеличивается координата X, а вниз — координата Y. Максимальное значение X на экране 640, а Y — 480 (рис. 1.4).

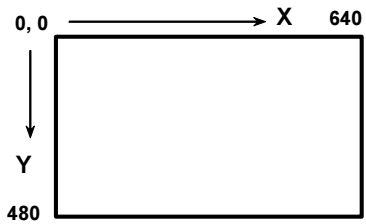


Рис. 1.4. Размеры и координаты экрана в режиме `SCREEN 12`

Замечание

Если хочется, чтобы координатная сетка экрана более привычно начиналась с левого нижнего угла экрана (т. е. чтобы Y увеличивался вверх), надо использовать перед рисованием команду

`WINDOW (0, 480) – (640, 0)`, т. е. задать координаты верхнего левого угла и правого нижнего.

Графические примитивы

Ну что ж, начнем с графических примитивов, т. е. элементарных объектов, на которые можно разбить любое сложное изображение.

Точка

Первый из графических примитивов — *точка*. Для изображения точки используется оператор `PSET` со следующим синтаксисом:

```
PSET(X, Y), C
```

где `X` и `Y` — координаты точки на экране, а `C` — ее цвет. Если цвет не указан, то точка будет изображена последним установленным цветом (это касается и всех прочих графических примитивов).

Рассмотрим пример.

```
SCREEN 12
PSET(320, 240), 4
```

На черном экране в этом случае появится красная точка в центре.

Сразу же хочется заметить, что из эстетических соображений лучше менять цвет фона, потому что черный экран выглядит мрачно. Например:

```
SCREEN 12
COLOR,1
PSET(320,175),4
```

Теперь та же точка будет в центре синего экрана. Уже лучше.

Отрезок и прямоугольник

Следующий примитив — *отрезок прямой линии*. Оператор записывается таким образом:

```
LINE (X1, Y1)-(X2, Y2), C
```

где X1, Y1 — координаты начала отрезка; X2, Y2 — координаты конца отрезка; C — как всегда цвет.

Например, мы хотим изобразить отрезок синего цвета на желтом экране с координатами (рис. 1.5).

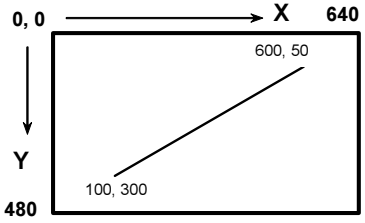


Рис. 1.5. Отрезок прямой линии

Программа будет выглядеть так:

```
SCREEN 12
COLOR 1, 14
LINE (100, 300)–(600, 50), 1
```

Не секрет, что по диагонали прямоугольника можно всегда его достроить. Известно это и Бейсику. Поэтому можно одним оператором построить *прямоугольник со сторонами, параллельными экрану*. В этом случае оператор LINE имеет следующий вид:

```
LINE (X1, Y1)–(X2, Y2), C, B
```

То есть к обычному оператору отрезка добавляется буква (это именно буква, а не число!) B — от английского слова box (ящик, коробка). А координаты — это начало и конец любой из диагоналей прямоугольника. Возьмем предыдущий пример и добавим в операторе LINE букву B:

```
LINE (100, 300)–(600, 50), 1, B
```

В результате выполнения у нас получится уже не отрезок, а прямоугольник, причем сама диагональ видна не будет (рис. 1.6).

Предупреждение

Будьте внимательны с координатами. Довольно часто вместо координат диагонали указывают диагонали стороны прямоугольни-

ка. В таком случае вы получите просто отрезок, параллельный экрану.

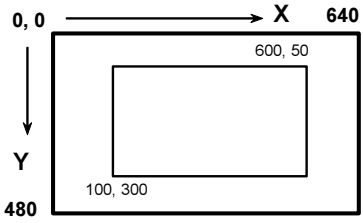


Рис. 1.6. Прямоугольник со сторонами, параллельными экрану

И еще один примитив, основанный на операторе `LINE`. Он позволяет не только строить *прямоугольник по диагонали*, но и сразу закрашивать его. Пишется он так:

```
LINE (X1, Y1)-(X2, Y2), C, BF
```

Обратите внимание, что добавилась еще буква `F` (без пробела после `B`), вместе с буквой `B` обозначающая "box full" — полная коробка. Это заставляет компьютер залить полученный прямоугольник указанным цветом `C`.

Попробуйте самостоятельно в предыдущем примере добавить букву `F` и посмотреть, что получится.

Замечание

Одним оператором `LINE` можно получать только прямоугольники и со сторонами, параллельными экрану. Другие фигуры, состоящие из отрезков прямых, можно рисовать только последовательностью операторов `LINE` без всяких букв `B` и `F`.

Ну что ж, для начала достаточно. Надо бы поупражняться. Но перед упражнениями хотелось бы дать совет, как облегчить себе жизнь.

Чтобы быстро и четко нарисовать заданный объект, рекомендуется взять листок миллиметровки или бумаги в клеточку, нарисовать экран в масштабе 1 см=40 экранных точек. Получится 16 см в ширину и 12 см в высоту. Затем изобразить там объект, который вы хотите получить, и тогда все координаты будут у вас, как на ладони (рис. 1.7).

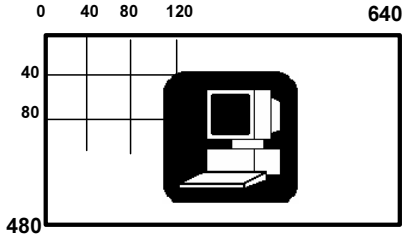


Рис. 1.7. Так надо делать!

К сожалению, опыт автора показывает, что многим из вас лень этим заниматься — отсюда пустая трата сил и времени и очень неказистые результаты.

Перейдем к упражнениям.

92. Напишите программу, изображающую на экране четыре точки разного цвета в центре желтого экрана. Точки расположите квадратом на расстоянии друг от друга 5 точек.

93. Напишите программу, выводящую ромб с диагоналями (рис. 1.8).

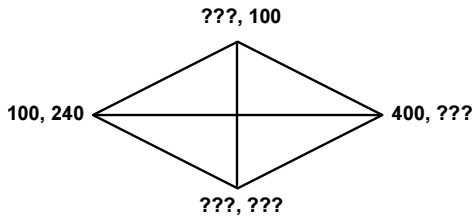


Рис. 1.8. Ромб с диагоналями

Пропущенные координаты определите сами, зная свойства ромба.

94. Создайте программу, рисующую Российский флаг[1]. Флаг состоит из трех полос, поэтому рисовать их надо, используя

[1] Есть, кстати, мнемоническое правило для запоминания цветов нашего флага — это магическое слово БеСиК — БЕлый, СИний, Красный — и на Бейсик похоже, и не забудешь никогда.

команду `LINE` с параметром `BF`. Здесь главное четко рассчитать координаты диагоналей всех трех прямоугольников.

95. Напишите программу, выводящую рамку с конвертом (рис. 1.9).

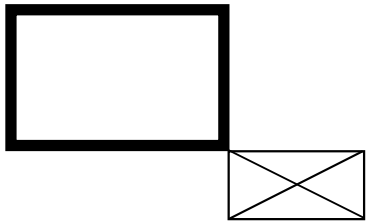


Рис. 1.9. Рамка с конвертом

Рамка рисуется двумя операторами `LINE` с `BF`, конверт — подумайте сами. Координаты тоже выберите сами. Только не рисуйте слишком мелкие объекты. Рисунок должен быть крупным и наглядным. Это любят преподаватели!

96. Создайте программу, выводящую на одном экране куб и пирамиду (рис. 1.10).

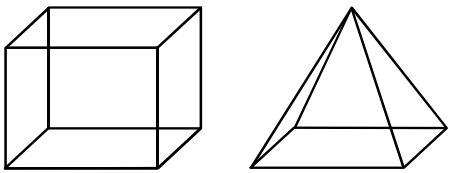


Рис. 1.10. Куб и пирамида

Окружность

Следующий этап построения — *окружность*. Все, что нужно знать об окружности для Бейсика, — это координаты ее центра и радиус. Оператор выглядит так:

```
CIRCLE (X, Y), R, C
```

где `X`, `Y` — координаты центра, `R` — радиус (в экранных точках), `C` — цвет.

Например, оператор

```
CIRCLE (320, 240), 50, 2
```

изобразит нам зеленую окружность радиусом 50 точек в центре экрана.

Эллипс

Очередной примитив — *эллипс*. Эллипс — это, по сути, окружность, которой слегка дали по "голове" или по "бокам", в результате чего она стала вытянутой по вертикали или горизонтали. И вместо одного радиуса, как у окружности, у эллипса стало два — по осям X и Y.

Частное от деления R_y на R_x дает нам так называемый коэффициент сжатия:

$$K = \frac{R_y}{R_x}.$$

Таким образом, для вытянутых по горизонтали эллипсов (рис. 1.11) коэффициент сжатия будет в пределах от 0 до 1.

А для эллипсов, вытянутых по вертикали (рис. 1.12), коэффициент сжатия будет больше 1.

Очевидно, что если коэффициент сжатия равен 1, то это будет уже никакой не эллипс, а просто окружность.

Оператор рисования эллипса записывается почти так же, как и для окружности:

```
CIRCLE (X, Y), R, C,,, K
```

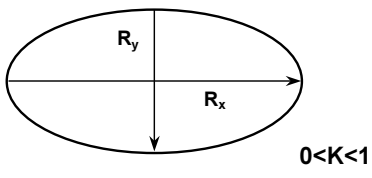


Рис. 1.11. Эллипс, сжатый по вертикали

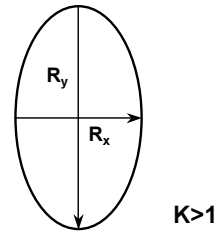


Рис. 1.12. Эллипс, сжатый по горизонтали

где `X`, `Y` — координаты центра эллипса, `R` — радиус той окружности, из которой этот эллипс получился, `C` — цвет, `K` — значение коэффициента сжатия.

Например, после выполнения оператора

```
CIRCLE (320, 240), 50, 2,,,.5
```

в центре экрана появится эллипс зеленого цвета, сжатый сверху и снизу, у которого R_y в два раза меньше, чем R_x.

Если же добавить еще один оператор

```
CIRCLE (320, 240), 50, 2,,,2
```

то сверху дорисуется дополнительный один эллипс, который будет сжат уже по вертикали, и у него уже R_y будет в два раза больше, чем R_x. Вместе они должны составить следующую картинку (рис. 1.13)

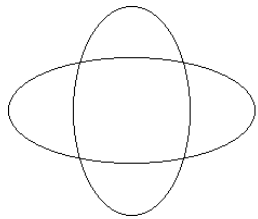


Рис. 1.13. Два пересекающихся эллипса

Ну, а теперь продолжим наши упражнения.

97. Напишите программу, которая изобразит квадрат и вписанную в него окружность.

98. Создайте программу, с помощью которой можно нарисовать мишень — пять концентрических (т. е. с одним центром) окружностей, вложенных друг в друга, и перекрестие двух отрезков прямых. Можно усложнить изображение, добавив попавшую в мишень стрелу (рис. 1.14).

99. Напишите программу, которая отобразит снеговика, состоящего из пяти окружностей разного радиуса и ведерка на голове из эллипса и двух отрезков (рис. 1.15).

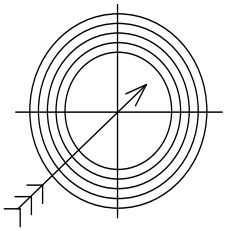


Рис. 1.14. Мишень

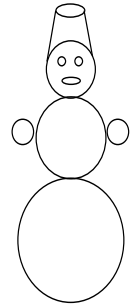


Рис. 1.15. Снеговик

100. Напишите программу построения цилиндра, состоящего из эллипсов и двух вертикальных линий (рис. 1.16).

101. При помощи Бейсика нарисуйте летающую тарелку (рис. 1.17).

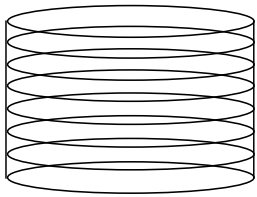


Рис. 1.16. Цилиндр

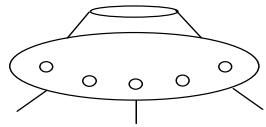


Рис. 1.17. Летающая тарелка

Ну что ж, если все получается, то пора постепенно переходить к более сложным вещам, а именно к рисованию дуг окружностей и эллипсов. Но прежде следует вспомнить, что в Бейсике используется радианная мера углов *(см. разд. "Стандартные функции Бейсика")*. Любая дуга имеет угол, от которого она начинается, и угол, где она заканчивается. Чтобы разобраться с этим, сначала я приведу оператор рисования дуг окружностей:

```
CIRCLE (X, Y), R, C, a, b
```

где появившиеся две новые величины `a` и `b` и обозначают эти углы. Правило рисования дуг на Бейсике звучит так: "дуга строится от угла `a` к углу `b` против часовой стрелки".

Для того чтобы показать это наглядно, лучше всего обратиться к тригонометрической окружности (рис. 1.18).

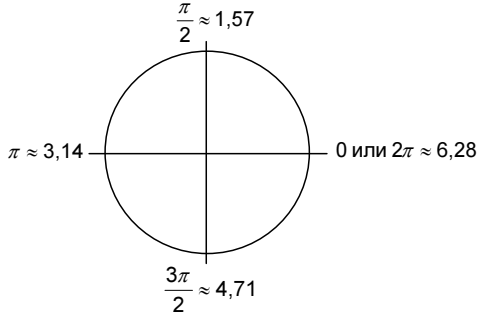


Рис. 1.18. Тригонометрическая окружность

Взглянув на тригонометрическую окружность, мы всегда довольно легко сможем определить начало и конец нужных нам дуг. Более того, Бейсик позволяет использовать в качестве операндов в своих командах арифметические выражения. Поэтому, например, если вы знаете угол начала дуги — 30°, но затрудняетесь определить его на тригонометрической окружности, то можете в соответствующем месте оператора дуги написать

```
3.14*30/180
```

и пусть компьютер сам считает.

Приведем несколько простых примеров.

Построим верхнюю половину окружности синего цвета с радиусом 30 (рис. 1.19), у которой центр находится в точке 150, 100.

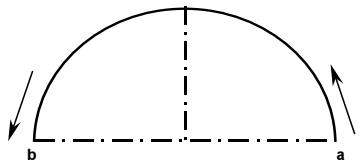


Рис. 1.19. Верхняя половина окружности

Первым делом, глядя на дугу, определяем направление против часовой стрелки. Это дает нам возможность выяснить, где нахо-

дится начало и конец дуги. Далее обращаемся к тригонометрической окружности. В этом простом примере сразу видно, что угол начала a=0, а угол конца дуги b=3,14. Таким образом, оператор построения этой дуги выглядит так:

```
CIRCLE (150, 100), 30, 1, 0, 3.14
```

Построим правую половину той же самой окружности (мы уже не будем сопровождать это построение рисунком, т. к. процесс аналогичен). По тригонометрической окружности определяем, что a=4,71, b=1,57. Записываем оператор построения дуги:

```
CIRCLE (150, 100), 30, 1, 4.71, 1.57
```

Предупреждение

Прежде чем мы приведем более сложный пример дуги, хочется сделать еще одно полезное замечание. Если в операторе построения дуг поставить знак минус перед значениями углов дуги, то автоматически будут проведены радиусы, соединяющие центр окружности с концами дуг. Нельзя ставить минус перед 0. В этом случае вместо нуля надо использовать –6,28 (т. е. 2π).

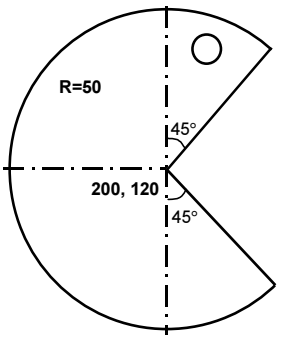


Рис. 1.20. CD-MAN

Построим изображение CD-MAN (рис. 1.20). Радиус "глаза" и координаты его центра определим исходя из здравого смысла.

Хотя на первый взгляд кажется, что данное построение очень сложно, но на самом деле мы знаем уже достаточно инструментов, чтобы сделать это двумя изящными легкими движениями.

Для начала, действительно давайте определим радиус "глаза". Диаметр его составляет примерно пятую часть радиуса "туловища", стало быть радиус — одну десятую, т. е. 5. Координаты центра примерно X=215, Y=85. Теперь займемся углами дуги "туловища". Угол `a` определяется довольно легко — он равен 45°, а вот для определения угла `b` надо пройти почти всю тригонометрическую окружность — первая четверть, вторая четверть — еще +90°, третья четверть — еще +90°, и, наконец, еще +45°. Итого b=315°. Ну что ж, все данные для построения есть. Приступим.

```
Rem "туловище"

CIRCLE (200, 120), 50, 1, -45*3.14/180, -315*3.14/180
CIRCLE (215, 85), 5, 1     Rem "глаз"
```

Еще раз напоминаю, что в программировании главное — это предварительная подготовка всех исходных данных!

Теперь рассмотрим рисование дуг эллипсов.

Когда мы говорили о построении самих эллипсов, то в операторе было несколько странно видеть три указанных подряд запятых. Сейчас все стало понятно — это место для углов `a` и `b`. Остальное — аналогично дугам окружности. Итак, оператор построения дуг эллипса таков:

```
CIRCLE (X, Y), R, C, a, b, K
```

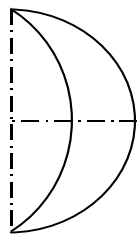


Рис. 1.21. Месяц

В качестве примера нарисуем "светит месяц, светит ясный..." (рис. 1.21).

Внешняя граница месяца — это дуга окружности, а внутренняя — дуга эллипса. Центр у них общий (X=500, Y=70). Коэффи-

циент сжатия эллипса равен 2. Начало и конец дуг тоже одинаковы (a=4,71, b=1,57). Радиус окружности возьмем 50, а вот с радиусом эллипса придется помучиться, поскольку, как мы уже говорили, экранные точки имеют разные ширину и высоту, определяющиеся примерным соотношением 3:4. Поэтому в качестве радиуса эллипса возьмем значение 50×3/4. Тогда месяц построится следующим образом:

```
CIRCLE (500, 70), 50, 14, 4.71, 1.57      Rem дуга окружности
CIRCLE (215, 85), 50*3/4, 4.71, 1.57, 2   Rem дуга эллипса
```

Закраска

Уже очень хочется попробовать свои силы? Еще немножко нужно подождать. Ведь мы хотим все делать красиво. И нам нужно изучить еще один шаг — *закраску замкнутых контуров*.

После того как мы научимся закрашивать, у нас в руках будут практически все необходимые инструменты для рисования сложных объектов, разработки больших линейных алгоритмов.

Итак, оператор закраски имеет следующий синтаксис:

```
PAINT(X, Y), C1, C2
```

где `X`, `Y` — координаты любой точки внутри закрашиваемого контура, `C1` — цвет, которым закрашивается контур, `C2` — цвет самого контура. Если эти цвета совпадают, то достаточно указать `C1`.

Правила закраски:

- Контур должен быть *замкнут*. Если в нем будет прокол хотя бы в одну экранную точку, то вся "краска" вытечет и зальет экран.
- Контур должен быть *одноцветен*. Если составляющие даже замкнутого контура разных цветов, то для компьютера эта ситуация аналогична разрыву. Будет закрашен весь экран.
- Координаты точки закраски должны лежать *внутри контура*.

Рекомендуется закрашивать контур непосредственно после того, как он нарисован. Часто рисуют все изображение, потом начинают закрашивать разные области, а линии разного цвета пересека-

ясь дают разрывы, и вот уже экран затекает самыми разными красками.

Если точка закраски попала *вне контура*, то закрасится весь экран, за исключением самого контура (иногда это бывает нужно, но подобные случаи редки).

Если точка закраски попала *на контур*, то ничего не закрасится.

Есть общая рекомендация из личного опыта автора, немного удлиняющая процесс отладки программы, но дающая 100%-ный правильный результат закраски. Перед закраской вместо оператора `PAINT` лучше написать оператор `PSET` с теми же координатами, но какого-нибудь контрастного цвета, чтобы ее было хорошо видно. Запустить программу и убедиться, что все вышеназванные правила соблюдены и точка находится внутри закрашиваемого контура. После этого возвращаемся в программу, изменяем оператор `PSET` на `PAINT`.

Рассмотрим пример закраски месяца из предыдущей программы белым цветом. После того как мы его нарисовали, у нас добавится всего одна команда:

```
PAINT(530, 70), 15, 14
```

Откуда взялось число 530 в качестве координаты X, я думаю, понятно, если посмотреть на рисунок, вспомнить координаты центра и радиус.

Ну что ж, а теперь упражнения на дуги и закраску. В них изображения уже более сложные, требующие достаточно основательной предварительной подготовки. Не забывайте о ней!

102. Напишите программу, выводящую на экран образ, аналогичный примеру CD-MAN, изменив лишь углы и координаты центра "глаза" (рис. 1.22).

Закрасьте "туловище" одним цветом, а "глаз" — другим.

103. Создайте программу, отображающую месяц в обратную сторону, чем в примере, рассмотренном выше. Закрасьте его красным цветом.

104. С помощью Бейсика нарисуйте закрашенный квадрат (одним оператором) с вписанными в него двумя разноцветными окружностями (рис. 1.23).

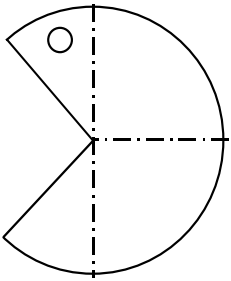


Рис. 1.22. CD-MAN, глядящий на запад

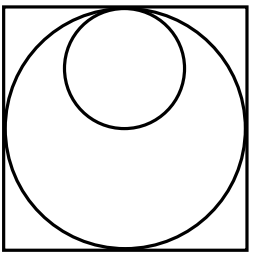


Рис. 1.23. Квадрат с окружностями

105. Напишите программу, выводящую на экран петлю гистерезиса, которая состоит из двух отрезков прямых линий и четырех дуг, являющихся четвертями окружности (рис. 1.24). Попробуйте сначала нарисовать изображение в масштабе на бумаге, рассчитать все исходные данные. Закрасьте получившееся изображение. (Будьте внимательны при состыковке дуг, зачастую там образуются разрывы.)

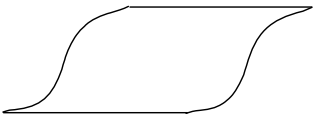


Рис. 1.24. Петля гистерезиса

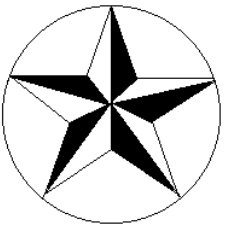


Рис. 1.25. Звезда

106. Выведите на экран пятиконечную звезду, вписанную в окружность и закрашенную как на рис. 1.25.

107. Нарисуйте корабль в ночном море под Российским флагом (рис. 1.26).

Это задание уже представляет собой достаточно сложное изображение. Попробуйте выполнить его так, как вы считаете нужным, а мы хотим изучить, как это нужно делать.

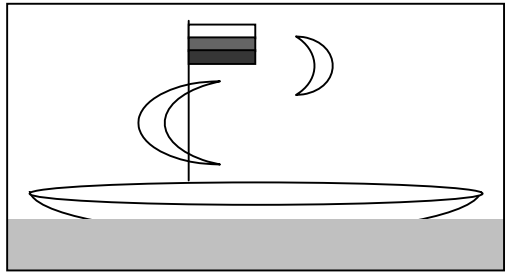


Рис. 1.26. Корабль в ночи

Правила построения сложных изображений

При рисовании придерживайтесь определенных правил, которые перечислены далее.

- Обязательно начертите свое изображение на бумаге в масштабе.
- Разбейте изображение на те самые графические примитивы, каждый из которых вы могли бы нарисовать каким-либо уже изученным оператором.
- Найдите и выделите на рисунке опорные точки — концы отрезков, начала и концы дуг, центры окружностей, эллипсов, отдельные точки (если они есть). Наметьте точки для оператора закраски `PAINT`. Если в изображении есть прямоугольники со сторонами, параллельными экрану, то необходимо выделить координаты их базовых диагоналей, по которым они будут строиться.
- Определите максимальный размер изображения в экранных точках, исходя из размеров экрана.
- Определите координаты всех опорных точек, значения радиусов окружностей и эллипсов, коэффициенты сжатия эллипсов, углы всех дуг.
- Определите последовательность построения и закраски замкнутых контуров.

- Напишите программу, введите ее, снабжая как можно большим количеством необходимых комментариев (можно сначала на бумаге), и отладьте ее.

Предупреждение

Запускать программу необходимо не после ее ввода целиком, а следом за первым логически законченным фрагментом изображения, чтобы иметь возможность сразу оценить правильность того или иного вашего программного решения.

108. Попробуйте самостоятельно построить сложное изображение по вышеизложенным правилам на примере рис. 1.27.

109. А вот еще несколько вариантов сложных изображений (рис. 1.28).

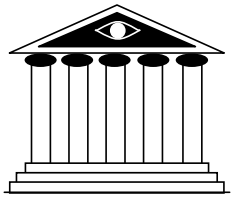


Рис. 1.27. Храм

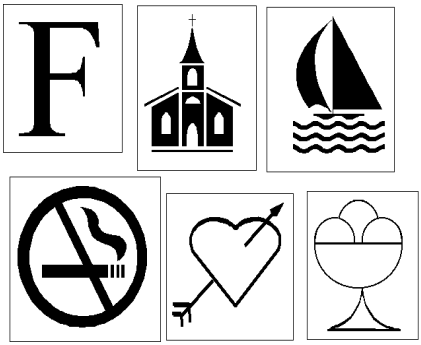


Рис. 1.28. Буква F, католический костел, яхта, знаки "Не курить" и "Любовь", мороженое

110. Так как программированием чаще занимаются представители сильного пола, а среди них, как известно, бывает довольно много любителей футбола, то предлагаю изобразить эмблему своей любимой команды. У меня это, конечно же, "Зенит"! Вот, кстати, как выглядит их эмблема (рис. 1.29). Сложно? Конечно, да! Но ведь ради любимой команды можно и постараться!

111. В качестве самостоятельного задания придумайте себе сами эмблему и нарисуй ее.

Рис. 1.29. Эмблема футбольной команды "Зенит"

Макроязык GML

Для расширения возможностей машинной графики Бейсика был дополнительно разработан специальный макроязык GML (Graphics Macro Language). Он позволяет строить довольно сложные изображения и быстро выводить их на экран. Каждая команда языка представляет собой латинскую букву, после которой следует один или два числовых параметра (как правило, целых числа). Команды этого языка перечислены в табл. 1.3.

Таблица 1.3. Команды языка GML

Команда	Действие
`Un`	Переместиться вверх на n точек
`Dn`	Переместиться вниз на n точек
`Ln`	Переместиться влево на n точек
`Rn`	Переместиться вправо на n точек
`En`	Переместиться по диагонали вверх и вправо на n точек
`Fn`	Переместиться по диагонали вниз и вправо на n точек
`Gn`	Переместиться по диагонали вниз и влево на n точек
`Hn`	Переместиться по диагонали вверх и влево на n точек
`M x, y`	Переместиться в точку с координатами x, y
`M ±n, ±m`	Переместиться по отношению к текущей позиции на ±n точек по оси X и на ±m точек по оси Y. Знаки плюс и минус проставлять обязательно
`At`	Поворот изображения против часовой стрелки вокруг точки, с которой начиналось рисование на 90°×t (t∈0,1,2,3). Действует во всех дальнейших командах до нового назначения

Таблица 1.3 *(окончание)*

Команда	Действие
`Cn`	Задание нового цвета. Действует во всех дальнейших командах до нового назначения
`Sn`	Расстояние, указанное в командах перемещения, умножается на n/4 (0≤n≤255)
`B`	Переместиться в новую позицию при помощи следующих за `B` команд, но рисование не производить. Отменяется установкой цвета `C`
`N`	Выполнить следующую команду перемещения и вернуться в исходную позицию. Может предшествовать любой команде перемещения
`Pc1,c2`	Команда заполнения контура цветом, `c1` — цвет заполнения, `c2` — цвет контура. Должны соблюдаться все вышеизложенные правила закраски, и курсор должен находиться внутри закрашиваемого контура

Основные команды перемещения легко запомнить с помощью рис. 1.30.

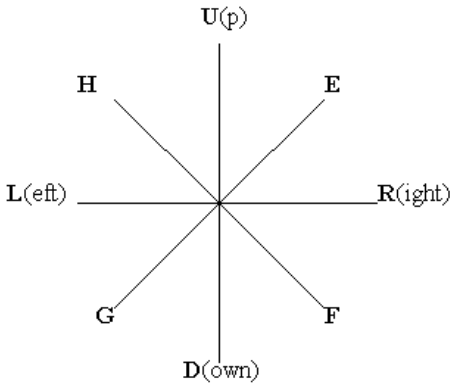


Рис. 1.30. Основные команды перемещения языка GML

Для приведения в действие последовательности команд языка GML необходимо использование оператора `DRAW`.

Например, команда

```
DRAW "R50 D50 L50 U50"
```

изобразит нам квадрат, начиная от последней графической точки.

Если такой точки не было, и вы сразу начинаете строить изображение с помощью оператора DRAW, то по умолчанию исходной точкой считается центр экрана.

Если вы подготовите сначала какое-либо изображение в строковой переменной, то потом его можно будет многократно вызывать на экран. Для этого используется функция "X". Ее работу рассмотрим на примере.

```
N$="R50 D50 L50 U50"
DRAW "C14 X"+VARPTR$(N$)
DRAW "BM500, 50 C2 X"+VARPTR$(N$)
```

С помощью первого оператора запоминаем последовательность команд построения квадрата в строковой переменной N$. Вторым оператором задаем цвет 14 и сообщаем компьютеру командой X, что необходимо выполнить последовательность команд N$. Это делает функция VARPTR$. Последним оператором перемещаемся без рисования в точку с координатами 500, 50, меняем цвет на 2 и снова выполняем последовательность команд N$.

В результате на экране в центре и в правом верхнем углу должны появиться одинаковые разноцветные квадраты со стороной 50.

Совместное использование графических примитивов и оператора DRAW позволит вам рисовать очень качественные изображения на вашем экране.

112. Для того чтобы опробовать эти возможности, нарисуйте елочку (рис. 1.31).

Рис. 1.31. Елочка

113. Если вам удалось предыдущее задание, то попробуйте вырастить на экране еловый бор из трех одинаковых елочек, пририсовать к ним траву, солнце, небо, облака — в общем, все, что подскажет вам фантазия.

Вывод текстовой информации в графике

Часто хочется сопроводить свои рисунки подписями. Не сдерживайте свои желания. Делайте это просто — с помощью операторов `LOCATE` и `PRINT`, как мы это уже многократно делали.

Предупреждение

Только надо помнить, что координаты в операторе `LOCATE` не графические (640×480), а текстовые (80×30).

А теперь несколько упражнений на совмещение арифметическо-алгебраических вычислений, графики и вывода текстовых сообщений на экран.

114. Запрашивается радиус круга (от 10 до 100 пикселов). Напишите программу, которая вычисляет площадь этого круга. Круг нарисован в зависимости от введенного радиуса, внутри него выводится вычисленная площадь с точностью до сотых.

115. Запрашивается радиус окружности (от 10 до 100 пикселов). Напишите программу, которая вычисляет длину этой окружности. Окружность нарисована в зависимости от введенного радиуса, внутри нее выводится вычисленная длина с точностью до тысячных.

116. Запрашиваются диагонали ромба. Создайте программу, вычисляющую площадь ромба. Ромб изображается, диагонали подписываются, площадь ромба выводится под ним.

117. Разработайте программу, которая находит площадь трапеции по ее основаниям и высоте. Трапеция должна быть нарисована, исходные данные — подписаны, площадь — выведена внутри.

118. Вычислите объем цилиндра с радиусом основания R и высотой H. Цилиндр, естественно, должен быть нарисован.

119. Определите координату середины отрезка (X, Y), если известны координаты концов отрезка: (X1, Y1) и (X2, Y2). Отрезок, как повелось, необходимо нарисовать, все координаты — вывести.

120. Даны декартовы координаты вершин треугольника (в плоскости). Разработайте программу, вычисляющую площадь и периметр этого треугольника. Треугольник должен присутствовать на экране в нарисованном виде, вычисленные длины сторон — подписаны, под треугольником — выведены его площадь и периметр с точностью до сотых.

121. Определите расстояние, пройденное физическим телом за время T, если тело движется с постоянным ускорением A и имеет в начальный момент времени скорость V. Нарисовать это расстояние в виде толстого горизонтального отрезка, его длину подписать.

Построение диаграмм

Вот уж где могут пригодиться графические примитивы, так это при построении диаграмм, иллюстрирующих различные достижения или сравнительные характеристики. Прежде чем дать задания, приведем два примера, где построение диаграммы предваряется расчетами.

Пример. Население земного шара по континентам: Европа — 800 млн чел., Азия — 4000 млн чел., Африка — 700 млн чел., Северная Америка — 500 млн чел., Южная Америка — 400 млн чел., Австралия — 40 млн чел.

Программа (пока мы не знаем циклов, потом это же можно будет проделать гораздо короче!) может быть записана так:

```
DATA 800, 4000, 700, 500, 400, 40
SCREEN 12
WINDOW (0, 480)-(640, 0)

LOCATE 1,4 : PRINT "РАСПРЕДЕЛЕНИЕ НАСЕЛЕНИЯ ЗЕМНОГО ШАРА ПО
КОНТИНЕНТАМ"
READ A
```

```
LINE (20, 60)-(120, 60 + A / 10), 2, BF
LINE (20, 60)-(120, 60 + A / 10), 15, B
LOCATE 28, 4: PRINT "EUROPE"

READ B
LINE (120, 60)-(220, 60 + B / 10), 14, BF
LINE (120, 60)-(220, 60 + B / 10), 15, B
LOCATE 28, 18: PRINT "ASIA"

READ C
LINE (220, 60)-(320, 60 + C / 10), 4, BF
LINE (220, 60)-(320, 60 + C / 10), 15, B
LOCATE 28, 31: PRINT "AFRICA"

READ D
LINE (320, 60)-(420, 60 + D / 10), 1, BF
LINE (320, 60)-(420, 60 + D / 10), 15, B
LOCATE 28, 43: : PRINT "N.AMERICA"

READ E
LINE (420, 60)-(520, 60 + E / 10), 6, BF
LINE (420, 60)-(520, 60 + E / 10), 15, B
LOCATE 28, 55: PRINT "S.AMERICA"

READ F
LINE (520, 60)-(620, 60 + F / 10), 13, BF
LINE (520, 60)-(620, 60 + F / 10), 15, B
LOCATE 28, 67: PRINT "AUSTRALIA"
```

Для улучшения внешнего вида диаграммы можете вывести над столбиками диаграммы числа, означающие население континента в млн чел.

А теперь несколько самостоятельных заданий:

122. Построить круговую диаграмму процентного соотношения богатых, людей среднего класса, бедных и людей, живущих за чертой бедности в России (данные найти самостоятельно).

123. Построить круговую диаграмму процентного распределения бюджета своей семьи за месяц (данные взять у родителей).

124. Построить столбиковую диаграмму распределения по росту учащихся вашего класса по следующей классификации: вы-

ше 175 см, от 170 до 175 см, от 165 до 170 см, от 160 до 165 см, ниже 160 см.

125. Построить столбиковую диаграмму, отражающую распределение своих оценок за месяц (данные взять в дневнике).

Разветвляющийся алгоритм

Ну что ж, надеюсь, что с линейными программами особых трудностей у вас не возникло. Пиши команды подряд, хотя, конечно, все сначала надо было продумать, и изображения у нас получались просто великолепные!

Однако надо двигаться вперед. А дальше у нас — разветвляющийся алгоритм. Что это такое, мы уже говорили. Например, витязь на перепутье: "Если налево пойдешь, коня потеряешь, если прямо пойдешь — смерть найдешь...", ну и т. д.

А вот как он реализуется на практике в Бейсике, мы поговорим сейчас.

Ветвление в алгоритме и программе осуществляется двумя способами:

- на основе *безусловного* перехода;
- на основе *условного* перехода.

Начнем с первого.

Безусловный переход

Безусловный переход оператором `GOTO` предписывает программе свернуть с линейного пути и, беспрекословно повинуясь, перейти к метке, расположенной в любом месте программы.

В качестве метки используются натуральные числа с двоеточием после них. Метка указывается только в начале строки (т. е. если в строке программы несколько операторов, то нельзя ставить метку, например, перед вторым). Приведем пример программы, рисующей на экране три символа звездочки по диагонали:

```
CLS
?"*"
?"?*"
?"??*"
```

Символ ? в данном случае означает пробел.

Добавим теперь метку 1 рядом со второй строкой и оператор GOTO в конце программы. Получаем:

```
CLS
1: ?"*"
?"?*"
?"??*"
GOTO 1
```

Казалось бы, изменения незначительные, но запустите программу и посмотрите что происходит! Вы в панике? Вы пытаетесь прекратить действие программы, но ничего не получается? Программа зациклилась! Что делать? Спокойно нажмите клавишу <Ctrl> и, не отпуская ее, клавишу <Break>. Это называется *прервать выполнения программы*. Переведите дух и давайте шаг за шагом разберемся, что же произошло.

Программа очищает экран, затем рисует первую звездочку, под ней с отступом в один пробел — вторую, потом ниже третью, а вот затем получает команду GOTO 1. Это приказ, которого невозможно ослушаться. Программа ищет метку, находит ее у команды рисования первой звездочки и начинает снова последовательно выполнять команды рисования — первую, вторую, третью и — снова приказ перехода, и так — до бесконечности (вернее, до прерывания программы). А поскольку оператор PRINT, не снабженный точкой с запятой, после выполнения всегда переводит курсор на следующую строку, то и получается эффект движения.

А теперь поупражняйтесь.

126. Попробуйте спрогнозировать, что произойдет, если метку 1 в программе рисования звездочек указать в первой строке перед оператором CLS? А что, если в четвертой строке? Или (кощунственная мысль!) перед самим оператором GOTO? Проверьте свои предположения на практике.

127. Напишите программу, которая заставит бежать по экрану следующий узор:

```
###
    #######
                #
                #
                #
    #######
###
```

128. Если вы обратили внимание на текст предыдущей программы, то должны были заметить, что узор легко прослеживается. Воспользовавшись этим, напишите программу, которая заставит бежать по экрану слово "РОК", а потом, если не лень, то и свое имя. Буквы этих слов надо изобразить символами клавиатуры (какими вам больше нравится).

129. Посмотрите на текст следующей программы:

```
X=0
1: X=X+1
? X
GOTO 1
```

Как вы думаете, при каком значении `X` программа закончит свою работу? Если у вас достаточно быстродействующий компьютер, можете рискнуть проверить свое предположение на практике. Да, впрочем, рискните и на медленном, ведь всегда можно прервать программу. Правда, в этом случае вы так и не узнаете тайны, какое самое большое целое число компьютер может вывести на экран.

Строго говоря, серьезное программирование не любит оператор `GOTO`. Считается, что он засоряет и путает программу. Вызывает зацикливание и зависание компьютера. Рекомендуют обходиться без него. А мне он все равно нравится. Но это мое личное даже не мнение, а просто чувство. При написании серьезных программ я тоже стараюсь обходиться без `GOTO`.

Условный переход

Мало того, что условный переход подразумевает выполнение тех или иных команд в зависимости от проверяемого условия, так Бейсик предоставляет нам еще и несколько вариантов действий.

Начнем с классической конструкции:

```
IF ... THEN ... ELSE
```

Или по-русски:

```
ЕСЛИ ... ТОГДА ... ИНАЧЕ
```

В записи условия можно использовать следующие символы:

- = (равно);
- > (больше);
- < (меньше);
- <> или >< (не равно);
- <= или =< (меньше или равно);
- >= или => (больше или равно).

Мне кажется, что уже что-то должно быть понятно из самой конструкции условного оператора. Но раскроем его смысл на примере следующей программы. Допустим, в некоторой стране живут люди, говорящие только правду. Но при этом и у них есть вредные привычки, и некоторые из них курят. Для продажи сигарет сделаны автоматы, которые запрашивают возраст покупателя и реагируют на это двумя различными способами в зависимости от названной правдиво величины.

```
CLS
INPUT "Как Вас зовут"; N$
INPUT "Сколько Вам полных лет"; S
IF S<18 THEN ?"Вам еще рановато курить, ";N$ ELSE ?'Минздрав
России предупреждает — курение опасно для вашего здоровья!"
```

Программа запрашивает имя, потом возраст (все это уже не вызывает у нас никаких затруднений), а вот потом вступает в действие условный оператор. Если возраст S меньше 18, то выводится надпись, что курить еще рановато (а чтобы не было очень обидно, обращается по имени). Иначе (т. е. если возраст больше либо

равен 18) автомат продает сигареты, но предупреждает об опасности курения.

Еще один маленький пример. Вычислим корень квадратный из X.

```
CLS
INPUT "Введите любое число"; X
? "Корень квадратный из X="; SQR(X)
```

Запускаем программу. Вводим число 4. Ответ 2. Вводим 121. Ответ 11. Вводим –9. Аварийное прерывание программы и сообщение об ошибке "Неверный вызов функции". Что такое? А все потому, что корень квадратный вычисляется только из положительного числа. Вот программа и "ругается". Но мы же умные, добавим в программу условный оператор, и будет она работать для любых чисел.

```
CLS
1:INPUT "Введите любое число";X
IF X>=0 THEN ? "Корень квадратный из X ="; SQR (X) ELSE
?"Корень квадратный из отрицательного числа вычислять
отказываюсь. Дайте положительное число!":GOTO 1
```

Теперь программа анализирует введенное значение и, в случае его положительности, вычисляет корень, а в случае отрицательности в легкой форме раздражается и просит у пользователя другое число.

И еще пример с графикой. Предположим, мы настолько дружественны к пользователю, что не говорим ему: "Вот, посмотри, какая картинка!", а предлагаем некий выбор. Для первого раза из двух возможностей.

```
SCREEN 12
INPUT "Введите 1, если хотите увидеть солнце и любую другую
цифру, если ничего не хотите"; X
IF X=1 THEN CIRCLE (320, 240), 50, 2: PAINT(320, 240), 14, 2
ELSE ?"Спасибо за внимание!"
```

Программа запрашивает у пользователя цифру (причем в случае ввода буквы просит осуществить ввод заново). Если это цифра 1, то рисуется желтый круг, в любом другом случае появляется надпись "Спасибо за внимание!".

Поупражняемся.

130. Напишите программы, которые в зависимости от введенного числа либо вычисляют функцию, либо выдают сообщение, что функция не определена:

$$y = \frac{1}{x}$$

$$y = \sqrt{x^2 - 1}$$

131. Напишите программу для вычисления функции:

$$y = \begin{cases} \dfrac{\cos x}{x}, & x > 0 \\ x \sin x, & x \le 0 \end{cases}$$

132. Напишите программу, определяющую четность или нечетность введенного с клавиатуры целого числа.

133. Напишите программу, находящую меньшее из двух, введенных с клавиатуры чисел.

134. По четырехзначному номеру года, запрошенному с клавиатуры определите номер столетия (например, для 1492 г. — ответ XIV век, для 1812 г. — XVIII век). Учесть, что началом века считается первый, а не нулевой год. (То есть, 2000-й год, из астрономии — последний год XX века.)

Выше мы рассмотрели *полную форму условного оператора*, когда в зависимости от истинности или ложности условия выполняются те или иные действия, после чего программа переходит к выполнению следующей за условным оператором команды.

Существует также *неполная форма условного оператора*. Она выглядит так

```
IF условие THEN
команды_выполняемые_в_случае_истинности_условия
```

В этом случае программа проверяет условие и в случае его истинности выполняет команды, указанные после служебного слова `THEN`. Если же условие ложно, то программа переходит к выполнению следующей за условным оператором команды.

Например, программу автомата по продаже сигарет можно модифицировать при помощи сокращенного варианта условного оператора:

```
CLS
INPUT "Как Вас зовут"; N$
INPUT "А сколько Вам полных лет"; S
IF S<18 THEN ?"Вам еще рановато курить, ";N$ : END
?'Минздрав России предупреждает — курение опасно для вашего
здоровья!"
```

Здесь появляется новый оператор END, выполнение которого в любом месте программы приводит к ее остановке.

Возможен еще также составной условный оператор, когда после служебных слов THEN или ELSE вновь следует условный оператор, когда выполнение тех или иных действий зависит не от одного, а от нескольких условий.

Например, допустим, что Преображенский полк гренадеров набирали только из тех новобранцев, рост которых был не менее 180 см, а вес — не менее 80 кг. Программа этого процесса выглядела бы следующим образом:

```
CLS
INPUT "Как Вас зовут?"; N$
INPUT "Каков Ваш рост в см"; R
INPUT "Каков Ваш вес в кг"; M
IF R>=180 THEN IF M>=80 THEN ?"Вы — достойный кандидат
в гренадеры, ";N$ ELSE ? N$; ", к сожалению, вы в гренадеры не
годитесь"
```

Здесь условный оператор сначала проверяет соответствие кандидата по росту, а затем еще и по весу, и только в случае выполнения обоих условий выдается сообщение о пригодности кандидата.

При использовании составных условных операторов надо быть очень осторожным, чтобы не запутаться самому в THEN и ELSE. Есть правило, по которому ELSE относится к ближайшему слева от него THEN. В рассматриваемом случае первый условный оператор, который проверяет рост, относится к сокращенной форме, у него нет ELSE, а второй является полным. Отсюда получается

такой казус, что если кандидат сразу не проходит по росту, то на экране вовсе не появится никакого сообщения, потому что программа должна перейти к следующей за условным оператором команде, а ее нет. Чтобы разобраться в этом, выполните упражнение. Вообще, лучше не злоупотреблять составными условными операторами, а разбивать их на несколько простых или использовать сложные условия, о которых мы поговорим чуть ниже.

135. Какое значение будет иметь переменная X после выполнения очередных операторов:

```
X=0
IF A>0 THEN IF B>0 THEN X=1 ELSE X=2
```

при значениях переменных A и B:

```
A=1, B=1
A=1, B=-1
A=-1, B=1
```

136. Запишите в виде одного условного оператора указанные действия:

- известно, что из трех чисел A1, A2 и A3 одно отлично от других, равных между собой. Присвоить номер этого числа переменной N;
- $y = \begin{cases} \cos^2 x, & 0 < x < 2 \\ 1 - \sin^2 x \end{cases}$.

137. Напишите программу, запрашивающую у пользователя три разных целых положительных числа и находящую сумму двух наименьших из них.

138. Если N=3, то какое значение будет иметь переменная S после выполнения следующего оператора:

```
S=1: K=2
1: IF K>N THEN GOTO 2
S=S*K
K=K+1
GOTO 1
2: ? "S="; S
```

Предупреждение

В сокращенной форме условного оператора при переходе к метке можно опускать оператор GOTO.

Таким образом, вторая строчка предыдущей программы могла бы выглядеть так:

```
1: IF K > N THEN 2
```

Сложные условия

Если для выполнения тех или иных действий нужно одновременное выполнение двух и более условий, то они объединяются логической связкой AND (И). Например, программу про набор в Преображенский полк можно было бы изменить таким образом:

```
CLS
INPUT "Как Вас зовут?"; N$
INPUT "Каков Ваш рост в см"; R
INPUT "Каков Ваш вес в кг"; M
IF R>=180 AND M>=80 THEN ?"Вы — достойный кандидат
в гренадеры, "; N$ ELSE ? N$; ", к сожалению, вы в гренадеры
не годитесь"
```

А программа для вычисления функции

$$y=\begin{cases} x^2, & x>1 \\ x, & 0\le x\le 1 \\ -2x+1, & x<0 \end{cases}$$

выглядела бы следующим образом:

```
CLS
INPUT "Введите значение X"; X
IF X > 1 THEN Y=X ^ 2
IF X <= 0 AND X <= 1 THEN Y=X ELSE Y=-2*X+1
? "Y (X)="; Y
```

Третье условие мы не рассматриваем потому, что если не выполняются первые два, то, очевидно, X находится в третьем интервале, поскольку описанные условия охватывают всю числовую ось.

Если есть несколько условий и для организации тех или иных действий достаточно выполнения хотя бы одного из этих условий, то применяется логическая связка OR (ИЛИ). Например, программа для вычисления следующей функции:

$$y=\begin{cases} x^3, & x<-1 \\ x^2, & -1\le x\le 1 \\ x^3, & x>1 \end{cases}$$

выглядит так:

```
CLS
INPUT "Введите значение X"; X
IF X<-1 OR X>=1 THEN Y=X ^ 3 ELSE Y=X ^ 2
? "Y (X)="; Y
```

Для различия применения AND и OR можно привести такой жизненный пример. Каждый человек хоть раз задумывался о спутнике жизни и, естественно, выдвигал какие-либо условия к будущему избраннику. Так вот, более требовательные могли бы выдвинуть два таких условия: пусть он будет красивым И (AND) богатым. А менее требовательные, предпочитающие синицу в руках, могли бы сказать так: пусть он будет красивым ИЛИ (OR) богатым. Чувствуете разницу? В первом случае избраннику не будет отказано только в одном случае из четырех возможных, а во втором, наоборот, три исхода благосклонны. Он может быть только красив, или только богат, ну а если и красив, и богат, то тем более.

Еще один пример. В стене существует квадратное отверстие N×N см. И имеется кирпич с измерениями A, B и C. Определить, пройдет он в отверстие или нет, если подавать его можно только параллельно стенкам отверстия.

Понятно, что кирпич пройдет в отверстие только в случае, если хотя бы два его измерения меньше N. Отсюда программа будет выглядеть так:

```
CLS
INPUT "Введите сторону отверстия N"; N
1: INPUT "Введите стороны кирпича A, B и C"; A, B, C
```

```
IF A<N AND B<N OR A<N AND C<N OR B<N AND C<N THEN ?"Кирпич
проходит в отверстие" ELSE ?"Кирпич не проходит в отверстие"
INPUT "Рассмотрим еще один кирпич? 1- да, 0 — нет"; X
IF X=1 THEN GOTO 1 ELSE ?"Спасибо за работу!"
```

По-русски это условие можно объяснить следующим образом. Если A и B меньше N, или A и C меньше N, или B и C меньше N, тогда печатать "Кирпич проходит в отверстие", иначе печатать "Кирпич не проходит в отверстие". Кроме того, в этой программе применен достаточно простой, но эффективный ход, управляющий совместной работой пользователя и программы. Чтобы каждый раз не запускать программу (если пользователь хочет рассмотреть несколько вариантов исходных данных), применен запрос с клавиатуры вариантов продолжения: либо рассматриваем еще один кирпич, либо — конец программы. Там же вы можете видеть применение безусловного перехода в условном операторе. Разберитесь и пользуйтесь этим для грамотного оформления своих программных продуктов.

Напишите программы для решения следующих задач, используя условный оператор и сложные условия.

139. По введенным с клавиатуры коэффициентам квадратного уравнения A, B и C найдите его корни. Рассмотрите шесть возможных вариантов:

- `A=B=C=0`, корней бесчисленное множество (X — любое);
- `A=B=0, C≠0`, уравнение не имеет корней;
- `A=0, B≠0, C≠0`, вырожденное квадратное уравнение, имеется один корень (формулу вычисления корня найдите сами);
- `D<0`, где `D` — дискриминант, который предварительно надо вычислить; уравнение не имеет вещественных корней;
- `D=0`, уравнение имеет два одинаковых корня (вывести их значения);
- `D>0`, уравнение имеет два различных вещественных корня (вычислить и вывести их значения).

Для проверки правильности работы программы предлагается шесть тестовых вариантов исходных данных:

- A=B=C=0;
- A=B=0, C=1;
- A=0, B=3, C=6 (должно получиться X=-2);
- A=5, B=3, C=2;
- A=1, B=2, C=1 (должно получиться X1=X2=-1);
- A=2, B=5, C=2 (должно получиться X1=-2, X2=-8).

140. Осуществите запрос трех целых различных чисел с клавиатуры. Выведите на экран наибольшее и наименьшее.

141. Осуществите запрос с клавиатуры у тренера олимпийской сборной России по марафону результаты в часах, минутах и секундах трех победителей чемпионата России. Если какие-то два результата различаются менее чем на 5 сек, выведите сообщение "Вот так шла борьба за _____ медаль". В сообщении должно быть указано достоинство медали (золотая, серебряная), за которую шла борьба. В противном случае, вычислить среднюю скорость победителя в км/час, если принять длину марафонской дистанции 42 км 195 м.

142. А сейчас мы попробуем сделать пока не очень красивый, но очень простой вариант телевизионной игры "О, счастливчик!". Придумайте пять любых вопросов, и к каждому из них четыре варианта ответов. Теперь я попробую словесно описать алгоритм, а вы — перевести его на Бейсик. Итак, запрашиваем у игрока имя, и узнаем, желает ли он играть. Если не желает, прощаемся, если желает — приветствуем и предлагаем первый вопрос с вариантами ответов. Запрашиваем у игрока с клавиатуры, какой вариант он выбирает. В случае правильного ответа начисляем ему сто очков и переходим ко второму вопросу. Если ответ неверен, то отправляемся на самое начало — снова регистрация и т. д. Первый вопрос — 100 очков, второй — 200, третий — 300, четвертый — 500, пятый — 1000. Если игрок правильно отвечает на все пять вопросов, то поздравляем его и заканчиваем программу.

143. Каждую пятницу члены Клуба толстяков выстраиваются в определенном порядке и взвешиваются. Напишите программу, которая хранит данные взвешивания 10-ти членов Клуба за прошлую неделю. Затем программа запрашивает новые данные взвешивания и для каждого члена Клуба либо выводит поздравление в случае похудения, либо величину прибавки веса с сожалением.

144. Напишите программу, запрашивающую у пользователя три разных целых положительных числа и находящую сумму двух наименьших из них.

145. В одном из заданий мы уже вычисляли площадь треугольника по формуле Герона. Теперь усложним нашу задачу. С клавиатуры запрашиваются целые числа a, b и c. Программа проверяет, можно ли, представив, что эти числа означают длины сторон, составить из них треугольник, затем рисует его на экране и вычисляет его площадь. Если треугольник с такими сторонами не существует, то на экране появляется соответствующее сообщение и картинка.

146. Определите и выведите на экран номер квадранта, в котором расположена точка A(x, y), x и y — заданные целые числа.

147. Составьте программу, которая определяет, принадлежит ли точка M(x, y) кругу с центром в точке *Z*(a, b) и радиусом равным r.

148. Составьте программу, которая определяет, принадлежит ли точка N(x, y, z) шару с центром в точке Z(a, b, c) и радиусом r.

149. Определите, какая из двух фигур (круг или квадрат) имеет большую площадь. Известно, что сторона квадрата равна a, радиус круга — r. Выведите на экран название и значение площади большей фигуры.

В рамках изучения условного оператора мы можем проанализировать основные принципы движения геометрических объектов по экрану.

Но прежде необходимо рассмотреть программу-сумматор, которая по мере ввода чисел вычисляла бы их сумму, пока мы не захотим прекратить этот процесс. Она выглядит так:

```
CLS
S=0       Rem S — переменная, в которую
          Rem будет заноситься накапливаемая сумма.
          Rem Естественно, пока мы еще не ввели
          Rem ни одного числа, она должна быть обнулена
1: INPUT "Введите любое число"; N
S=S+N
?"Сумма стала равна"; S
INPUT "Еще число? (1 — да, 0 — нет"; R
IF R=1 THEN GOTO 1 ELSE ?"Спасибо за работу!"
```

Выполните упражнение.

150. Попробуйте сами написать программу, вычисляющую произведение последовательно вводимых с клавиатуры, отличных от нуля, чисел. Что изменится по сравнению с предыдущей программой?

Движение геометрических объектов по экрану

Так как интерпретатор Бейсика достаточно медлителен, то очень красивых программ с движением нам не сделать, но зная принципы, легко потом добиться большего на других языках.

Итак, движение. На самом деле, конечно, ничего там не перемещается, а создается иллюзия движения за счет того, что сначала рисуется изображение, потом оно стирается (т. е. рисуется на том же месте цветом фона), а затем вновь выводится в новом месте экрана. Компьютер достаточно быстро проделывает это, и нам кажется, что шарик, например, двигается. Начнем мы с точки, которая смещается по горизонтали, отражаясь от вертикальных сторон экрана.

```
SCREEN 12
X=300: Y=175  ' начальные координаты точки
DX=1      ' приращение координаты X.
          ' Так как мы его сделали положительным,
          ' координата X будет увеличиваться, а, следовательно,
          ' точка сначала будет двигаться вправо
1: PSET (X, Y), 14    ' вывод точки на экран
PSET (X, Y), 0        ' стирание точки цветом фона
```

```
IF X=0 OR X=640 THEN DX=-DX  ' в случае достижения точкой
                ' вертикальных сторон экрана, приращение по X
                ' меняется на противоположное, а, значит, точка
                ' начнет двигаться в обратную сторону
X=X+DX    ' изменение координаты точки
GOTO 1    ' переход на метку 1, где происходит рисование точки
```

Программа, как вы видите, получилась зацикленной, т. е. точка будет двигаться бесконечно, пока мы не нажмем клавиши <Ctrl>+<Break>.

Предупреждение

Если у вас уже очень хороший компьютер, то точка может двигаться достаточно быстро, и вы не сможете насладиться процессом ее перемещения. В таком случае мы искусственно замедлим программу.

Для этого между двумя операторами `PSET` надо вставить два оператора:

```
FOR I=1 to 100: NEXT I
```

Это оператор цикла, о котором мы поговорим позже; пока просто поймите, что в данном месте компьютер посчитает до 100 в своем уме, и на это время в программе возникнет пауза. Подставляя вместо числа 100 другие значения, вы сможете управлять величиной паузы.

151. Напишите программу, которая заставит точку двигаться по вертикали. Потом замените точку на закрашенную окружность, а затем — на закрашенный прямоугольник.

Теперь рассмотрим организацию движения точки, которая меняет свое положение на экране и отражается от всех его четырех сторон под углом 45°, как это показано на рис. 1.32.

```
SCREEN 12
X=300: Y=240  ' начальные координаты точки
DX=1: DY=1    ' приращение координат X и Y.
      ' Так как мы их сделали положительными,
      ' координаты X и Y будут увеличиваться,
      ' а, следовательно, точка сначала будет двигаться
      ' вправо вниз под углом 45 градусов
```

```
1: PSET (X, Y), 14     ' вывод точки на экран
PSET (X, Y), 0         ' стирание точки цветом фона
IF X=0 OR X=640 THEN DX=-DX   ' в случае достижения точкой
       ' вертикальных сторон экрана, приращение по X меняется
       ' на противоположное, а, значит, точка начнет двигаться
       ' в обратную по X сторону
IF Y=0 OR Y=480 THEN DY=-DY   ' в случае достижения точкой
             ' горизонтальных сторон экрана, приращение по Y
             ' меняется на противоположное, а, значит, точка
             ' начнет двигаться в обратную по Y сторону
X=X+DX: Y=Y+DY      ' изменение координаты точки
GOTO 1              ' переход на метку 1, где происходит
                    ' рисование точки
```

Не забывайте про паузу!

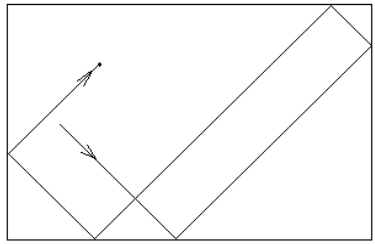


Рис. 1.32. Движение точки по экрану под углом 45°

152. Замените в предыдущей программе точку на закрашенную окружность. Получится подобие биллиардного стола. А еще можно сопровождать удар шарика о границу экрана звуковым сигналом `BEEP`, который нужно добавить после изменения приращения на противоположное.

153. Усложним задачу. На экране появляется прямоугольник, от которого точка тоже должна отражаться (рис. 1.33).

Допустим, координаты углов прямоугольника следующие:

- ❒ левый верхний 100, 180;
- ❒ правый верхний 300, 180;
- ❒ левый нижний 100, 230;
- ❒ правый нижний 300, 230.

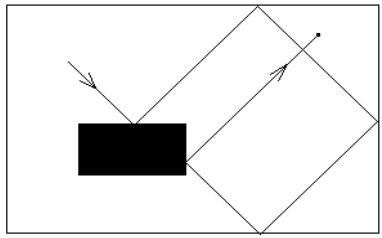


Рис. 1.33. Движение точки по экрану с препятствием

Тогда, по сравнению с предыдущим примером, к условиям отражения от сторон экрана добавятся еще четыре условия отражения от сторон прямоугольника. Мы приведем два — для верхней и левой сторон, а вы напишите еще два — для нижней и правой.

Для верхней:

```
IF Y=180 AND 100<X<300 THEN DY=-DY
```

То есть если точка достигла горизонтали 180, а по X попала в интервал]100; 300[, то изменяем направление движения по Y на противоположное.

Для левой:

```
IF X=100 AND Y > 180 AND Y<230 THEN DY=-DY
```

Рассуждения аналогичны.

154. Теперь вы готовы к более объемному и сложному заданию. Называться оно будет "Муха в графине". Сначала на экране вы рисуете прямоугольный графин с горлышком (рис. 1.34).

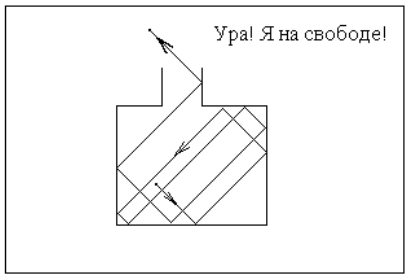


Рис. 1.34. Муха в графине

Затем располагаете там точку и заставляете ее двигаться внутри графина, отражаясь от его стенок (будет интересней, если вы не станете ее стирать, тогда мы сможем видеть траекторию ее полета). Меняя начальное расположение точки, можно добиться, что через какое-то время она вылетит из графина. Пусть в этот момент на экране появится надпись "Ура! Я на свободе!", а муха продолжит полет, отражаясь от сторон экрана. Графин можно сделать закрашенным.

155. Если вы обратили внимание, "муха" при полете слегка пробивает стенки графина. Что нужно изменить в условиях, чтобы этого избежать? Сделайте из "мухи" шарик, и снова добейтесь того, чтобы он не портил стенки графина.

Давайте сделаем простейший графический редактор. У некоторых в детстве была, наверное, игрушка "Волшебный экран", где можно было крутить колесики и рисовать отрезками прямых линий. Почему бы то же самое не сделать на компьютере? Для этого нам понадобится использование оператора `INKEY$`, который ожидает нажатия любой символьной клавиши и запоминает ее значение. Например:

```
CLS
1:K$=INKEY$
? K$
GOTO 1
```

На экране будет воспроизводиться символ той клавиши, которую вы нажмете (попробуйте поставить после `PRINT K$` точку с запятой и посмотрите, что изменилось).

Теперь, совместно используя `INKEY$` и условный оператор, мы можем сделать полезную прикладную программу-развлечение для младших друзей. Приведем ее фрагмент.

```
SCREEN 12
X=300: Y=200: C =15    ' начальные координаты и цвет точки
                       ' рисования.
                        ' Вы можете задать им свои,
                        ' произвольные значения
1: PSET(X,Y), C
K$=INKEY$
```

```
IF K$="w' THEN Y=Y -1  ' эта строка заставляет программу
                       ' после нажатия клавиши "w"
                       ' уменьшить координату Y на 1,
                       ' т. е. осуществить передвижение точки
                       ' на одну экранную единичку вверх
Rem Здесь вы программируете другие клавиши для движения
Rem в другие стороны, а также и изменение цвета,
Rem и, возможно, стирание. Все в ваших руках!
GOTO 1      ' зацикливание программы на постоянный опрос
            ' клавиатуры
```

Теперь, когда программа готова, нарисуйте сами с ее помощью какую-нибудь осмысленную картинку — домик в деревне, или звездные войны, или что хотите.

Замечание

Хорошо бы предусмотреть рисование цветом фона для перемещения точки без следа. А чтобы потом определить, где находится точка, достаточно изменить ее цвет.

Оператор выбора

Для организации меню, т. е. выбора из нескольких возможностей, применяется *оператор выбора*, который позволяет, в зависимости от значения переменной, делать те или иные действия.

Конструкция оператора несколько тяжеловесна, но ничего страшного в ней нет.

Она выполняет один из нескольких блоков операторов в зависимости от значения выражения.

```
SELECT CASE тест_выражение
  CASE список_выражений1
    [блок_операторов_1]
  [CASE список_выражений2
    [блок_операторов_2]]
  ...
  [CASE ELSE
    [блок_операторов_n]]
END SELECT
```

Здесь *тест_выражение* — любое числовое или строковое выражение; *список_выражений1*, *список_*выражений*2* — одно или несколько выражений для сравнения с параметром *тест_выражение*. В выражении ключевое слово IS должно стоять перед любым знаком отношения. *блок_операторов_1*, *блок_операторов_2*, *блок_операторов_n* — один или несколько операторов в одной или нескольких строках.

Аргументы списка выражений могут принимать любую из следующих форм или их комбинацию и должны разделяться запятыми:

- *выражение*[,*выражение*]...
- *выражение* TO *выражение*
- IS *оператор_отношения выражение*

Здесь *выражение* — любое числовое или строковое выражение, совместимое с параметром *тест_выражение*; *знак_отношения* — один из знаков отношения <, <=, >, >=, <> или =.

Рассмотрим пример:

```
CLS
INPUT "Введите уровень риска (1-5): "; RISK
SELECT CASE RISK
  CASE IS >= 5
    PRINT "Максимальный риск, шансов на возвращение
           практически нет."
    PRINT "Пишите завещание."
  CASE 2 TO 4
    PRINT "Высокий риск, шансов на возвращение немного."
    PRINT "Проверьте снаряжение, оружие и припасы."
  CASE 1
    PRINT "Риск отсутствует, возврат гарантирован."
    PRINT "В Вас отсутствует дух авантюризма,
           это слишком скучно."
END SELECT
```

Еще пример. По заданному номеру дня недели требуется напечатать его название:

```
CLS
INPUT "Введите номер дня недели"; N
```

```
SELECT CASE N
  CASE 1
    ? "Понедельник"
  CASE 2
    ? "Вторник"
END SELECT
```

Ну что ж, теория без практики суха. Задания.

156. По введенному номеру месяца определить его название и время года и загрузить соответствующую картинку. Надо использовать два Case: один для определения названия месяца, другой — для времени года.

157. По введенной цифровой оценке, выставите ученику отметку (5 — "отлично", 4 — "хорошо" и т. д.).

158. Напишите программу, которая по знаку арифметической операции выводит ее название.

159. Напишите программу, которая определяет по заданному числу месяца и по дню недели первого числа этого месяца день недели для заданного числа. (Пример: первое число — вторник, тогда 17 — четверг, задали число 17.)

160. Напишите программу, которая выводит на экран меню, содержащее список трех поэтов под номерами, затем запрашивает у пользователя номер поэта и выводит на экран какое-нибудь его четверостишие, после чего возвращается в меню. Последним пунктом меню предусмотреть выход из программы.

161. В восточных календарях принят 60-летний цикл, состоящий, в свою очередь, из пяти 12-летних подциклов. Подциклы обозначались цветом: зеленый, красный, желтый, белый и черный. Внутри каждого подцикла годы носили названия животных: крысы, быка, тигра, кролика (зайца или кота), дракона, змеи, лошади, овцы (барана или козы), обезьяны, петуха, собаки и свиньи. Создайте программу с использованием операторов выбора, запрашивающую номер года нашей эры и печатающую его название по восточному календарю. Для проверки: 1966 г. — год красной лошади, 1984 г. — зеленой крысы.

162. Усложненный вариант предыдущего задания. Теперь требуется определить не только название введенного с клавиатуры года, но и год наступления "светлого будущего", до которого вы, безусловно, доживете! Делаем это поэтапно. Вот примерный словесный алгоритм. После введения исходного четырехзначного года с клавиатуры вам необходимо выделить составляющие его цифры и записать их в переменные A1, A2, A3 и A4. (Если в номере года есть нули, то в соответствующие переменные записать 4.) Затем необходимо взять от них синусы по модулю. Из полученных значений выделить по две цифры после запятой и сложить их. Если сумма больше либо равна 10, то сложить еще раз. Должно получиться 4 цифры, из которых надо составить четырехзначное число. Это и будет номер года наступления светлого будущего. Осталось определить его название по восточному календарю и вывести результаты на экран. Для проверки: при исходном годе 1986 годом светлого будущего будет 3589.

Циклический алгоритм

В тех случаях, когда нам необходимо выполнить одинаковые действия, в которых изменяется только какая-либо величина (так называемый *параметр*), то мы применяем операторы цикла. Это дает мощный толчок программированию, позволяя выполнять очень сложные алгоритмы.

Оператор с заранее известным числом повторений

Когда мы работали с графикой, то в одном из заданий вам предлагалось построить мишень — пять концентрических окружностей. Мы это легко сделали, пять раз написав оператор построения окружности `CIRCLE`, изменяя в нем каждый раз радиус.

Ну а если бы таких окружностей надо было бы 100? Неужели 100 раз писать практически одно и то же? Очень нелегко, даже если пользоваться копированием строк. Неужели нельзя как-нибудь

полегче? Конечно можно! Нам на помощь приходит оператор цикла `FOR...NEXT`.

Итак, правила пользования оператором цикла `FOR...NEXT`:

- ❒ Рассмотреть повторяющиеся действия и выделить в них равномерно изменяющуюся величину (параметр).
- ❒ Дать параметру *имя*.
- ❒ Определить для параметра *начальное значение*, *конечное значение* и *шаг изменения*, т. е. насколько за один раз увеличивается (или уменьшается) параметр.
- ❒ Написать оператор цикла, состоящий из трех частей:
  - Заголовок цикла

    `FOR` *параметр=нач_значение* `TO` *кон_значение* `STEP` *шаг*
  - Тело цикла

    В теле цикла указываются один или несколько операторов, предназначенных для повторяющихся действий, причем вместо конкретных значений изменяющейся величины указывают имя параметра.
  - `NEXT` *параметр*

Рассмотрим пример построения мишени. Действуем по правилам. При рисовании мишени изменяется радиус, это и будет параметр. Обозначим его буквой `R`. Пусть начальное значение `R=20`, конечное значение `R=60`, шаг изменения `?R=10`.

Записываем оператор цикла (предварительно, конечно, включив графический режим):

```
FOR R=20 TO 60 STEP 10
  CIRCLE (320, 175), R, 14
NEXT R
```

Если мы также четко будем и в дальнейшем следовать этим правилам, то нам не составит труда работать с оператором цикла.

Необходимо отметить, что можно записывать оператор цикла в одну строку, это не ошибка:

```
FOR R=20 TO 60 STEP 10: CIRCLE (320, 175), R, 14: NEXT R
```

Но если в теле цикла много операторов, то такая запись будет плохо читаемой.

Как же работает оператор цикла на примере приведенной выше программы? Встретив заголовок цикла FOR, программа присваивает параметру начальное значение и переходит в тело цикла. Там находит команду CIRCLE и исполняет ее, подставляя вместо параметра его начальное значение. Встретив команду NEXT программа добавляет к предыдущему значению параметра величину шага, затем проверяет условие, не стало ли значение параметра больше конечного его значения, и, если нет, продолжает выполнение цикла. Если да, переходит к выполнению следующей за оператором цикла команды или заканчивает выполнение программы, если таковая отсутствует.

Таким образом, последнее значение параметра в нашем примере будет 70, после чего цикл закончится. (Попробуйте к программе добавить строку `? "R="; R` и посмотрите, что появилось на экране.)

Теперь следует сказать несколько слов о составной части оператора цикла — шаге изменения параметра STEP. Довольно часто преподаватели провокационно спрашивают — а какие значения может принимать STEP? Может ли он быть отрицательным? Дробным? Вот ответы на эти вопросы. Шаг может быть любым вещественным числом, за исключением нуля, иными словами, и целым, и дробным, и отрицательным. Но отрицательным он может быть только в случае, если начальное значение параметра больше конечного, т. е. когда мы идем от большего к меньшему. Например, тот же оператор цикла из примера про мишень с тем же результатом можно было бы записать и так:

```
FOR R=60 TO 20 STEP -10: CIRCLE (320, 175), R, 14: NEXT R
```

Кроме того, если шаг изменения параметра равен единице, то при написании оператора цикла его можно опускать, например:

```
FOR W=12 TO 24 STEP 1
```

можно было бы записать просто

```
FOR W=12 TO 24
```

Но это не касается минус единицы!

Использование оператора цикла очень наглядно в графике (хотя это, как раз, не главное применение данного очень полезного инструмента). Поэтому давайте освоим правила управления оператором цикла, выполнив несколько упражнений.

163. Напишите программу, рисующую на экране горизонтальную линию, состоящую из точек, расстояние между которыми 8 (рис. 1.35).

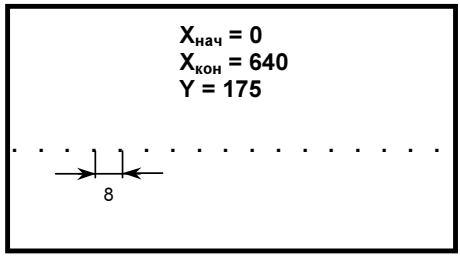


Рис. 1.35. Горизонтальная линия из точек

164. Заполните экран горизонтальными линиями (через 10), а затем, с помощью еще одного оператора цикла, вертикальными линиями другого цвета (тоже через 10). Должна получиться решетка, как на рис. 1.36.

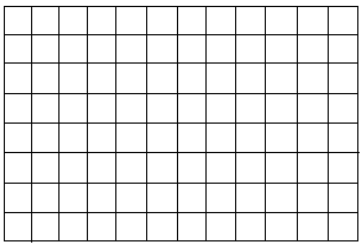


Рис. 1.36. Решетка

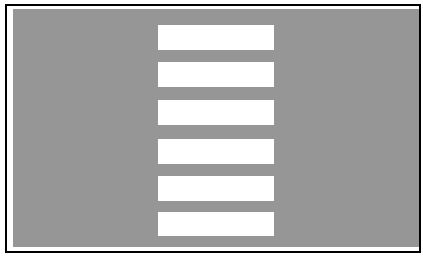


Рис. 1.37. Пешеходный переход "зебра"

165. Нарисуйте пятиконечную звезду, вписанную в окружность, как на рис. 1.25. Используйте для рисования и закраски операторы цикла.

166. Постройте пешеходный переход — "зебру", как на рис. 1.37.

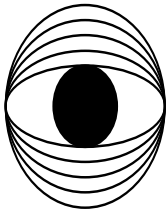


Рис. 1.38. Фантастический глаз

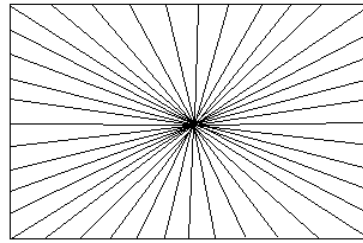


Рис. 1.39. Взрыв сверхновой

167. Нарисуйте фантастический глаз, как на рис. 1.38.

168. Изобразите взрыв сверхновой звезды (если это задание делать по-простому, то понадобится четыре оператора цикла, можно сложнее — но и короче). Сделайте лучи во всех четырех случаях разного цвета. Это придаст живости (рис. 1.39).

169. Заставьте окружность надуваться как воздушный шарик.

170. Мы уже перемещали по экрану геометрические фигуры с использованием условного оператора и оператора безусловного перехода. Теперь наши фигуры будут двигаться равномерно по вертикали или горизонтали. Сместите:

- закрашенный прямоугольник;
- закрашенную окружность.

Все это замечательно. Но давайте вспомним, что Бейсик — все-таки язык в большой степени не графический, а вычислительный, и попробуем применить оператор цикла к вычислениям.

Например, напечатать значения $y = \sin x$ в интервале [–30°; 30°] с шагом 5°. Действуя по правилам, несложно понять, что параметр — это X, так мы его и обозначим. Начальное, конечное значения параметра и шаг указаны непосредственно в самом задании. Пишем оператор цикла:

```
FOR X=-30 TO 30 STEP 5
  Y=SIN (X*3.14/180)
  ? "SIN("; X; ")="; Y
NEXT X
```

На что, кроме оператора цикла, здесь хочется обратить ваше внимание, так это на оператор `PRINT`. Разберитесь, пожалуйста, как он работает, и что дает такая, казалось бы, запутанная запись.

Еще пример. Необходимо вывести на экран четные числа от 2 до 20. Здесь меняется само число, обозначаем его N, начальное значение 2, конечное 20, шаг 2. Пишем:

```
FOR N=2 TO 20 STEP 2: ? N: NEXT N
```

А если от 20 до 2, то будет, соответственно:

```
FOR N=20 TO 2 STEP -2: ? N: NEXT N
```

Проверим ваше понимание, как всегда, при помощи ряда заданий.

171. Выведите на экран в строку все числа первой сотни, оканчивающиеся на пять.

172. Определите значение переменной `F` после выполнения следующих операторов:

```
F=1: N=1
FOR I=2 TO N: F=F+1/I: NEXT I
```

173. Напишите программу, запрашивающую возраст пользователя, а затем печатающую текст "Да ты крут!" по числу прожитых лет. Обратите внимание, что здесь в теле цикла не будет использоваться параметр. Такое тоже возможно.

174. С клавиатуры запрашивается любая цифра от 2 до 9, а затем компьютер печатает таблицу умножения на эту цифру.

175. Напишите программу, выводящую на экран степени числа 2 от 2 до 10 включительно.

176. Распечатайте в табличном виде (с аргументами) значение функции квадратного корня на интервале [2; 4] с шагом 0,1.

177. Напишите программу, которая в центре чистого экрана на одном и том же месте выведет последовательно цифры от 1 до 3600 (аналог электронного секундомера).

Когда мы двигали геометрические объекты, то нам приходилось иногда искусственно замедлять движение, чтобы оно было более плавным. Тогда без объяснений был введен *пустой цикл* для ор-

ганизации пауз. Пустым будем называть такой цикл, в котором нет тела. Например:

```
FOR I=1 TO 100: NEXT I
```

Просто компьютер считает в уме до 100. А мы его ждем (ну, правда, если до 100 — это быстро, а вот до 10 000 уже придется и потерпеть).

Следующая группа упражнений поможет вам закрепить работу с оператором цикла в графике на примере орнаментов. Сначала рассмотрим пример.

Орнамент — это повторяющийся геометрический узор, например, как на рис. 1.40.

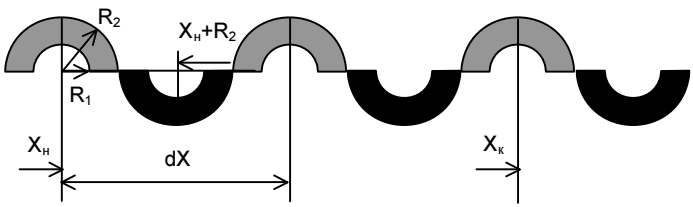


Рис. 1.40. Орнамент

Правила построения орнаментов:

- ❒ выявить в орнаменте повторяющийся фрагмент;
- ❒ выделить опорную изменяющуюся величину (параметр), ее начальное, конечное и шаг;
- ❒ дать параметру имя;
- ❒ определить смещение других изменяющихся величин относительно опорной;
- ❒ написать оператор цикла.

178. Напишите программу для рисования орнамента (рис. 1.40).

179. Закрепите успех, написав программу для построения греческого орнамента (рис. 1.41).

180. Когда мы изучали операторы `DATA` и `READ`, то мы говорили, что используем их при изучении оператора цикла. Этот час

пробил. Задав в операторе DATA координаты всех точек концов отрезков, из которых состоит рис. 1.42, напишите программу, рисующую Буратино при помощи оператора цикла.

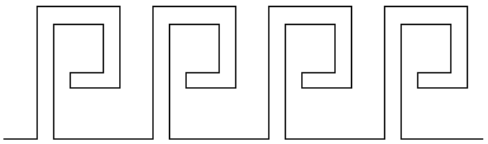


Рис. 1.41. Греческий орнамент

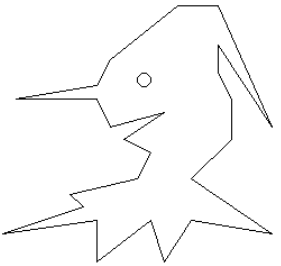


Рис. 1.42. Буратино

Совет

При рисовании непрерывной ломаной линии применяется сокращенная форма оператора LINE, а именно LINE –(X, Y), C. То есть линия будет рисоваться от последней графической точки на экране до точки с координатами X, Y. Поэтому при выполнении предыдущего задания рекомендуется поставить первую точку оператором PSET, а затем в цикле считывать поочередно координаты следующих точек и рисовать отрезки при помощи сокращенной формы оператора LINE.

Оператор цикла *WHILE...WEND*

Этот оператор позволяет сочетать свойства условного оператора и оператора цикла. С их помощью можно выполнять повторяющиеся действия *с заранее неизвестным количеством повторений*.

Работает он следующим образом. Сначала идет оператор WHILE с условием, при котором цикл выполняется. Ключевое слово WEND аналогично по своему назначению слову NEXT, т. е. является последней строкой цикла. Всякий раз, доходя до WEND, компьютер проверяет, выполняется ли условие, указанное оператором WHILE. Если оно не выполняется, то программа переходит к исполнению операторов, следующих за WEND. Если же выполняется, то цикл повторяется снова. Следующая программа, прежде чем допус-

тить пользователя до продолжения работы, запрашивает у него пароль.

```
WHILE WORD$<>"TERMINATOR"
  INPUT "ВВЕДИТЕ ПАРОЛЬ"; WORD$
WEND
? "ПРАВИЛЬНО! ВЫ ДОПУЩЕНЫ К ПРОДОЛЖЕНИЮ РАБОТЫ"
```

Вот как эта программа будет работать после запуска:

```
ВВЕДИТЕ ПАРОЛЬ? MOON
ВВЕДИТЕ ПАРОЛЬ? ARNOLD
ВВЕДИТЕ ПАРОЛЬ? TERMINATOR
ПРАВИЛЬНО! ВЫ ДОПУЩЕНЫ К ПРОДОЛЖЕНИЮ РАБОТЫ
```

181. При помощи оператора `WHILE...WEND` вычислите C — наибольший общий делитель введенных с клавиатуры натуральных чисел X и Y.

182. Определите, какие из представленных ниже операторов Бейсика написаны правильно, а какие нет. В случаях неправильной записи объясните допущенные ошибки. Все операторы рассматривайте по отдельности, потому что они не являются частями одной программы.

- `FOR D=R TO S STEP H`
- `IF W=13 THEN ? "ЧЕРТОВА ДЮЖИНА"`
- `IF W=13 THEN X=X+1`
- `FOR S=2—6`
- `FOR S=4 TO 12 STEP 2`
- `FOR S=12 TO 4 STEP 2`
- `WEND S=45`

Что будет выведено на экран в результате выполнения следующих программ при указанных исходных данных.

183. Программа:

```
INPUT W
FOR R=1 TO W
```

```
  ? "WAR"
  ?
NEXT R
IF R=W THEN GOTO 1
? "SUNDAY"
1: "PEACE"
```

при `W=3`.

184. Программа:

```
W=27.6
1: IF INT(W)=W THEN GOTO 2
? "ЧЕРТ ПОБЕРИ!!!"
W=W * 2 — 5.2
GOTO 1
2: ? "МИХАИЛ СВЕТЛОВ"
A=W -1
IF A <> 50 THEN 3
GOTO 1
3: END
```

185. Программа:

```
FOR H=5 TO 21 STEP 4
  ? H — 3
  ? H+3
NEXT H
```

186. Программа:

```
1: ? "НА СТАРТ! ВНИМАНИЕ!"
FOR V=5 TO 1 STEP -1
? V; "СЕКУНД"
GOTO 1
? "МАРШ!"
```

187. Программа:

```
DATA 10, 100, 1000
READ A, B
? B — A
READ C
```

```
RESTORE
READ X, Y, Z
IF C=Y THEN 1
IF B=Y THEN 2
GOTO 3
1: ? "ПЛЮС"
GOTO 3
2: ? "МИНУС"
3: END
```

188. Программа:

```
X=13
Y=17
Z=2
2: IF Z >=5 THEN 1
? X, Y, Z
X=X — 1
Y=X+Y
Z=Z+1
GOTO 2
1: END
```

189. Программа:

```
FOR R=1 TO 49 STEP 6
  W=W+R*2
NEXT R
? R
```

190. Программа:

```
FOR K=7 TO 28 STEP 7
  ? "УРА!!!"
NEXT K
? "ВПЕРЕД!!!"
```

191. Программа:

```
X=1
1: ? X
X=X+2
IF X <= 13 THEN 1
END
```

192. Программа:

```
FOR B=1 TO 4
  D=B*2
  ? D, B
NEXT B
```

193. Программа:

```
FOR C=2 TO 11 STEP 2
  ? C, C^2
NEXT C
? "ВЗРЫВ!"
```

194. Программа:

```
X=13
Y=52
Z=99
FOR U=100 TO 1 STEP -2
  IF U=X THEN ? U
  IF U=Y THEN ? U
  IF U=Z THEN ? U
NEXT U
```

195. Программа:

```
WHILE S < 10
  ? S
WEND
```

196. Программа:

```
WHILE D$ <> "ИНФОРМАТИКА"
  READ D$, B
  ? "ПРЕДМЕТ"; D$
  ? B
WEND
DATA МАТЕМАТИКА, 4, ФИЗИКА, 3, ИНФОРМАТИКА, 5, ХИМИЯ, 3
```

Случайные числа

В процессе изучения циклических алгоритмов уместно рассказать о генерации случайных чисел программным путем. При кажу-

щейся простоте получить случайные числа не так легко. Попробуйте продиктовать своему другу 100 цифр. В идеале каждая из них должна бы повториться по теории вероятности около 10 раз, но у каждого человека встречаются какие-то пристрастия к тем или иным цифрам и в ряду из сотни цифр не будет абсолютной случайности. Получить на компьютере случайные числа тоже было нелегко, но в конце концов это удалось. В качестве основы для создания какой-либо последовательности случайных чисел служат показания встроенного в компьютер таймера. А поскольку они в каждый момент времени различны, то мы и получаем великое множество практически неповторяющихся случайных чисел в заданном диапазоне.

Случайные числа нужны, прежде всего, в компьютерных играх, чтобы обеспечить непредсказуемость игры для человека. Кроме того, они используются в учебных задачах и моделировании различных математических, физических и других процессов на компьютере.

Итак, новый оператор `RND`.

Чтобы получать случайные числа, вы прежде всего должны инициировать процесс генерации последовательности случайных чисел компьютером при помощи оператора `RANDOMIZE TIMER`. Он указывается в программе только один раз и сообщает компьютеру, что надо создавать случайные числа, беря за основу показания таймера.

Затем, чтобы получить из этой последовательности какое-либо значение, мы должны использовать оператор `RND(N)`, который выдает случайное число в диапазоне от 0 до 1. Минимальное полученное число будет 0,0000001, а максимальное 0,9999999. В качестве `N` может служить любое целое или действительное число. Этот параметр также влияет на выбор компьютером случайных чисел. Например, программа

```
RANDOMIZE TIMER
X=RND(1)
? X
```

может вывести на экран число 0,367423, а может 0,0034289, а может 0,8912314. Каждый раз при запуске программы мы будем видеть новое значение.

Но часто встает проблема получения случайных чисел в заданном диапазоне. Как это сделать, мы рассмотрим на примерах.

Моделирование бросания монеты

Нам необходимо при каждом запуске программы случайным образом получать либо число 0 ("решка"), либо 1 ("орел"). Если мы будем просто брать целую часть от полученного при помощи `RND(N)` случайного числа, то всегда будем получать 0.

Попробуем умножить `RND(N)` на 2. В результате имеем числа от 0,0000002 до 1,9999998. Если мы теперь будем брать целую часть, то как раз и получим 0 или 1. Программа выглядит следующим образом:

```
CLS
RANDOMIZE TIMER
X=INT(RND(1)*2)
IF X=0 THEN ? "У вас выпала решка" ELSE ? "У вас выпал орел"
```

Моделирование бросания игрального кубика

Казалось бы надо пойти по уже проторенному пути первого примера и, раз нам надо получить цифры от 1 до 6, просто умножить `RND(N)` на 6. Но если разобраться, то обнаружится, что при этом мы будем получать числа в диапазоне от 0,0000006 до 5,9999994. Взятая целая часть даст нам числа от 0 до 5. На 6 никак не выходим. Хорошо, умножим на `RND(N)` 7. Тогда получим числа от 0,0000007 до 6,9999993. Теперь получается 6, но нам ведь совсем не нужен 0. Что же делать? Очень просто — умножаем на 6 и прибавляем 1!

```
CLS
RANDOMIZE TIMER
X=INT(RND(1)*6)+1
? X
```

Так как в дальнейшем нам придется часто сталкиваться с заданиями, требующими использование случайных чисел, то необходимо потренироваться.

Напишите операторы для получения случайных целых чисел в интервалах.

197. От 1 до 10.

198. От –5 до +5.

199. От 10 до 20.

200. От 50 до 100.

201. От –35 до 65.

Применим теперь оператор случайных чисел на практике и, как всегда, для наглядности начнем с графики.

Программа "Звездное небо"

Изобразим звездное небо пятьюстами белыми точками на черном фоне. Координаты каждой из точек должны быть в пределах по X от 0 до 640 и по Y от 0 до 480.

```
SCREEN 12
RANDOMIZE TIMER
FOR N =1 TO 500
  X=INT(RND(1)*641)
  Y=INT(RND(1)*481)
  PSET(X, Y), 15
NEXT N
```

Здесь число 500 в заголовке цикла указывает на количество "звезд". Можете поэкспериментировать, изменяя это значение.

А теперь мы хотим нарисовать разноцветные "звезды".

Добавим для этого перед `PSET` оператор случайных чисел для цвета

```
C=INT(RND(1)*16)
```

а в самом операторе `PSET` вместо цвета 15 укажем цвет `C`, и можем наслаждаться разноцветными "звездами".

Но давайте уж доведем дело до конца. Ведь при использовании оператора `C=INT(RND(1)*16)` у нас будет получаться и 0 — черный цвет, т. е. на черном фоне своеобразные "черные дыры". Давайте

избежим и этого, заставив компьютер при получении 0 не брать его в расчет, пока не получится другой цвет:

```
SCREEN 12
RANDOMIZE TIMER
FOR N =1 TO 500
  X=INT(RND(1)*641)
  Y=INT(RND(1)*481)
  1: C=INT(RND(1)*16)
  IF C=0 THEN 1
  PSET(X, Y), C
NEXT N
```

Со звездным небом разобрались. Теперь задания.

202. Когда-то на петербургском 5-м канале была скандально известная информационная программа "600 секунд" с телеведущим Александром Невзоровым. Начиналась она с заставки, где экран телевизора покрывался сеткой разноцветных отрезков прямых линий со случайными координатами концов (рис. 1.43). Попробуйте воспроизвести эту заставку и покажите ее своим старшим родственникам или друзьям. Это навеет на них воспоминания.

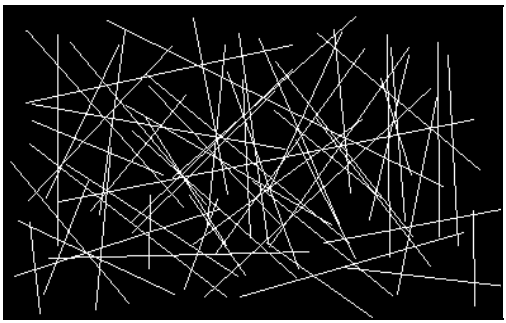


Рис. 1.43. "600 секунд"

203. Видоизмените (весьма, надо сказать, незначительно, легким движением руки) предыдущую программу, чтобы она заполняла экран сотней цветных прямоугольников случайных

размеров (рис. 1.44). Постарайтесь, чтобы на черном фоне не получались черные прямоугольники.

Рис. 1.44. Разноцветные прямоугольники

204. Теперь настал черед заполнить экран полусотней окружностей. Закраска пусть будет разноцветная, а контур окружностей только белый (рис. 1.45).

205. Давайте устроим в центре экрана маленький взрыв. Он будет изображен сотней отрезков разноцветных прямых, сходящихся в точке X=320, Y=175. Длина их пусть лежит случайным образом в пределах от 50 до 100 экранных точек (рис. 1.46).

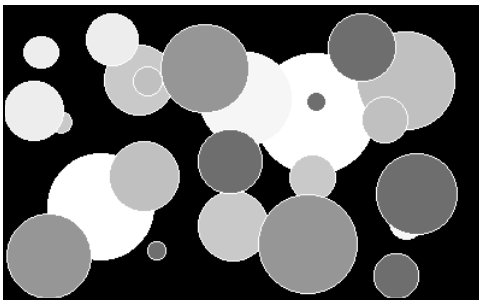


Рис. 1.45. Разноцветные окружности

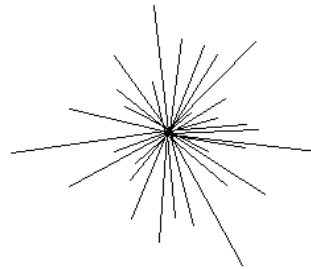


Рис. 1.46. Взрыв

206. Создайте программу, показывающую на форме игральную кость со случайным числом точек на верхней грани — от 1 до 6.

207. "Угадайка". Программа задумывает случайное число от 1 до 10, не выводя его на экран. Человек должен угадать его за три попытки. В каждой попытке компьютер выводит сообщение о том, больше его число или меньше. В случае отгадывания выводится поздравление, иначе сожаление и загаданное число. Все сообщения пользователю выводятся с обращением по имени, запрошенному в начале с клавиатуры.

208. Усложните предыдущее задание, предоставив человеку еще одну попытку, и, в случае угадывания, выведите на экран количество попыток.

209. Сделайте в предыдущем задании запрашиваемым максимальное число, которое может задумать компьютер (например, 100 или даже 1 000 000), и ограничьте количество попыток угадывания, чтобы компьютер тоже мог выиграть!

Построение графиков функций

Строить графики в Бейсике мы будем точками, а задавать компьютеру функцию для построения оператором `DEF FN`.

Кроме того, координатная сетка экрана компьютера, как мы должны помнить, является перевернутой по отношению к привычной геометрической. Отсюда и последуют преобразования по приведению координатной сетки к нормальному виду. Я не хочу утомлять читателя длинными математическими выкладками, а призываю поверить на слово. Если же в ком-то проснется любознательность, то разобраться во всем можно самому.

Итак, оператор `DEF FN` определяет для компьютера ту функцию, график которой мы хотим построить. Работает он следующим образом:

```
DEF FN имя_функции(параметр_функции)=функция
```

Например, определим функцию $y = \sin x$.

```
DEF FNY(X)=SIN(X)
```

Здесь `Y` — имя функции, `X` — ее параметр, а `SIN(X)` — непосредственно сама функция.

Определим функцию $y = \cos 2x + \mathrm{tg}\frac{x}{2}$.

```
DEF FNY(X)=COS(2*X)+TAN(X/2)
```

Предупреждение

В одной и той же программе нельзя использовать для разных функций одно и то же имя. Необходимо давать разные имена, например `Y(X)`, `Z(X)`, `R(X)` и т. д.

Теперь непосредственно о построении графиков на примере функции синуса.

Сначала нарисуем оси координат — две взаимно перпендикулярные линии, пересекающиеся в центре экрана.

```
SCREEN 12
LINE (0, 240)—(640, 240), 15          Rem ось абсцисс
LINE (320, 0)—(320, 480), 15          Rem ось ординат
```

Теперь определим функцию, как мы это уже делали.

```
DEF FNY(X)=SIN(X)
```

Далее самое интересное — загадочный цикл непосредственного построения графика.

```
FOR X=0 TO 640
  Y=240—30*FNY((X — 320)/30)
  PSET(X, Y), 15
NEXT X
```

После запуска программы должна получиться следующая картинка (рис. 1.47).

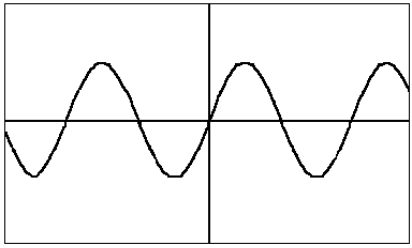


Рис. 1.47. График функции синуса

Значение X изменяется от 0 до 640, давая возможность строить график на всю ширину экрана. Достаточно сложно выражение для Y. Это связано с переносом начала координат в центр экрана (отсюда числа 240 и 320). А вот число 30 в данном случае несколько произвольно — это коэффициент растяжения. Попробуйте его поменять сначала на 10, а потом на 50 (только менять всегда надо в двух местах) и посмотрите на возникающие отличия. На мой взгляд, коэффициент 30 наиболее оптимален.

Построение иного графика сводится к замене определения функции на другую.

Давайте поупражняемся. Напишите программы построения графиков перечисленных ниже функций.

210. `Y=COS(X).`

211. `Y=ABS(X).`

212. `Y=INT(X).`

213. `Y=1/(1+X^2).`

214. `Y=X^2.`

215. `Y=2*SIN(X/2)+0.5*COS(2*X).`

216. `Y=TAN(X).`

Графики представляют собой семейства точек, между которыми могут быть значительные разрывы (как, например, при построении графика тангенса). Чтобы избежать этого, уплотнить график, можно в заголовке цикла установить шаг изменения по X меньше 1. Попробуйте.

Теперь попытаемся построить график функции `Y=1/X` (кажется, это называется гиперболой). Если вы будете рисовать его как описано выше, ничем не дополняя, то возникнет ошибка, о которой мы уже говорили — "Деление на ноль". Придется в очередной раз вспомнить математику и область допустимых значений. В данном случае, X не может быть равен 0, но поскольку начало координат у нас искусственно перенесено в центр экрана, то для графика X будет обращаться в 0 при координате экрана, равной 320. Поэтому программа будет выглядеть так:

```
SCREEN 12
LINE (0, 240)—(640, 240), 15        Rem ось абсцисс
LINE (320, 0)—(320, 480), 15        Rem ось ординат
DEF FNY(X)=1/X
FOR X=0 TO 640
  IF X=320 THEN X=X+1
  Y=240—30*FNY((X — 320)/30)
  PSET(X, Y), 15
NEXT X
```

А график должен выглядеть так (рис. 1.48).

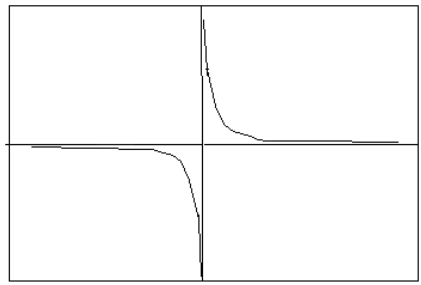


Рис. 1.48. График гиперболы

Постройте теперь графики следующих функций.

217. `Y=1/(X+1)`.

218. `Y=SQR(X)`.

219. `Y=X/(X^2-1)`.

Сейчас задания посложнее, но и интереснее. Изобразите на экране поверхность, образованную вращением вокруг оси X графиков представленных ниже функций.

220. `Y=1/(1+X^2)`.

221. `Y=1/X`.

222. Нарисуйте поверхность, образованную вращением вокруг оси Y, графика функции $y = x^2$.

Совет

Попробуйте строить график не точками, а эллипсами, радиус которых... Вот именно!

Циклы с несколькими зависимыми параметрами

Бывает, что при попытке выполнить задание с помощью оператора цикла выясняется, что изменяется не одна, а две или три величины. В том случае, если вы можете установить между ними зависимости, следует использовать следующие правила:

- определить, какие же величины изменяются, и обозначить их буквами;
- назначить одну из этих величин независимым аргументом;
- определить закономерности, связующие изменяющиеся величины, и выразить эти величины через независимый аргумент;
- написать программу с оператором цикла, основным параметром которого будет независимый аргумент, а в командах тела цикла, на месте других изменяющихся величин, необходимо подставить их выражения через этот независимый аргумент.

Приведем пример. Предположим, мы хотим построить семейство окружностей, как на рис. 1.49. Не правда ли, напоминает рупор?

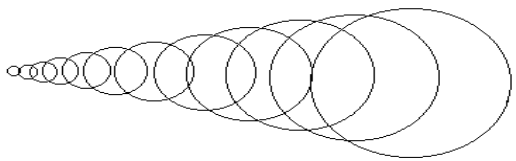


Рис. 1.49. "Рупор"

Действуем по правилам. Все окружности лежат на одной горизонтали, т. е. у их центров меняются не координаты Y, а координаты X. Кроме этого, они имеют разный радиус. Причем он изменяется не спонтанно, плавно увеличиваясь. Итак, обозначим изменяющиеся величины: X — координата центров окружностей по оси абсцисс, R — радиус окружностей.

Назначим независимым аргументом координату X. Попробуем установить закономерность изменения радиуса R от аргумента X. X изменяется от 0 до 640. Допустим, шаг изменения 10. Обычно устанавливают такое взаимоотношение X и R:

```
R=X
```

Но в этом случае получится, что радиус последней окружности будет равен 640, диаметр, соответственно, 1280. А ведь экран высотой всего 350! Значит, если мы хотим видеть на экране все наши окружности, то последний радиус не должен превышать 175, а лучше был бы меньше. Попробуем избрать такое соотношение:

```
R=X/4
```

Максимальное значение радиуса в таком случае равно 160.

Напишем программу:

```
SCREEN 12
FOR X=0 TO 640 STEP 10
  CIRCLE (X, 240), X/4, 15
NEXT X
```

Посмотрите, что получается на экране. Похоже? Не совсем, скажете вы и будете правы, потому что последние окружности выходят за пределы экрана справа. Но с этой проблемой я предлагаю разобраться вам самостоятельно.

Потренируемся.

223. Напишите программу, изображающую "орбиты", соприкасающиеся все в крайней правой точке и имеющие разные радиусы (рис. 1.50).

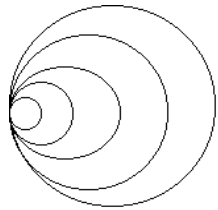


Рис. 1.50. "Орбиты"

224. Создайте программу, выводящую на экран "труба" — окружности одного радиуса, выходящие из левого верхнего угла экрана и идущие под углом 45° вправо вниз (рис. 1.51). Окружности нарисованы одним цветом, а закрашены другим.

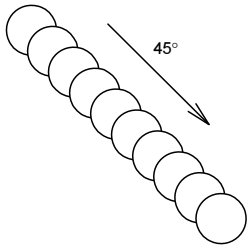


Рис. 1.51. "Труба" под углом 45°

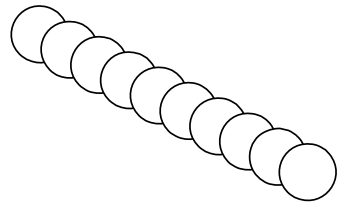


Рис. 1.52. "Труба" по диагонали

225. Напишите программу, аналогичную предыдущей, только окружности разместите из левого верхнего угла в правый нижний по диагонали экрана (рис. 1.52). Надо использовать соотношения сторон экрана.

226. С помощью Бейсика постройте две лестницы (рис. 1.53 и 1.54).

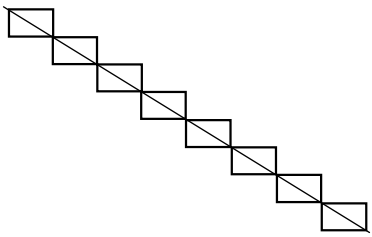


Рис. 1.53. Лестница 1

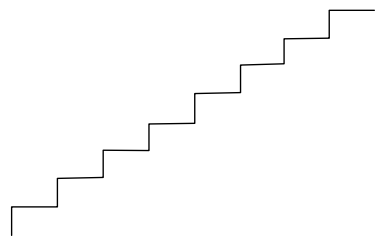


Рис. 1.54. Лестница 2

227. Нарисуйте две пирамиды индейцев майя — вид спереди и вид сверху (рис. 1.55 и 1.56).

228. Вернитесь к примеру с "рупором" из увеличивающихся окружностей и попробуйте сделать картинку, на которой подобные рупоры выходят из всех четырех углов экрана и встречаются в центре.

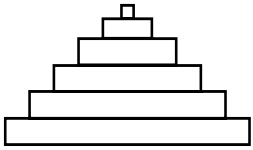


Рис. 1.55. Пирамида: вид спереди

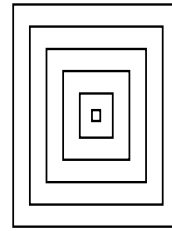


Рис. 1.56. Пирамида: вид сверху

229. Напишите программу построения "эллипсирующих окружностей" (рис. 1.57).

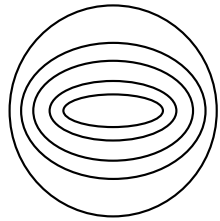


Рис. 1.57. "Эллипсирующие окружности"

Вложенные циклы

В предыдущем разделе мы рассмотрели случаи, когда в цикле изменяются несколько величин, но их можно выразить друг через друга. Сейчас нам надо рассмотреть ситуации, когда изменяются несколько *независимых* величин.

Приведем пример. Напишем программу заполнения экрана точками с расстоянием между ними по горизонтали и вертикали 10.

```
SCREEN 12
FOR X=0 TO 640 STEP 10
  FOR Y=0 TO 480 STEP 10
    PSET(X, Y), 4
  NEXT Y
NEXT X
```

Работает программа таким образом. В первом заголовке цикла переменной X присваивается значение 0. Потом программа переходит к выполнению следующего оператора, т. е. в нашем случае второго цикла, который и исполняется, пока Y не превысит значение 480. После этого берется следующее значение X, и действия с Y повторяются. И так до тех пор, пока X не станет больше 640. То есть программа строит вертикальные цепочки точек, пока не заполнится весь экран.

Все это не так сложно, если понять и запомнить следующее. Циклы должны быть вложены друг в друга, как матрешки. Параметры, упоминаемые в заголовках циклов, в их окончаниях должны быть перечислены в обратном порядке.

Правильное использование циклов:

```
FOR X=0 TO 640 STEP 10
  FOR Y=0 TO 480 STEP 10
    Rem Тело цикла
  NEXT Y
NEXT X
```

Неправильное применение циклов:

```
FOR X=0 TO 640 STEP 10
  FOR Y=0 TO 480 STEP 10
    Rem Тело цикла
  NEXT X
NEXT Y
```

Кроме того, в окончании вложенных циклов можно указывать только один оператор `NEXT` с перечислением параметров, например:

```
NEXT Y, X
```

Приступим к упражнениям.

230. Напишите программу, заполняющую экран лоскутным ковром. Лоскутки должны быть плотно прилегающими друг к другу разноцветными квадратиками со стороной 10 (рис. 1.58).

231. Заполните экран светофорными кругами (рис. 1.59). Самые большие круги красные, средние — желтые, маленькие — зеленые.

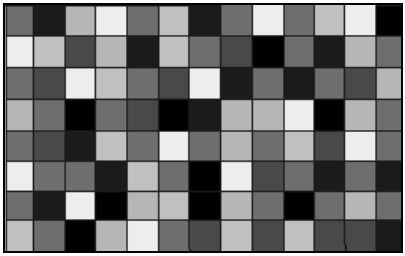


Рис. 1.58. Лоскутный ковер

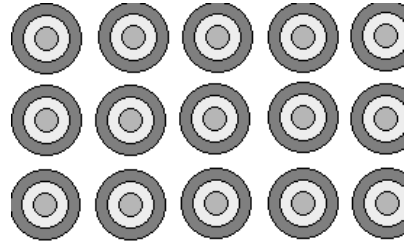


Рис. 1.59. Светофорные круги

232. Воспользовавшись предыдущим заданием, замените окружности на квадраты — получатся "светофорные" квадраты.

233. Изобразите шахматную доску.

Мы все работаем с графикой. Это наглядно. Но не будем забывать о том слове, от которого произошел термин "компьютер" — вычислять.

234. Напишите программу, выводящую на экран таблицу умножения от 2 до 10 в следующем виде (рис. 1.60).

	2	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
2	4	6	8	10	12	14	16	18	20
3	6	9	12	15	18	21	24	27	30
4	8	12	16	20	24	28	32	36	40
5	10	15	20	25	30	35	40	45	50
6	12	18	24	30	36	42	48	54	60
7	14	21	28	35	42	49	56	63	70
8	16	24	32	40	48	56	64	72	80
9	18	27	36	45	54	63	72	81	90
10	20	30	40	50	60	70	80	90	100

Рис. 1.60. Таблица умножения

Прежде чем выдать вам следующее задание, мне бы хотелось ознакомить вас с весьма полезным оператором `SLEEP`. Он приостанавливает выполнение программы на указанное количество секунд. Например:

```
? "Поспите 10 сек."
SLEEP 10
? "Подъем! Учитель пришел!"
```

Если значение секунд указано 0 или не указано вовсе, то программа будет ожидать нажатия любой клавиши, например:

```
? "Прочитайте внимательно нижеследующее слово и нажмите
   любую клавишу, когда надоест..."
? "Вы — просто гений!"
SLEEP
? "Но от скромности вы не умрете..."
```

235. Итак, используя полученные знания про вложенные циклы и оператор `SLEEP`, напишите программу, которая в центре чистого экрана будет выводить показания хронометра — часы, минуты и секунды, разделенные двоеточием, начиная с 0 час. 0 мин. 0 с. Отрегулируйте свой хронометр, чтобы он шел правильно.

236. Усложним предыдущее задание. Нарисуем будильник, сделаем в нем два поля. В одно выведем текущую дату, а во второе — наш уже получившийся хронометр. Если очень захочется, то можно сделать из него будильник, добавив в нужный момент оператор `BEEP`, который производит отрывистый, не совсем приятный звук (рис. 1.61).

Рис. 1.61. Электронный будильник

А теперь два интегрированных задания, где надо применить знания, приобретенные вами ранее.

237. Напишите программу "Уничтожение астероида". Экран заполняется пятьюстами звездами. Астероид изображается красным кругом белым контуром, центром в точке 470, 160 и радиусом 10. По траектории функции $y = 0{,}5\sin 2x + 2\cos\frac{x}{2}$

от правой стороны экрана движется боевой звездолет, изображаемый зеленым кругом с желтым контуром и радиусом 3. Каждый момент продвижения сопровождается проверкой условия, не находится ли звездолет в опасной близости от астероида (опасным считается расстояние менее 50 экранных точек). Если это расстояние в некоторый момент времени стало меньше критического, то звездолет уничтожает астероид, т. е. он взрывается (см. задание 179). На экране появляется торжествующая надпись. Все радуются (в том числе и учитель).

Совет

Расстояние между двумя точками на плоскости вычисляется по формуле: корень квадратный из суммы квадратов разностей координат X1, X2 и Y1, Y2. Я специально не пишу эту формулу. Проверяем понимание сложных инструкций.

238. Это задание носит прикладной характер и позволяет опытным путем вычислить число π. Да, безусловно, практически все из вас знают это число с точностью по крайней мере до двух знаков. Но предлагаемый метод очень хорош. Называется он методом Монте-Карло. Монте-Карло — европейская столица игорного бизнеса, а значит, там владычествует Его Величество Случай. Вот мы и попробуем поставить его себе на службу.

Сначала забудьте, чему равно π, и послушайте теорию вопроса. Представьте себе окружность радиусом $R=1$, вписанную в квадрат. Из этого следует, что сторона квадрата будет $2R$, а его площадь $SK=(2R)^2=4R^2$. Площадь круга $SO=\pi R^2$. Далее берем и равномерно посыпаем квадрат песком. Затем нанимаем бригаду рабочих, которые считают, сколько песчинок всего (N1) и сколько из них попало в круг (N2). Потом составляется простая пропорция — площадь квадрата так относится к площади круга, как общее количество песчинок к количеству песчинок попавших в круг.

$$\frac{SK}{SO}=\frac{N1}{N2}\Rightarrow\frac{4\times R^2}{\pi\times R^2}=\frac{N1}{N2}\Rightarrow\pi=\frac{4\times N2}{N1}$$

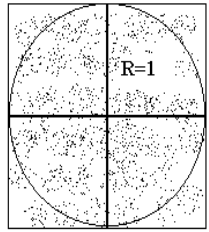


Рис. 1.62. Вычисление числа π методом Монте-Карло

Отсюда сенсационный вывод — радиус окружности не имеет никакого значения, она должна быть лишь вписана в квадрат (рис. 1.62).

Но где ж мы найдем песок, а главное тех, кто все это будет считать? Поэтому поставим компьютерный эксперимент. Нарисуем квадрат и впишем в него окружность. Координаты опорных точек (если сами рисовали) знаем. Уравнение окружности $X^2 + Y^2 = R^2$ тоже знаем. Задаем цикл до 1000, в котором случайным образом определяем координаты "песчинок" так, чтобы они лежали внутри квадрата. Тут же проверяем условие, а не попала ли "песчинка" в круг (используя уравнение окружности), и если попала, подсчитываем их количество. Кроме того, рисуем их на экране разными цветами (попавшие и не попавшие). По окончании цикла подсчитываем и выводим на экран число π. Понятно, что чем больше количество "песчинок", тем более точным будет результат. Для того чтобы знать, когда закончится эксперимент, рекомендуется выводить на экран счетчик "песчинок" (как мы делали с хронометром). Но все-таки экспериментировать с миллионом "песчинок" не надо — замучаетесь ждать сами, да и компьютер, хотя и железный, но все же "живой".

Наращивание переменной

О том, что такое переменная и об операторе присваивания мы уже говорили. И обращали внимание на такое абсурдное с точки зрения математики выражение:

```
X=X+1
```

Применяется оно обычно в Бейсике для работы всевозможных счетчиков, а также для подсчета сумм или произведений рядов чисел. Например, необходимо вычислить сумму всех четных чисел от 1 до 100 включительно. Программа будет выглядеть так:

```
CLS
S=0    'Обнуление переменной, где будет накапливаться сумма
FOR I=2 TO 100 STEP 2
  S=S+I
NEXT I
? "Сумма четных чисел от 1 до 100 равна"; S
```

Программа работает эффективно и просто. Параметром цикла являются сами четные числа, которые нам остается накапливать в переменной `S`. Разберем алгоритм по шагам:

- `S=0`;
- `I=2`;
- `S=0+2=2`;
- `I=4`;
- `S=2+4=6`;
- `I=6`;
- `S=6+6` и т. д. до 100.

На экране в результате увидим надпись:

```
Сумма четных чисел от 1 до 100 равна 2550.
```

Объяснять здесь вроде больше нечего. Иногда лучше решать, чем говорить. Итак, вам необходимо написать программы для нахождения сумм.

239. Всех чисел, делящихся на 13 в интервале [1; 1000].

240. $1^2 + 2^2 + 3^2 + ... + 10^2$.

241. $\frac{1}{\sqrt{1}} + \frac{2}{\sqrt{2}} + \frac{3}{\sqrt{3}} + ... + \frac{10}{\sqrt{10}}$.

242. Напишите программу для нахождения суммы пяти произвольных чисел, вводимых с клавиатуры (сумматор).

243. Создайте программу для нахождения суммы пяти заданных чисел, указанных в операторе `DATA`.

244. Напишите программу вычисления произведения

$$\cos 5^0 \times \cos 10^0 \times \cos 15^0 \times ... \times \cos 85^0 .$$

245. Напишите программу вычисления суммы

$$\frac{1}{1+a^2} + \frac{1}{1+(a+1)^2} + ... + \frac{1}{1+b^2} .$$

246. Напишите программу, рисующую цепочку соприкасающихся окружностей с заданными в операторе `DATA` радиусами (рис. 1.63).

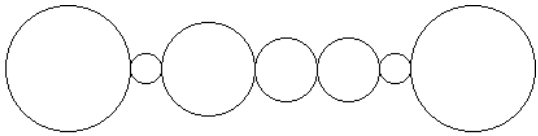


Рис. 1.63. Цепочка соприкасающихся окружностей

247. Создайте программу вычисления среднего роста десяти человек, данные о которых занесены в оператор `DATA`.

248. Напишите программу вычисления среднего балла при поступлении в институт по результатам четырех экзаменов, которые вводятся с клавиатуры.

249. С помощью Бейсика вычислите среднее геометрическое трех произвольных чисел. Среднее геометрическое есть корень степени N из произведения N чисел.

Оператор *DO...LOOP*

Оператор `DO...LOOP` чем-то похож на оператор `WHILE...WEND`, но возможности его применения несколько шире — проверяемое условие может быть использовано не только в начале оператора, но и в конце.

Поэтому возможны четыре различные формы написания оператора:

- первая форма

```
DO UNTIL условие
    тело_цикла
LOOP
```

- вторая форма

```
DO WHILE условие
    тело_цикла
LOOP
```

- третья форма

```
DO
    тело_цикла
LOOP UNTIL условие
```

- четвертая форма

```
DO
    тело_цикла
LOOP WHILE условие
```

Если используется ключевое слово `UNTIL`, цикл будет выполняться, только когда условие имеет значение "ложь".

При указании ключевого слова `WHILE` цикл будет выполняться до тех пор, пока условие будет иметь значение "истина".

Размещение условий в начале или конце цикла приводит к разным результатам: при проверке условия в конце оператора цикла, конструкции, составляющие его тело, будут исполняться хотя бы один раз, в то время как при проверке условия в начале возможен вариант, что цикл исполняться вообще не будет. Например:

```
K=0
? "Значение K в начале цикла равно"; K
DO WHILE K < 10
  K=K+1
LOOP
? "Значение K в конце цикла равно"; K
```

Кстати, чему же будет равно значение `K` в конце цикла?

250. Напишите программы — одну с оператором `DO … LOOP WHILE`, а другую с `DO … LOOP UNTIL`, — которые бы выводили на экран расположенные по диагонали экрана круги, закрашенные цветами с 1 по 6 (рис. 1.64). Параметр цикла — номер цвета.

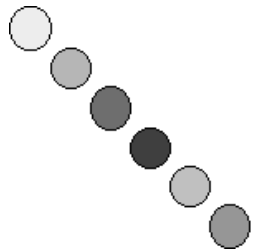


Рис. 1.64. Разноцветные круги

Разные задания на операторы цикла:

251. Вычислить значение n!.

252. Определить, существуют ли такие четыре последовательных натуральных числа, сумма квадратов которых равна сумме квадратов трех следующих натуральных чисел.

253. Написать программу вычисления значения S при вводимых с клавиатуры x и n:

S=x+2x2+3x3+4x4+...+nxn

254. Вычислить наибольший общий делитель натуральных чисел A и B.

255. Вычислить $P=(1-\frac{1}{2})(1-\frac{1}{3})...(1-\frac{1}{n})$, n>2.

256. Подсчитать k — количество цифр в десятичной записи целого неотрицательного числа N.

257. Логической переменной `T` присвоить значение `TRUE` или `FALSE` в зависимости от того, является натуральное число k степенью 3 или нет.

258. Вычислить S=1! + 2! + 3! +...+ n! (n > 1).

259. Числа Фибоначчи f n определяются формулами f 0 = f1 = 1; f n = fn – 1 + fn – 2 при n = 2, 3,... Написать программу, которая по введенному натуральному числу n находит n-е число Фибоначчи.

260. Найти первое число Фибоначчи, большее *m* (*m*>1).

261. Вычислить s — сумму всех чисел Фибоначчи, которые не превосходят 1001.

262. Логической переменной `K` присвоить значение `TRUE`, если целое `N` (`N`>1) — простое число, и значение `FALSE` в противном случае.

263. Дано целое n>2. Напечатать все простые числа из диапазона [2,n].

264. Найти сумму цифр заданного натурального числа.

265. Определить число, получаемое выписыванием в обратном порядке цифр заданного натурального числа.

266. Определить, является ли заданное натуральное число палиндромом, т. е. таким, десятичная запись которого читается одинаково слева направо и справа налево.

267. В каких двузначных числах удвоенная сумма цифр равна их произведению? (Для проверки: 36, 44, 63).

268. Найти двузначное число, равное утроенному произведению его цифр. (Для проверки: 15, 24).

269. Найти двузначное число, обладающее тем свойством, что куб суммы его цифр равен квадрату самого числа. (Для проверки: 27).

270. Найти все трехзначные числа, представимые в виде сумм факториалов своих цифр. (Для проверки: 145).

271. Найти все двузначные числа, сумма квадратов цифр которых делится на 17. (Для проверки: 14, 28, 29, 35, 41, 53, 67, 76, 82, 92).

272. Найти все трехзначные числа, которые можно представить разностью между квадратом числа, образованного первыми двумя цифрами, и квадратом третьей цифры. (Для проверки: 100, 147).

273. Найти все трехзначные числа, средняя цифра которых равна сумме двух крайних.

274. Найти все трехзначные числа, сумма цифр которых равна данному целому числу.

275. Вывести все трехзначные числа, в десятичной записи которых нет одинаковых цифр (операции деления не использовать!).

276. Найти все делители числа 1234.

277. Найти все двузначные числа, сумма цифр которых не меняется при умножении числа на 2, 3, 4, 5, 6, 7, 8, 9.

278. Даны целое число a и натуральное число n. Вычислить P = a(a + 1)...(a + n – 1).

279. Найти первую степень числа 3, превышающую данное целое число K.

280. Проверить, содержит ли квадрат данного натурального числа n цифру 3 в своей записи.

281. Привести дробь вида a/b (b не равно 0) к несократимому виду.

282. Найти среднее арифметическое последовательности целых чисел произвольной длины.

283. Сколько существует натуральных чисел n меньших 1000, для которых $2^n - n$ делится на 7.

284. Дана последовательность из N целых чисел. Напишите программу, которая определяет, какое число встретится раньше, положительное или отрицательное.

285. Дано натуральное число n. Вычислить произведение первых n сомножителей и распечатать:

$$P = \frac{1}{2} \cdot \frac{3}{4} \cdot \frac{5}{6} \cdots$$

286. Вычислить сумму и распечатать для данного натурального n:

$$S = \sum_{1}^{n} \frac{(-1)^{i+1}}{i(i+1)}.$$

287. Вычислить сумму и распечатать для данного натурального n:

$$S = \sum_{1}^{n} \frac{(-1)^{i+1}}{i!i^2}.$$

288. Вычислить для данного натурального n:

$$S = \sum_{1}^{n} \frac{(-1)^{i+1}}{i(i+1)(i+2)}.$$

289. Определить, какова дальность видения с самолета, набирающего высоту над океаном, в моменты, когда эта высота составляет 1, 2, 3, ... N км. Считать поверхность океана идеальной сферой с радиусом R=6350 км, а атмосферу — идеально прозрачной.

290. Последовательно вычислить объемы N цилиндрических нефтехранилищ, внутренние диаметры которых имеют значения D, D+1, D+2, ..., D+N–1, а высоты равны диаметрам.

291. Вычислить величину угла между минутной и часовой стрелками, если время суток равно 12 часов и 5, 10, 15, ..., 55 мин., а циферблат обычный, т. е. содержит 12 делений.

292. Вычислить площади N эллипсов, у которых большие оси одинаковы и равны S, а длины меньших осей образуют последовательность S/(C–1), S/(C–2), ... , S/(C–N), где C = N+2. Площадь эллипса равна произведению длин полуосей, умноженному на число π.

293. Найти суммарную площадь N колец, внутренние радиусы которых одинаковы, а значения внешних радиусов образуют последовательность: 5, 10, 15, ... , 5*N.

294. Игрушка состоит из N матрешек, вложенных в самую большую матрешку, причем объем самой маленькой известен и равен V. Объемы остальных образуют последовательность: (1,2)3V, (1,4)3V, (1,603V, ..., (1+2N/10)3V. Определить, во сколько раз увеличится занимаемый матрешками объем, после того, как все они будут извлечены друг из друга.

Символы и строки

Когда мы говорили о дружественном интерфейсе, то упоминали о так называемых строковых переменных, имена которых отличаются от обычных добавлением знака $. В таких переменных могут содержаться как отдельные символы, так и их последовательности длиной до 255 символов. К ним в Бейсике применимы специальные операции, о которых мы здесь и расскажем.

Для начала надо немного отвлечься, чтобы сообщить о том, что каждый символ, представленный на клавиатуре для компьютера, переводится в числовой код. Эти коды объединены в стандартную международную таблицу кодов ASCII *(см. Приложение)*. Коды с 0 по 32 не имеют изображения на экране и служат для функций управления (пробел, клавиши управления курсором и т. д.). Далее следуют знаки препинания, цифры, строчные и прописные буквы латинского алфавита и другие символы, которые вы можете найти на клавиатуре. Всего их 128. А еще 128 кодов (от 129 до 255) служат для расширения возможностей клавиатуры, например для генерации национальных символов — в нашем случае для кириллицы. Учить их наизусть ни в коем случае не надо — они есть в таблице. Но если ее не окажется под рукой, то вы должны определить код любого символа, используя специальные функции ASC и CHR$.

Функции *ASC* и *CHR$*

Функция ASC определит нам код ASCII для первого символа этой строковой переменной и имеет следующую форму записи:

```
ASC(строковая_переменная)
```

Например:

```
N=ASC("F")
? " Код заглавной буквы F —"; N
```

В результате получим:

```
Код заглавной буквы F — 70
```

Еще пример.

```
X$="YAHOO"
N=ASC(X$)
? N
```

В результате получим код первого символа, входящего в слово "YAHOO", т. е. "Y", который равен 89.

Замечание

Следует помнить, что коды заглавных и строчных букв — разные.

Кроме того, если мы напрямую указываем в функции ASC символ или текст, то он берется в кавычки (первый пример), а если это строковая переменная, то без кавычек (второй пример).

Функция CHR$ определит нам символ, код которого указан в скобках. Форма записи функции:

```
CHR$(код)
```

Например:

```
CLS
1: INPUT "Введите любой код от 33 до 128"; N
IF N < 33 OR N >=128 THEN ? "Обратите внимание на числовые
границы для кода": GOTO 1
? "Символ с кодом "; N; "- это"; CHR$(N)
```

Обратите внимание на оформление программы. Сначала выполняется очистка экрана. Затем — запрос кода. Если он введен не в требуемых пределах, то программа возвращает человека к запросу — простейший, но очень полезный способ помочь пользователю.

295. Опробуйте представленную выше программу и узнайте, что за символы скрываются под кодами 33, 66, 99, 100, 128.

296. Примените функции ASC и CHR$ к примеру простейшей шифровки информации, когда символы вводятся побуквенно, а программа определяет их код, добавляет к ним 1 и выводит на экран вместо введенного символа символ с новым получившимся кодом. Слово для тестовой проверки такой программы — "CAT", после его побуквенного введения должно получиться "DBU".

297. Напишите программу-дешифратор для предыдущего задания. Тестовая проверка: из слова "DBU" должно получиться слово "CAT".

Эти программы грамотно работают для первых стандартизированных 128 кодов. Чтобы правильно работать, например, с русским текстом, надо знать коды строчных и прописных букв кириллицы, которые скрываются в интервале от 129 до 255. Поэтому еще одно задание.

298. Напишите программу, выводящую на экран символы, скрывающиеся за кодами 129—255. Распечатайте или выпишите коды строчных и прописных букв кириллицы.

Но всякий раз вводить текст побуквенно — большая морока. Нельзя ли как-нибудь в Бейсике обрабатывать слова и строки? Конечно, можно. Для этого существуют специальные функции.

Функция *INPUT$*

Эта функция ожидает ввода N символов, которые и будут обрабатываться программой сразу после их набора на клавиатуре, не дожидаясь нажатия клавиши <Enter>. В этом заключается принципиальное отличие функции от оператора `INPUT`.

Например:

```
CLS
N$=INPUT$(2)
IF N$="да" THEN ?N$ ELSE ?"нет"
```

В данном случае программа после ввода с клавиатуры последовательности символов анализирует ее, выделяя только два первых символа, и действует в зависимости от проверяемого условия.

Функция *LEN*

Следующая функция — `LEN`. Она определяет длину введенной или существующей в переменной строковой переменной в символах. Синтаксис:

`LEN(`*`строковая_переменная`*`)`

Например,

```
CLS
INPUT "Введите Вашу фамилию"; F$
N=LEN(F$)
? "В вашей фамилии "; N; "букв"
```

Представленная программа выясняет количество букв во введенной пользователем фамилии. Причем обратите внимание, что функция LEN учитывает не только буквы, но и символы, т. е. она распознает и пробелы, и знаки препинания, и цифры, содержащиеся во введенном тексте. Например:

```
CLS
INPUT "Введите Ваш адрес"; F$
N=LEN(F$)
? "В вашем адресе "; N; "символов"
```

299. Определите с помощью предыдущего примера, сколько символов будет в следующем адресе:

197110, Россия, Санкт-Петербург, Чкаловский пр., 78-33

Функции *LEFT$, RIGHT$* и *MID$*

Для получения фрагмента строки (или значения строковой переменной) применяются специальные функции.

Функция LEFT$ выделяет из введенной строковой переменной N символов *слева*:

```
LEFT$(строковая_переменная, N)
```

Рассмотрим пример.

```
CLS
F$="ГАЗОНОКОСИЛЬЩИК"
L$=LEFT$(F$, 5)
? L$
```

На экране появится слово "ГАЗОН", т. е. первые пять символов слева исходной строковой переменной.

Функция RIGHT$ вырезает из введенной строковой переменной N символов *справа*:

```
RIGHT$(строковая_переменная, N)
```

Например:

```
CLS
F$="ГАЗОНОКОСИЛЬЩИК"
R$=RIGHT$(F$, 9)
? R$
```

На экране появится слово "КОСИЛЬЩИК", т. е. первые девять символов справа исходной строковой переменной.

Наконец, функция MID$ извлекает N2 символов, начиная с N1 исходной строковой переменной:

```
MID$(строковая_переменная, N1, N2)
```

Например,

```
CLS
F$="ГАЗОНОКОСИЛЬЩИК"
M$=MID$(F$, 7, 4)
? M$
```

На экране появится слово "КОСИ", т. е. четыре символа, начиная с седьмого исходной строковой переменной.

Используя эти функции для начала можно с их помощью поиграть в игру "Наборщик", когда из букв, составляющих какое-либо слово, нужно составить другие слова. Для этого предназначена функция конкатенации или, по-простому, слияния, которая записывается просто знаком +. Например:

```
CLS
F$="ГАЗОНОКОСИЛЬЩИК"
W1$=MID$(F$, 4, 2)+RIGHT$(F$, 7)
W2$=MID$(F$, 4, 2)+LEFT$(F$, 2)
W3$=MID$(F$, 9, 1)+MID$(F$, 7, 2)+MID$(F$, 11, 2)+MID$(F$,7,2)
? W1$
? W2$
? W3$
```

Выполните упражнения.

300. Определите, какие слова получатся в результате выполнения приведенной выше программы?

301. Напишите программу, которая выдаст на экран пять слов максимальной длины из слова "ЭЛЕКТРИЧЕСТВО". Побеждает тот, у кого сумма букв во всех словах наибольшая.

В качестве очередного примера приведем задачу подсчета слов во введенном тексте. Как известно, для компьютера словом является последовательность символов, заключенная в пробелы с двух сторон. Подчеркиваю, это не обязательно слово, в привычном для нас понимании, а любой набор символов, например, 45ро9) или ВАР56+УР47. Поэтому, в простейшем случае, подсчет количества слов во введенном тексте сводится к подсчету количества пробелов и добавлении к полученному значению единицы. (Почему так? Очень просто: слов два, а пробел между ними один; слов три — а пробелов два и т. д.) Получаем программу.

```
CLS
INPUT "Введите текст телеграммы"; W$
N=LEN(W$) : K=0
FOR I=1 TO N
  P$=MID$(W$, I, 1)
  IF P$="  " THEN K=K+1
NEXT I
? "В вашей телеграмме — "; K+1; "слов"
```

Программа работает очень просто. Она определяет длину текста в символах и заносит это число в переменную N. Затем устанавливает счетчику пробелов K нулевое значение. После чего, в цикле вырезает из текста последовательно по одному символу и проверяет, а не является ли он пробелом. Если это так, то увеличивает счетчик пробелов K на единицу, а если нет — берет следующий символ. По завершении цикла в переменной K хранится количество пробелов в тексте, и мы выводим ответ о количестве слов на экран, добавляя к K еще единичку.

302. Используя пример с подсчетом слов в телеграмме, напишите программу, имитирующую отделение связи с очень хорошим обслуживанием. Программа должна выяснить имя кли-

ента и в дальнейшем обращаться к нему только по имени. Запрашивается также регион, куда посылается телеграмма. Их три — Россия (коэффициент 1), страны СНГ (стоимость одного слова умножается на 2) и дальнее зарубежье (стоимость одного слова умножается на 5). По России стоимость одного слова составляет 3 руб. 50 коп. (причем неважно, какой длины слово). Затем у клиента запрашивается текст телеграммы и денежная сумма, определяется количество слов, стоимость телеграммы. Если денег ровно столько, сколько надо, его благодарят и прощаются. Если больше, чем надо, то ему предлагают сдачу и прощаются. Если — меньше, то просят добавить необходимую сумму, а затем, после расчета, с клиентом прощаются. А для пущей красоты я обычно прошу нарисовать окошко телеграфа, в прорезях которого и происходит диалог компьютера с пользователем (рис. 1.65).

Рис. 1.65. Телеграф

Сравнение строковых переменных

Над строковыми переменными тоже можно производить операции сравнения. Больше будет та переменная, которая начинается с символов, более близких к концу алфавита, т. е. имеющих больший код, а если символы совпадают, то более длинное слово. Строковые переменные считаются идентичными, если они полностью тождественны. Если они отличаются пробелами в начале или конце, то они уже не идентичны!

Например,

```
"DOG" > "CAT", но
"ELEFANT" < "MOUSE"
```

```
"TIGER2" > "TIGER1"
"M16  " > "M16"
```

303. Напишите программу, проверяющую, является ли введенное слово или фраза палиндромом, т. е. читающемся слева направо и справа налево одинаково (например, шалаш, казак, А роза упала на лапу Азора). Программа сообщает "Да, это палиндром", или "Нет, это не палиндром" и выводит на экран введенный текст в варианте слева направо и справа налево. Здесь необходим цикл посимвольного чтения от N (длины текста) до 1.

304. Напишите программу, подсчитывающую количество слогов во введенном слове.

305. В операторе `DATA` перечислены 10 слов. Напишите программу, которая:

- напечатает все слова из списка, отличные от слова "SUN";
- напечатает слово, ближайшее к началу алфавита в списке (считаем, что все слова различны; выполнять аналогично поиску минимального из ряда чисел);
- слово, составленное из последних символов всех слов списка;
- все слова из списка, содержащие три буквы.

306. В тексте, содержащем между словами от 1 до 3 пробелов, оставить только по одному.

307. Подсчитать, сколько раз входит каждый символ в данную строку.

308. Написать программу шифратор и дешифратор, ставящую в соответствие русским символам соответствующие латинские и наоборот (аналог так называемого транслита).

Преобразование строчных и прописных букв

Если ваш текст напечатан строчными буквами, вы хотите заменить его прописными или наоборот, не надо заново его набирать.

Для этого есть две функции:

- UCASE$(*строковая_переменная*) — преобразует все буквы строки в прописные.
- LCASE$(*строковая_переменная*) — преобразует все буквы строки в строчные.

К сожалению, эти функции применимы только к латинскому алфавиту.

Пример:

```
CLS
N$="I have 5"
R$=" рублей"
? N$;R$
N1$=UCASE$(N$)
R1$=UCASE$(R$)
? N1$, R1$
```

Результатом работы программы будет следующее:

```
I have 5 рублей
I HAVE 5 рублей
```

Функции эти очень полезны, когда мы просим пользователя ввести один из возможных ответов, например "YES" или "NO", или просто "Y" или "N", а пользователь, естественно, может ввести ответ как строчными, так и прописными буквами. В таком случае, с помощью функций UCASE$ или LCASE$ сначала надо привести ответ к требуемому виду, а потом проверять условие. Например:

```
INPUT "Будете еще играть? (Y/N); N$
IF UCASE$(N$)="N" THEN ?"До свидания"
```

Функция определения вхождения подстроки

Допустим, мы хотим найти в тексте какое-либо слово. Нам на помощь приходит функция INSTR, имеющая следующий синтаксис:

```
INSTR(N, F$, R$)
```

где N — позиция, с которой вы хотите начинать поиск (необязательный параметр), F$ — строковая переменная, в которой будет производиться поиск, R$ — подстрока, поиск которой осуществляется. В случае отсутствия этой позиции поиск начнется с первого символа строковой переменной. Функция INSTR укажет нам номер позиции, с которой начинается вхождение искомой подстроки, или 0 в следующих случаях:

- подстрока не найдена;
- значение N больше длины исходной строковой переменной;
- длина строковой переменной нулевая.

Если подстрока пустая, то результатом будет N, а при его отсутствии — 1. Поиск прекращается с первым нахождением подстроки. Например:

```
CLS
F$="Свиноводство, овцеводство, пчеловодство"
R$="чело"
N=INSTR (F$, R$)
IF N <> 0 THEN ? "Слово 'чело' в исходной строке есть
                 и начинается с"; N; "позиции"
```

Результатом выполнения программы будет фраза:

```
Слово 'чело' в исходной строке есть и начинается с 29 позиции
```

309. Заменить в исходном тексте "photo, graph, philophon, cophe" все сочетания "ph" на символ "f".

Функция *INKEY$*

Вернемся еще раз к организации пауз и задержек в вашей программе. Мы уже делали это с помощью пустого оператора цикла, а также оператора SLEEP. Есть еще одна полезная функция INKEY$, которая ожидает нажатия определенной клавиши, анализирует поступающую информацию об уже нажатых клавишах, и, в зависимости от результатов анализа, программа следует по тому или иному пути. Например:

```
CLS
? "Нажмите любую клавишу для продолжения..."
```

```
WHILE INKEY$=""
WEND
? "Продолжение..."
```

Можно задать и определенную клавишу для продолжения выполнения программы:

```
CLS
? "Нажмите клавишу ESC для продолжения..."
WHILE INKEY$<>CHR$(27)
WEND
```

Замечание

Функция `INKEY$` работает только вместе с проверкой условия.

Разные задания на работу со строковыми переменными:

310. Напишите программу для вычеркивания из данного слова всех букв "K" и "G".

311. Напишите программу для вычеркивания в данном слове всех букв, стоящих на нечетных местах после буквы "а".

312. Напишите программу для вычеркивания из данного слова всех букв "р", перед которыми стоит буква "а".

313. Напишите программу для вычеркивания из данного слова каждой третьей буквы.

314. Вычеркните из данного слова все буквы "с" и "л", стоящие на нечетных местах.

315. Напишите программу, проверяющую, все ли буквы данного слова одинаковы.

316. Напишите программу, выясняющую, можно ли из букв данного слова N составить слово M.

317. Напишите программу для проверки, есть ли в данном слове буквы "в". Если есть, то найдите номер первой из них.

318. Напишите программу, выясняющую, есть ли в данном слове буква "к", и если есть, то замените все буквы "а" в этом слове на "с".

319. Напишите программу, проверяющую, все ли буквы данного слова, стоящие на четных местах, одинаковы.

320. Определите, есть ли в данных словах N и M одинаковые буквы.

321. Выясните, есть ли в данном слове буква "в", стоящая на нечетном месте.

322. Определите, имеются ли в данном слове две одинаковые буквы, идущие подряд.

323. Заменить в исходном тексте "photo, graph, philophon, cophe" все сочетания "ph" на символ "f".

324. Подсчитать, сколько раз словосочетание ph входит в исходный текст предыдущего задания.

325. Подсчитать, по сколько раз входит каждая буква русского алфавита в следующую фразу:

"Санкт-Петербург отметил свое трехсотлетие в обстановке великого подъема настроения всего населения города и теплого отношения со стороны других стран и народов!"

326. Вычислить сумму кодов всех символов, входящих в выражение "Я уже кое-что понимаю в QuickBasic!"

327. С клавиатуры вводится некое двузначное число. Вывести его на форму в словесной записи. Например 87 — "восемьдесят семь".

328. Усложните предыдущую задачу — пусть это будет трехзначное число.

329. Пусть с клавиатуры вводится четырехзначное число, означающее количество копеек. Представьте его в словесной форме, описывающей количество рублей и копеек, например:

2142 — "Двадцать один рубль сорок две копейки"

Обратите внимание на падежи!

330. С клавиатуры вводится дата в формате ДД.ММ.ГГГГ (например, 29.06.2001). Написать программу, прописью выдающую на экран название даты следующего календарного дня (в данном случае — тридцатое июня две тысячи первого года).

Помните про високосные годы!

331. В заданном тексте каждую запятую прижать к предыдущему слову и отделить от последующего пробелом.

332. В заданном тексте, содержащем наряду с другой информацией целые числа, определить самое большое и самое маленькое целое число.

333. Распечатать из заданного текста первые слова только тех предложений, которые содержат количество слов не больше заданного.

334. Выбрать из заданного текста слова, содержащие только те буквы, которые содержит заданное слово.

335. Дан непустой текст, в который входят только цифры и буквы. Определить, удовлетворяет ли он следующим свойствам:

- текст является записью десятичного числа, кратного пяти;
- текст начинается с ненулевой цифры, за которой следуют только буквы, и их количество равно числовому значению этой цифры;
- текст состоит только из цифр;
- текст состоит только из букв;
- сумма числовых значений цифр, входящих в текст, равна длине текста.

Массивы

Мы подобрались к одной из самых сложных, на мой взгляд, тем в программировании для начинающих. Именно из-за массивов я остался на второй год в институте (потому что тогда в школах еще этим не занимались). Теперь, когда я объясняю эту тему своим ученикам, то стараюсь сделать это как можно более доходчиво, пусть не совсем научными терминами, но понятно, поскольку без представления, что такое массив, дорога в программирование будет закрыта.

Но и без умных определений не обойтись.

Итак, *массив* — это набор однородных данных (чисел, символов, слов), имеющий имя и последовательную нумерацию его элементов. Например, список фамилий учеников вашего класса — массив, численные данные о среднесуточной температуре за месяц — массив, буквы русского алфавита — массив.

Как физически массив представляется в компьютере, я рассказывать не буду, а как формально — это надо себе представлять.

Описание массива

Если мы знаем, что в программе предстоит работать с большим объемом каких-то данных, то мы должны этот массив в программе объявить с помощью специального оператора `DIM` (от англ. "dimention"), после которого указывается имя массива, а потом в скобках следует так называемый размер массива, т. е. количество его элементов. Например, пусть в группе четыре человека. Массив — это фамилии учеников. Мы тогда должны записать так:

```
DIM FAM$ (4)
```

Знак `$` добавляется, т. к. в массиве будут храниться строковые переменные. В этом случае компьютер в памяти отводит некую область из четырех ячеек, которую всю и называет `FAM$`. Кроме того, эти ячейки нумеруются натуральными числами, начиная с 1. Я всегда подобную процедуру, да и сам массив, сравниваю с улицей одноэтажных домов в деревне или маленьком городке. Построили на улице четыре дома, назвали улицу `FAM$` (имя массива дается по тем же правилам, что и имя переменной), пронумеровали дома и заселили туда жильцов (рис. 1.66).

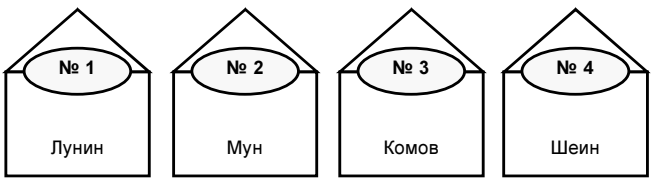


Рис. 1.66. Массив `FAM$` из четырех элементов

Из этого следует, что:

- у массива есть *имя*, которое дает ему программист;
- у массива есть *тип*, который определяется именем — числовой (имя без `$`) и текстовый (или символьный, или строковый — имя с символом `$`);
- у массива есть *размер*, т. е. количество составляющих его элементов;
- у массива есть *сквозная последовательная индексация* составляющих его элементов;
- у каждого элемента массива есть *значение* (в нашем случае это фамилия).

Предупреждение

Оператор `DIM` для каждого конкретного массива должен задаваться только один раз в программе до первого к нему обращения.

Продолжая аналогию с улицей одноэтажных домов, что надо сделать, чтобы обратиться к какому-либо конкретному жильцу? Знать его адрес!

Предположим, мы хотим потревожить господина Муна — указываем его адрес — `FAM$(2)`, т. е. название улицы и дом. Зачастую начинающие программисты путают индекс элемента массива (его номер) и значение элемента массива, т. е. "кто-кто в тереме живет". Итак, еще раз: индекс или номер элемента массива — величина постоянная, а значение (как и жильцы в доме) с легкостью может меняться.

Вначале мы рассмотрим одномерные массивы — такие, в которых адрес элемента массива определяется только одним индексом (номером "дома").

Предупреждение

На самом деле, нумерация ячеек-"домиков" в Бейсике начинается с нуля, но с единицы нам привычнее и удобнее, поэтому нулевой "домик" мы пропускаем. Возможен более законный вариант — обязать Бейсик нумеровать "домики" с единицы оператором `OPTION BASE 1`.

Заполнение одномерных массивов и вывод их на экран

Первая задача, встающая перед программистом прежде чем обработать массив, — заполнить его "жильцами". Для этого в Бейсике существует несколько способов, которые мы и рассмотрим. Кроме того, для контроля правильности заполнения лучше бы сразу выводить массив на экран, чтобы потом можно было проверить правильность решения поставленной задачи.

Для всех случаев мы рассмотрим один и тот же пример — заполнить массив N целыми числами, каждое из которых не более 100. Мы не берем конкретное значение N, чтобы вы понимали: программа не должна зависеть от исходных данных. Сейчас мы хотим создать массив из 10 чисел, а в следующий раз — из 1000. Программа останется той же.

Заполнение одномерного массива с клавиатуры

Рассмотрим следующий пример:

```
CLS
INPUT "Введите количество элементов массива"; N
DIM MASS(N)
FOR I=1 TO N
  ? "Введите"; N; "элемент массива"
  INPUT MASS(I)
NEXT I
?
FOR I=1 TO N
  ? MASS(I);
NEXT I
```

Программа требует некоторых пояснений. Первая команда традиционна — очистка экрана. Далее идет запрос с клавиатуры количества элементов массива. Потом цикл, в котором от 1 до N программа последовательно запрашивает у пользователя ввод очередного элемента массива и записывает его значение по указанному адресу `MASS(I)`. После первого цикла выполняется оператор `PRINT` без параметров. Он отображает пустую строку между

вводом значений и их выводом. Последний оператор цикла выводит значения массива на экран в строчку, что обеспечивается добавлением к оператору PRINT точки с запятой. Результаты работы программы для трех элементов будут выглядеть так:

```
Введите 1 элемент массива
? 23
Введите 2 элемент массива
? 13
Введите 3 элемент массива
? 98

23    13    98
```

Заполнение одномерного массива заранее известными значениями из оператора *DATA*

Часто встречаются задачи, когда данные для обработки уже известны и содержатся в операторе DATA. (Однако, если их много, тяжелый труд — их туда заносить. Обычно — работа для молодых программистов.)

```
DATA 23, 13, 98, 77, 45, 56, 32, 10, 90, 55
CLS
INPUT "Введите количество элементов массива"; N
DIM MASS(N)
FOR I=1 TO N
  READ MASS(I)
  ? MASS(I);
NEXT I
```

Программа стала несколько короче, т. к. в этом случае можно совместить чтение данных из DATA и одновременный вывод их на экран — ведь пользователь освобожден от необходимости вводить данные с клавиатуры. Мы сразу увидим массив на экране.

Этот способ экономичней первого, и, кроме того, при отладке программы нет нужды всякий раз заново вводить данные.

Заполнение массива при помощи стандартных функций

Так как чаще мы решаем учебные задачи, то конкретные числовые или строковые значения элементов массива нас мало интересуют. Важно, чтобы программа работала правильно. Поэтому часто массив заполняется случайными значениями.

```
CLS
RANOMIZE TIMER
INPUT "Введите количество элементов массива"; N
DIM MASS(N)
FOR I=1 TO N
  MASS(I)=INT(RND(1)*100)
  ? MASS(I);
NEXT I
```

Мы вычисляем какую-либо функцию, например синус, и значения этой функции заносим в массив. Пусть дана функция $y = \sin x$, где x меняется от 1 до 10 с шагом 0,5. Здесь мы должны сначала решить для себя, а сколько будет значений вычислено. Мы это уже делали, но повторение — мать учения.

```
N=(Хнач. — Хкон.)/шаг+1
```

В нашем случае:

```
N=(10-1)/0.5 +1=19
```

Тогда программа будет выглядеть так:

```
CLS
INPUT "Введите начальное, конечное значения аргумента
и шаг приращения"; XN, XK, DX
N=INT((XK — XN)/DX)+1
DIM MASS(N)
I=1
FOR X=XN TO XK STEP DX
  MASS(I)=SIN(X)
  ? MASS(I);
  I=I+1
NEXT X
```

Здесь нам уже приходится вручную наращивать индекс `I`, т. к. параметром цикла в данном случае является `X`.

После заполнения массива настает пора его обрабатывать. Чаще всего приходится обрабатывать все элементы массива, поэтому действия выполняются *в цикле*.

Изменение значений элементов массива ведется с помощью оператора присваивания, например:

```
MASS(1)=13
MASS(3)=12
MASS(3)=SQR(MASS(1)+MASS(3))
? MASS(3)
```

Вопрос. Какое значение будет выведено на экран? Чему равны значения `MASS(1)` и `MASS(3)`?

Задания. Прежде всего попрактикуемся в заполнении массивов и выводе на экран не только численных значений элементов массива, но и графической их интерпретации.

336. Заполните массив десятью случайными целыми числами, каждое из которых лежит в пределах от 50 до 200, и выведите на экран их численные значения, а также графическое представление в виде вертикальных закрашенных прямоугольников шириной 30 и высотой, соответствующей их значению. Нижние стороны прямоугольников лежат на линии с координатой Y=300, левой стороне первого прямоугольника соответствует координата X=100 (рис. 1.67).

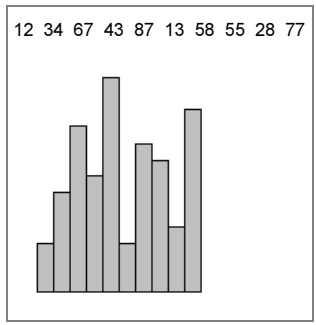


Рис. 1.67. Столбиковая интерпретация одномерного массива

337. Заполните массив десятью случайными целыми числами, каждое из которых лежит в пределах от 5 до 30, и выведите на экран их численные значения, а также графическое представление в виде закрашенных соприкасающихся кругов, радиусы которых равны значениям элементов массива (рис. 1.68).

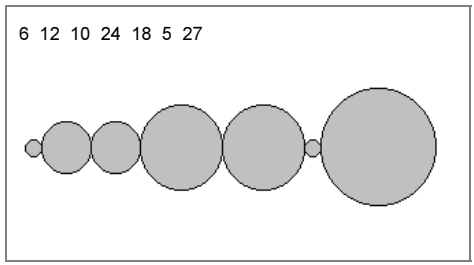


Рис. 1.68. Круговая интерпретация одномерного массива

338. Напишите программу вычисления среднего арифметического содержащихся в операторе `DATA` следующих десяти чисел: 31, 19, 52, 65, 6, 8, 13, 16, 97, 33.

339. Напишите программу вычисления выражения:

$$\frac{(x_1 - S)^2 + (x_2 - S)^2 + ... + (x_N - S)^2}{N},$$

где x_i — числа из оператора `DATA` предыдущего задания; N — их количество; S — среднее арифметическое элементов массива x.

340. Найти сумму 1, 4, 9, 16 и... 81 элементов массива, состоящего из 100 целых случайных чисел, каждое из которых лежит в пределах от 2 до 22.

341. Замените в массиве из 10 случайных целых чисел, каждое из которых лежит в пределах от 1 до 10, все четные элементы нулями и выведите полученный массив на экран.

342. Массив состоит из 60 случайных двузначных целых чисел. Выведите их на экран в обратном порядке по 6 чисел в строке.

343. В массиве содержатся 10 букв — С, Ф, О, И, К, Л, О, И, Л, Н. Выведите на экран слово, образованное буквами с четными индексами, и слово, образованное буквами с нечетными индексами.

344. Массив состоит из 20 целых положительных и отрицательных чисел, модуль каждого из которых в пределах от 2 до 12. Выведите на экран сначала отрицательные, а затем положительные числа. Определите, модуль суммы каких чисел больше — положительных или отрицательных.

345. Найдите максимальный и минимальный элементы массива из 10 случайных целых двузначных чисел и разность между ними. Представьте графическую столбиковую интерпретацию этого массива, выделив максимальный элемент красным, а минимальный — зеленым цветом. Остальные прямоугольники должны быть желтого цвета.

Если в программе используется несколько массивов, то они могут быть описаны через запятую, например:

```
DIM X(10), Y(10), W$(20)
```

Теперь задания с несколькими массивами.

346. Даны два массива, заполненные каждый десятью случайными целыми числами, каждое из которых от 1 до 9 включительно. Сложите массивы поэлементно, результаты запишите в третий массив. На экран вывести все три массива.

347. Даны три массива с одинаковым количеством элементов 5, в которых содержатся стороны треугольников Ai, Bi и Ci. Определите периметр Pi и площадь Si каждого треугольника по формуле Герона. (Так как для того, чтобы треугольник существовал, необходимо определенное соотношение его сторон, то лучше элементы массивов задать в операторе `DATA`.)

348. Найдите скалярное произведение двух массивов A и B, состоящих из 5 элементов каждый, которые содержат случайные числа от 2 до 9 включительно. Воспользуйтесь формулой:

```
P=A1*B1+A2*B2+ ... +AN*BN
```

где `N` — размер массива.

349. Найдите соотношение SX/SY, где SX и SY — средние арифметические значения массивов X и Y соответственно. (Массивы из 10 элементов содержат случайные двузначные целые числа.)

350. Определите объем каждого из 10 цилиндров, для которых заданы радиусы оснований Ri (случайные целые числа от 5 до 25 см) и высоты Hi (случайные целые числа от 10 до 30 см).

351. Заданы 10 пар координат Xi, Yi одних точек на плоскости и 10 пар координат Ai, Bi других точек на плоскости. Вычислите попарно расстояния между точками по формуле:

$$S_i = \sqrt{(X_i - A_i)^2 + (Y_i - B_i)^2} .$$

Занесите эти расстояния в массив S. Проиллюстрируйте задачу графически (соответствующими отрезками на экране). Выделите разными цветами наибольшую и наименьшую длину.

352. Дан одномерный массив W из 10 случайных целых чисел, каждое из которых лежит в пределах от 1 до 100. Получите новый массив R, где каждый элемент создается из массива W делением соответствующего элемента на его индекс.

353. Вычислите и представьте в виде массива последовательность первых 20 чисел Фибоначчи, если X_1=1, X_2=2, а каждый последующий элемент равен сумме двух предыдущих.

354. В операторе DATA содержатся данные о среднесуточной температуре в течение февраля 2005 г. по Санкт-Петербургу. Вычислите среднюю температуру февраля, наибольшую и наименьшую температуры. Постройте линейный график изменения среднесуточной температуры.

355. Школьники неохотно носят одежду, отличающуюся по цвету от одежды одноклассников. Напишите программу, выбирающую 2 цвета (для мальчиков и для девочек) из 12 возможных цветов, которые сегодня будут носить все. Выбор производится случайным образом и сопровождается выводом на экран прямоугольников соответствующих цветов, внутри

одного из которых написано "Сегодня этот цвет для мальчиков", а внутри другого "Сегодня этот цвет для девочек".

356. В фирме Green Beavis Ltd работают семь сборщиков компьютеров. Для того чтобы повысить производительность их труда, в конце недели обрабатывают сведения о количестве компьютеров, собранных каждым из них ежедневно. Напишите программу, которая выдаст на экран следующие данные:

- наибольшее количество компьютеров, собранных одним служащим за неделю;
- среднее количество собранных за день компьютеров;
- лучший результат за один день;
- номер служащего, показавшего этот результат и день, в который он был достигнут.

357. Дан массив X, состоящий из 100 целых случайных чисел, каждое из которых лежит в пределах от 3 до 13. С клавиатуры вводится целое число N, также лежащее в этих пределах. Определите количество элементов массива, равных числу N.

358. Даны два числовых массива X и Y с количеством элементов 10 и 20 соответственно. Получите массив Z из 30 элементов, составленный добавлением массива X в конец массива Y.

359. Дан массив из 20 случайных целых чисел, каждое из которых лежит в пределах от 10 до 50. Найдите максимальное число и его номер, а если таких чисел несколько, то подсчитайте, сколько их.

360. Дан массив из 20 случайных целых чисел, каждое из которых лежит в пределах от 10 до 50. Определите среднее арифметическое элементов массива, а также значение элемента, ближайшего к среднему, и его номер.

361. Дан массив среднемесячных температур за год. Определите, в каком месяце была самая высокая температура, а в каком самая низкая, а также среднесезонные температуры.

362. У автора этой книги была копилка, в которой было 100 советских монеток достоинством в 1, 2, 3, 5, 10, 15, 20 и 50 ко-

пеек. Задайте массив M(100) случайным образом из этого набора, а затем подсчитайте, сколько в копилке было пятачков и полтинников, и какова общая сумма накопленного.

363. Дан массив из десяти целых двузначных случайных чисел. Найдите сумму трех максимальных из них.

364. Заполните массив R(25) случайными целыми двузначными числами так, чтобы числа не повторялись.

365. Программа "Пожиратель звезд". Когда мы говорили о циклах и построении графиков функций, то писали программу о движении "звездолета", который, пролетая по траектории заданной тригонометрической функции, уничтожал планету, находящуюся от него в опасной близости. Теперь, со знанием массивов, мы можем написать более красивую и более сложную программу. Сначала заполним экран тысячью разноцветных звезд, изображенных либо точками, либо окружностями радиусом 1. Координаты всех звезд запоминаются в массивах X(1000) и Y(1000). Затем по траектории $y = 2\sin\frac{x}{2} + \frac{1}{2}\cos 2x$ начинает двигаться кавалькада. Впереди сторожевой звездолет (закрашенный круг радиусом 2), проверяющий, не находится ли какая-нибудь звезда в опасной близости от армады (S<=30), и уничтожающий ее в таком случае (звезда заменяется точно такой же, но цветом фона). Позади движутся три крейсерских звездолета (закрашенные круги радиусом 4). В результате выполнения программы на экране должен быть проложен коридор по траектории функции с шириной 60 экранных точек (рис. 1.69).

Рис. 1.69. "Пожиратель звезд"

366. Напишите программу "Сторож", которая бы заставила змейку (рис. 1.70) оббегать стороны экрана по часовой стрелке (это можно написать и без массива). Усложните программу, взяв на службу еще одну "змейку". Теперь они ползают друг за другом (рис. 1.71). Еще сложнее — программа для движения змейки, управляемой стрелками или буквенными клавишами и совершающей повороты под прямым углом. А если и это все по плечу, то заставьте змейку поворачивать еще и под углом 45°.

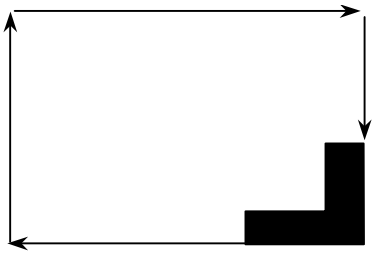


Рис. 1.70. Сторож-змейка

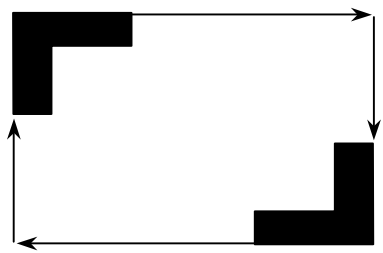


Рис. 1.71. Две змейки

367. Случайным образом сформировать вектор B(4), элементы которого являются двузначными числами и делятся на 3.

368. Дана последовательность чисел Фибоначчи, определяемая соотношениями: u[1]=1, u[2]=1, u[n]=u[n–1]+u[n–2], n>2. Проверьте на нескольких примерах, будет ли u[5k] (k=1,2,...) делиться на 5.

369. Найти k простых чисел Фибоначчи.

370. Для заданного целого числа m найти среди первых k чисел Фибоначчи хотя бы одно, делящееся на m.

371. Найти среди первых 20 чисел Фибоначчи все четные числа.

372. Модой массива называется число M, которое встречается в массиве наиболее часто. Если в массиве имеется несколько наиболее часто встречающихся элементов и число их вхождений совпадает, то считается, что массив не имеет моды. Напишите программу, которая либо вычисляет моду массива, либо устанавливает, что последний ее не имеет.

373. При заданных A и B подсчитать, сколько кругов с заданными радиусами R1, R2, ... , RN имеют большую площадь, чем прямоугольник со сторонами A, B.

Простейшие сортировки

Одной из основных операций, производимых над массивами, являются операции сортировки или упорядочивания элементов массива по какому-либо признаку: чаще по возрастанию или убыванию — для чисел, и по алфавиту — для символов и строк.

Сортировок придумано множество, и, говорят, тому, кто придумает новый эффективный метод сортировки, сразу будет вручена Нобелевская премия.

С древних времен, однако, до нас дошли два самых простых (но, конечно, не самых эффективных) способа сортировки, которые мы здесь и рассмотрим.

Сортировка выбором

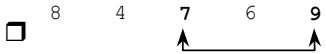
Допустим, дан числовой массив из N элементов. Надо отсортировать его по возрастанию.

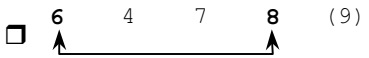
Суть способа в следующем. Находим наибольший элемент в массиве и меняем его местами с последним. Уменьшаем количество рассматриваемых элементов на 1 (т. к. последний элемент уже на своем месте). Повторяем операцию для уменьшенного на единицу массива. И так — N–1 раз.

Пусть дан массив из пяти элементов:

```
8   4   9   6   7
```

Рассмотрим процесс упорядочивания по шагам.





На этом шаге третий элемент поменялся сам с собой.

374. Напишите программу для упорядочивания массива по возрастанию методом выбора. Измените программу так, чтобы она упорядочивала массив по убыванию.

375. Дан массив из 13 чисел. Расположите числа по возрастанию. Введите с клавиатуры число M так, чтобы оно вошло в массив и получившийся массив также был бы упорядочен по возрастанию.

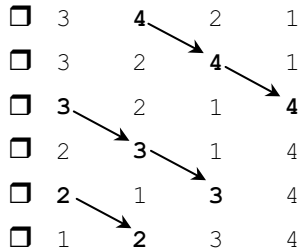
376. Дан массив из двадцати английских слов. Упорядочьте его по убыванию (от Z до A).

Метод обмена или "пузырька"

Название данного метода часто вызывает нездоровый смех у молодежи, хотя никакого тайного смысла у этого метода нет. Просто он выполняется таким образом, что максимальное число после каждого шага сортировки как бы всплывает в конец массива, на свое заслуженное место. Заключается метод в следующем. Программа, начиная с первых элементов массивов, сравнивает эти элементы попарно, и, в случае, если они расположены не по возрастанию, меняет их местами. В результате $(N-1)^2$ перестановок (в случае самого плохого расположения элементов массива — все элементы по убыванию) массив окажется упорядочен по возрастанию. Например, есть массив из четырех элементов:

```
4 3 2 1
```

Рассмотрим по шагам метод "пузырька" для этого массива.



377. Напишите программу, реализующую метод "пузырька". Для уменьшения количества ненужных сравнений может служить счетчик, подсчитывающий количество обменов за один

полный пробег вдоль всего массива. Как только его значение станет равно нулю (т. е. ни одного нового обмена не будет произведено), это будет означать, что массив упорядочен. Для наглядности оформите исходный и получившийся массив в виде столбиковых интерпретаций друг под другом.

378. Дан массив букв, составляющих английский алфавит, но размещенных не по порядку. Напишите программу, преобразующую этот массив в алфавит английского языка.

379. Дан массив из 13 четырехбуквенных русских слов (существительных и нарицательных), в единственном числе, в именительном падеже. Упорядочить их по алфавиту.

Двумерные массивы

Что такое двумерный массив? Это такой набор однотипных данных, местоположение каждого элемента которого определяется не одним индексом, а двумя. Например, для тех, кто с детства играл в "морской бой", не будет открытием, что каждая клеточка игрового поля обозначается двумя символами — буквой и цифрой, например, А5 — "мимо", И10 — "попал", Ж7 — "убит". Только в Бейсике принято в качестве индексов использовать все же целые числа. Жизненный пример использования двумерных массивов — билеты в кино или театр, имеющие для каждого зрителя две координаты — ряд и место[1].

Описываются подобные массивы в Бейсике тем же оператором `DIM`, после которого в скобках указываются две размерности массива — количество строк и количество столбцов. Например, массив 5×3 объявим так:

[1] Кстати, маленькое лирическое отступление. Откуда пошла традиция указывать на билетах ряд и место? Оказывается из Франции. Когда тамошние дворяне при шпагах и гордом характере приходили в театр, имея просто билет без указания места, то нередко возникали смертоубийственные стычки из-за этих самых мест. Королю это надоело, и он обратился за помощью к ученому Декарту, который и предложил систему — "ряд—место". Эта система в дальнейшем трансформировалась в привычную нам декартову систему координат — ось X, ось Y.

```
DIM X(5, 3)
```

X (1, 1)	X (1, 2)	X (1, 3)	X (1, 4)	X (1, 5)
X (2, 1)	X (2, 2)	X (2, 3)	X (2, 4)	X (2, 5)
X (3, 1)	X (3, 2)	X (3, 3)	X (3, 4)	X (1, 5)

Заполнение двумерных массивов и вывод их на экран

В обработке двумерных массивов есть своя специфика — использование вложенных циклов.

Заполним двумерный массив X(3, 5) целыми случайными числами, лежащими в интервале от 1 до 20 и выведем массив на экран в виде таблицы.

```
CLS : RANDOMIZE TIMER
DIM X(3, 5)
FOR I =1 TO 3
  FOR J=1 TO 5
    X(I, J)=INT(RND(1)*20)+1
    ? X(I, J);
  NEXT J
  ?
NEXT I
```

На что надо обратить внимание. Для начала, при выводе массива во внутреннем цикле после оператора `PRINT` стоит точка с запятой. Это дает возможность отображать массив построчно. А оператор `PRINT` без параметров, указанный после внутреннего цикла, позволяет после вывода каждой строки элементов массива переводить курсор на новую строчку.

380. Найдите в массиве из приведенного выше примера максимальный и минимальный элементы и, при выводе массива на экран, выделите их красным цветом.

381. Дан двумерный массив 5×5. Определите сумму элементов каждой строки и ту строку, в которой сумма элементов максимальна.

382. Дан массив A(2, 10). В первом столбце содержатся координаты X точек плоскости экрана, а во втором столбце — координаты Y тех же точек. Определите количество точек, попадающих в нижнюю правую четверть экрана, выведите их на экран, а искомые точки выделите другим цветом.

383. Определите наименьший элемент в массиве X(10, 10). Выделите его другим цветом.

384. Дан массив W$(5, 4), в котором каждая строка состоит из четырех символов, составляющих английское слово. Отсортируйте массив таким образом, чтобы слова были расположены по алфавиту.

385. В массиве R (5×5) поменяйте местами первую и последнюю строки.

386. В массиве R (5×5) замените элементы, стоящие ниже главной диагонали, нулями.

387. В массиве R (5×5) замените элементы главной диагонали нулями.

388. В массиве R (5×5) вычислите сумму элементов главной диагонали.

389. В массиве R (5×5) упорядочьте строки по возрастанию элементов главной диагонали.

390. Определите, является ли заданный массив 3×3 магическим квадратом, т. е. таким, суммы элементов которого в строках, столбцах и главных диагоналях равны между собой.

391. Выведите на экран номера строк массива 5×5, сумма элементов которых четна.

392. Выведите на экран изображение Андреевского флага, если у данного массива 5×5 суммы элементов диагоналей равны, и флаг Японии — в обратном случае (рис. 1.72 и 1.73).

393. Одномерный массив из N элементов свернуть по спирали в квадратную матрицу размерностью корень квадратный из N по следующему образцу.

4	6	8	3	**2**
6	**9**	5	**3**	4
6	0	**1**	8	7
5	**8**	2	**3**	5
7	6	0	3	**4**

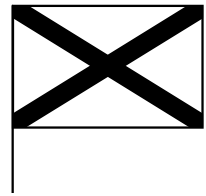


Рис. 1.72. Андреевский флаг (суммы элементов диагоналей равны)

4	6	8	3	**2**
6	**8**	5	**3**	4
6	0	**1**	8	7
5	**8**	2	**3**	5
7	6	0	3	**4**

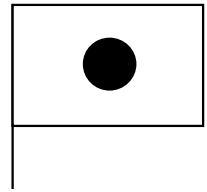


Рис. 1.73. Флаг Японии (суммы элементов диагоналей не равны)

394. Исходный массив S1(16) состоит из следующих элементов:

3, 5, 9, 7, 12, 34, 21, 13. 6, 89, 54, 66, 2, 10, 99, 55.

Создайте массив S2(4, 4), вид которого представлен на рис. 1.74.

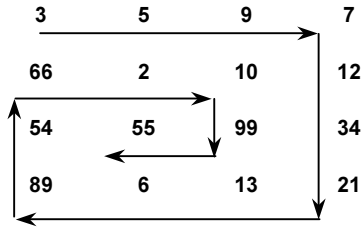


Рис. 1.74. Преобразованный в спираль одномерный массив

395. Сформируйте двумерный массив 7 на 7 по следующему правилу: элементы главной диагонали равны 1, а все остальные элементы — нулевые. Распечатайте полученную матрицу.

396. Дан двумерный массив A(m, n). Напишите программу построения одномерного массива B(m), элементы которого, соответственно, равны суммам элементов строк.

397. Транспонируйте (т. е. поменяйте строки и столбцы местами) произвольный двумерный "квадратный" массив. Дополнительные массивы не используйте.

398. Напишите программу замены элементов двумерного массива, находящихся в четной строке и четном столбце на число 100.

399. Дан массив A(N,N). Напишите программу, которая прибавляла бы к каждому элементу данной строки элемент, принадлежащий этой строке и главной диагонали.

400. Из одномерного массива длины 8 сформируйте двумерный массив так, чтобы первая строка нового массива содержала четные по номеру элементы исходного массива, а вторая — нечетные.

401. Напишите программу, находящую в двумерном массиве номера строк с наибольшей суммой элементов.

402. Дана матрица из 3 строк и 6 столбцов. Элементы каждого столбца представляют собой длины трех сопряженных ребер одного из шести параллелепипедов. Вывести на экран номер каждого столбца, в котором задан параллелепипед большего объема, чем объем шара с заданным радиусом R, и число таких столбцов.

403. Дана матрица из 3 строк и 7 столбцов. Первый элемент каждого столбца представляет собой длину наибольшей стороны, а две другие — стороны одного из семи треугольников, заданных столбцами исходной матрицы. Вывести на экран номера столбцов, которыми заданы тупоугольные треугольники, и количество таких столбцов.

Подпрограммы

Иногда в определенных местах программы приходится выполнять практически одни и те же последовательности действий с разными исходными данными. Такие последовательности действий можно оформить в виде так называемых *подпрограмм* (от англ. "subroutine"). Подпрограммы сокращают текст программы,

существенно уменьшают время их исполнения, облегчают жизнь программистам, которые могут создавать программы модульно, т. е. собирая сложную программу из законченных кусочков более простых составляющих. Это позволяет создавать большие программы группой программистов, разрабатывать и реализовать группе школьников какие-либо глобальные проекты.

Для примера использования процедуры приведем программу вычисления следующего выражения:

$$Z = \frac{N!}{M! \times (N-M)!}$$

Переменные N и M, а также выражение (N–M) снабжены восклицательным знаком, который в математике означает не что иное, как факториал (например, N-факториал). *Факториал* — это произведение натуральных чисел от 1 до N включительно. Например, 3!=1×2×3=6, а, например, 6! — это уже 1×2×3×4×5×6=720.

В выражении нам три раза придется вычислять факториал, производя при этом одни и те же действия с разными числами. Нам облегчит работу подпрограмма, которую мы разместим в конце основной программы и назовем, например, `FACT`. В подпрограмме мы опишем, как вычислить факториал вообще, а затем из основной программы обратимся к подпрограмме три раза с различными параметрами. Вот как будет выглядеть программа:

```
CLS
1 : INPUT "Введите значения N и M"; N, M
K=N
GOSUB FACT
X1=P
K=M
GOSUB FACT
X2=P
K=(N-M)
GOSUB FACT
Z=X1/(X2*P)
PRINT "Искомое выражение равно"; Z
PRINT "Еще одно выражение?(Y/N): INPUT a$
```

```
IF UCASE$(a$)="Y" THEN 1 ELSE PRINT "До новых встреч
                  у монитора..."
END

' Описание подпрограммы FACT для вычисления факториала
FACT:
P=1
FOR i=1 TO K
  P=P*i
NEXT i
RETURN
```

Рассмотрим работу программы. Начало стандартное: очистка экрана, запрос с клавиатуры исходных данных. Самое интересное начинается потом. Подпрограмма FACT описана после слова END основной программы, что обусловлено простым желанием, чтобы программа все же не зациклилась, а когда-нибудь все-таки завершилась. Параметром подпрограммы является переменная K, для которой и вычисляется факториал.

В основной программе мы присваиваем переменной K значение переменной N, для которой мы хотим вычислить первый факториал из трех. После чего необходимо обратиться к подпрограмме. Для этого существует специальное слово GOSUB, после которого указывается имя подпрограммы (в нашем случае FACT). Управление переходит на первый оператор, содержащийся в подпрограмме, затем выполняются все операторы подпрограммы, вплоть до ключевого слова RETURN. Это означает, что подпрограмма закончилась, и управление передается в основную программу.

Предупреждение

Ключевое слово RETURN передает управление в основную программу на оператор, следующий за обращением к подпрограмме GOSUB.

Результатом выполнения подпрограммы в нашем случае является переменная P, в которой сохраняется значение факториала. Запомним первое полученное значение (фактически N-факториал) в переменной X1. Это необходимо сделать, т. к. подпрограмма

будет выполняться несколько раз, и значение переменной `P` будет меняться.

Затем мы повторяем действия для переменных `M` и `(N-M)` соответственно. Причем в последнем случае уже не надо запоминать полученное значение факториала в новой переменной, т. к. больше обращений к подпрограмме не будет. Осталось вывести на экран полученное значение и запросить пользователя, будет ли он работать с программой еще.

Если непонятно, запустите пошаговый режим исполнения этой программы, разберитесь, как она работает (клавиша <F8>).

Предупреждение

Привыкайте оформлять самостоятельные куски программ в виде подпрограмм. Это способствует четкой структуризации, описанию внутренних логических связей и понимаю со стороны коллег-программистов, что не маловажно. Программирование с использованием подпрограмм — это хороший стиль!

Итак, если все ясно, то надо бы выполнить ряд упражнений с использованием подпрограмм.

404. Для каждого двумерного массива X(3, 4), Y(5, 3), Z(4, 6) определите номер строки с максимальной суммой положительных элементов.

405. Напишите программу, предлагающую пользователю меню из десяти функций и строящую по ним графики этих функций в зависимости от выбора пользователя.

406. Напишите программу для младших школьников, проверяющую знание ими таблицы умножения от 2 до 12. Учащемуся задаются 5 примеров перемножения случайных чисел в заданном интервале. Оценкой является количество правильных ответов. Используйте подпрограмму для печати замечаний в ответ на каждый результат, вводимый пользователем. За правильный ответ — замечание должно быть поощрительным, за неправильный — сожалеющим. Чтобы сделать опрос более интересным, необходимо заготовить по десять замечаний для правильных и неправильных ответов и выбирать их случайным образом, обращаясь при этом к пользова-

телю по имени, запрошенному в начале программы. Сделайте красочную заставку, легкое музыкальное сопровождение тоже не будет лишним.

407. Напишите программу, отображающую цветное кольцо. Используя ее в качестве подпрограммы, нарисуйте олимпийский флаг.

408. Напишите программы, находящие минимальное и максимальное значения из трех чисел X, Y и Z, введенных с клавиатуры. Используя их в качестве подпрограмм, напишите программу, вычисляющую значение следующей функции:

$$S = \frac{\max(x, y, z) - 2^x \min(x, y, z)}{\sin 2 + \dfrac{\max(x, y, z)}{\min(x, y, z)}}.$$

409. Даны два одномерных массива из 20 элементов каждый. Элементом является случайное целое двузначное число. Напишите программу с использованием подпрограммы, которая изменяет исходный массив путем деления четных чисел на их индексы. Используя эту подпрограмму, определите, в каком из массивов было произведено больше замен.

410. P принимает значения (n!+4)3, если n>=5, и значения sin(n!), если n<5. Напишите программу вычисления P.

411. Даны натуральные числа m и n. Вычислите max(min(m, n), min(2, 6)).

412. По вещественному значению x вычислите значение функции:

sh(x)*tg(x+1) – tg2(2+sh(x–1)).

413. Опишите функцию Stepen(x, n), зависящую от вещественного x и натурального n и вычисляющую (посредством умножения) величину xn. Используйте ее для вычисления значения выражения 2.4k+(a+1)–5.

414. Даны три числа. Определите их наибольший общий делитель.

415. Даны отрезки a, b, c, d. Для каждой тройки этих отрезков, из которых можно построить треугольник, найдите площадь

данного треугольника. Определите процедуру, определяющую площадь треугольника со сторонами x, y и z, если такой треугольник существует.

416. Даны координаты вершин двух треугольников. Определите, какой из них имеет большую площадь.

417. Найдите наименьшее общее кратное четырех заданных натуральных чисел.

418. Дано натуральное число n. Выясните, является ли оно полным квадратом. Определите функцию, позволяющую распознавать полные квадраты.

419. Дано натуральное число n. Выясните, является ли оно степенью пятерки. Определите функцию, позволяющую распознавать степени пятерки.

420. Дано натуральное число n. Выясните, является ли оно простым. Определите функцию, позволяющую распознавать простые числа.

421. Даны три натуральных числа. Определите их наибольший общий делитель.

422. Числа Фибоначчи U0, U1, U2, ... определяются следующим образом:

U0=1, U1=2, Un=Un–1+Un–2 (n=2, 3, ...).

Напишите программу вычисления Un для данного неотрицательного целого n. Воспользуйтесь функцией.

423. Даны длины a, b и c сторон треугольника. Найти медианы треугольника, сторонами которого являются медианы исходного треугольника.

Медиана, проведенная к стороне a, вычисляется по формуле:

$m = 0,5\sqrt{2b^2 + 2c^2 - a^2}$.

424. Даны две квадратные вещественные матрицы. Напечатать квадрат максимального элемента той из них, в которой наименьший след (сумма диагональных элементов), считая, что такая матрица одна.

В заключение несколько слов о подпрограммах в Russian QuickBasic. Мы привели самый простой способ написания и использования подпрограмм. Но если вы чувствуете в себе силы, то попробуйте самостоятельно разобраться с такими мощными инструментами Бейсика по написанию подпрограмм, как `DECLARE SUB` и `CALL SUB`. Не пожалеете!

Задачи на нахождение корня функционального уравнения на заданном отрезке с заданной точностью методом половинного деления

Общие сведения. Дано уравнение f(x)=0, x принадлежит интервалу [a;b]. Метод состоит в последовательном приближении к корню за счет уменьшения отрезка, на котором находится корень. Каждое новое приближение x находится как середина текущего отрезка. Концы текущего отрезка выбираются из условия противоположности знака f(x) на его концах. Вычисление корня заканчивается, когда длина отрезка станет меньше заданной точности E.

Пример:

Уравнение y=3cos(2x+4) имеет единственный корень на отрезке [1;3]. Решим это уравнение с точностью до 0,001 методом половинного деления на компьютере.

```
DEFFN F(X)= 3*COS(2*X+4)
INPUT A,B,E
1: C=(A+B)/2
IF FN F(C)=0 THEN A=C: B=C: GOTO 2
IF FN F(A)* FN F(C)<0 THEN B=C ELSE A=C
2: IF B-A>E THEN 1
X=(A-B)/2
PRINT X
END
```

Для следующих функций задать интервал изменения от 0 до 2, точность — 10^{-4}.

425. Найти корень уравнения с заданной точностью:

а) $\frac{\sqrt[3]{4}-\sin^2(x/10)}{\sqrt{x}}-x=0$;

б) $\sqrt[3]{0,07}-2x+\operatorname{arctg}\sqrt{x}=0$;

в) $(\ln(1+x)+\frac{10}{3}e^{0,01x})/2\sqrt{x-x=0}$;

г) $(\sin x+\cos x)^2/\sqrt[3]{33,5^2+\sqrt{\frac{3}{7}}}-x=0$;

д) $e^{-\sin^2 x}+\frac{3/7x}{1+\sqrt{x}}-x=0$.

Работа с файлами

Сейчас, когда в ваших головах и руках уже есть практически все необходимые инструменты для написания сложных программ, осталось немножко поднапрячься и узнать, что хранить исходные данные для больших программ очень удобно в виде отдельных файлов. Но, чтобы это понять, надо иметь минимальное представление о файловой организации информации на дисках.

Файловая система

Информация, вводимая с клавиатуры или обрабатываемая с помощью программных средств Бейсика размещается в оперативной памяти компьютера, которая является энергозависимой, и, как только мы выключим питание, вся наша информация погибнет. Чтобы избежать это и донести наши гениальные программы по крайней мере до преподавателя (чтобы он их достойно оценил), а лучше до далеких потомков, необходимо сохранять наши работы на диске в виде файлов.

Файл — это поименованная область на магнитном или лазерном диске. Поименованная — значит, имеющая имя. В файлах могут содержаться тексты, графические и видеоизображения, звуки и

музыка, таблицы и базы данных и многое другое, но нас интересуют прежде всего программы, написанные на Бейсике, и данные для этих программ.

Мы уже объясняли в самом начале, как сохранить программу в виде файла и открыть уже существующий файл. Имя файлу мы даем сами, но к нему есть некоторые требования:

- имя не должно быть больше чем 8 символов;
- имя может состоять из букв латинского алфавита, цифр и символов, например, _, -, (,), $ и некоторых других. Впрочем, злоупотреблять специальными символами не стоит — букв и цифр вполне хватает;
- в имени файла запрещены символы <Пробел>, *, точка, запятая, кавычки, двоеточие.

У файла также есть расширение имени. Оно имеет длину не более трех символов, указывается через точку после имени и характеризует тип файла. Изучаемая нами версия Бейсика (как, впрочем, и практически все другие) автоматически при сохранении добавляет к имени файла расширение bas, что упрощает поиск написанных нами программ и работу с ними.

Для того чтобы хранить свои файлы и результаты работы с ними на диске, лучше завести свой личный каталог (или папку — это одно и то же). Надеюсь, вы знаете как это делается.

В файлах вы можете хранить как исходные данные для обработки, так и результаты работы вашей программы.

Для работы в Бейсике нам чаще нужны файлы, хранящие однородные по типу или структуре сведения о каких-либо объектах. Набор данных о каком-либо одном объекте называется записью.

За самой последней записью находится невидимый символ конца файла, который устанавливается автоматически. Файл может быть пустым, т. е. содержать 0 байт информации, но имя файла и символ конца файла будут присутствовать всегда. Таков закон.

Записи могут содержать данные разных типов, но должны быть обязательно одинаковы по структуре, например:

```
"Запорожец", "4067 ЛДЕ", "1972", "100$"
"ГАЗ-34", "6666 ЛАА", 1989, "3500$"
```

В соответствии со способом доступа к файлам они делятся на два вида.

Способы доступа к файлам

В изучаемом нами Бейсике существуют два метода доступа к информации, хранящейся в файлах:

- последовательный доступ;
- прямой доступ.

Файлы последовательного доступа наиболее просты как в организации, так и в работе с ними. Записи обрабатываются последовательно одна за другой. Информация в таких файлах хранится в виде текста в кодах ASCII. Подобные файлы легко просмотреть на экране, используя любой простейший редактор, или в самом Бейсике. Но, как всегда, у каждой медали две стороны. Простота — хорошо, а последовательность в данном случае — плохо. Если информация об интересующих меня объектах упорядочена в файле по алфавиту, то мне всякий раз придется перебирать практически весь файл, чтобы добраться до нужной записи. Отсюда, при большом информационном объеме файла обработка его резко замедляется.

Файлы прямого доступа хранят информацию в специальном формате, в котором каждая запись занимает строго фиксированную одинаковую с остальными длину. То, что такие файлы могут занимать на диске больше места, чем файлы последовательного доступа, с лихвой компенсируется скоростью работы с ними.

Предупреждение

Если при каждом обращении к файлу вы собираетесь использовать почти все данные, а менять их содержимое часто не предполагается, то выбирайте метод последовательного доступа. Его применение будет и более оптимальным и облегчит вам программирование. Прямой доступ к файлу целесообразен в том случае, когда требуется часто менять содержимое записей и просматривать их в произвольном порядке. Так адреса для рассылки корреспонденции уместнее хранить в последовательном файле, а бронированные места в зале театра, меняющиеся от спектакля к спектаклю, — в файле прямого доступа.

Операции над файлами

Независимо от того, какие действия мы проделываем с информацией, хранящейся в файле, мы должны будем производить следующие обязательные операции:

- открытие файла;
- чтение и запись обрабатываемых данных;
- закрытие файла.

Предупреждение

Нельзя одновременно пытаться читать и записывать в открытый файл. Сначала надо открыть файл для чтения, прочитать нужную информацию, обработать ее и закрыть файл. Потом открыть файл для записи, записать туда результаты обработки и закрыть файл.

Открытие файла

Для открытия файла предназначен оператор `OPEN`, имеющий следующий формат:

```
OPEN имя_файла FOR режим AS # номер файла
```

С именем файла должно быть уже все понятно. Режим определяет доступ к данным файла. Возможны следующие режимы:

- `INPUT`. Это режим чтения информации из файла. В случае, если указывается несуществующее имя файла, возникнет сообщение об ошибке "Файл не найден".
- `OUTPUT`. Режим записи информации в файл. Обычно при этом создается новый файл. Если же открывается для записи уже существующий файл, то ранее хранимая в нем информация будет безвозвратно утеряна.
- `APPEND`. Режим добавления информации в файл. Новая информация будет размещена в конце файла, за последней записью.

Номер файла предваряется необязательным знаком `#`, после которого следует целое число от 1 до 255.

Запись в файл

Рассмотрим пример записи в файл.

```
OPEN "capitals.dat" FOR OUTPUT AS #1
FOR X=1 TO 5
  INPUT "ВВЕДИТЕ НАЗВАНИЕ СТОЛИЦЫ"; F$
  PRINT #1, F$
NEXT X
CLOSE #1
END
```

В результате работы программы мы получим (полужирным шрифтом выделены данные, вводимые пользователем с клавиатуры):

```
ВВЕДИТЕ НАЗВАНИЕ СТОЛИЦЫ? МОСКВА
ВВЕДИТЕ НАЗВАНИЕ СТОЛИЦЫ? САНКТ-ПЕТЕРБУРГ
ВВЕДИТЕ НАЗВАНИЕ СТОЛИЦЫ? ТАЛЛИН
ВВЕДИТЕ НАЗВАНИЕ СТОЛИЦЫ? РИГА
ВВЕДИТЕ НАЗВАНИЕ СТОЛИЦЫ? ВИЛЬНЮС
```

При этом на диске в текущем каталоге образуется файл, содержащий пять строковых значений. Имя файла будет capitals.dat.

Удобнее всего при работе с файлами сделать текущим каталог, где эти файлы содержатся или будут содержаться, а при обращении к ним указывать только их имена.

В качестве параметра *имя_файла* можно использовать переменную текстового типа. Это позволяет вводить имя файла с клавиатуры и является универсальным способом работы с файлами. Например:

```
INPUT " Введите имя файла "; FileName$
OPEN FileName$ FOR OUTPUT AS #1
```

После ключевого слова AS указывается номер файла. Больше открыть файл с таким номером в данной программе нельзя. Для каждого последующего файла должен быть указан свой собственный неповторимый номер в пределах от 1 до 255. Сколько всего может быть открыто файлов, зависит от файла конфигурации вашего компьютера config.sys, в котором число одновременно открытых файлов определяется командой FILES.

Предупреждение

Во избежание неприятностей не открывайте одновременно слишком много файлов. Рекомендуется по окончании работы с файлом сразу закрыть его.

Представленный далее пример демонстрирует программу создания небольшой базы данных автомобилей, предназначенных для продажи. О каждом автомобиле заносится информация о его марке, номере, цвете, годе производства и продажной цене.

```
' Программа создания файла данных об автомобилях
CLS
INPUT " Введите имя файла"; F$
' Открытие файла
OPEN F$ FOR OUTPUT AS #1
DO
  INPUT "Марка автомобиля? (Для окончания работы введите
        QWE.); M$
  IF UCASE$(M$)="QWE" OR UCASE$(M$)="ЙЦУ" THEN 1
  INPUT "Номер автомобиля?"; N$
  INPUT "Цвет автомобиля?"; C$
  INPUT "Год производства автомобиля?"; G$
  INPUT "Продажная цена автомобиля?"; S$
  WRITE #1, M$, N$, C$, G$, S$
LOOP
' Закрытие файла
1 : CLOSE #1
PRINT "Файл сформирован"
```

Программа действует следующим образом. Запрашивает имя файла, открывает его для записи, запрашивает информацию, записывает ее в файл до введения пользователем сочетания букв "QWE" или "ЙЦУ" (эти символы расположены на одних клавишах, вследствие чего пользователь может случайно набрать как одну комбинацию, так и другую). Запись в файл может производиться операторами

```
PRINT # номер_файла, выражение
```

или

```
WRITE # номер_файла, выражение
```

Результат работы этих операторов одинаков.

Для закрытия файлов применяется оператор

```
CLOSE # номер_файла
```

Предупреждение

Если номер файла в операторе CLOSE указан, то будет закрыт именно этот, вполне определенный файл. Если же номер не указан, будут закрыты все открытые файлы.

Наберите эту программу с именем создаваемого файла avto.dat.

Занесите сведения о пяти автомобилях: двух "Москвичах" и трех "Волгах". После чего просмотрите созданный вами файл в редакторе Бейсика. Для этого надо в меню **Файл** выбрать команду **Открыть**, затем в появившемся диалоговом окне ввести имя файла avto.dat и нажать клавишу <Enter>. Вы увидите, что данные в файле записаны в кавычках, через запятую, т. е. все они текстового типа.

Чтение из файла

Чтение из файла производится аналогично записи, но — вместо режима OUTPUT используется режим INPUT. Будьте внимательны! Прочитаем занесенные нами данные из файла avto.dat.

```
' Программа чтения файла данных об автомобилях
CLS
INPUT " Введите имя файла"; F$
' Открытие файла
OPEN F$ FOR INPUT AS #1
PRINT "База данных автомобилей на 17 декабря 2000 года"
I=1
DO
  PRINT "Вывести данные об"; I; "автомобиле?"
  INPUT "Для окончания введите QWE, для продолжения —
        <Enter>"; M$
  IF UCASE$(M$)="QWE" OR UCASE$(M$)="ЙЦУ" THEN 1
  INPUT #1, M$, N$, C$, G$, S$
  PRINT M$, N$, C$, G$, S$
  I=I+1
LOOP UNTIL EOF(1)
```

```
' Закрытие файла
1 : CLOSE #1
PRINT "Файл закрыт"
```

Обратите внимание на оператор `LOOP UNTIL EOF(1)`. Он означает, что считывание ведется до тех пор, пока не будет обнаружен символ конца файла (end of file), а в скобках указан номер открытого файла.

Выведите с помощью этой программы данные о первых трех автомобилях.

Изменения данных в файле

Для изменения какой-либо записи, удаления старых или добавления новых данных в последовательном файле необходимо открыть два файла: подлежащий изменению и новый, в котором создается обновленная версия исходного файла. Старый файл в дальнейшем можно удалить. Приведенная ниже программа в файле avto.dat изменяет "МОСКВИЧ" на "МЕРСЕДЕС". В первых строках открываются исходный файл avto.dat и новый файл avto2.dat, сначала пустой. Очередная запись считывается из файла avto.dat и, при условии, что это не "МОСКВИЧ", переписывается без изменения в новый файл. Если же встречается значение "МОСКВИЧ", то оно заменяется на "МЕРСЕДЕС" путем присваивания нового значения переменной `M$`. В следующей строке данное значение попадает в выходной файл. После того как весь входной файл просмотрен, оба файла закрываются.

```
OPEN "avto.dat" FOR INPUT AS #1
OPEN "avto2.dat" FOR OUTPUT AS #2
FOR I=1 TO 5
  INPUT #1, M$, N$, C$, G$, S$
  IF UCASE$(M$)="МОСКВИЧ" THEN M$="МЕРСЕДЕС"
  PRINT #2, M$, N$, C$, G$, S$
NEXT I
CLOSE 1, 2
KILL "avto.dat"
NAME "avto2.dat" AS "avto.dat"
END
```

Заключительный этап — удаление исходного и переименование нового файла, которому придается прежнее имя, что обеспечивает и в дальнейшем наличие на дискете файла avto.dat.

Добавление данных в файл

Указание `FOR APPEND` в операторе `OPEN` подготавливает файл для вывода данных и смещает указатель на конец файла. Последующие операторы приписывают новую информацию к уже имеющейся. В предложенной далее программе в файл данных об автомобилях добавляются сведения о двух новых поступлениях.

```
' Программа создания файла данных об автомобилях
CLS
INPUT " Введите имя файла"; F$
' Открытие файла
OPEN F$ FOR APPEND AS #1
DO
  INPUT "Марка автомобиля? (Для окончания работы введите
        QWE.); M$
  IF UCASE$(M$)="QWE" OR UCASE$(M$)="ЙЦУ" THEN 1
  INPUT "Номер автомобиля?"; N$
  INPUT "Цвет автомобиля?"; C$
  INPUT "Год производства автомобиля?"; G$
  INPUT "Продажная цена автомобиля?"; S$
  WRITE #1, M$, N$, C$, G$, S$
LOOP
' Закрытие файла
1 : CLOSE #1
PRINT "Файл дополнен"
```

Выполните упражнения.

426. Школе необходим последовательный файл для учета выпускников.

- Создайте последовательный файл для канцелярии по учету выпускников. Храните в нем фамилию, имя, год выпуска, любимый вид спорта и нынешний род занятий выпускника. Для образца составьте файл на десять человек.

- Воспользуйтесь этим файлом и напечатайте приглашения на очередной домашний матч "Зенита" тем выпускникам, которые назвали футбол своим любимым видом спорта.

427. Компьютерная фирма ведет файл со сведениями о двадцати своих сотрудниках.

- Создайте последовательный файл, содержащий имя и адрес каждого сотрудника (с указанием улицы, дома, квартиры и почтового индекса).
- По содержимому файла напечатайте почтовые адреса для рассылки чеков еженедельной заработной платы.

428. Гидрометцентр ведет статистику выпадения снега по регионам, для каждого из которых заведен последовательный файл. Во всех файлах присутствуют три элемента данных: имя метеоролога, название региона, количество выпавшего за зиму снега в мм.

- Напишите программу ввода данных; заполните файлы для трех регионов.
- Просмотрите все три файла и подсчитайте средний уровень снежных осадков по трем областям. Результат выведите на экран.

429. Налоговая инспекция поощряет налогоплательщиков, вносящих подоходный налог до истечения апрельского контрольного срока, делая им скидку.

- Создайте файл, в котором содержались бы имена, сведения о сроках уплаты и размере налога для каждого налогоплательщика (ограничьтесь группой из шести человек).
- Пусть ваша программа читает файл и делает скидку в 10% для тех, кто уплатил налог досрочно, а также выводит на экран их имена и размер скидки в рублях.

430. Фабрика игрушек ведет учет фирм розничной торговли, сбывающих ее продукцию. Файл контрагентов содержит названия этих фирм, сведения об их местоположении и индекс кредитоспособности: низкая или высокая.

- Напишите программу, которая создала бы последовательный файл контрагентов.

- Напишите программу, которая создала бы два последовательных файла с именами good.dat и bad.dat соответственно для фирм с высокой и низкой кредитоспособностью.
- Пусть ваша программа спрашивает у бухгалтера, какой из двух списков ему представить, а затем выдает названия фирм и их местоположение из соответствующего файла.

431. Предположим, адвокат Михаил Бурщевский с помощью компьютера ведет учет своих клиентов и их дел (табл. 1.4).

- Напишите программу, которая позволяла бы ему вводить в последовательный файл следующие сведения: имя клиента, обвинение, исход дела.
- Мицкевич "из огня попадает в полымя". Напишите программу, которая заменяла бы неопределенное решение суда на "Проиграно".
- Напечатайте обновленный файл.

Таблица 1.4. Исходные данные задачи

Имя клиента	Обвинение	Исход дела
Сердюков	Клевета	Выиграно
Прохоров	Оскорбление	Проиграно
Мицкевич	Поджог	?????
Максимова	Взлом	Выиграно
Лерман	Взятка	Проиграно

432. Хоккейные команды "Черные ястребы" и "Красные крылья" хранят в последовательных файлах имена всех своих двенадцати нападающих, число заброшенных ими шайб, сделанных голевых передач и заработанное штрафное время.

- Создайте файлы black.dat и red.dat, содержащие информацию о каждой из двух команд.
- Ваша программа по данным, извлеченным из этих файлов, должна создавать новый файл allstars.dat, в котором

содержались бы имя, команда и сумма очков (голы и передачи) для шести лучших игроков обеих команд. Пусть имена и показатели результативности хоккеистов выводятся на экран.

433. Имена и адреса всех, кто обращается за информацией в фирму, попадают в список рекламной рассылки.

- Создайте основной файл master.dat из десяти записей в качестве списка рассылки и меньший файл family.dat из пяти записей для вновь обратившихся с запросами в фирму. Добавьте данные из второго файла в конец первого.
- Напишите программу, которая случайным образом выбирала бы из основного файла одну запись и посылала бы адресату письмо с уведомлением о выигрыше приза.

Инспектор колледжа ведет файл академических занятий студентов.

- Создайте последовательный файл и заполните его фамилиями, названиями академических курсов и оценочным коэффициентом студентов. Воспользуйтесь данными, перечисленными в табл. 1.5.

Таблица 1.5. Исходные данные

Фамилия студента	Курс	Оценочный коэффициент
Югов	Программирование	78
Северов	Японский язык	91
Западов	Психология	56
Востоков	Психология	45
Зюйдов	Корейский язык	89
Вестов	Программирование	66
Полюсов	Психология	90

- Выберите "умных" студентов, т. е. тех, кто имеет оценку выше 88, и запишите сведения о них в файл best.dat.

Пусть программа помогает инспектору формировать на основе этого файла группы углубленного обучения. По названию курса она должна выдавать список "умных" студентов, зачисленных в такую группу.

434. Дешифровальщик. Возьмите в качестве исходного файла последовательного доступа страницу текста вашего любимого писателя. Проведите так называемый частотный анализ, т. е. установите, сколько раз каждая буква алфавита встречается в тексте.

Для частотного анализа рекомендуется создать файл, содержащий буквы русского алфавита.

Затем попросите своего друга зашифровать какой-нибудь текст так, чтобы каждая буква была заменена какой-нибудь другой или числом (это, конечно, тоже лучше сделать программно). Используя частотный анализ, попробуйте расшифровать текст.

435. Попробуйте сделать процесс дешифрации максимально автоматическим — может быть у вас получится эвристический анализ! ☺

Задания повышенной трудности, интегрированные, азартные (с примерами выполнения)

Если вы добрались до сих слов, дорогой читатель, то дальше, полагаю, вы сможете идти один, гордо неся знамя российского программирования над миром (ведь только в США 80% разработчиков программного обеспечения и математиков — наши соотечественники).

Напоследок я хочу предложить вам ряд разнообразных задач, поломав голову над которыми вы приобретете неоценимую закалку и опыт. Они будут без решений. Кроме того, подобные задания всегда хочется видеть не просто работающими, но и оформленными, и снабженные такими "FOOL PROOF" (защитой от дурака, т. е. выдерживающими случайные нажатия не тех клавиш не-

опытными пользователями), чтобы с ними было приятно работать.

Угадайка (математика и программирование)

Компьютер случайным образом загадывает число от 1 до 100. Задача пользователя — за минимальное количество попыток, но не более шести, угадать это число, предлагая компьютеру свои варианты, в ответ на которые программа указывает номер попытки, и сообщение о том, больше загаданное число или меньше введенного с клавиатуры. При правильной стратегии (делении интервала чисел пополам) наверняка угадать число можно за 7 попыток (2^7=128). Именно поэтому человеку дается только шесть попыток, чтобы у компьютера тоже был шанс. Игра состоит из пяти партий, и, как показывает практика, человек не всегда побеждает!

Анаграммы (русский язык и программирование)

Игра со словами. Анаграмма — это слово, в котором перепутаны буквы, например, "ШАДОЛЬ" — это "ЛОШАДЬ", а "ТИВОНКР" — это "ВТОРНИК". Программа рассчитана на двух игроков, соревнующихся друг с другом. В качестве слов используются только имена существительные, нарицательные, в единственном числе, в именительном падеже. Первый игрок вводит с клавиатуры слово длиной не менее пяти, но не более восьми букв (постарайтесь сделать так, чтобы во время ввода на экране вместо букв отображались символы *). Затем компьютер определяет длину введенного слова, разбирает его на отдельные символы, заносит их в массив, откуда случайным образом выводит на экран. Соперник в течение 2-х минут (время определяет компьютер) должен определить это слово. В случае правильного ответа (который он вводит с клавиатуры) он получает 1 очко, и сам загадывает слово первому игроку. Игра идет до 5 очков. За неправильный ответ очки не начисляются. Безусловно, приветствуется дружествен-

ный интерфейс: запрос и обращение к игрокам по именам, вывод на экран правил игры, графическое и звуковое оформление.

Стрельба из пушки (физика, математика и программирование)

Вариант 1

В замке в заточении томится прекрасная принцесса. В стене замка имеется небольшое окошко-бойница. У нас есть пушка. Необходимо передать принцессе план побега, выстрелив им из пушки (и, естественно, попав). Дается всего три попытки. Исходными данными являются угол наклона ствола пушки по отношению к горизонту и начальная скорость полета ядра. Они запрашиваются с клавиатуры. Предусмотреть примитивное изображение пушки (рис. 1.75), поворот ствола в зависимости от введенного угла, вычерчивание траектории полета ядра, реакцию замка на попадание в стену ядра, вывод на экран текста плана побега в случае правильного попадания.

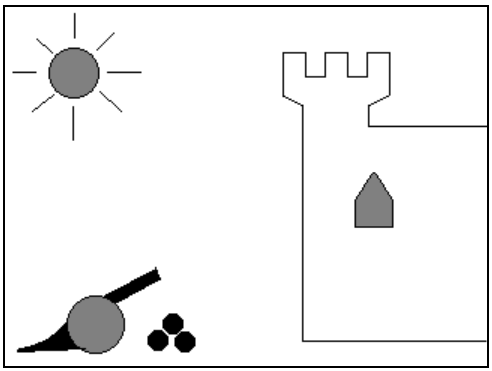


Рис. 1.75. Стрельба из пушки: вариант 1

Вариант 2

Дуэль на пушках через каменную стену — кто кого. Пушки теперь есть у двух игроков. Они небольшого размера. Стреляют по очереди, также задавая угол и скорость полета ядра в пределах:

угол от 10 до 90°, начальная скорость от 30 до 100 м/с. Попадание засчитывается, если расстояние от центра ядра до центра колеса пушки не более 20 экранных точек. В таком случае противник проигрывает. Высота стены каждый раз выбирается случайным образом в пределах от 50 до 200 экранных точек (рис. 1.76). Игра идет до трех побед.

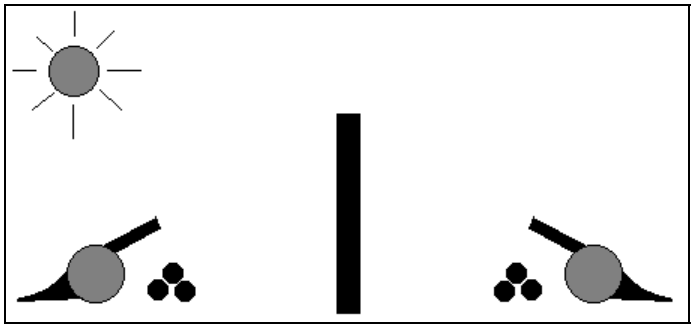


Рис. 1.76. Стрельба из пушки: вариант 2

Царь-пушка (математика, физика, экономика, история, русский язык и программирование)

Выполните расчет рентабельности изготовления ядер для "Царь-пушки".

Царь выделил 10 млн рублей на изготовление наиболее рентабельных ядер для Царь-пушки. Необходимо написать программу, рассчитывающую рентабельность изготовления ядер для 17 различных материалов, сэкономив таким образом для государственной казны значительные средства, которые могли бы быть затрачены на проверку рентабельности в полевых условиях с использованием натуральных материалов.

Исходные данные таковы. Отпущено 10 млн руб. Калибр Царь-пушки, фамилию мастера и год изготовления выясните дома. Взяв из табл. 1.6 исходные данные, создайте файл pushka.dat.

Таблица 1.6. Исходные данные для задания "Царь-пушка"

№	Материал	Плотность г/см^3	Себестоимость руб./кг	Коэффициент убойной силы
1	Платина	21,5	3636,36	1000,0
2	Золото	19,3	3090,91	950,0
3	Свинец	11,3	178,18	900,0
4	Серебро	10,5	454,55	800,0
5	Медь	9	81,82	300,0
6	Никель	8,9	92,73	340,0
7	Железо	7,9	29,09	500,0
8	Олово	7,3	67,27	550,0
9	Титан	4,5	203,64	450,0
10	Алюминий	2,7	21,82	450,0
11	Глина	2,2	3,64	0,1
12	Каучук	0,9	25,45	130,0
13	Цинк	7,1	60,00	610,0
14	Графит	2,5	9,09	2,5
15	Латунь	8,24	73,09	230,0
16	Мельхиор	5,42	84,00	200,0
17	Чугун	7,63	28,09	440,0

Здесь необходимо перевести плотности в систему СИ. Далее вычислить (все вычисления производить в системе СИ):

- объем ядра (ядро идеально круглое);
- массу ядра (M);
- начальную скорость (по формуле $V_n = \frac{3000}{\sqrt{M}}$);
- цену одного ядра (C);
- количество ядер (N);
- дальность полета (S), если выстрел производится под углом 45°;

- убойную силу ($US = \frac{M \times K \times V_n}{S}$, где K — коэффициент убойной силы);
- рентабельность ($R = \frac{N \times US}{C}$).

Напишите программу, которая поможет:

- найти пять наиболее рентабельных материалов, их рентабельности занести в массив и упорядочить по возрастанию;
- из самой большой рентабельности взять вторую цифру;
- из следующей по убыванию — число, составленное из первых двух цифр, поделить на два;
- в очередной по убыванию — сложить цифры целой части и прибавить число, которое получится, если число, составленное из двух первых цифр пятой рентабельности, поделить пополам;
- из четвертой рентабельности взять разность второй и третьей цифр.

Полученные четыре значения соответствуют порядковым номерам букв русского алфавита, в котором отсутствует буква "е". Необходимо определить эти буквы и составить из них четырехбуквенное слово, которое является первым результатом выполнения задания. Второе слово получаем следующим образом: из фамилии мастера берем три последние буквы, добавляем букву, чей порядковый номер — вторая цифра года создания Царь-пушки и букву, чей порядковый номер получается делением на 2 числа, образуемого второй и четвертой цифрами года создания Царь-пушки. Составляем пятибуквенное слово.

Кинотеатр "Кристалл-Палас" (математика, экономика и программирование)

И еще одно очень объемное задание. Представьте себе, что вас приглашают в кинотеатр "Кристалл-Палас", расположенный на Невском проспекте, на должность директора. Когда вы пытаетесь

узнать, сколько же вам будут платить, то говорят, что сейчас заодно и проверят ваши деловые способности, и предлагают написать программу, используя следующие исходные данные. В кинотеатре "Кристалл-Палас" три зала: "Красный", "Зеленый" и "Синий". Залы одинаковы и представляют собой квадраты 11×11 кресел. В результате маркетинговых исследований были установлены показатели по заполнению зала в среднем (рис. 1.77).

8	9	7	9	9	0	8	9	7	2	5
1	5	8	9	3	9	1	1	0	7	0
5	8	8	4	0	4	6	0	0	4	0
4	1	2	3	7	3	1	7	9	7	3
4	8	9	8	3	6	5	0	3	7	3
6	8	1	6	8	1	4	3	5	5	7
3	1	0	7	5	2	6	3	9	6	5
1	1	6	7	9	6	3	1	4	3	3
8	6	3	6	9	3	1	2	5	3	6
9	6	5	9	5	2	2	0	1	2	1
7	3	7	1	2	6	9	2	5	7	3

Рис. 1.77. Рассадка зала

Первым делом, чтобы облегчить себе жизнь, создадим файл исходных данных, в котором будет содержаться массив 11×11, элементами которого будут цифры из таблицы. Здесь цифры 1 и 2 будут обозначать мужчин, 3 и 4 — женщин, 5 и 6 — детей, 7 и 8 — тех, кто имеет право на льготные билеты, 0 — пустые места и 9 — знакомые и родственники служащих кинотеатра, которые ходят в кино бесплатно. К исходным данным также относится курс доллара на сегодняшний день. Цены на билеты в зависимости от времени сеанса и принадлежности к той или иной категории зрителей приведены в табл. 1.7, а затраты от общей прибыли — в табл. 1.8.

Таблица 1.7. Цены билетов в кинотеатре "Кристалл-Палас"

Категория зрителей	Цена билетов (руб.)		
	Утро (1 сеанс)	День (3 сеанс)	Вечер (2 сеанс)
Мужчины	70	100	200
Женщины	50	70	150
Дети	30	50	100
Льготники	10	20	50

Таблица 1.8. Затраты из общей прибыли за месяц

Затраты	%	Затраты	%
Амортизация	12	Охранные структуры	25
Налоги	22	Новые фильмы	22
Зарплата сотрудникам	16	Директор	3

Необходимо вычислить прибыль кинотеатра за день, за месяц (считая, что в месяце 30 дней), расходы по статьям в рублях и долларах. Вывести эти данные на экран, выделив зарплату директора цветом.

Тараканьи бега (математика, дизайн и программирование)

Конечно, сейчас уже далеко не все из вас смотрели фильм "Бег", поставленный по роману М. Булгакова. В частности, в нем есть сюжет о том, как наши эмигранты первой волны зарабатывали себе в Париже на жизнь. Одним из источников их доходов были тараканьи бега. Попробуем реализовать их компьютерную версию.

- Часть 1. Красочная заставка.
- Часть 2. Представление кличек участвующих тараканов, запрос количества делающих ставки и их фамилий, выдача всем кредита в 200 единиц, прием ставок (не более 100, но не менее 30).

Рис. 1.78. Игровое поле для "тараканьих бегов"

- Часть 3. Оформление игрового поля примерно такое, как на рис. 1.78.
- Часть 4. Непосредственно забег. Скорость каждого таракана за один такт — случайное целое число от 1 до 5. Забег заканчивается, как только какой-либо таракан коснулся "головой" финишной черты. Если никто из сделавших ставки не выиграл, то деньги переходят в доход казино. В случае выигрыша игрока, ему начисляется двойная ставка. Игра идет до тех пор, пока кто-либо не разорится или кто-либо не наберет сумму в размере двойного кредита.
- Часть 5. После каждого забега выводится информационное сообщение о финансовом положении участников и статистике побед тараканов.

Рекомендуется оформить забег музыкально.

Тесты (психология, русский язык и программирование)

Для этого задания вы можете взять любой тест из многочисленных выпускаемых сейчас журналов. Создайте файл исходных

данных — вопросов и вариантов ответов. Напишите программу, знакомящуюся с пользователем, сообщающую ему цель теста и инструкцию по работе с программой, проводящую тест, анализирующую результаты и выводящую итоги, взятые из другого файла данных.

Такая программа может использоваться как для психологических тестов, так и для проверки знаний в какой-либо области.

Если с одним тестом у вас получилось, то можно написать целую гибкую систему тестирования, где пользователю предлагается выбор из нескольких тестов, а также непосредственное создание тестов самому.

Итак, вы уже имеете некий багаж знаний, который никогда не бывает лишним, вы уже пользуетесь уважением у друзей и преподавателей, вы получаете только пятерки по программированию, но впереди — моря и океаны новых знаний. Ничего не бойтесь, и все у вас получится!

Удачи!

Примеры готовых и почти готовых ☺ программ

Эти примеры даны мною, конечно, не только и не столько для того, чтобы вы насладились результатами чужого труда, но затем, чтобы вы в них разобрались, улучшили и сделали бы новые, более совершенные, интересные и поучительные программы!

Почти все эти программы были написаны моими учениками — семи- и восьмиклассниками.

Стрельба из пушки

Пушка слева имеет три попытки для того, чтобы поразить случайно располагаемую на экране цель. Пользователь задает угол наклона ствола и начальную скорость.

```
SCREEN 12
RANDOMIZE TIMER
WINDOW (0, 480)-(640, 0)
```

```
FOR i = 1 TO 300
x = INT(RND(1) * 640)
y = INT(RND(1) * 400) + 80
PSET (x, y), 15
NEXT i

LINE (0, 0)-(640, 80), 10, BF
LINE (0, 0)-(640, 80), 10, BF
CIRCLE (60, 91), 10, 4: PAINT (60, 91), 6, 4
LINE (40, 80)-(80, 100), 15

x1 = INT(RND(1) * 300) + 300
y1 = INT(RND(1) * 250) + 100
CIRCLE (x1, y1), 10, 14: PAINT (x1, y1), 9, 14


FOR j = 1 TO 3
INPUT "angle="; a
IF a = 0 THEN END
INPUT "speed="; v
IF v = 0 THEN END

FOR t = 0 TO 20 STEP .01
x = 60 + v * COS(a * 3.14 / 180) * t
y = 91 + v * SIN(a * 3.14 / 180) * t - 9.81 * t * t / 2
IF y <= 80 OR y >= 480 OR x >= 640 OR x <= 0 THEN GOSUB boom:
t = 21 ELSE PSET (x, y), 15
IF ABS(x - x1) <= 5 AND ABS(y - y1) <= 5 THEN GOSUB boom: END
'LOCATE 2, 2: PRINT "x="; x, "y="; y

NEXT t
NEXT j
END

boom:
FOR i = 1 TO 50
x2 = x - 20 + INT(RND(1) * 50)
y2 = y - 20 + INT(RND(1) * 50)
LINE (x, y)-(x2, y2), 4
NEXT i
RETURN
```

Танчики

```
Дуэль двух танков. Кто кого?
'TANKWARS
'
TIMER ON
ts = TIMER
FOR s = 1 TO 1000 STEP .1
ss = SIN(s)
NEXT s
ts = TIMER — ts
speed = CINT((ts / .269531) * 10000) / 10000
speed = speed * .6
clr = 63
SCREEN 9
COLOR 1, 15
PAINT (320, 240), 15, 15
INPUT "player 1 name"; plr1$
INPUT "player 2 name"; plr2$
snd:
i$ = INKEY$
IF i$ <> "" THEN IF ASC(i$) = 27 THEN SYSTEM
LOCATE 3, 1: PRINT "sound on?"
IF LCASE$(i$) = "y" THEN snd = 1: GOTO begin
IF LCASE$(i$) = "n" THEN snd = 0: GOTO begin
GOTO snd
begin:
COLOR 15
SCREEN 12
CLS
en1 = 20
en2 = 20
DIM expl(5, 8)
DIM expl2(2, 8)
tank = 0
c = 1
CONST pi = 3.141593
bom$ = "hl4d3r4e "
bit$ = "g2f2e2h2"
```

```
boom$ = "c4bl6e2u2r2e2f2r2d2f2g2d2l2g2h2l2u2h2br6p4,4c14bl3
eurefrdfgdlghluhbr3s4p14,14"
arrow$ = "ta0c1h5u2r2u10r6d10r2d2g5bu4p1,1"
tank$ = "l7h2u2e2r14f2d2g2l11bu6u3er6fd3rg2df2l10e2uh2"
RANDOMIZE TIMER
by2 = 240
y = 240
lscape:
yr = 80 * RND(1)
xr = 6 * RND(1)
FOR la = 0 TO 6.283 STEP .1
by = y + SIN(la) * yr
x = x + xr
FOR p = 1 TO 5
LINE (x, by + p)-(x2, by2 + p), 2
NEXT p
sy = (480 — by) * RND(1)
PSET (x, by + sy), 2
by2 = by: x2 = x
IF x > 640 THEN GOTO fin
NEXT la
GOTO lscape
fin:
FOR d = 0 TO 30 STEP speed
PALETTE 8, INT(d)
PAINT (320, 470), 8, 2
NEXT
FOR skel = 1 TO 5
sk:
sx = INT(640 * RND(1))
sy = INT(100 * RND(1)) + 300
IF POINT(sx, sy) <> 8 THEN GOTO sk
PSET (sx, sy), 7: DRAW "c7dfr2dr2u6r2d8r2u8r2d8r2u6r2d3m+9,+2"
PSET (sx, sy), 7: DRAW "c7u2er2f2d4g4br15h4"
NEXT
LINE (1, 440)-(320, 460), 0, BF
LINE (0, 439)-(321, 461), 2, B
LINE (400, 420)-(500, 430), 6, BF
LINE (400, 433)-(500, 445), 6, BF
```

```
LINE (399, 419)-(501, 431), 2, B
LINE (399, 433)-(501, 445), 2, B
LINE (350 + en1 * 5, 420)-(450, 430), 0, BF
LINE (350 + en2 * 5, 433)-(450, 445), 0, BF
COLOR 2
LOCATE 27, 65: PRINT plr1$
LOCATE 28, 65: PRINT plr2$
tank2:
tx = INT(620 * RND(1)) + 10
FOR ty = 200 TO 480
IF POINT(tx, ty) > 0 THEN GOTO start
NEXT ty
start:
IF tank = 1 AND plr = 0 THEN IF SQR(((tx - qx) * (tx - qx)) +
((ty - qy) * (ty - qy))) < 200 THEN GOTO tank2 ELSE plr = 1
drw:
FOR fobj = ty - 20 TO 480
ft% = POINT(tx + 5 + (12 * SIN(ang + 1.570796)), fobj)
IF ft% > 0 THEN GOTO bk
NEXT fobj
bk:
FOR bobj = ty - 20 TO 480
bk% = POINT(tx - 5 - (12 * SIN(ang + 1.570796)), bobj)
IF bk% > 0 THEN GOTO calc
NEXT bobj
calc:
ang = -ATN((fobj - bobj) / ((tx + 12) - (tx - 12)))
ta$ = "ta" + STR$(INT(ang / (pi / 180)))
ty = ((bobj + fobj) / 2)
tch:
IF POINT(tx, ty) > 0 THEN ty = ty - 2: GOTO tch
IF POINT(tx, ty + 2) = 0 THEN ty = ty + 2: GOTO tch
IF tank = 1 THEN x = tx: y = ty: a$ = ta$
IF tank = 0 AND plr = 0 THEN tank = 1: qx = tx: qy = ty: qa$ =
ta$: GOTO tank2
IF tank = 0 AND plr > 0 THEN tank = 1: qx = tx: qy = ty: qa$ =
ta$: tx = x: ty = y: GOTO drw:
PSET (qx, qy)
DRAW "c11" + STR$(c) + qa$ + tank$
```

```
PSET (x, y)
DRAW "c12" + STR$(c) + a$ + tank$
game:
IF plr = 1 THEN arx = qx: ary = qy — 20: ang$ = qa$: nary =
y — 20: narx = x: plr$ = plr1$
IF plr = 2 THEN arx = x: ary = y — 20: ang$ = a$: nary =
qy — 20: narx = qx: plr$ = plr2$
by = ary + 20: bx = arx
COLOR 12
LOCATE 25, 7: PRINT "WIND"
LOCATE 27, 35: PRINT "POWER"
wind = ((RND(1) — .5) / 60) * speed
LINE (20, 400)-(120, 410), 0, BF
LINE (19, 399)-(121, 411), 10, B
FOR w = 1 TO wind * 5000 / speed STEP (wind * 3) * speed
LINE (70, 400)-(70 + w, 405), 12
LINE (70, 410)-(70 + w, 405), 12
LINE (70, 400)-(70, 410), 10
NEXT w
arrow:
FOR ac = 63 TO 1 STEP -speed
PALETTE 1, ac
LINE (narx — 5, nary)-(narx + 5, nary — 20), 0, BF
PRESET (arx, ary), 0
DRAW arrow$
LOCATE INT((30 / 480) * ary) — 3, INT((80 / 640) * arx)
COLOR 1
PRINT plr$
COLOR 0
LOCATE INT((30 / 480) * nary) — 3, INT((80 / 640) * narx)
PRINT plr2$
i$ = INKEY$
IF i$ <> "" THEN IF ASC(i$) = 27 THEN SYSTEM
IF i$ <> "" THEN GOTO cont
NEXT
GOTO arrow
cont:
PALETTE 1, 0
PSET (arx, ary), 0
```

```
DRAW arrow$
LINE (arx — 5, ary)-(arx + 5, ary — 20), 0, BF
LOCATE INT((30 / 480) * ary) — 3, INT((80 / 640) * arx)
COLOR 0
PRINT plr$
ang$ = RIGHT$(ang$, 3)
ang = VAL(ang$) * (pi / 180)
gx = bx + COS(-ang — pi / 2) * 10
gy = by + SIN(-ang — pi / 2) * 10
a = ang
COLOR 11
gun:
bx2 = bx: by2 = by
bx = gx + COS(-a) * 14
by = gy + SIN(-a) * 14
LINE (gx, gy)-(bx, by), 15
IF bx2 <> bx OR by2 <> by THEN LINE (gx, gy)-(bx2, by2), 0
i$ = INKEY$
IF i$ <> "" THEN IF ASC(i$) = 27 THEN SYSTEM
IF i$ = "8" THEN a = a + .1 * speed
IF i$ = "2" THEN a = a — .1 * speed
LOCATE 27, 1: PRINT "ANGLE:"; INT((a / (pi / 180)))
IF i$ = " " THEN bx = gx + COS(-a) * 12: by = gy + SIN(-a) *
12: LINE (gx, gy)-(bx2, by2), 0: GOTO shoot
IF LCASE$(i$) = "t" THEN GOTO tport
IF (a — ang) < 0 THEN a = a + .1 * speed
IF (a — ang) > pi THEN a = a — .1 * speed
GOTO gun
shoot:
i$ = INKEY$
IF i$ <> "" THEN IF ASC(i$) = 27 THEN SYSTEM
IF pwr > 320 OR i$ = " " THEN byv = (SIN(a) * pwr / 750) *
speed: bxv = (COS(a) * pwr / 750) * speed
IF pwr > 320 OR i$ = " " THEN IF snd = 1 THEN SOUND 200, 2
IF pwr > 320 OR i$ = " " THEN drag = bxv + wind: GOTO fire
pwr = pwr + .15 * speed
LINE (1, 440)-(pwr, 460), c, BF
c = 2
IF pwr > 100 THEN c = 12
```

```
IF pwr > 250 THEN c = 4
GOTO shoot:
fire:
bx2 = bx: by2 = by
bx = bx + bxv
bxv = bxv + wind / 100
IF bxv = drag THEN wind = 0
by = by — byv
byv = byv — (.0002 * speed)
IF bxv <> 0 AND byv <> 0 THEN ag = ATN(byv / bxv) + pi / 2
IF bxv < 0 THEN ag = ag + pi
ag2$ = ag$
ag$ = "ta" + STR$(INT((ag — (pi / 2)) / (pi / 180)))
PSET (bx, by), 13: DRAW ag$ + bom$
IF bx <> bx2 THEN PRESET (bx2, by2), 0: DRAW ag2$ + bom$
IF bx > 640 OR bx < 0 OR by > 420 OR POINT(bx + SIN(ag) * 4,
by + COS(ag) * 4) > 0 THEN GOTO expl
GOTO fire
expl:
i$ = INKEY$: IF i$ <> "" THEN GOTO expl
IF bx > 640 THEN bx = 639
IF bx < 0 THEN bx = 1
IF snd = 1 THEN SOUND 100, 2
FOR boom = 4 TO 8 STEP 2 * speed
PSET (bx, by + 10), 0
DRAW "s" + STR$(INT(boom)) + "ta" + STR$(INT(boom * 44)) +
boom$
FOR del = 1 TO 5000: NEXT del
CIRCLE (bx, by + 10), boom * 1.5, 0
PAINT (bx, by + 10), 0, 0
NEXT
pwr = 0
LINE (1, 440)-(320, 460), 0, BF
FOR shp = 1 TO 5
expl(1, shp) = bx
expl(2, shp) = by
expl(3, shp) = ((RND(1) — .5) / 2) * speed
expl(4, shp) = (-RND(1) — .25) * speed
expl(5, shp) = CINT(5 * RND(1)) + 7
NEXT shp
```

```
erk:
FOR xpl = 1 TO 5
expl(1, xpl) = expl(1, xpl) + expl(3, xpl)
expl(2, xpl) = expl(2, xpl) + expl(4, xpl)
expl(4, xpl) = expl(4, xpl) + .004 * speed
PSET (expl(1, xpl), expl(2, xpl)), expl(5, xpl)
DRAW bit$
PRESET (expl2(1, xpl), expl2(2, xpl)), 0
DRAW bit$
expl2(1, xpl) = expl(1, xpl)
expl2(2, xpl) = expl(2, xpl)
IF POINT(expl(1, xpl) + 4, expl(2, xpl) + 6) > 0 OR
expl(2, xpl) > 400 OR expl(1, xpl) > 640 OR expl(1, xpl) < 0
THEN expl(3, xpl) = 0: expl(4, xpl) = 0: expl(5, xpl) = 1
IF expl(5, 1) = 1 AND expl(5, 2) = 1 AND expl(5, 3) = 1 AND
expl(5, 4) = 1 AND expl(5, 5) = 1 THEN PRESET (expl(1, xpl),
expl(2, xpl)), 0: DRAW bit$: gy% = by: GOTO ground
NEXT
GOTO erk
ground:
FOR gx = bx - 25 TO bx + 25
FOR gy = by + 25 TO by - 100 STEP -1
ns:
gd% = POINT(gx, gy)
IF gd% > 0 AND gd% < 15 THEN GOTO fall
NEXT
NEXT
GOTO plr
fall:
FOR fall% = gy TO 420
PSET (gx, fall%), 0:
PSET (gx, fall% + 1), gd%: ld% = POINT(gx, fall% + 2):
IF ld% > 0 AND ld% < 15 THEN GOTO ns
NEXT fall%
plr:
dist1 = SQR(((qx - bx) * (qx - bx)) + ((qy - by) * (qy - by)))
dist2 = SQR(((x - bx) * (x - bx)) + ((y - by) * (y - by)))
IF dist1 / 100 THEN en1 = en1 - (1 / (dist1 / 100))
IF dist2 / 100 THEN en2 = en2 - (1 / (dist2 / 100))
IF en < 0 THEN en = 0
```

```
IF en2 < 0 THEN en2 = 0
LINE (400 + en1 * 5, 420)-(500, 430), 0, BF
LINE (400 + en2 * 5, 433)-(500, 445), 0, BF
LINE (399, 419)-(501, 431), 2, B
LINE (399, 433)-(501, 446), 2, B
IF en2 = 0 AND en1 = 0 THEN dr = 1: GOTO win
IF en1 <= 0 THEN dx = qx: dy = qy — 10: dang$ = qang$: winner$
= plr2$: loser$ = plr1$: GOTO dead
IF en2 <= 0 THEN dx = x: dy = y — 10: dang$ = ang$: winner$ =
plr1$: loser$ = plr2$: GOTO dead
GOTO xtport:
tport:
IF plr = 1 THEN PRESET (qx, qy): DRAW "c0" + qa$ + tank$: ELSE
PRESET (x, y): DRAW "c0" + a$ + tank$
LINE (gx, gy)-(bx, by), 0
px = INT(620 * RND(1)) + 10
IF snd = 1 THEN PLAY "l32o6c<g>f<<b>gc<e>>g"
FOR py = 160 TO 480
IF POINT(px, py + 5) > 0 THEN GOTO mov
NEXT
mov:
FOR cl = 1 TO 63 STEP 2
PALETTE 4, 64 — cl
CIRCLE (gx, gy — 5), 11 — cl2 / 6, 0
PAINT (gx, gy — 5), 0, 0
CIRCLE (gx, gy — 5), 11 — cl / 6, 4
PAINT (gx, gy — 5), 4, 4
PALETTE 4, cl
CIRCLE (px, py — 5), (cl / 6) + 1, 0
CIRCLE (px, py — 5), cl / 6, 4
PAINT (px, py — 5), 4, 4
CIRCLE (px, py — 5), cl / 6, 4
cl2 = cl
NEXT
CIRCLE (px, py — 5), (cl / 6), 0
PAINT (px, py — 4), 0, 0
PSET (gx, gy — 5), 0
PAINT (gx, gy — 5), 0, 0
IF plr = 1 THEN qx = px: qy = py: ELSE x = px: y = py
```

```
xtport:
PRESET (qx, qy)
DRAW "c0" + qa$ + tank$
PRESET (x, y)
DRAW "c0" + a$ + tank$
IF plr = 1 THEN plr = 2: tank = 0: tx = qx: ty = qy: GOTO drw
IF plr = 2 THEN plr = 1: tank = 0: tx = qx: ty = qy: GOTO drw
dead:
FOR f = 1 TO 3
FOR d = 1 TO 63 STEP 5 * speed
PALETTE 1, d
IF snd = 1 THEN SOUND 37 + 20 * d, 1
LINE (dx — 7, dy — 10)-(dx + 7, dy — 25), 0, BF
PSET (dx, dy + 10), 1
DRAW "ta" + dang$ + tank$
NEXT
NEXT
PRESET (dx, dy + 10), 0
DRAW "ta" + dang$ + tank$
FOR shp = 1 TO 8
expl(1, shp) = dx
expl(2, shp) = dy
expl(3, shp) = (RND(1) — .5) * 4
expl(4, shp) = (-RND(1) — 1) * 2
expl(5, shp) = 3
NEXT shp
cr = 64
IF snd = 0 THEN GOTO derk
FOR blw = 1 TO 5
SOUND 200, 1
SOUND 37, 1
NEXT
derk:
cr = cr — .5
PALETTE 3, INT(cr)
FOR xpl = 1 TO 8
expl(1, xpl) = expl(1, xpl) + expl(3, xpl)
expl(2, xpl) = expl(2, xpl) + expl(4, xpl)
expl(4, xpl) = expl(4, xpl) + .04
```

```
PSET (expl(1, xpl), expl(2, xpl)), 3
1 DRAW bit$
expl2(1, xpl) = expl(1, xpl)
expl2(2, xpl) = expl(2, xpl)
IF cr = 1 THEN GOTO win
NEXT
GOTO derk
win:
SLEEP 1
CLS
COLOR 15
FOR cl = 1 TO 63 STEP 2 * speed
PALETTE 2, cl
COLOR 2
IF dr = 0 THEN LOCATE 10, 30: PRINT winner$ + " wins the game.
Another game(y/n)?"
IF dr = 1 THEN LOCATE 10, 30: PRINT "round drawn: Another
game?(y/n)"
i$ = INKEY$
IF i$ <> "" THEN IF ASC(i$) = 27 THEN SYSTEM
IF LCASE$(i$) = "y" OR LCASE$(i$) = "n" THEN GOTO dec
NEXT
GOTO win
dec:
IF LCASE$(i$) = "y" THEN RUN
SYSTEM
```

Заставка "Звездные войны"

Экранная заставка.

```
CLS : SCREEN 12

RANDOMIZE TIMER
FOR i = 1 TO 1000
PSET (RND * 1000, RND * 1000), INT(RND * 15 + 1)
NEXT i

c = 1
FOR a = 1 TO 2 STEP .1
c = c + 1
```

```
FOR x = .01 TO 6.28 STEP .01
x1 = a * SIN(x) * 100 + 300
y = COS(x) * 100 + 250
PSET (x1, y), c

NEXT x
NEXT a

c = 1
FOR a = 1 TO 2 STEP .1
c = c + 1
FOR x = .01 TO 6.28 STEP .01
x1 = SIN(x) * 100 + 300
y = a * COS(x) * 100 + 250
PSET (x1, y), c
NEXT x
NEXT a


CIRCLE (300, 250), 200, 2
CIRCLE (300, 250), 210, 2
PAINT (91, 250), 2


m1:

c = INT(RND * 15) + 1
c$ = "<-- SPACE WARS -->"
COLOR c: LOCATE 2, 30: PRINT c$
COLOR c: LOCATE 5, 2: PRINT c$
COLOR c: LOCATE 5, 60: PRINT c$
COLOR c: LOCATE 26, 2: PRINT c$
COLOR c: LOCATE 26, 60: PRINT c$

FOR i = 0 TO 99 STEP .5
CIRCLE (300, 250), i, 3
PSET (RND * 1000, RND * 1000), INT(RND * 15 + 1)
IF INKEY$ <> "" THEN END
NEXT i

FOR i = 99 TO 0 STEP -.5
CIRCLE (300, 250), i, 0
```

```
PSET (RND * 1000, RND * 1000), INT(RND * 15)
IF INKEY$ <> "" THEN END
NEXT i
GOTO m1
```

Еще заставка. "Снег"

"Снег кружится, летает, летает..."

```
SCREEN 12
DIM x(1000), y(1000)
RANDOMIZE TIMER
n = 1000

FOR i = 1 TO n
x(i) = INT(RND(1) * 640)
NEXT i

FOR i = 1 TO n
y(i) = INT(RND(1) * 480)
NEXT i

FOR i = 1 TO n
PSET (x(i), y(i))
NEXT i

k = 1
WHILE k < 100000
i = INT(RND(1) * 1000)
PSET (x(i), y(i)), 0
i = INT(RND(1) * 1000)
PSET (x(i), y(i)), 15
k = k + 1
FOR m = 1 TO 500: NEXT m

WEND


'WHILE y(1) < 480
'FOR k = 1 TO n
'PSET (x(k), y(k)), 15
```

```
'FOR j = 1 TO 5000: NEXT j
'PSET (x(k), y(k)), 0
'NEXT k
'FOR k = 1 TO n
'y(k) = y(k) + RND(1) * 30
'NEXT k
'WEND
'NEXT i
```

А теперь заставка "Смайлики!

Создайте хорошее настроение!

```
DIM happy%(1 TO 300)
CLS : SCREEN 8
x1% = 30: y1% = 40: x2% = 100: y2% = 75: x3% = 230: y3% = 35
x4% = 30: y4% = 40: x5% = 100: y5% = 75: x6% = 230: y6% = 35
x7% = 30: y7% = 40: x8% = 100: y8% = 75: x9% = 230: y9% = 35
x10% = 30: y10% = 40: x11% = 100: y11% = 75: x12% = 230: y12%
= 35
x13% = 30: y13% = 40: x14% = 100: y14% = 75: x15% = 230: y15%
= 35
x16% = 30: y16% = 40: x17% = 100: y17% = 75: x18% = 230: y18%
= 35
CIRCLE (50, 50), 20, 14
PAINT (50, 50), 14
CIRCLE (44, 47), 4, 0
CIRCLE (56, 47), 4, 0
CIRCLE (50, 50), 15, 0, 3, .2
GET (x1%, y1%)-(70, 60), happy%
PUT (x2%, y2%), happy%
PUT (x3%, y3%), happy%
PUT (x4%, y4%), happy%
PUT (x5%, y5%), happy%
PUT (x6%, y6%), happy%
d! = 1 / 100
CIRCLE (300, 95), 150, 14
PAINT (300, 95), 14
CIRCLE (250, 70), 15, 0
PAINT (250, 70), 0
```

```
CIRCLE (350, 70), 15, 0
PAINT (350, 70), 0
CIRCLE (300, 95), 110, 0, 3, .1
DO
PUT (x1%, y1%), happy%, XOR
x1% = RND * 580
y1% = RND * 175
PUT (x1%, y1%), happy%
PUT (x2%, y2%), happy%, XOR
x2% = RND * 580
y2% = RND * 175
PUT (x2%, y2%), happy%
PUT (x3%, y3%), happy%, XOR
x3% = RND * 580
y3% = RND * 175
PUT (x3%, y3%), happy%
PUT (x4%, y4%), happy%, XOR
x4% = RND * 580
y4% = RND * 175
PUT (x4%, y4%), happy%
PUT (x5%, y5%), happy%, XOR
x5% = RND * 580
y5% = RND * 175
PUT (x5%, y5%), happy%
PUT (x6%, y6%), happy%, XOR
x6% = RND * 580
y6% = RND * 175
PUT (x6%, y6%), happy%
PUT (x7%, y7%), happy%, XOR
x7% = RND * 580
y7% = RND * 175
PUT (x7%, y7%), happy%
PUT (x8%, y8%), happy%, XOR
x8% = RND * 580
y8% = RND * 175
PUT (x8%, y8%), happy%
PUT (x9%, y9%), happy%, XOR
x9% = RND * 580
y9% = RND * 175
```

```
PUT (x9%, y9%), happy%
PUT (x10%, y10%), happy%, XOR
x10% = RND * 580
y10% = RND * 175
PUT (x10%, y10%), happy%
PUT (x11%, y11%), happy%, XOR
x11% = RND * 580
y11% = RND * 175
PUT (x11%, y11%), happy%
PUT (x12%, y12%), happy%, XOR
x12% = RND * 580
y12% = RND * 175
PUT (x12%, y12%), happy%
PUT (x13%, y13%), happy%, XOR
x13% = RND * 580
y13% = RND * 175
PUT (x13%, y13%), happy%
PUT (x14%, y14%), happy%, XOR
x14% = RND * 580
y14% = RND * 175
PUT (x14%, y14%), happy%
PUT (x15%, y15%), happy%, XOR
x15% = RND * 580
y15% = RND * 175
PUT (x15%, y15%), happy%
PUT (x16%, y16%), happy%, XOR
x16% = RND * 580
y16% = RND * 175
PUT (x16%, y16%), happy%
PUT (x17%, y17%), happy%, XOR
x17% = RND * 580
y17% = RND * 175
PUT (x17%, y17%), happy%
PUT (x18%, y18%), happy%, XOR
x18% = RND * 580
y18% = RND * 175
PUT (x18%, y18%), happy%
CurrentTimer! = TIMER
```

```
WHILE TIMER < (CurrentTimer! + d!)
WEND
LOOP WHILE INKEY$ = ""
```

Магическая заставка

Смотрите — не засмотритесь!

```
'looks impressive, sloppy, and symmetric all at the same time.
WOOHOO!

CLS
SCREEN 12
RANDOMIZE TIMER

pal = INT(RND * 5) + 1
IF pal = 2 THEN pal = 256
IF pal = 3 THEN pal = 257
IF pal = 4 THEN pal = 65536
IF pal = 5 THEN pal = 65537

FOR i = 1 TO 60 STEP 4
x = x + 1
PALETTE x, i * pal
NEXT
PALETTE 1, 0

size = 15
delay = 0
winX = 320
winY = 240
col1 = 1
col2 = 9
accel = .2
maxsp = 15
minsp = -maxsp
rr = 1
DIM x(size), y(size)
DIM xx(size), yy(size), nx(size), ny(size)
DIM col(size)
DIM r(size)
DIM t(size), v(size), u(size)
```

```
maxX = 640 — winX
maxY = 480 — winY
x = maxX
y = maxY
FOR i = 1 TO size
xx(i) = -RND * accel — accel
yy(i) = -RND * accel — accel
x = x — 400 / size
y = y — 400 / size
x(i) = x
y(i) = y
xx(i) = minsp
yy(i) = minsp
r = r + rr
r(i) = r
IF col = col1 THEN col = col2 ELSE col = col1
IF i < 16 THEN col(i) = i ELSE col(i) = 15
NEXT

DO
FOR i = 1 TO delay STEP 1: NEXT

FOR i = 1 TO size

IF i < size THEN
LINE (x(i), y(i))-(x(i + 1), y(i + 1)), col(i)
LINE (640 — x(i), 480 — y(i))-(640 — x(i + 1), 480 — y(i +
1)), col(i)
LINE (640 — x(i), y(i))-(640 — x(i + 1), y(i + 1)), col(i)
LINE (x(i), 480 — y(i))-(x(i + 1), 480 — y(i + 1)), col(i)

'LINE (x1, y1)-(x2, y2), 0
LINE (x(size — i), y(size — i))-(640 — x(i + 1), 480 —
y(i + 1)), col(i)
LINE (640 — x(size — i), y(size — i))-(x(i + 1), 480 —
y(i + 1)), col(i)
'x1 = x(size — 1): y1 = y(size — 1): x2 = 640 — x(i + 1):
y2 = 480 — y(i + 1)
END IF
t(i) = x(i): v(i) = y(i): u(i) = r(i)

NEXT
```

```
FOR i = 1 TO size
IF i = size THEN
   xx(i) = xx(i) + nx(i)
   yy(i) = yy(i) + ny(i)
   IF xx(i) > maxsp THEN xx(i) = maxsp
   IF xx(i) < minsp THEN xx(i) = minsp
   IF yy(i) > maxsp THEN yy(i) = maxsp
   IF yy(i) < minsp THEN yy(i) = minsp

   x(i) = x(i) + xx(i)
   y(i) = y(i) + yy(i)
   IF x(i) > maxX THEN nx(i) = -RND * accel — accel
   IF x(i) < winX THEN nx(i) = RND * accel + accel
   IF y(i) > maxY THEN ny(i) = -RND * accel — accel
   IF y(i) < winY THEN ny(i) = RND * accel + accel
END IF
IF i < size THEN
   x(i) = x(i + 1)
   y(i) = y(i + 1)
END IF

NEXT

LOOP UNTIL INKEY$ <> ""
```

Казино приглашает на тараканьи бега

Программа реализует алгоритм, предложенный в заданиях повышенной сложности этой книги. Но поработать там есть над чем!

```
'--------------- PROGRAM "ROAHDROM" -------------------

CLS
SCREEN 12
RANDOMIZE TIMER


'----------------------RULES--------------------------

'FOR S = 1 TO 5
'SOUND 25000, 3
```

```
'SOUND 400, 5
'NEXT S


'---------------------- ZASTAVKA ----------------------

LINE (50, 175)-(575, 450), 6, BF
CIRCLE (312, 175), 263, 1, 0, 3.14, .5
LINE (50, 175)-(575, 175), 1
PAINT (300, 100), 1
LINE (450, 375)-(500, 450), 8, BF
CIRCLE (475, 375), 50, 8, 0, 3.14, 2
PAINT (475, 370), 8
LINE (450, 375)-(500, 375), 0
CIRCLE (475, 375), 50, 0, 0, 3.14, 2
LINE (450, 375)-(500, 450), 0, B
LINE (475, 375)-(475, 450), 0

DRAW "BM 100,350 R 50 U100 L50 D100"
DRAW "BM 200,350 R 50 U100 L50 D100"
DRAW "BM 300,350 R 50 U100 L50 D100"
DRAW "BM 400,300 R 150 U100 L150 D100"

PAINT (120, 340), 3, 0
PAINT (220, 340), 3, 0
PAINT (320, 340), 3, 0
PAINT (420, 210), 3, 0
LOCATE 7, 34: PRINT "CASINO ";
LOCATE 9, 32: PRINT "ROACHDROM";


LOCATE 28, 10: PRINT "TO START THE GAME PRESS <6>"
WHILE INKEY$ <> "6": WEND
'PLAY "G2 D2 G8 A4"


RANDOMIZE TIMER


FOR X = 0 TO 640 STEP 10
FOR y = 0 TO 480 STEP 10
C = INT(RND(1) * 15) + 1
```

```
LINE (X, y)-(X + 10, y + 10), C, BF
LINE (X, y)-(X + 10, y + 10), 15, B
FOR i = 1 TO 1000: NEXT i
NEXT y

NEXT X

FOR y = 100 TO 400
LINE (100, y)-(500, y), 8
FOR i = 1 TO 1000: NEXT i
NEXT y
'LINE (100, 100)-(500, 400), 8, BF

'------------ AMOUNT AND NAMES OF PLAYERS ------------

LOCATE 10, 15: PRINT "WELCOME TO ROACHDROM TANG-TANG"
3 : LOCATE 12, 15: INPUT "HOW MANY PLAYERS WILL PLAY THE GAME
(1-5)"; N
IF N < 1 OR N > 5 THEN LOCATE 15, 15: PRINT "SORRY, IMPOSSIBLE
AMOUNT!": GOTO 3
DIM A$(N), J(N), B(N), C(N)
FOR Q = 1 TO N
J(Q) = 1000
LOCATE 17, 15: PRINT "WHAT IS NAME OF PLAYER "; Q; : INPUT
A$(Q)
NEXT Q

LOCATE 19, 10: PRINT "WAIT A MINUTE, PLEASE!";

'PLAY "MB  A2 B4 C3 D2 E1 F5 G7 G9 A4 "

k = 1   '--------------- ü ZABEGA -------------------

'-----------------

4 : CLS :     PAINT (34, 6), 8

'------------------------ FIELD ---------------

DRAW "BM 100,125 C0 U85 L50 D35 R50 "
PAINT (75, 60), 4, 0
LOCATE 4, 8: PRINT "START"
```

```
DRAW "BM 600,125 C0 U85 L60 D35 R60 "
PAINT (575, 60), 4, 0
LOCATE 4, 70: PRINT "FINISH"

LINE (100, 125)-(600, 275), 1, BF
DRAW "BM 100,155 C0 R500"
DRAW "BM 100,185 C0 R500"
DRAW "BM 100,215 C0 R500"
DRAW "BM 100,245 C0 R500"
LINE (100, 125)-(600, 275), 0, B
LINE (25, 300)-(625, 480), 7, BF

LOCATE 19, 34: PRINT "ZABEG ü "; k

y = 22
FOR i = 1 TO N
LOCATE y + i, 5: PRINT A$(i)
LOCATE y + i, 20: PRINT J(i)
NEXT i


LOCATE 21, 5: PRINT "PLAYER'S NAME"
LOCATE 21, 20: PRINT "ACCOUNT"
LOCATE 21, 35: PRINT "ROACH'S NUMBER"
LOCATE 21, 60: PRINT "BET"
LOCATE 3, 31: PRINT "**ROACHDROM TANG-TANG**"

'------------------------ ROACHS -------------

LOCATE 9, 3: PRINT "1 ROCK"
LOCATE 11, 3: PRINT "2 RAP"
LOCATE 13, 3: PRINT "3 HIP"
LOCATE 15, 3: PRINT "4 HOP"
LOCATE 17, 3: PRINT "5 POP"

FOR Z = 1 TO N

LOCATE 25, 40: PRINT "MISTER "; A$(Z)
6 : LOCATE 26, 40: INPUT "ROACH'S NUMBER(ü) "; B(Z)
IF B(Z) < 1 OR B(Z) > 5 THEN LINE (475, 400)-(600, 415), 7,
BF: GOTO 6
7 : LOCATE 27, 40: INPUT "INPUT YOUR BET "; C(Z)
```

```
IF C(Z) < 1 OR C(Z) > J(Z) THEN LINE (450, 416)-(600, 432), 7,
BF: GOTO 7
LINE (310, 350)-(625, 480), 7, BF

NEXT Z

LINE (310, 350)-(625, 480), 7, BF
y = 22
FOR i = 1 TO N
LOCATE y + i, 45: PRINT B(i)
LOCATE y + i, 65: PRINT C(i)
NEXT i

LOCATE 28, 35: PRINT "FOR START PRESS <5>"
WHILE INKEY$ <> "5": WEND

S1 = 0: S2 = 0: S3 = 0: S4 = 0: S5 = 0

X1 = 100: X2 = 100: X3 = 100: X4 = 100: X5 = 100
Y1 = 140: Y2 = 170: Y3 = 200: Y4 = 205 + 25: Y5 = Y3 + 60

1 : CIRCLE (X1, Y1), 10, 13, , , .5
PAINT (X1, Y1), 13
LINE (X1 — 15, Y1)-(X1 + 15, Y1), 13
LINE (X1 — 12, Y1 + 6)-(X1 + 12, Y1 — 6), 13
LINE (X1 — 12, Y1 — 6)-(X1 + 12, Y1 + 6), 13

CIRCLE (X2, Y2), 10, 5, , , .5
PAINT (X2, Y2), 5
LINE (X2 — 15, Y2)-(X2 + 15, Y2), 5
LINE (X2 — 12, Y2 + 6)-(X2 + 12, Y2 — 6), 5
LINE (X2 — 12, Y2 — 6)-(X2 + 12, Y2 + 6), 5

CIRCLE (X3, Y3), 10, 9, , , .5
PAINT (X3, Y3), 9
LINE (X3 — 15, Y3)-(X3 + 15, Y3), 9
LINE (X3 — 12, Y3 + 6)-(X3 + 12, Y3 — 6), 9
LINE (X3 — 12, Y3 — 6)-(X3 + 12, Y3 + 6), 9

CIRCLE (X4, Y4), 10, 2, , , .5
PAINT (X4, Y4), 2
LINE (X4 — 15, Y4)-(X4 + 15, Y4), 2
```

```
LINE (X4 — 12, Y4 + 6)-(X4 + 12, Y4 — 6), 2
LINE (X4 — 12, Y4 — 6)-(X4 + 12, Y4 + 6), 2

CIRCLE (X5, Y5), 10, 12, , , .5
PAINT (X5, Y5), 12
LINE (X5 — 15, Y5)-(X5 + 15, Y5), 12
LINE (X5 — 12, Y5 + 6)-(X5 + 12, Y5 — 6), 12
LINE (X5 — 12, Y5 — 6)-(X5 + 12, Y5 + 6), 12

'TIMER
'FOR I = 0 TO 50
'FOR k = 0 TO 99
'LOCATE 5, 45: PRINT I; "."; k
'NEXT k, I

FOR i = 1 TO 53000: NEXT i
CIRCLE (X1, Y1), 10, 1, , , .5
PAINT (X1, Y1), 1

CIRCLE (X2, Y2), 10, 1, , , .5
PAINT (X2, Y2), 1

CIRCLE (X3, Y3), 10, 1, , , .5
PAINT (X3, Y3), 1

CIRCLE (X4, Y4), 10, 1, , , .5
PAINT (X4, Y4), 1

CIRCLE (X5, Y5), 10, 1, , , .5
PAINT (X5, Y5), 1

DX1 = INT(RND(1) * 10) + 1
IF X1 >= 600 THEN LOCATE 5, 30: PRINT "  ROCK HAS WON!  ": P =
1: GOTO 2
X1 = X1 + DX1

DX2 = INT(RND(1) * 10) + 1
IF X2 >= 600 THEN LOCATE 5, 30: PRINT "  RAP HAS WON!  ": P =
2: GOTO 2
X2 = X2 + DX2

DX3 = INT(RND(1) * 10) + 1
IF X3 >= 600 THEN LOCATE 5, 30: PRINT "  HIP HAS WON! ":
P = 3: GOTO 2
X3 = X3 + DX3
```

```
DX4 = INT(RND(1) * 10) + 1
IF X4 >= 600 THEN LOCATE 5, 30: PRINT "  HOP HAS WON! ":
P = 4: GOTO 2
X4 = X4 + DX4

DX5 = INT(RND(1) * 10) + 1
IF X5 >= 600 THEN LOCATE 5, 30: PRINT "  POP HAS WON! ": P =
5: GOTO 2
X5 = X5 + DX5

S1 = S1 + DX1: S2 = S2 + DX2: S3 = S3 + DX3: S4 = S4 + DX4: S5
= S5 + DX5

LOCATE  9, 76: PRINT 500 — S1
LOCATE 11, 76: PRINT 500 — S2
LOCATE 13, 76: PRINT 500 — S3
LOCATE 15, 76: PRINT 500 — S4
LOCATE 17, 76: PRINT 500 — S5

LINE (200, 60)-(500, 90), 8, BF

IF S1 > S2 AND S1 > S3 AND S1 > S4 AND S1 > S5 THEN LOCATE 5,
30: PRINT "<ROCK> IS LEADER..."
IF S2 > S1 AND S2 > S3 AND S2 > S4 AND S2 > S5 THEN LOCATE 5,
30: PRINT "<RAP> IS LEADER..."
IF S3 > S1 AND S3 > S2 AND S3 > S4 AND S3 > S5 THEN LOCATE 5,
30: PRINT "<HIP> IS LEADER..."
IF S4 > S1 AND S4 > S2 AND S4 > S3 AND S4 > S5 THEN LOCATE 5,
30: PRINT "<HOP> IS LEADER..."
IF S5 > S1 AND S5 > S2 AND S5 > S3 AND S5 > S4 THEN LOCATE 5,
30: PRINT "<POP> IS LEADER... "


GOTO 1

2 : CIRCLE (X1, Y1), 10, 13, , , .5
PAINT (X1, Y1), 13

CIRCLE (X2, Y2), 10, 5, , , .5
PAINT (X2, Y2), 5

CIRCLE (X3, Y3), 10, 9, , , .5
PAINT (X3, Y3), 9
```

```
CIRCLE (X4, Y4), 10, 2, , , .5
PAINT (X4, Y4), 2

CIRCLE (X5, Y5), 10, 12, , , .5
PAINT (X5, Y5), 12

ER = 0
FOR L = 1 TO N
IF P = B(L) THEN LOCATE 6, 30: PRINT "MISTER "; A$(L); " HAS
WON IN THIS ZABEG!": J(L) = J(L) + C(L) ELSE ER = ER + 1: J(L)
= J(L) — C(L): IF J(L) <= 0 THEN LOCATE 23, 5: PRINT "BYE,
MISTER "; A$(L); " YOU ARE LOOSER :("
NEXT L
IF ER = N THEN LOCATE 6, 30: PRINT "NOBODY HAS WON :( "

y = 22
FOR i = 1 TO N
LOCATE y + i, 20: PRINT J(i)
NEXT i

LOCATE 7, 30: INPUT "PLAY NEXT ZABEG (Y/N)"; N$
IF N$ = "Y" OR N$ = "¤" OR N$ = "y" OR N$ = "„"
THEN k = k + 1: GOTO 4
PAINT (1, 1), 0

CLS

LOCATE 12, 35: PRINT "BYE!"

'PLAY "A2 A1 A4 A6 A7 "

WHILE INKEY$ <> "": WEND

END
```

Пинг-понг простой

Совсем простенький пинг-понг для двух игроков.

```
SCREEN 12

RANDOMIZE TIMER
'********* ZASTAVKA *********

PAINT (2, 3), 2
```

```
'********* PING — PONG ********

FOR X = 20 TO 260 STEP 40
LINE (X, 120)-(X + 20, 320), 1, BF
NEXT X

FOR X = 340 TO 580 STEP 40
LINE (X, 120)-(X + 20, 320), 1, BF
NEXT X

LINE (20, 120)-(80, 140), 1, BF
LINE (200, 200)-(220, 220), 1, BF
LINE (260, 120)-(300, 140), 1, BF
LINE (300, 200)-(320, 220), 1, BF
LINE (340, 120)-(400, 140), 1, BF
LINE (420, 120)-(480, 140), 1, BF
LINE (420, 320)-(480, 300), 1, BF
LINE (520, 200)-(540, 220), 1, BF
LINE (580, 120)-(620, 140), 1, BF
LINE (120, 300)-(140, 280), 1
LINE (120, 320)-(140, 300), 1
PAINT (122, 314), 1, 1

LOCATE 27, 28: PRINT " ALEX & ALEXANDR PRESENT....."

CIRCLE (200, 423), 4, 4, 1.57 / 2, (4.71 + 6.28) / 2
CIRCLE (200, 423), 7, 1

k$ = INPUT$(1)
CLS
'********** ZAPROS IMEN IGROKOV **********

LOCATE 14, 30: INPUT "1 IGROK KAK TVOE IMYA"; P1$
LOCATE 16, 30: INPUT "2 IGROK KAK TVOE IMYA"; P2$

LINE (0, 0)-(640, 480), 2, BF
LINE (0, 0)-(639, 359), 15, B

'************ TABLO ************
LINE (0, 360)-(640, 480), 4, BF
'************* SCHET *************
Z1 = 0: Z2 = 0
```

```
'************* VYVOD IMENI NA EKRAN *************
LOCATE 26, 70: PRINT P1$

LOCATE 26, 10: PRINT P2$
'******** LEVAYA TRIBUNA *********
Y = 380
FOR X = 100 TO 300 STEP 20
CIRCLE (X, Y), 10, 15
PAINT (X, Y), 2, 15
CIRCLE (X — 3, Y — 3), 1, 0
CIRCLE (X + 3, Y — 3), 1, 0

LINE (X — 3, Y + 3)-(X + 3, Y + 3), 4
NEXT X

Y = 405
FOR X = 110 TO 310 STEP 20
CIRCLE (X, Y), 10, 15
PAINT (X, Y), 2, 15
CIRCLE (X — 3, Y — 3), 1, 0
CIRCLE (X + 3, Y — 3), 1, 0

LINE (X — 3, Y + 3)-(X + 3, Y + 3), 4
NEXT X

Y = 430
FOR X = 100 TO 300 STEP 20
CIRCLE (X, Y), 10, 15
PAINT (X, Y), 2, 15
CIRCLE (X — 3, Y — 3), 1, 0
CIRCLE (X + 3, Y — 3), 1, 0

LINE (X — 3, Y + 3)-(X + 3, Y + 3), 4
NEXT X

'******** PRAVAYA TRIBUNA *********
Y = 380
FOR X = 340 TO 540 STEP 20
CIRCLE (X, Y), 10, 15
PAINT (X, Y), 14, 15
CIRCLE (X — 3, Y — 3), 1, 0
CIRCLE (X + 3, Y — 3), 1, 0
```

```
LINE (X — 3, Y + 3)-(X + 3, Y + 3), 4
NEXT X

Y = 405
FOR X = 330 TO 530 STEP 20
CIRCLE (X, Y), 10, 15
PAINT (X, Y), 14, 15
CIRCLE (X — 3, Y — 3), 1, 0
CIRCLE (X + 3, Y — 3), 1, 0

LINE (X — 3, Y + 3)-(X + 3, Y + 3), 4
NEXT X

Y = 430
FOR X = 340 TO 540 STEP 20
CIRCLE (X, Y), 10, 15
PAINT (X, Y), 14, 15
CIRCLE (X — 3, Y — 3), 1, 0
CIRCLE (X + 3, Y — 3), 1, 0

LINE (X — 3, Y + 3)-(X + 3, Y + 3), 4
NEXT X

'********* LEVAYA RAKETKA *********

X2 = 20
Y2 = 160
'********** pravaya roketka **************
X1 = 620
Y1 = 160

2 : '********** sharik **************
XM = 320: YM = 180: DX = (-1) ^ (INT(RND(1) * 2) + 1): DY = (-
1) ^ (INT(RND(1) * 2) + 1)
CIRCLE (XM, YM), 3, 4: PAINT (XM, YM), 4: CIRCLE (XM, YM), 3,
15
k$ = INPUT$(1)

1 : k$ = INKEY$

LINE (320, 0)-(320, 359), 15

'********** pravaya roketka **************

LINE (X1, Y1)-(X1 + 10, Y1 + 50), 1, BF
```

```
IF k$ = "O" OR k$ = "o" THEN LINE (X1, Y1)-(X1 + 10, Y1 + 50),
2, BF: Y1 = Y1 — 10

IF k$ = "L" OR k$ = "l" THEN LINE (X1, Y1)-(X1 + 10, Y1 + 50),
2, BF: Y1 = Y1 + 10
'*********** ogranichenie *************
IF Y1 >= 308 THEN Y1 = 308
IF Y1 <= 1 THEN Y1 = 1
'********* LEVAYA ROKETKA *********

LINE (X2, Y2)-(X2 + 10, Y2 + 50), 1, BF

IF k$ = "W" OR k$ = "w" THEN LINE (X2, Y2)-(X2 + 10, Y2 + 50),
2, BF: Y2 = Y2 — 10

IF k$ = "S" OR k$ = "s" THEN LINE (X2, Y2)-(X2 + 10, Y2 + 50),
2, BF: Y2 = Y2 + 10

'*********** ogranichenie *************
IF Y2 >= 308 THEN Y2 = 308
IF Y2 <= 1 THEN Y2 = 1

'********** sharik **************

CIRCLE (XM, YM), 3, 4: PAINT (XM, YM), 4: CIRCLE (XM, YM), 3, 15
IF XM = 4 AND YM >= 4 AND YM <= 355 THEN DX = -DX
IF XM = 635 AND YM >= 4 AND YM <= 355 THEN DX = -DX
IF YM = 4 AND XM >= 4 AND XM <= 635 THEN DY = -DY
IF YM = 355 AND XM >= 4 AND XM <= 635 THEN DY = -DY
IF XM = X2 + 15 AND YM >= Y2 AND YM <= Y2 + 50 THEN DX = -DX
IF XM = X1 — 5 AND YM >= Y1 AND YM <= Y1 + 50 THEN DX = -DX
IF XM >= 635 THEN Z2 = Z2 + 1: CIRCLE (XM, YM), 3, 2: PAINT
(XM, YM), 2: CIRCLE (XM, YM), 3, 2: GOSUB SMILE1: GOTO 2
IF XM <= 4 THEN Z1 = Z1 + 1: CIRCLE (XM, YM), 3, 2:
PAINT (XM, YM), 2: CIRCLE (XM, YM), 3, 2: GOTO 2

FOR I = 1 TO 3000: NEXT I
CIRCLE (XM, YM), 3, 2: PAINT (XM, YM), 2: CIRCLE (XM, YM), 3, 2
XM = XM + DX: YM = YM + DY
LOCATE 28, 70: PRINT Z1
LOCATE 28, 10: PRINT Z2

GOTO 1

END
```

```
SMILE1:

'************ YLIBKA LEVIH ********
'******** LEVAYA TRIBUNA *********
Y = 380
FOR X = 100 TO 300 STEP 20
CIRCLE (X, Y), 10, 15
PAINT (X, Y), 2, 15
CIRCLE (X — 3, Y — 3), 1, 0
CIRCLE (X + 3, Y — 3), 1, 0

CIRCLE (X, Y + 3), 4, 4, 3.14, 0
NEXT X

Y = 405
FOR X = 110 TO 310 STEP 20
CIRCLE (X, Y), 10, 15
PAINT (X, Y), 2, 15
CIRCLE (X — 3, Y — 3), 1, 0
CIRCLE (X + 3, Y — 3), 1, 0

CIRCLE (X, Y + 3), 4, 4, 3.14, 0
NEXT X

Y = 430
FOR X = 100 TO 300 STEP 20
CIRCLE (X, Y), 10, 15
PAINT (X, Y), 2, 15
CIRCLE (X — 3, Y — 3), 1, 0
CIRCLE (X + 3, Y — 3), 1, 0

CIRCLE (X, Y + 3), 4, 4, 3.14, 0
NEXT X

'******** PRAVAYA TRIBUNA *********
Y = 380
FOR X = 340 TO 540 STEP 20
CIRCLE (X, Y), 10, 15
PAINT (X, Y), 14, 15
CIRCLE (X — 3, Y — 3), 1, 0
CIRCLE (X + 3, Y — 3), 1, 0
```

```
CIRCLE (X, Y + 3), 4, 4, 0, 3.14
NEXT X

Y = 405
FOR X = 330 TO 530 STEP 20
CIRCLE (X, Y), 10, 15
PAINT (X, Y), 14, 15
CIRCLE (X — 3, Y — 3), 1, 0
CIRCLE (X + 3, Y — 3), 1, 0

CIRCLE (X, Y + 3), 4, 4, 0, 3.14
NEXT X

Y = 430
FOR X = 340 TO 540 STEP 20
CIRCLE (X, Y), 10, 15
PAINT (X, Y), 14, 15
CIRCLE (X — 3, Y — 3), 1, 0
CIRCLE (X + 3, Y — 3), 1, 0

CIRCLE (X, Y + 3), 4, 4, 0, 3.14
NEXT X
RETURN
```

Игра "Реверси"

А это уже для продвинутых пользователей.

```
DEFINT A-Z
CLS
PRINT "Othello Reversi": PRINT
PRINT "Select Video Type"
PRINT
PRINT "1. EGA"
PRINT "2. VGA"
LOCATE , , 1
DO
 G$ = INPUT$(1)
LOOP UNTIL G$ = "1" OR G$ = "2"
IF G$ = "1" THEN SCREEN 7
IF G$ = "2" THEN SCREEN 13
```

```
BEGINGAME:
X = 0: Y = 0: W = 230: Z = 190
IF G$ = "2" THEN
FOR C = 16 TO 31
 LINE (X, Y)-(W, Z), C, B
 X = X + 1: Y = Y + 1: W = W — 1: Z = Z — 1
NEXT C
ELSE
 LINE (15, 15)-(215, 175), 15, B
 LINE (14, 14)-(216, 176), 7, B
 LINE (13, 13)-(217, 177), 8, B

END IF
IF G$ = "2" THEN PAINT (150, 100), 7, 31 ELSE PAINT (150,
100), 7, 15
FOR X = 35 TO 205 STEP 20
 LINE (X, 15)-(X, 175), 15
NEXT X
FOR Y = 31 TO 180 STEP 16
 LINE (15, Y)-(215, Y), 15
NEXT Y
PC = 12
PLAYERC = 9
CIRCLE (105, 87), 7, PLAYERC
PAINT (105, 87), PLAYERC
CIRCLE (125, 87), 7, PC
PAINT (125, 87), PC
CIRCLE (105, 103), 7, PC
PAINT (105, 103), PC
CIRCLE (125, 103), 7, PLAYERC
PAINT (125, 103), PLAYERC

LOCATE 2, 32: COLOR 15: PRINT "OTHELLO"
LINE (95, 79)-(115, 95), 0, B
X = 95: Y = 79: W = 115: Z = 95
CIRCLE (275, 30), 7, PLAYERC
PAINT (275, 30), PLAYERC
LOCATE 9, 32: PRINT "PLAYERS"
LOCATE 11, 31: PRINT "1. ONE"
LOCATE 13, 31: PRINT "2. TWO"
```

```
DO
 A$ = INPUT$(1)
LOOP UNTIL A$ = "1" OR A$ = "2"
LOCATE 9, 32: PRINT SPACE$(7)
LOCATE 11, 31: PRINT SPACE$(6)
LOCATE 13, 31: PRINT SPACE$(6)
IF A$ = "1" THEN PLAYER = 1 ELSE PLAYER = 2
LOCATE 20, 32: PRINT CHR$(34); : COLOR 10: PRINT "P"; : COLOR
15: PRINT CHR$(34); "ass"
LOCATE 22, 32: PRINT CHR$(34); : COLOR 10: PRINT "Q"; : COLOR
15: PRINT CHR$(34); "uit"
BOX = 4
DO
 IF PLAYERC = 12 AND PLAYER = 1 THEN GOSUB COMPUTER
 V$ = INKEY$
  IF V$ = CHR$(0) + "P" THEN GOSUB DOWN
  IF V$ = CHR$(0) + "K" THEN GOSUB LEFT
  IF V$ = CHR$(0) + "M" THEN GOSUB RIGHT
  IF V$ = CHR$(0) + "H" THEN GOSUB UP
  IF UCASE$(V$) = "P" THEN SWAP PLAYERC, PC: PAINT (275, 30),
PLAYERC, 0
  IF UCASE$(V$) = "Q" THEN GOTO OTHELLOEND
  IF V$ = CHR$(13) THEN GOSUB PUTSQUARE
  IF V$ = CHR$(27) THEN GOTO OTHELLOEND
LOOP

LEFT:
 IF X = 15 THEN RETURN
 LINE (X, Y)-(W, Z), 15, B
 X = X — 20: W = W — 20
 LINE (X, Y)-(W, Z), 0, B
RETURN

RIGHT:
 IF W = 215 THEN RETURN
 LINE (X, Y)-(W, Z), 15, B
 X = X + 20: W = W + 20
 LINE (X, Y)-(W, Z), 0, B
RETURN
```

```
DOWN:
 IF Z = 175 THEN RETURN
 LINE (X, Y)-(W, Z), 15, B
 Y = Y + 16: Z = Z + 16
 LINE (X, Y)-(W, Z), 0, B
RETURN

UP:
 IF Y = 15 THEN RETURN
 LINE (X, Y)-(W, Z), 15, B
 Y = Y — 16: Z = Z — 16
 LINE (X, Y)-(W, Z), 0, B
RETURN

COMPUTER:
LOCATE 9, 31: COLOR 14: PRINT "COMPUTING"
BACKX = X
BACKY = Y
AGAIN = 0
DO
 V$ = INKEY$
 IF V$ = CHR$(27) THEN GOTO DECIDEWINNER
 RANDOMIZE TIMER
 RNY = INT(RND * 10) + 1
 RNX = INT(RND * 10) + 1
 IF RNY = 1 THEN Y = 15 ELSE Y = 15 + ((RNY — 1) * 16)
 IF RNX = 1 THEN X = 15 ELSE X = 15 + ((RNX — 1) * 20)
 GOSUB PUTSQUARE

AGAIN = AGAIN + 1
 IF AGAIN = 250 THEN
  LOCATE 9, 31: COLOR 11: PRINT " PASS! "
  SLEEP 1
  SWAP PLAYERC, PC
  PAINT (275, 30), PLAYERC, 0
  LOCATE 9, 31: PRINT SPACE$(9)
  X = BACKX
  Y = BACKY
  RETURN
 END IF
```

```
LOOP UNTIL VALID = 1
IF VALID <> 1 THEN
 FOR Y = 15 TO 170 STEP 16
  FOR X = 15 TO 200 STEP 20
   GOSUB PUTSQUARE
   IF VALID = 1 THEN EXIT FOR
  NEXT X
  IF VALID = 1 THEN EXIT FOR
 NEXT Y
 X = BACKX
 Y = BACKY
END IF

LOCATE 9, 31: PRINT SPACE$(9)
X = BACKX
Y = BACKY
RETURN

PUTSQUARE:
 VALID = -1
 Colour = POINT(X + 10, Y + 10)
 IF Colour <> 7 THEN
  IF PLAYERC = 12 AND PLAYER = 1 THEN
   RETURN
  ELSE
   LOCATE 10, 31: COLOR 10: PRINT "OCCUPIED!": SLEEP 1: LOCATE
10, 31: PRINT SPACE$(9): RETURN
  END IF
 END IF

 ' Check side on Right
  N = X + 20
  M = N
  C = 0
  FOR R = N TO 200 STEP 20
   C = POINT(R + 10, Y + 10)
   IF C = 7 THEN EXIT FOR
   IF C = PLAYERC THEN EXIT FOR
  NEXT R
```

```
 IF C = PLAYERC AND N <> R THEN
  VALID = 1
  N = R — 20
  FOR R = M TO N STEP 20
   PAINT (R + 10, Y + 10), PLAYERC, 7
  NEXT R
 END IF

' Check side on Left
 N = X — 20
 M = N
 C = 0
 FOR R = N TO 10 STEP -20
  C = POINT(R + 10, Y + 10)
  IF C = 7 THEN EXIT FOR
  IF C = PLAYERC THEN EXIT FOR
 NEXT R
 IF C = PLAYERC AND N <> R THEN
  VALID = 1
  N = R + 20
  FOR R = N TO M STEP 20
   PAINT (R + 10, Y + 10), PLAYERC, 7
  NEXT R
 END IF

' Check Up
 N = Y — 18
 M = N
 C = 0
 FOR R = N TO 10 STEP -16
  C = POINT(X + 10, R + 10)
  IF C = 7 THEN EXIT FOR
  IF C = PLAYERC THEN EXIT FOR
 NEXT R
 IF C = PLAYERC AND N <> R THEN
  VALID = 1
  N = R + 16
  FOR R = N TO M STEP 16
   PAINT (X + 10, R + 10), PLAYERC, 7
  NEXT R
 END IF
```

```
' Check Down
 N = Y + 14
 M = N
 C = 0
 FOR R = N TO 170 STEP 16
  C = POINT(X + 10, R + 10)
  IF C = 7 THEN EXIT FOR
  IF C = PLAYERC THEN EXIT FOR
 NEXT R
 IF C = PLAYERC AND N <> R THEN
  VALID = 1
  N = R — 14
  FOR R = M TO N STEP 16
   PAINT (X + 10, R + 10), PLAYERC, 7
  NEXT R
 END IF

' Check Diagonals
 ' Right & Down
 N = Y + 14
 F = X + 20
 M = N
 G = F
 C = 0
 FOR R = N TO 170 STEP 16
  C = POINT(F + 10, R + 10)
  IF C = 7 THEN EXIT FOR
  IF C = PLAYERC THEN EXIT FOR
  F = F + 20
 NEXT R
 IF C = PLAYERC AND N <> R THEN
  VALID = 1
  N = R — 14
  F = X + 20
  FOR R = M TO N STEP 16
   PAINT (F + 10, R + 10), PLAYERC, 7
   F = F + 20
  NEXT R
 END IF
```

```
' Right & Up
 N = Y — 18
 F = X — 20
 M = N
 G = F
 C = 0
 FOR R = N TO 10 STEP -16
  C = POINT(F + 10, R + 10)
  IF C = 7 THEN EXIT FOR
  IF C = PLAYERC THEN EXIT FOR
  F = F — 20
 NEXT R
 IF C = PLAYERC AND N <> R THEN
  VALID = 1
  N = R + 16
  F = F + 20
  FOR R = N TO M STEP 16
   PAINT (F + 10, R + 10), PLAYERC, 7
   F = F + 20
  NEXT R
 END IF

' Left & Down
 N = Y + 14
 F = X — 20
 M = N
 G = F
 C = 0
 FOR R = N TO 170 STEP 16
  C = POINT(F + 10, R + 10)
  IF C = 7 THEN EXIT FOR
  IF C = PLAYERC THEN EXIT FOR
  F = F — 20
 NEXT R
 IF C = PLAYERC AND N <> R THEN
  VALID = 1
  N = R — 14
  F = X — 20
  FOR R = M TO N STEP 16
```

```
    PAINT (F + 10, R + 10), PLAYERC, 7
    F = F — 20
   NEXT R
  END IF

' Left & Up
  N = Y — 18
  F = X + 20
  M = N
  G = F
  C = 0
  FOR R = N TO 10 STEP -16
   C = POINT(F + 10, R + 10)
   IF C = 7 THEN EXIT FOR
   IF C = PLAYERC THEN EXIT FOR
   F = F + 20
  NEXT R
  IF C = PLAYERC AND N <> R THEN
   VALID = 1
   N = R + 16
   F = F — 20
   FOR R = N TO M STEP 16
    PAINT (F + 10, R + 10), PLAYERC, 7
    F = F — 20
   NEXT R
  END IF
  IF VALID = -1 THEN
   IF PLAYER = 1 AND PLAYERC = 12 THEN
   ELSE
    LOCATE 9, 32: COLOR 14: PRINT "ILLEGAL"
    LOCATE 11, 33: PRINT "MOVE!"
    SLEEP 1
    LOCATE 9, 32: PRINT SPACE$(9)
    LOCATE 11, 33: PRINT SPACE$(5)
   END IF
   RETURN
  ELSE
   BOX = BOX + 1
  END IF
```

```
  CIRCLE (X + 10, Y + 8), 7, PLAYERC
  PAINT (X + 10, Y + 8), PLAYERC
  SWAP PLAYERC, PC
  PAINT (275, 30), PLAYERC, 0
 IF BOX = 100 THEN LOCATE 9, 31: PRINT SPACE$(9): GOTO
DECIDEWINNER
RETURN

DECIDEWINNER:
 BLUE = 0
 RED = 0
 LOCATE 20, 31: PRINT SPACE$(9)
 LOCATE 22, 31: PRINT SPACE$(9)
 LOCATE 8, 33: COLOR 10: PRINT "FINAL"
 LOCATE 10, 33: PRINT "SCORE"
 LOCATE 13, 31: COLOR 9: PRINT "BLUE:"
 LOCATE 15, 31: COLOR 12: PRINT "RED: "
 COLOR 15
 Y = 24
 DO
  FOR X = 25 TO 220 STEP 20
   CL = POINT(X, Y)
   IF CL = 9 THEN BLUE = BLUE + 1: LOCATE 13, 36: PRINT BLUE
   IF CL = 12 THEN RED = RED + 1: LOCATE 15, 36: PRINT RED
  NEXT X
  Y = Y + 16
 LOOP UNTIL Y >= 170

COLOR 14
 IF BLUE > RED THEN LOCATE 19, 31: PRINT "BLUE WINS!"
 IF RED > BLUE THEN LOCATE 19, 31: PRINT "RED WINS!"
 IF BLUE = RED THEN LOCATE 19, 31: PRINT "IT'S A TIE"
 LOCATE 25, 1: PRINT "ANOTHER GAME [Y/N] "; : COLOR 12
 DO
  A$ = INPUT$(1)
  ANS$ = UCASE$(A$)
 LOOP UNTIL ANS$ = "Y" OR ANS$ = "N"
 IF UCASE$(A$) = "Y" THEN CLS : GOTO BEGINGAME

OTHELLOEND:
SCREEN 0, 0, , 0: CLS
WIDTH 80, 25
```

Игра "Виселица" (со словами)

Почти "Поле чудес" ☺.

```
DECLARE SUB MouseInit ()
DECLARE SUB MouseReset ()
DECLARE SUB MouseHide ()
DECLARE SUB MouseShow ()
DECLARE FUNCTION MouseX% ()
DECLARE FUNCTION MouseY% ()
DECLARE FUNCTION MouseButtons% ()
DIM SHARED a%(34)
DIM B%(15), c%(31)
DEF SEG = VARSEG(a%(0))
FOR i% = 0 TO 63
READ d%
POKE VARPTR(a%(0)) + i%, d%
NEXT i%
DEF SEG
DATA 0, 0
DATA 0, 0
DATA 0, 0
DATA &hb8,0,0
DATA &hcd,&h33
DATA &h3d,&hff,&hff
DATA &h75,&h0d
DATA &h0e
DATA &h07
DATA &hba,&h24,0
DATA &hb9,&hff,&hff
DATA &hb8,&h0c,0
DATA &hcd,&h33
DATA &hcb
DATA &hb8,0,0
DATA &hcd,&h33
DATA &hcb
DATA &h2e,&h89,&h0e,0,0
DATA &h2e,&h89,&h16,2,0
DATA &h2e,&h89,&h1e,4,0
```

```
DATA &hcb
DATA &hb8,1,0
DATA &hcd,&h33
DATA &hcb
DATA &hb8,2,0
DATA &hcd,&h33
DATA &hcb
SCREEN 12
p1% = 0: p2% = 0
CALL MouseInit
na: CLS
tur% = tur% + 1: s% = 0
FOR i = 0 TO 15: B%(i) = 0: NEXT
FOR i = 0 TO 31: c%(i) = 0: NEXT
FOR B% = 0 TO 31
PRINT CHR$(B% + 128);
NEXT
B% = 0
FOR y% = 1 TO 16
FOR x% = 0 TO 32 * 8
IF POINT(x%, y%) <> 0 THEN LINE (a, B% + 430)-(a + 2, B% +
432), 10, BF
a = a + 639 / (32 * 8)
NEXT
a = 0: B% = B% + 3
NEXT
LINE (0, 428)-(639, 428), 5
LINE (0, 470)-(639, 470), 5
LOCATE 1, 1: PRINT "                          "
RANDOMIZE (TIMER)
t = INT(RND(1) * 9 + 1)
IF t = 1 THEN RESTORE dat1
IF t = 2 THEN RESTORE dat2
IF t = 3 THEN RESTORE dat3
IF t = 4 THEN RESTORE dat4
IF t = 5 THEN RESTORE dat5
IF t = 6 THEN RESTORE dat6
IF t = 7 THEN RESTORE dat7
IF t = 8 THEN RESTORE dat8
```

```
IF t = 9 THEN RESTORE dat9
READ tem$
kol% = 0
DO
kol% = kol% + 1
READ m$
LOOP WHILE m$ <> "-1"
IF t = 1 THEN RESTORE dat1
IF t = 2 THEN RESTORE dat2
IF t = 3 THEN RESTORE dat3
IF t = 4 THEN RESTORE dat4
IF t = 5 THEN RESTORE dat5
IF t = 6 THEN RESTORE dat6
IF t = 7 THEN RESTORE dat7
IF t = 8 THEN RESTORE dat8
IF t = 9 THEN RESTORE dat9
READ tem$
LOCATE 1, 37: PRINT tur%; "Тур"
LOCATE 3, 31: PRINT "ТЕМА ИГРЫ: "; tem$
t = RND(1) * kol% + 1: FOR i = 1 TO t — 1: READ a$: NEXT
GOSUB nadp
z = 639 / (32 * 8) * 8
MouseShow
LOCATE 1, 75: PRINT "Выход!"
LINE (591, 0)-(639, 16), 10, B
LOCATE 1, 1: PRINT "http:\\www.kas-cor.boom.ru и
http:\\www.kas-cor.narod.ru"
DO
x% = MouseX%
y% = MouseY%
k% = MouseButtons%
IF k% = 1 AND x% > 591 AND y% > 0 AND x% < 639 AND y% < 16
THEN END
IF k% = 1 AND y% > 430 AND c%(x% \ z) = 0 THEN
n$ = CHR$(x% \ z + 128)
u% = 0
FOR i% = 1 TO LEN(a$)
 IF MID$(a$, i%, 1) = n$ THEN B%(i%) = 1: u% = 1
NEXT
```

```
CALL MouseHide
LINE ((x% \ z) * z, 430)-((x% \ z) * z + z, 468), 0, BF:
c%(x% \ z) = 1
GOSUB nadp
IF u% = 0 THEN LOCATE 15, 1: PRINT STRING$(80, " "):
s% = s% + 1: ON s% GOSUB v1, v2, v3, v4, v5, v6, v7
d% = 0
FOR i% = 1 TO LEN(a$)
 IF B%(i%) = 0 THEN d% = 1: EXIT FOR
NEXT
IF d% = 0 THEN GOTO en
CALL MouseShow
END IF
LOOP WHILE INKEY$ <> CHR$(27)
f$ = INPUT$(1)
END
nadp:
a1% = 319 — (8 * LEN(a$) * 2) / 2
LOCATE 1, 1: PRINT a$
FOR i% = 1 TO LEN(a$)
IF B%(i%) = 0 THEN LOCATE 1, i%: PRINT CHR$(254)
NEXT
FOR y% = 0 TO 16
FOR x% = 0 TO LEN(a$) * 8
LINE (x% * 2 + a1%, y% * 2 + 100)-(x% * 2 + a1% + 2, y% * 2 +
102), POINT(x%, y%)
NEXT
NEXT
LOCATE 1, 1: PRINT "          "
RETURN
v1: DRAW "c8bm279,250d90u90r10d90g30r10e25h5g5h5g5f5bf5f5
d10l10u10e5f15r10h20l10"
LOCATE 15, 1: PRINT "               Вот и первый столб."
RETURN
v2: DRAW "c8bm349,250d90u90r10d90g30r10e25h5g5h5g5f5bf5
f5d10l10u10e5f15r10h20l10"
LOCATE 15, 1: PRINT "       Вот и второй уже поставили,
недолго тебе осталось."
RETURN
v3: DRAW "c8bm269,260r100d10l100u10r10f10l10e10r60f10l10e10"
```

```
LOCATE 15, 1: PRINT "         А это перекладина, на которой
тебя повесят."
RETURN
v4: LINE (319, 260)-(319, 290), 8: CIRCLE (319, 295), 5, 8
LOCATE 15, 1: PRINT "              Вот и петлю повесили."
RETURN
v5: CIRCLE (319, 310), 10, 8, , , 1.7: LINE (319, 300)-(306,
307), 8: LINE (320, 300)-(331, 306), 8
LOCATE 15, 1: PRINT "            Да-а-а еще чуть-чуть
осталось."
RETURN
v6: LINE (317, 319)-(309, 335), 8: LINE (321, 319)-(329, 335),
8
LOCATE 15, 1: PRINT "         Вот ты уже и висишь, остался
последний шанс."
RETURN
v7:
FOR i = 0 TO 30
LINE (309 + i / 3, 370 — i)-(329 — i / 3, 370 — i), 4
FOR t = 0 TO 2000: NEXT
NEXT
FOR i% = 1 TO LEN(a$): B%(i%) = 1: NEXT: GOSUB nadp: LOCATE 8,
10: PRINT "Это:"
LOCATE 15, 1: PRINT "            Т ы  п о г о р е л ."
p2% = p2% + 1
GOTO fi
en:
LOCATE 13, 1: PRINT "   П о з д р а в л я ю  с  п о б е д о й
! ! !"
p1% = p1% + 1
fi: f$ = INPUT$(1): IF tur% <> 5 THEN GOTO na
CLS
PRINT "ИТОГИ ЭТОЙ ИГРЫ:"
PRINT "Побед:   "; p1%
PRINT "Поражений: "; p2%
IF p1% > p2% THEN PRINT "Поздравляю с окончательной победой
!!!" ELSE PRINT "К сожалению, у вас побед меньше, чем
поражений, значит вы проиграли все."
END
dat1: DATA Животные,ЛИСА,ВОЛК,МЕДВЕДЬ,ЛОСЬ,ЗАЯЦ,КУНИЦА,ОЛЕНЬ,
ГИЕНА,БУЙВОЛ,РЫСЬ,ЛЕВ,ВОЛ,МУЛ,ЛОШАДЬ,КАБАН,ТИГР,ЛЕОПАРД,-1
```

```
dat2: DATA Мебель,СТОЛ,СТУЛ,ШКАФ,КРОВАТЬ,ТАБУРЕТКА,КРЕСЛО,
СОФА,ТУМБОЧКА,ПОДСВЕЧНИК,ЧАСЫ,ПОЛКА,ЗЕРКАЛО,-1
dat3: DATA Деревья,БЕРЕЗА,ДУБ,СОСНА,ЕЛЬ,БАОБАБ,ПАЛЬМА,ЯСЕНЬ,
КЛЕН,КЕДР,ЛИСТВЕННИЦА,ИВА,-1
dat4: DATA Оргтехника,МОНИТОР,КЛАВИАТУРА,ПРИНТЕР,СКАНЕР,МЫШЬ,
КОВРИК,ПРОЦЕССОР,ДИСКОВОД,ДИСК,ДЖОЙСТИК,ТРЕКБОЛ,КОЛОНКИ,-1
dat5: DATA Страны,РОССИЯ,США,АНГЛИЯ,ФРАНЦИЯ,ИТАЛИЯ,ЮАР,ФРГ,
НИГЕРИЯ,АЛЖИР,АВСТРАЛИЯ,ВАТИКАН,ГРЕЦИЯ,-1
dat6: DATA Работа,СТРОИТЕЛЬ,МЕНЕДЖЕР,ШВЕЯ,УЧИТЕЛЬ,ВАХТЕР,
ДОКТОР,МЕДСЕСТРА,МЕДБРАТ,МИЛИЦИОНЕР,ВРАЧ,ГИНЕКОЛОГ,ПРОГРАММИСТ,
ДВОРНИК,МЯСНИК,ДОМОХОЗЯЙКА,БЕЗРАБОТНЫЙ,ПОВАР,ОФИЦИАНТ,
СЕКРЕТАРЬ,-1
dat7: DATA Газеты,КМ,ТРУД,ПРАВДА,ЛЕНИН,-1
dat8: DATA Кинофильмы,ЗОМБИ,НАПАРНИК,НАВОДНЕНИЕ,МАКСИМКА,
ТЕРМИНАТОР,РОККИ,ФАНТОМАС,МАМА,ПРОВОКАТОР,ПРИЗРАК,ИГЛА,МЕНТЫ,
КАМЕНСКАЯ,ПОЛИЦЕЙСКИЙ,ГОРЕЦ,ЧАРОДЕИ,БЛИЗНЕЦЫ,ЭПИДЕМИЯ,
АСТЕРОЙД,АРМАГЕДДОН,ВОЛКОДАВ,МАЛЫШ,ЗЛОДЕЙКА,МАСКА,ГРЕШНИК,
НАСТЯ,БЕГЛЕЦ,БЕГЛЕЦЫ
DATA РОЗЫГРЫШ,ПОБЕГ,-1
dat9: DATA Спорт,ФУТБОЛ,БАСКЕТБОЛ,ВОЛЕЙБОЛ,ХОККЕЙ,БЕЙСБОЛ,-1
'dat10: DATA
'dat11: DATA

FUNCTION MouseButtons%
MouseButtons% = a%(2)
END FUNCTION

SUB MouseHide
DEF SEG = VARSEG(a%(0))
CALL absolute(VARPTR(a%(0)) + &H3A)
DEF SEG
END SUB

SUB MouseInit
DEF SEG = VARSEG(a%(0))
CALL absolute(VARPTR(a%(0)) + 6)
DEF SEG
END SUB

SUB MouseReset
DEF SEG = VARSEG(a%(0))
CALL absolute(VARPTR(a%(1)) + &H1E)
```

```
DEF SEG
END SUB

SUB MouseShow
DEF SEG = VARSEG(a%(0))
CALL absolute(VARPTR(a%(0)) + &H34)
DEF SEG
END SUB

FUNCTION MouseX%
MouseX% = a%(0)
END FUNCTION

FUNCTION MouseY%
MouseY% = a%(1)
END FUNCTION
```

Игра "15"

Очень хорошо сделанная игра "15".

```
DECLARE SUB MouseInit ()
DECLARE SUB MouseReset ()
DECLARE SUB MouseHide ()
DECLARE SUB MouseShow ()
DECLARE FUNCTION MouseX% ()
DECLARE FUNCTION MouseY% ()
DECLARE FUNCTION MouseButtons% ()
DIM SHARED a%(34)
DIM b%(6, 6), k%(1 TO 10000), im$(10), r%(10)
DEF SEG = VARSEG(a%(0))
FOR i% = 0 TO 63
READ d%
POKE VARPTR(a%(0)) + i%, d%
NEXT i%
DEF SEG
DATA 0, 0
DATA 0, 0
DATA 0, 0
DATA &hb8,0,0
```

```
DATA &hcd,&h33
DATA &h3d,&hff,&hff
DATA &h75,&h0d
DATA &h0e
DATA &h07
DATA &hba,&h24,0
DATA &hb9,&hff,&hff
DATA &hb8,&h0c,0
DATA &hcd,&h33
DATA &hcb
DATA &hb8,0,0
DATA &hcd,&h33
DATA &hcb
DATA &h2e,&h89,&h0e,0,0
DATA &h2e,&h89,&h16,2,0
DATA &h2e,&h89,&h1e,4,0
DATA &hcb
DATA &hb8,1,0
DATA &hcd,&h33
DATA &hcb
DATA &hb8,2,0
DATA &hcd,&h33
DATA &hcb
SCREEN 12
ON ERROR GOTO s1
OPEN "record3.dat" FOR INPUT AS #1
INPUT #1, maxig%
FOR i% = 0 TO maxig%: INPUT #1, im$(i%), r%(i%): NEXT
CLOSE #1
s1: CLS
PRINT "Кто сегодня играет ?"
FOR i% = 1 TO maxig%
PRINT i%; ". "; im$(i%); ", личный рекорд:"; r%(i%); "ход."
NEXT
PRINT "-1 . Новый игрок"
INPUT ig%
IF ig% = -1 THEN maxig% = maxig% + 1: INPUT "Напишите, как
зовут нового игрока"; im$(maxig%): GOTO s1
PRINT "Здравствуйте "; im$(ig%)
```

```
FOR t = 0 TO 90000: NEXT
CLS
CALL MouseInit
na: h = 0: CLS
FOR y% = 0 TO 3
FOR x% = 1 TO 4
LINE (x% * 110 + 2, y% * 110 + 2)-(x% * 110 + 108, y% * 110 +
108), 10, B
LINE (x% * 110 + 10, y% * 110 + 10)-(x% * 110 + 100, y% * 110
+ 100), 10, B
LINE (x% * 110 + 2, y% * 110 + 2)-(x% * 110 + 10, y% * 110 +
10), 10
LINE (x% * 110 + 108, y% * 110 + 108)-(x% * 110 + 100, y% *
110 + 100), 10
LINE (x% * 110 + 108, y% * 110 + 2)-(x% * 110 + 100, y% * 110
+ 10), 10
LINE (x% * 110 + 2, y% * 110 + 108)-(x% * 110 + 10, y% * 110 +
100), 10
CIRCLE (x% * 110 + 55, y% * 110 + 55), 30, 10
PAINT (x% * 110 + 12, y% * 110 + 12), 3, 10
PAINT (x% * 110 + 4, y% * 110 + 3), 7, 10
PAINT (x% * 110 + 3, y% * 110 + 4), 7, 10
PAINT (x% * 110 + 105, y% * 110 + 107), 8, 10
PAINT (x% * 110 + 107, y% * 110 + 105), 8, 10
NEXT
NEXT
LINE (0, 460)-(639, 460), 10
PAINT (0, 0), 2, 10
LINE (0, 460)-(639, 460), 0
LINE (109, -1)-(551, 441), 0, B
RANDOMIZE (TIMER)
FOR y% = 1 TO 4
FOR x% = 1 TO 4
b%(x%, y%) = 0
NEXT
NEXT
FOR i% = 0 TO 15
DO
x% = INT(RND(1) * 4 + 1)
y% = INT(RND(1) * 4 + 1)
```

```
LOOP WHILE b%(x%, y%) <> 0
b%(x%, y%) = i%
LOCATE y% * 7 — 3, x% * 14 + 5: PRINT i%
NEXT
FOR y% = 1 TO 4
FOR x% = 1 TO 4
IF b%(x%, y%) = 0 THEN x1% = x%: y1% = y%
NEXT
NEXT
LINE (x1% * 110, (y1% — 1) * 110)-(x1% * 110 + 110, (y1% — 1)
* 110 + 110), 0, BF
CALL MouseShow
LOCATE 1, 75: PRINT "Выход!"
LINE (591, 0)-(639, 16), 10, B
DO
x% = MouseX%
y% = MouseY%
k% = MouseButtons%
IF k% = 1 AND x% > 591 AND y% > 0 AND x% < 639 AND y% < 16
THEN END
a% = (x% \ 110) — x1%: b% = (y% \ 110) — (y1% — 1)
IF a% = 0 OR b% = 0 THEN
 IF b%(x% \ 110, y% \ 110 + 1) <> 0 AND k% = 1 THEN
 CALL MouseHide
 x4% = x%: y4% = y%
 IF a% < 0 AND b% = 0 THEN GET ((x% \ 110) * 110, (y% \ 110) *
110)-(x1% * 110, (y1% — 1) * 110 + 110), k%: p% = 1: x2% = (x%
\ 110) * 110: y2% = (y% \ 110) * 110
 IF a% > 0 AND b% = 0 THEN GET ((x% \ 110) * 110 + 110, (y% \
110) * 110 + 110)-(x1% * 110 + 110, (y1% — 1) * 110), k%: p% =
2: x2% = x1% * 110 + 110: y2% = (y1% — 1) * 110
 IF a% = 0 AND b% < 0 THEN GET ((x% \ 110) * 110, (y% \ 110) *
110)-(x1% * 110 + 110, (y1% — 1) * 110), k%: p% = 3: x2% = (x%
\ 110) * 110: y2% = (y% \ 110) * 110
 IF a% = 0 AND b% > 0 THEN GET ((x% \ 110) * 110 + 110, (y% \
110) * 110 + 110)-(x1% * 110, (y1% — 1) * 110 + 110), k%: p% =
4: x2% = x1% * 110: y2% = (y1% — 1) * 110 + 110
 x3% = x2%: y3% = y2%
 IF p% = 1 THEN r% = ABS(a%)
 IF p% = 2 THEN r% = a%
 IF p% = 3 THEN r% = ABS(b%)
```

```
 IF p% = 4 THEN r% = b%
 PUT (x3%, y3%), k%
 FOR i% = 1 TO 110 STEP 5
 PUT (x3%, y3%), k%
 FOR t = r% * 500 TO 1500: NEXT
 PUT (x3%, y3%), k%
 IF p% = 1 THEN x3% = x3% + 5
 IF p% = 2 THEN x3% = x3% — 5
 IF p% = 3 THEN y3% = y3% + 5
 IF p% = 4 THEN y3% = y3% — 5
 NEXT
 PUT (x3%, y3%), k%
 x3% = x1%: y3% = y1%
 x2% = x2% \ 110: y2% = y2% \ 110
 FOR i% = 1 TO r%
 IF p% = 1 THEN b%(x3%, y3%) = b%(x3% — 1, y3%): x3% = x3% — 1
 IF p% = 2 THEN b%(x3%, y3%) = b%(x3% + 1, y3%): x3% = x3% + 1
 IF p% = 3 THEN b%(x3%, y3%) = b%(x3%, y3% — 1): y3% = y3% — 1
 IF p% = 4 THEN b%(x3%, y3%) = b%(x3%, y3% + 1): y3% = y3% + 1
 NEXT
 b%(x4% \ 110, y4% \ 110 + 1) = 0: x1% = x4% \ 110: y1% = y4%
\ 110 + 1
 CALL MouseShow
 h = h + 1
 LOCATE 29, 38: PRINT " Ход "; CHR$(252); h;
END IF
END IF
w% = 0: o% = 0
FOR y% = 1 TO 4
FOR x% = 1 TO 4
 w% = w% + 1
 IF w% = 16 THEN w% = 0
 IF b%(x%, y%) <> w% THEN o% = 1
NEXT
NEXT
IF o% = 0 THEN GOTO pobeda
LOOP WHILE INKEY$ <> CHR$(27)
CHAIN "games3.bas"
pobeda:
CALL MouseHide
```

```
LOCATE 30, 1: PRINT "       П о з д р а в л я ю
с   п о б е д о й";
IF record% > h THEN
r%(im%) = h
OPEN "record3.dat" FOR OUTPUT AS #1
WRITE #1, maxig%
FOR i% = 0 TO maxig%: WRITE #1, im$(i%), r%(i%): NEXT
CLOSE #1
ELSE f$ = INPUT$(1)
END IF
CLS
LOCATE 16, 5: PRINT "Абсолютный рекорд принадлежит "; a$; "он
победил за"; record%; "ходов."
LOCATE 17, 35: PRINT "Еще хочешь ?"; : f$ = INPUT$(1)
IF f$ = "d" OR f$ = "l" OR f$ = "д" OR f$ = "в" THEN GOTO na
CHAIN "games3.bas"

FUNCTION MouseButtons%
MouseButtons% = a%(2)
END FUNCTION

SUB MouseHide
DEF SEG = VARSEG(a%(0))
CALL absolute(VARPTR(a%(0)) + &H3A)
DEF SEG
END SUB

SUB MouseInit
DEF SEG = VARSEG(a%(0))
CALL absolute(VARPTR(a%(0)) + 6)
DEF SEG
END SUB

SUB MouseReset
DEF SEG = VARSEG(a%(0))
CALL absolute(VARPTR(a%(1)) + &H1E)
DEF SEG
END SUB

SUB MouseShow
DEF SEG = VARSEG(a%(0))
CALL absolute(VARPTR(a%(0)) + &H34)
```

```
DEF SEG
END SUB

FUNCTION MouseX%
MouseX% = a%(0)
END FUNCTION

FUNCTION MouseY%
MouseY% = a%(1)
END FUNCTION
```

ЧАСТЬ 2
Решения

Во второй части приведены ответы к заданиям, по образцам которых вы сможете выполнить остальные упражнения.

Арифметика в Бейсике

Задача 5

```
10*(25+11)^.5
```

Задача 6

```
1/(2+3/(4+5/(6+7/8)))
```

Задача 12

$$3+\frac{8}{4}-7\times 3^2$$

Задача 19

1

Задача 20

0

Оператор присваивания

Задача 21

Неверно. Слева от знака равенства не может стоять выражение — только переменная.

Задача 22

Верно.

Задача 23

Нет. Нет знака равенства.

Задача 24

Неверно. То же, что и в задаче 21.

Задача 25

Верно.

Задача 26

Неверно. Нет знака равенства.

Задача 27

Верно.

Задача 28

Верно.

Задача 29

Неверно. Переменные связаны двумя знаками равенства.

Задача 30

Нет. Слева от знака равенства стоит арифметическое выражение.

Задача 31

```
S = (A + B + C) / 3
```

Задача 32

```
S = ((X1 — X2) ^ 2 + (Y1 — Y2) ^ 2) ^ (1/2)
```

Задача 33

```
C = (A ^ 2 + B ^ 2) ^ (1/2)
```

Задача 34

```
P = (X + Y + Z) /2
S = (P*(P — A)*(P — B)*(P — C)) ^ (1/2)
```

Задача 35

```
R=-R
```

Задача 36

```
X=6, Y=6
```

Задача 37

Необходимо ввести вспомогательную переменную `A`.

```
A=X
X=Y
Y=A
```

Задача 38

```
W = K\100 + (K MOD 100)\10 + (K MOD 100) MOD 10
```

Задача 39

```
S=3*10^5*365*24*60*60
```

Задача 40

```
V = A * B * C
```

Задача 41

```
RUB = DOL * 28
```

Выводим результаты

Задача 42

Будет 4,5, т. к программа сначала (по приоритету действий) возведет 9 в степень 1, а потом результат поделит на 2.

Задача 43

```
PRINT 5 ^ (1/2)
```

Задача 44

```
PRINT (12^2 + 5^2) ^ (1/2)
```

Задача 45

```
PRINT ((3^(1/4)+2)^(1/3))^.5
```

Задача 46

```
PRINT 6^(1/4)/((5-8^.5)^3)
```

Задача 47

```
PRINT 87 ^ -.25
```

Задача 48

```
PRINT (((3^.5+3)^.5+3)^.5+3)^.5
```

Задача 51

Решение для последнего варианта:

```
CLS: PRINT "Результат равен"; 2*3*4*5*6
```

Задача 52

```
CLS: PRINT "Печать":PRINT: PRINT "через":PRINT:PRINT "строку"
```

Стандартные функции Бейсика

Задача 54

```
SQR (1 + Z^3)
```

Задача 55

```
ABS (X + B*Y)
```

Задача 56

```
COS(X^3)^2
```

Задача 59

```
(X^7+7^X)^(1/7)
```

Задача 60

```
(A+SIN(B^2)^3)/(COS(25)+ABS(1/TAN(60)))
```

Задача 61

$$\frac{-R+\sqrt{S^2-6AB}}{3A}$$

Задача 62

$$\frac{X}{Y}(D-F)+\frac{\frac{X+Y}{D}}{F}$$

Задача 63

$$\frac{X1+\mathrm{tg}(F2-V3)}{3\left|X2-\log 4*Y3\right|}\div e^{-2}$$

Вывод данных в заданном месте экрана

Задача 67

```
CLS
LOCATE 10, 35: ?"СУММА КУБОВ"
LOCATE 12, 38: ? 2^3+3^3+4^3+5^3
END
```

Задача 68

```
CLS
LOCATE 11, 35: ? "*********"
LOCATE 12, 35: ? "*СЕРЕДИНА*"
LOCATE 13, 35: ? "*********"
END
```

Вводим данные

Задача 70

```
CLS
INPUT "Введите, пожалуйста, Ваш год рождения"; YEAR1
INPUT "Введите, пожалуйста, год рождения Вашей мамы"; YEAR2
? "Ваша мама родила Вас, когда ей было"; YEAR2 — YEAR1; "лет"

END
```

Задача 71

```
CLS
INPUT "Введите свое имя"; NAME$
CLS
PRINT NAME$; ", введите существительное" : INPUT S$
PRINT NAME$; ", введите прилагательное" : INPUT P$
PRINT NAME$; ", введите наречие" : INPUT N$
PRINT NAME$; ", введите глагол" : INPUT G$
PRINT
PRINT S$; "  "; P$; "  "; N$; "  " G$
END
```

Задача 72

```
CLS
INPUT "Введите три стороны треугольника"; A, B, C
P=(A+B+C)/2
S=SQR(P*(P-A)*(P-B)*(P-C))
? "Площадь данного треугольника равна"; S
END
```

Задача 75

```
CLS
INPUT "Сколько страниц в книге"; P
INPUT "Сколько строк на странице"; L
INPUT "Сколько символов в каждой строке"; S
N = INT ((P*L*S)/1024/1024/1.44) + 1
PRINT "Вам потребуется "; N ; " дискет"
END
```

Задача 76

```
CLS
INPUT "Сколько строк в документе"; L
INPUT "Сколько символов в каждой строке"; S
N = (L*S)/1024 + 10*15*600*4/8/1024
PRINT "Документ занимает "; N ; " килобайт"
END
```

Задача 77

```
CLS
INPUT "Сколько мегабайт нужно для установки игры"; A
INPUT "Сколько есть мегабайт на жестком диске"; B
INPUT "Каков объем вашей карты памяти в мегабайтах"; V
N = INT((A — B) / V ) + 1
PRINT "Вам понадобится "; N ; " таких карт памяти"
END
```

Задача 78

```
CLS
INPUT "Сколько свободных гигабайт на жестком диске"; F
INPUT "Сколько страниц в книге"; P
INPUT "Сколько строк на странице"; L
INPUT "Сколько символов в каждой строке"; S
N = INT (F/(P*L*S/1024/1024/1024))
H = N * 3/100
PRINT "На ваш диск можно разместить "; N ; " таких книг"
PRINT "Сложенные в стопку они будут высотой "; H ; " метров"
END
```

Задача 79

```
CLS
INPUT "Какова скорость передачи данных в млн бит в секунду"; F
INPUT "Сколько минут качалась программа"; T
INPUT "Сколько копеек стоит один закачанный мегабайт"; K
N = F * T * 60/ 8/1024/1024/1024))
D = INT(N /1.44) +1
M = (N — 1) * 1024 *K /100
PRINT "Закачанная программа занимает"; N ; " гигабайт"
PRINT "Ее можно разместить на "; D ; " дискет"
PRINT "Трафик будет стоить "; M ; " рублей"
END
```

Задача 80

```
CLS
INPUT "Какова длина ребра куба в см"; A
SG = A^2
SP = 6 * SG
V = A^3
PRINT "Площадь грани куба "; SG ; " кв.см"
PRINT "Площадь боковой поверхности куба "; SP ; " кв.см"
PRINT "Объем куба "; V ; " куб.см"
END
```

Операторы *DATA* и *READ*

Задача 81

```
' Исходные данные для прямоугольников
' В каждом DATA координаты диагоналей и цвет
DATA 30, 50, 130, 150, 1
DATA 160, 100, 280, 170, 4
DATA 300, 180, 450, 270, 14
' Считывание данных и построение прямоугольников
SCREEN 9
READ X1, Y1, X2, Y2, C
LINE (X1, Y1)-(X2, Y2), C, BF
READ X1, Y1, X2, Y2, C
LINE (X1, Y1)-(X2, Y2), C, BF
```

```
READ X1, Y1, X2, Y2, C
LINE (X1, Y1)-(X2, Y2), C, BF
END
```

Линейный алгоритм

Задача 83

```
' Полет к ближайшей звезде
CLS
' Длина светового года
S = 3*10^5*365*24*60*60
' Количество секунд полета
NSEC = S/100
' Количество часов полета
NHOUR = NSEC/3600
' Количество суток полета
NDAYS = NHOUR/24
' Количество лет полета
NYEARS = NDAYS/365
' Вывод результата
LOCATE 25, 12
?"С такой скоростью звездолет долетит до звезды за"; NYEARS;
"лет"
END
```

Задача 87

```
' Притча о шахматах
CLS
' Расчет производим по формуле суммы геометрической прогрессии
S = 2^64 — 1
PRINT " Понадобится "; S ;"зерен"
END
```

Задача 88

```
' Расплата
YEARS = INT ((2^64-1) / (9E7*10^6/5))
PRINT "Для расплаты потребуется "; YEARS ;" лет"
END
```

Задача 89

```
' Деньги в банке
CLS
M=52000
N1=M+M*.01           ' Первый месяц
N2=N1+N1*.01         ' Второй месяц
N3=N2+N2*.01         ' Третий месяц
N4=N3+N3*.01         ' Четвертый месяц
N5=N4+N4*.01         ' Пятый месяц
N6=N5+N5*.01         ' Шестой месяц
N7=N6+N6*.01         ' Седьмой месяц
N8=N7+N7*.01         ' Восьмой месяц
N9=N8+N8*.01         ' Девятый месяц
N10=N9+N9*.01        ' Десятый месяц
N11=N10+N10*.01      ' Одиннадцатый месяц
N12=N11+N11*.01      ' Двенадцатый месяц
' Вывод результата
PRINT "Через год у меня на счету будет"; N12; "$"
END
```

Конечно, красивее было бы решение с циклами. Когда их изучите, решите эту задачу снова.

Задача 91

```
' Стрельба
' Напишем программу, не зависящую от исходных данных
CLS
INPUT "Какова начальная скорость снаряда (м/с)?"; VN
INPUT "Каков угол наклона ствола орудия к горизонту (град)?"; A
L = VN^2*SIN (2*A*3.14/180) / 9.81
PRINT "Снаряд пролетит "; L ; " метров"
END
```

Графика в Бейсике

Задача 94

```
' Российский флаг
CLS
SCREEN 9
```

```
' Белая полоса
LINE (100, 50)-(500, 130), 15, BF
' Синяя полоса
LINE (100, 130)-(500, 210), 1, BF
' Красная полоса
LINE (100, 210)-(500, 290), 4, BF
END
```

Задача 101

```
' Летающая тарелка
SCREEN 9
' Основа тарелки
CIRCLE (320, 175), 200, 2, , , .3
' Рубка
CIRCLE (320, 70), 100, 2, , , .2
LINE (198, 125)-(220, 70), 2
LINE (420, 70)-(440, 125), 2
' Иллюминаторы
CIRCLE (150, 175), 10, 4
CIRCLE (490, 175), 10, 4
CIRCLE (200, 200), 10, 4
CIRCLE (440, 200), 10, 4
CIRCLE (320, 220), 10, 4
' Ноги
LINE (320, 235)-(320, 280), 2
LINE (197, 223)-(165, 310), 2
LINE (443, 223)-(475, 310), 2
END
```

Задача 105

```
' Гистерезис
SCREEN 9
' Горизонтальные составляющие
LINE (300, 126)-(500, 126), 13
LINE (100, 274)-(300, 274), 13
' Левая верхняя четверть дуги
CIRCLE (300, 200), 100, 13, 1.57, 3.14
' Левая нижняя четверть дуги
CIRCLE (100, 200), 100, 13, 4.71, 0
```

```
' Правая верхняя четверть дуги
CIRCLE (500, 200), 100, 13, 1.57, 3.14
' Правая нижняя четверть дуги
CIRCLE (300, 200), 100, 13, 4.71, 0
' Закраска
PAINT (300, 200), 14, 13
END
```

Задача 109

```
' Мороженое
SCREEN 12
WINDOW (0, 0)-(640, 480)
' Заливка фона
LINE (0, 0)-(640, 480), 9, BF
' Подставка
LINE (200, 40)-(440, 40), 15
CIRCLE (200, 160), 120, 15, 4.71, 0
CIRCLE (440, 160), 120, 15, 3.14, 4.71
PAINT (320, 80), 8, 15
' Чаша
CIRCLE (320, 280), 120, 15, -3.14, -6.28
PAINT (320, 240), 8, 15
' Левый шарик мороженого
CIRCLE (260, 280), 60, 15, 0, 3.14
PAINT (260, 310), 4, 15
' Правый шарик мороженого
CIRCLE (380, 280), 60, 15, 0, 3.14
PAINT (380, 310), 2, 15
' Верхний шарик мороженого
CIRCLE (380, 280), 60, 15, 0, 3.14
PAINT (380, 310), 2, 15
END
```

Задача 112

```
' Елочка
SCREEN 9
PRESET (320, 300)
DRAW "C2 L60 E26 L25 E26 L25 E55 F55 L25 F26 L25 F26 L60"
```

```
' Закраска
PAINT (320, 250), 2
' Ствол
LINE (300, 300)-(340, 340), 6, BF
END
```

Задача 114

```
' Площадь круга
INPUT "Введите радиус окружности (от 10 до 100)"; R
SCREEN 12
CIRCLE (320,240), R, 15
PAINT (320, 240), 1,15
S = 3.14*R^2
LOCATE 15, 37: PRINT INT(S*100)/100
END
```

Задача 118

```
' Цилиндр
SCREEN 12
INPUT "Введите радиус основания цилиндра и его высоту"; R, H
WINDOW (0, 0)-(640, 480)
'Будем считать начальной точкой левой нижней точки 'высоты
цилиндра точку с координатами (100, 100)
LINE (100, 100)-(100, 100 + H)
LINE (100 + 2 * R, 100)-(100 + 2 * R, 100 + H)
CIRCLE (100 + R, 100), R, , 3.14, 6.28, .3
CIRCLE (100 + R, 100 + H), R, , , , .3
V = 3.14 * R ^ 2 * H
LOCATE 5, 10: PRINT "V="; Задача
END
```

Задача 119

Сначала напомню, что если известны координаты концов отрезка X1, Y1 и X2, Y2, то координаты точки середины такого отрезка можно вычислить как:

Xср = (X2 + X1)/2

Yср = (Y2 + Y1)/2

Дальнейшее очевидно:

```
' Координаты точки середины отрезка
INPUT "Введите координаты концов отрезка"; X1, Y1, X2,Y2
SCREEN 12
WINDOW (0, 0)-(640, 480)
Xmid = INT((X2+X1)/2)
Ymid = INT((Y2+Y1)/2)
LINE(X1,Y1) — (X2,Y2)
' Изображение середины отрезка в виде "узелка"
CIRCLE(Xmid, Ymid), 2
' Подписи координат
LOCATE 30 — INT(Y1/16), INT(X1/8): PRINT "X1="; X1; " Y1="; Y1
LOCATE 30 — INT(Y2/16), INT(X2/8): PRINT "X2="; X2; " Y2="; Y2
LOCATE 30 — INT(Ymid/16), INT(Xmid/8)+2: PRINT "Xmid="; Xmid;
" Ymid="; Ymid
END
```

Задача 120

Для начала вспомним формулу вычисления длины отрезка, зная координаты его концов X1, Y1, X2,Y2:

$$l = \sqrt{(X1 - X2)^2 + (Y1 - Y2)^2}$$

и формулу Герона для вычисления площади треугольника по его сторонам:

$$S = \sqrt{P(P-A)(P-B)(P-C)}\,,$$

где P — полупериметр.

А дальше — дело техники:

```
CLS
INPUT "Введите координаты вершин треугольника"; X1, Y1, X2,Y2,
X3, Y3
SCREEN 12

WINDOW (0, 0)-(640, 480
A=SQR((X1-X2)^2+(Y1-Y2)^2)
B=SQR((X1-X3)^2+(Y1-Y3)^2)
```

```
C=SQR((X3-X2)^2+(Y3-Y2)^2)
P=(A+B+C)/2
S=SQR(P*(P-A)*(P-B)*(P-C))
Xmid = INT((X2+X1)/2)
Ymid = INT((Y2+Y1)/2)
LINE(X1,Y1) - (X2,Y2)
LINE(X1,Y1) - (X3,Y3)
LINE(X2,Y2) - (X3,Y3)
' Подписи длин сторон
LOCATE 30 - INT(Y1+Y2)/2/16, (X1+X2)/2/8: PRINT "A="; A
LOCATE 30 - INT(Y1+Y3)/2/16, (X1+X3)/2/8: PRINT "B="; B
LOCATE 30 - INT(Y2+Y3)/2/16, (X2+X3)/2/8: PRINT "C="; C
' Вывод результатов
LOCATE 28, 35: PRINT "Периметр="; P*2
LOCATE 29, 35: PRINT "Площадь="; S
END
```

Задача 121

Расстояние, пройденное телом при равноускоренном движении, рассчитывается по формуле:

$$l = V_0 t + \frac{at^2}{2}$$

Пусть на исходные данные будут наложены ограничения (с тем, чтобы путь в экранных пикселах укладывался в ширину экрана). Начальная скорость — не более 50, время — не более 5, ускорение — не более 10.

Тогда программа будет иметь следующий вид:

```
CLS
INPUT "Введите начальную скорость, ускорение и время
движения"; V0, a, t
SCREEN 12
WINDOW (0, 0)-(640, 480)
L = V0*T + A*T^2/2
' Исходная точка "старта" пусть имеет координаты X=50, Y=50
LINE (50,50) - (50+L, 60), 4, BF
LOCATE 1, 10: PRINT "Пройдено расстояние "; L ; "пикселов"
END
```

Разветвляющийся алгоритм

Задача 127

```
' Бегущий узор
1 : PRINT "***"
PRINT "   *******"
PRINT "          *"
PRINT "          *"
PRINT "          *"
PRINT "   *******"
PRINT "***"
GOTO 1
END
```

Условный переход

Задача 132

```
' Чет-нечет
CLS
INPUT "Введите целое число"; N
IF N MOD 2 = 0 THEN LOCATE 12, 35: PRINT "Это число четное"
ELSE LOCATE 12, 35: PRINT "Это число нечетное"
END
```

Задача 133

```
' Определение меньшего из двух
CLS
INPUT "Введите два целых разных числа"; A, B
IF A<B THEN LOCATE 12, 35: PRINT "Меньшее число -->"; A ELSE
LOCATE 12, 35: PRINT " Меньшее число -->"; B
END
```

Задача 135

Переменная X примет следующие значения:

- X=1
- X=2
- X=0

Задача 136

а)

```
' Определение номера числа, отличного от двух других
CLS
INPUT "Введите три целых числа, два из которых одинаковы"; A,
B, C
IF A=B THEN LOCATE 12, 35: PRINT "Номер искомого числа --> 3";
A ELSE IF A=C THEN LOCATE 12, 35: PRINT "Номер искомого числа
--> 2" ELSE LOCATE 12, 35: PRINT "Номер искомого числа --> 1"
END
```

б)

```
' Вычисление функции
CLS
INPUT "Введите X"; X
IF X>0 AND X<2 THEN Y = COS(X)^2 ELSE Y = 1-SIN(X)^2
LOCATE 12, 35: PRINT "Y="; Y
END
```

Задача 137

```
' Нахождение суммы двух меньших чисел
CLS
INPUT "Введите три разных целых числа"; A, B, C
IF A>B AND A>C THEN LOCATE 12, 35: PRINT "Искомая сумма =";
B+C ELSE IF B>A AND B>C THEN LOCATE 12, 35: PRINT "Искомая
сумма ="; A+C ELSE LOCATE 12, 35: PRINT "Искомая сумма ="; A+B
END
```

Задача 138

```
S=6
```

Задача 145

И вновь формула Герона у нас на пути. Задание сложнее, может быть, чем вы ожидаете. Ну что ж, помогу. Сложность возникает при рисовании треугольника на экране. Условимся, что вершина A будет лежать на экране всегда в точке с координатами X1 = 100, Y1 = 100, а сторона *a* треугольника будет отходить вправо и лежать параллельно нижней стороне экрана, тогда Y1 = Y2, а X2 = X1 + a.

Путем несложных умозаключений можно вывести, что

$X3 = ((b^2 – c^2)/(X2 – X1) + X1 + X2)/2,$

А Y3 найдется из решения уравнения

$(Y3 – Y1)^2 = b^2 – (X3 – X1)^2.$

Итак, начали:

```
CLS
INPUT "Введите три стороны треугольника"; A, B, C
IF A+B<=C OR A+C<=B OR B+C<=A THEN
LOCATE 12,35: PRINT "ТАКОГО ТРЕУГОЛЬНИКА НЕ СУЩЕСТВУЕТ!!!":END
X1 = 100:Y1 = 100
X2 = X1+A: Y2 = Y1
X3 = ((B^2- C^2)/(X2-X1)+X1+X2)/2
Y3= SQR (C^2-(X3-X2)^2)+Y1
P =(A+B+C)/2
S = SQR(P*(P-A)+(P-B)+(P-C))
SCREEN 12
WINDOW(0,0)-(640,480)
LINE(X1,Y1)-(X2,Y2)
LINE(X1,Y1)-(X3,Y3)
LINE(X3,Y3)-(X2,Y2)
LOCATE 12,35: PRINT "ПЛОЩАДЬ ТРЕУГОЛЬНИКА РАВНА -->"; S
END
```

Задача 150

Переменная, в которой будет накапливаться произведение, вначале должна быть равна не нулю, а единице (на ноль ведь что ни умножь, ноль и получишь ☹).

Задача 161

```
' Старояпонский календарь
CLS
' Запрос года
INPUT "Введите год"; YEAR
' Вычисление номера цвета и животного
C =((9910-YEAR) MOD 60)\12
K =(YEAR-3) MOD 12
```

```
' Определение по номеру цвета года
SELECT CASE C
  CASE 0
    PRINT "ЗЕЛЕНЫЙ";
  CASE 1
    PRINT "КРАСНЫЙ";
  CASE 2
    PRINT "ЖЕЛТЫЙ";
  CASE 3
    PRINT "БЕЛЫЙ";
  CASE 4
    PRINT "ЧЕРНЫЙ";
END SELECT
' Определение по номеру животного года
SELECT CASE K
  CASE 0
    PRINT " КАБАН"
  CASE 1
    PRINT " КРЫСА"
  CASE 2
    PRINT " БЫК"
  CASE 3
    PRINT " ТИГР"
  CASE 4
    PRINT " КОТ"
  CASE 5
    PRINT " ДРАКОН"
  CASE 6
    PRINT " ЗМЕЯ"
  CASE 7
    PRINT " ЛОШАДЬ"
  CASE 8
    PRINT " ОВЦА"
  CASE 9
    PRINT " ОБЕЗЬЯНА"
  CASE 10
    PRINT " ПЕТУХ"
  CASE 11
    PRINT " СОБАКА"
END SELECT
PRINT "УДАЧИ ВАМ!"
END
```

Циклический алгоритм

Задача 164

```
' Пешеходный переход "зебра"
FOR Y = 10 TO 320 STEP 30
  LINE (200, Y)-(400, Y+20), 15, BF
NEXT Y
END
```

Задача 169

```
' Воздушный шарик
SCREEN 9
' Изменение цвета фона
COLOR , 1
FOR I = 1 TO 201
  ' Рисование окружности изменяющегося радиуса
  CIRCLE (320, 175), I, 14
  ' Пауза
  FOR T = 1 TO 2000: NEXT T
  ' Стирание окружности изменяющегося радиуса
  CIRCLE (320, 175), I, 1
NEXT I
END
```

Задача 171

```
' числа первой сотни, оканчивающиеся на 5
CLS
FOR I=1 TO 95
IF I MOD 10 = 5 THEN PRINT I;
NEXT I
END
```

Задача 172

```
F=1
```

Цикл вообще не будет ни разу исполняться, т. к. начальное значение параметра больше конечного, а шаг по умолчанию принят +1.

Задача 173

```
' Да ты крут!
CLS
INPUT "Введи свой возраст"; N
FOR J = 1 TO N
  PRINT "ДА ТЫ КРУТ!"
  FOR T = 1 TO 5000: NEXT T
NEXT J
END
```

Задача 174

```
' Таблица умножения
1:INPUT "Введите целое число от 2 до 9"; N
IF N<2 OR N>9 THEN ?"От 2 ДО 9!": GOTO 1
? "Таблица умножения на "; N
FOR I = 2 TO N
  ? I; "*"; N; "="; I*N
NEXT I
END
```

Задача 177

```
' Электронный секундомер
FOR T = 1 TO 3600
  LOCATE 12, 38:? T
NEXT T
END
```

Задача 178

```
' Орнамент с полукольцами
SCREEN 9
' Задание радиусов окружностей
R1 = 20: R2 = 10
FOR X = 50 TO 600 STEP 4*R1
  ' Первое полукольцо
  CIRCLE (X, 175), R1, 14, 0, 3.14
  CIRCLE (X, 175), R2, 14, 0, 3.14
  LINE (X — R1, 175)-(X — R2, 175), 14
  LINE (X + R1, 175)-(X + R2, 175), 14
  PAINT (X, 162), 2, 14
```

```
  ' Второе полукольцо
  CIRCLE (X + 2 * R1, 175), R1, 14, 3.14, 0
  CIRCLE (X + 2 * R1, 175), R2, 14, 3.14, 0
  LINE (X + R1, 175)-(X + R1 + R2, 175), 14
  LINE (X + 2 * R1 + R2, 175)-(X + 3 * R1, 175), 14
  PAINT (X + 2 * R1, 188), 4, 14
NEXT X
END
```

Задача 181

Вычислим наибольший общий делитель по алгоритму Евклида:

1. Вычислим r — остаток от деления числа a на b, a = bq + r, 0 <= r < b.
2. Если r = 0, то b есть искомое число.
3. Если r ≠ 0, то заменим пару чисел (a, b) парой (b, r) и перейдем к шагу 1.

```
CLS
INPUT "введите два целых числа"; a, b
while (a<>0) and (b<>0)
if (a>=b) then a=a mod b else b=b mod a
nod=a+b
wend
print "НАИБОЛЬШИЙ ОБЩИЙ ДЕЛИТЕЛЬ ДАННЫХ ЧИСЕЛ РАВЕН"; nod
END
```

Задача 182

- ❒ Правильно, если переменные `R`, `S` и `H` ранее получили значения.
- ❒ Правильно.
- ❒ Правильно.
- ❒ Неправильно. Отсутствует ключевое слово `TO`.
- ❒ Правильно.
- ❒ Неправильно. Цикл выполняться не будет, т. к. начальное значение параметра больше конечного, а шаг неотрицателен.
- ❒ Неправильно, т. к. оператор `WEND` не должен содержать параметр.

Задача 183

```
WAR
WAR
WAR
SUNDAY
PEACE
```

Так как после окончания цикла параметр имеет значение большее конечного значения параметра, то проверяемое условие `R=w` ложно, следовательно, выполняется следующий оператор печати слова "SUNDAY", а затем "PEACE".

Задача 184

ЧЕРТ ПОБЕРИ!!!

МИХАИЛ СВЕТЛОВ

Задача 195

Будут бесконечно печататься нули. Значение `S` не меняется, поэтому цикл будет длиться бесконечно.

Задача 197

```
X=INT(RND(1)*10)+1
```

Задача 198

```
X=INT(RND(1)*11)-5
```

Задача 199

```
X=INT(RND(1)*11)+10
```

Задача 200

```
X=INT(RND(1)*51)+50
```

Задача 201

```
X =-INT(RND(1)*101)+65
```

Задача 205

```
' Маленький взрыв
RANDOMIZE TIMER
SCREEN 12
FOR I=1 TO 100
  X=INT(RND(1)*200)+220
  Y=INT(RND(1)*200)+75
  C=INT(RND(1)*15)+1
  LINE (X, Y)-(320, 240), C
NEXT I
END
```

Задача 207

```
'Угадайка
CLS
RANDOMIZE TIMER
INPUT "КАК ВАС ЗОВУТ, УВАЖАЕМЫЙ -->"; N$
PRINT "Здравствуйте, уважаемый "; N$
PRINT "ПОПРОБУЙТЕ ЗА 3 ПОПЫТКИ УГАДАТЬ ЗАДУМАННОЕ МНОЮ ЧИСЛО"
X = INT (RND(1)*10)+1
FOR I=1 TO 3
PRINT "ПОПЫТКА НОМЕР "; I
PRINT "ВВЕДИТЕ ВАШЕ ЧИСЛО "; N$;
INPUT S
IF X=S THEN PRINT "ПОЗДРАВЛЯЮ ВАС, "; N$; ", ВЫ — ЯСНОВИДЕЦ!":
END
NEXT I
PRINT "НУ, НИЧЕГО, ДОРОГОЙ "; N$ ;", В СЛЕДУЮЩИЙ РАЗ ВСЕ
ПОЛУЧИТСЯ..."
PRINT "А Я ЗАДУМАЛ ЧИСЛО "; X
END
```

Задача 218

```
' Построение графика Y=SQR(X)
SCREEN 12
LINE (0, 240)—(640, 240), 15        Rem ось абсцисс
LINE (320, 0)—(320, 480), 15        Rem ось ординат
DEF FNY(X)=SQR(X)
```

```
FOR X = 320 TO 640
  Y = 240—30*FNY*((X—320)/30)
  PSET (X, Y), 15
NEXT X
END
```

Пояснение: корень квадратный извлекается только из неотрицательных чисел, следовательно, аргумент функции `(X-320)/30)` не должен быть меньше нуля, что и достигается его начальным значением 320.

Задача 220

```
' Поверхность вращения функции y = 1/1+x2
SCREEN 12
LINE (0, 240) — (640, 240), 15        Rem ось абсцисс
LINE (320, 0) — (320, 480), 15        Rem ось ординат
DEF FNY(X) = 1/(1+x^2)
FOR X = 320 TO 640 STEP 5
  Y = 240 — 30*FNY((X — 320)/30)
  CIRCLE (X, 240), 240 — Y,15,,,3
NEXT X
END
```

Задача 226

```
' Лестница 1
SCREEN 12
FOR X = 0 TO 640 STEP 40
  LINE(X, X*48/64)-(X + 40, (X + 40)*48/64),15,B
NEXT X
END
```

Так как экран представляет собой не квадрат, а прямоугольник, то, чтобы серия прямоугольников располагалась вдоль диагонали, приходится вводить коэффициент соотношения сторон экрана 480/640 или 48/64.

Задача 227

```
' Пирамида. Вид сверху
SCREEN 12
```

```
FOR X = 20 TO 220 STEP 10
  LINE (X, X)-(220-X, 220-X),15, B
NEXT X
END
```

Задача 228

```
' Четыре рупора
Screen 12
For x = 0 To 320 Step 5
Circle (x, 3 * x / 4), x / 5
Next
For x = 0 To 320 Step 5
Circle (640 — x, 3 * x / 4), x / 5
Next
For x = 0 To 320 Step 5
Circle (x, 480 — 3 * x / 4), x / 5
Next
For x = 0 To 320 Step 5
Circle (640 — x, 480 — 3 * x / 4), x / 5
Next
END
```

Задача 230

```
' Лоскутный ковер
SCREEN 12
RANDOMIZE TIMER
FOR X = 0 TO 640 STEP 10
  FOR Y = 0 TO 480 STEP 10
   ' Выбор случайного цвета для закраски кв. лоскута
   C = INT(RND(1)*15) + 1
   ' Рисование закрашенного квадратика
   LINE (X, Y)-(X + 10, Y + 10), C, BF
   ' Обводка квадратика белым контуром
   LINE (X, Y)-(X + 10, Y + 10), 15, B
  NEXT Y
NEXT X
END
```

Задача 233

```
' Шахматная доска
screen 12
n = 2
For x = 50 To 400 Step 50
For y = 50 To 400 Step 50
If n Mod 2 = 0 Then
Line (x, y)-(x + 50, y + 50), 4, BF
Line (x, y)-(x + 50, y + 50), , B
Else
Line (x, y)-(x + 50, y + 50), 14, BF
Line (x, y)-(x + 50, y + 50), , B
End If
n = n + 1
Next
n = n + 1
Next
END
```

Задача 235

```
' Хронометр
CLS
' Цикл для часов
FOR HOURS = 0 to 23
  ' Цикл для минут
  FOR MINUTES = 0 to 59
    ' Цикл для секунд
    FOR SECONDS = 0 to 59
      LOCATE 12, 35
      ? HOURS; ":"; MINUTES; ":"; SECONDS
      SLEEP 1
    NEXT SECONDS
  NEXT MINUTES
NEXT HOURS
END
```

Задача 238

```
' Вычисление числа Пи методом Монте-Карло
CLS : RANDOMIZE TIMER
LOCATE 3, 20
```

```
PRINT " Вычисление числа Пи методом Монте-Карло"
SCREEN 12
' Рисование квадрата со стороной 200
LINE (100, 100)-(300, 300), 14, B
' Рисование вписанной в квадрат окружности радиусом 100
CIRCLE (200, 200), 100, 0, , , 1.01
' Задание общего количества песчинок N
' и обнуление счетчика попавших в круг KN=5000: K=0
FOR I=1 TO N
  X=INT(RND(1)*200)+100
  Y=INT(RND(1)*200)+100
' Вычисление расстояния от выпавшей точки до центра
  S =SQR((X-200)^2+(Y-200)^2)
' Проверка условия, попала ли точка в круг
' и изображение ее красным или белым цветами
IF S<=100 THEN K=K+1: PSET (X, Y), 4 ELSE PSET (X, Y), 15  '
Вывод на экран счетчиков песчинок
LOCATE 5, 20: PRINT "Выпала песчинка №"; I
  LOCATE 6, 20: PRINT "В круг уже попало"; K; "песчинок"
NEXT I
' Вычисление и вывод на экран значения числа Пи
LOCATE 23, 15: PRINT "Число ПИ для"; N; "точек ="; 4*K/N
END
```

Задача 239

```
' Числа, делящиеся на 13
CLS
S = 0
FOR I=1 TO 1000
IF I MOD 13=0 THEN PRINT I; :S=S+I
NEXT I
PRINT "СУММА ИСКОМЫХ ЧИСЕЛ РАВНА "; S
END
```

Задача 242

```
' Сумматор
CLS
' Запрос с клавиатуры количества суммируемых чисел
INPUT "Сколько чисел будем складывать"; N
```

```
' Обнуление переменной для накапливания суммы
S = 0
FOR I=1 TO N
  ? "Введите"; I; "число";
  INPUT A
  S=S+A
NEXT I
' Вывод результата на экран
?"Сумма ваших"; N; "чисел равна"; S
END
```

Задача 245

```
' Нахождение суммы ряда
CLS
' Запрос значений переменных A и B
INPUT "Введите значения A и B"; A, B
' Обнуление переменной для накапливания суммы
S=0
I=0
WHILE (A+I)<>B
  S=S+1/(1+(A+I)^2)
  I=I+1
WEND
' Вывод результата на экран
?"Сумма ряда равна"; S
END
```

Задача 246

```
' Соприкасающиеся окружности
' Занесение значений радиусов
DATA 50, 20, 10, 25, 40, 15, 60
SCREEN 12
' Задание начальной координаты X
X=0
FOR I=1 TO 7
  READ R
  CIRCLE (X+R, 240), R, 14
  X=X+2*R
NEXT I
END
```

Задача 250

```
' Разноцветные окружности. Вариант 1
' Программа с циклом DO...LOOP WHILE
SCREEN 12
' Задание значения первого цвета
C=1
DO
  ' Определение координаты X центра окружности
  X=10+40*C
  CIRCLE (X, X), 30, 15
  PAINT (X, X), C, 15
  C=C+1
LOOP WHILE C<7
END

' Разноцветные окружности. Вариант 2
' Программа с циклом DO...LOOP UNTIL
SCREEN 12
' Задание значения первого цвета
C=1
DO
  ' Определение координаты X центра окружности
  X=10+40*C
  CIRCLE (X, X), 30, 15
  PAINT (X, X), C, 15
  C=C+1
LOOP UNTIL C=7
END
```

Задача 257

```
input "введите любое натуральное число "; m
k=0
n=m
do
x=m mod 10
k=k+1
m=(m-x)/10
Loop while m>0
print "кол-во цифр в десятичной записи числа"; n;"=";k
END
```

Символы и строки

Задача 296

```
' Шифровка
CLS
' Запрос количества букв в шифруемом слове
INPUT "Сколько букв в слове"; N
' Объявление массива для зашифрованного слова
DIM D$(N)
FOR I=1 TO N
  ' Побуквенный запрос исходного слова
  PRINT "Введите"; I; "букву";
  INPUT C$
  ' Побуквенная шифровка
  D$(I)=CHR$(ASC(C$)+1)
NEXT I
' Вывод на экран зашифрованного слова
FOR I = 1 TO N
  PRINT D$(I);
NEXT I
END
```

Обратите внимание, что при введении буквы `Z` в результате получается не буква, а символ квадратной скобки. Как избежать этого? Подумайте сами.

Задача 303

```
' Палиндром
CLS
' Запрос слова с клавиатуры
INPUT "Введите ваше слово"; W$
' Определение длины введенного слова
N=LEN(W$): W1$=""
' Цикл переворачивания исходного слова задом наперед
FOR I=N TO 1 STEP -1
  W1$=W1$+MID$(W$, I, 1)
NEXT I
```

```
' Вывод исходного и получившегося слов
PRINT "Исходное слово — "; W$
PRINT "Исходное слово наоборот — "; W1$
' Вывод на экран результата
IF W$=W1$ THEN PRINT "Да, это палиндром" ELSE PRINT "Нет, это
не палиндром"
END
```

Обратите внимание, что программа для фразы "А РОЗА УПАЛА НА ЛАПУ АЗОРА" не сработает. Разберитесь, почему, и внесите необходимые поправки.

Задача 305

```
' 10 слов
DATA STUFF, CAR, SKI, QUIKE, SUN
DATA CARD, PRESS, POOH, GULI, COP
CLS
PRINT "Все слова исходного списка"
FOR I=1 TO 10
  READ W$
  PRINT W$; " ";
NEXT I
PRINT : PRINT
RESTORE
PRINT "Слова списка, отличные от слова SUN"
FOR I=1 TO 10
  READ W$
  IF W$<>"SUN" THEN PRINT W$; " ";
NEXT I
PRINT : PRINT
RESTORE
' Определение слова, ближайшего к началу алфавита
READ W$
ALFA$=W$
FOR I=1 TO 9
  READ W$
  IF W$<ALFA$ THEN ALFA$=W$
NEXT I
PRINT "Слово в списке, ближайшее к началу алфавита — "; ALFA$
```

```
PRINT
RESTORE
' Определение слова, составленного из последних букв
' всех слов списка
R$=""
FOR I=1 TO 10
  READ W$
  R$=R$+RIGHT$(W$, 1)
NEXT I
PRINT "Слово, составленное из последних букв всех слов
списка — "; R$
PRINT
RESTORE
PRINT "Все трехбуквенные слова из списка"
FOR I=1 TO 10
  READ W$
  IF LEN(W$)=3 THEN PRINT W$; " ";
NEXT I
END
```

Задача 309

```
' Замена буквосочетаний
DATA PHOTO, GRAPH, PHILOPHON, COPHE
CLS
FOR I=1 TO 4
  READ W$
  FOR J=1 TO LEN(W$)
    P$=MID$(W$, J, 1): H$=MID$(W$, J + 1, 1)
    IF P$<>"P" THEN PRINT P$;  ELSE IF H$<>"H" THEN PRINT H$;
ELSE
    PRINT "F"; : J=J+1
  NEXT J
  PRINT
NEXT I
END
```

Задача 336 (б)

```
' Столбиковая интерпретация массива
CLS : RANDOMIZE TIMER
INPUT "Сколько чисел будет в массиве"; N
```

```
DIM S(N)
' Формирование массива
FOR I=1 TO N
  S(I)=INT(RND(1)*150)+50
NEXT I
' Графическая интерпретация массива
SCREEN 12
X=100
FOR I=1 TO 10
  ' Столбик, интерпретирующий элемент массива
  LINE (X, 300)-(X+10, 300-S(I)), 2, BF
  ' Белая рамка, обрамляющая столбик
  LINE (X, 300)-(X+10, 300-S(I)), 15, B
  X=X+10
NEXT I
' Вывод элементов массива на экран
FOR I=1 TO N
  PRINT S(I);
NEXT I
END
```

Задача 338

```
' Вычисление среднего арифметического и среднего квадратичного
элементов заданного массива
DATA 31, 19, 52, 65, 6, 8, 13, 16, 97, 33
CLS
DIM S(10)
' Формирование массива и вывод его элементов на экран
FOR I=1 TO 10
  READ S(I)
  ? S(I);
NEXT I
' Вычисление среднего арифметического элементов массива
T=0
FOR I=1 TO N
  T=T+S(I)
NEXT I
TR=T/N
```

```
' Вычисление среднего квадратичного элементов массива
T=0
FOR I=1 TO N
  T=T+(S(I)-TR)^2
NEXT I
SRKV=S/N
? "Среднее арифметическое элементов массива = "; TR
? "Среднее квадратичное элементов массива = "; SRKV
END
```

Задача 342

```
' Вывод массива в строки по 6 элементов, начиная с последнего
CLS: RANDOMIZE TIMER
INPUT "Сколько чисел будет в массиве"; N
DIM S(N)
' Формирование массива и вывод его на экран
FOR I=1 TO N
  S(I)=INT(RND(1)*90)+10
  ? S(I);
NEXT I
?: ?
' Вывод массива в строки по 6 элементов, начиная с последнего
I = N
WHILE I<>6
  FOR I=N TO N-5 STEP -1
    ? S(I);
  NEXT I
  N=N-6
  ?
WEND
END
```

Задача 345

```
' Столбиковая интерпретация массива, его максимума и минимума
CLS : RANDOMIZE TIMER
' Объявление и формирование массива
INPUT "Сколько чисел будет в массиве"; N
DIM S(N)
```

```
FOR I=1 TO N
  S(I)=INT(RND(1)*150)+50
NEXT I
' Поиск максимального элемента массива и его индекса
MAX=S(1): NMAX=1
FOR I=2 TO N
  IF S(I)>MAX THEN MAX=S(I): NMAX=I
NEXT I
' Поиск минимального элемента массива и его индекса
MIN=S(1): NMIN=1
FOR I=2 TO N
  IF S(I)<MIN THEN MIN=S(I): NMIN=I
NEXT I
' Графическая интерпретация массива
SCREEN 12
X=100
FOR I=1 TO 10
  LINE (X, 300)-(X+10, 300-S(I)), 14, BF
  IF I = NMAX THEN LINE (X, 300)-(X+10, 300-S(I)), 4, BF
  IF I = NMIN THEN LINE (X, 300)-(X+10, 300-S(I)), 2, BF
  LINE (X, 300)-(X+10, 300-S(I)), 15, B
  X=X+10
NEXT I
' Вывод на экран элементов массива, его максимума и минимума
FOR I=1 TO N
  PRINT S(I);
NEXT I
PRINT
PRINT "Максимальный элемент массива — "; MAX; "Его индекс — ";
NMAX
PRINT "Минимальный элемент массива — "; MIN; "Его индекс — ";
NMIN
END
```

Задача 347

```
' Формула Герона
DATA 13, 48, 35, 90, 62
DATA 35, 40, 20, 71, 55
DATA 27, 68, 41, 54, 29
```

```
CLS
DIM A(5), B(5), C(5), P(5), S(5)
FOR I=1 TO 5: READ A(I): NEXT I
FOR I=1 TO 5: READ B(I): NEXT I
FOR I=1 TO 5: READ C(I): NEXT I
FOR I=1 TO 5
  P(I)=A(I) + B(I) + C(I)
  PP=P(I)/2
  S(I)=SQR (PP*(PP-A(I))*(PP-B(I))*(PP-C(I)))
  ? "Периметр"; I; "треугольника равен"; P(I); "см"
  ? "Площадь"; I; "треугольника равна"; S(I); "кв.см"
  ?
NEXT I
END
```

Задача 351

```
' Расстояния между точками
DATA 120, 58, 280, 440, 157, 99, 350, 290, 500,159
DATA 271, 279, 35, 100, 160, 234, 45, 170, 200, 66
DATA 271, 279, 35, 100, 360, 234, 545, 170, 200, 266
DATA 120, 58, 280, 240, 157, 99, 150, 290, 50, 159
SCREEN 12
' Объявление и формирование массивов
DIM X1(10), Y1(10), X2(10), Y2(10), S(10)
FOR I=1 TO 10: READ X1(I): NEXT I
FOR I=1 TO 10: READ Y1(I): NEXT I
FOR I=1 TO 10: READ X2(I): NEXT I
FOR I=1 TO 10: READ Y2(I): NEXT I
' Формирование массива расстояний между точками
FOR I=1 TO 10
  S(I)=SQR((X1(I)-X2(I))^2+(Y1(I)-Y2(I))^2)
NEXT I
' Поиск максимальной и минимальной длины
MAX=S(1): NMAX=1
FOR I=2 TO 10
  IF S(I)>MAX THEN MAX=S(I): NMAX=I
NEXT I
MIN=S(1): NMIN=1
```

```
FOR I=2 TO 10
  IF S(I)<MIN THEN MIN=S(I): NMIN=I
NEXT I
' Графическая интерпретация
FOR I=1 TO 10
  LINE (X1(I), Y1(I))-(X2(I), Y2(I)), 15
  IF I=NMAX THEN LINE (X1(I), Y1(I))-(X2(I), Y2(I)), 4
  IF I=NMIN THEN LINE (X1(I), Y1(I))-(X2(I), Y2(I)), 2
NEXT I
END
```

Задача 356

```
' Сборщики компьютеров
' Понедельник
DATA 37, 14, 48, 24, 80, 60, 56
' Вторник
DATA 77, 34, 45, 23, 45, 39, 51
' Среда
DATA 58, 65, 49, 49, 56, 45, 38
' Четверг
DATA 61, 57, 55, 89, 33, 52, 60
' Пятница
DATA 80, 67, 77, 70, 55, 76, 81
CLS
' Объявление и формирование массивов для каждого сборщика
DIM A(5), B(5), C(5), D(5), E(5), F(5), G(5), S(7), SR(5)
FOR I=1 TO 5
  READ A(I), B(I), C(I), D(I), E(I), F(I), G(I)
NEXT I
' Общее количество компьютеров за неделю для каждого сборщика
FOR J=1 TO 7: S(J)=0: NEXT J
J=1: FOR I=1 TO 5: S(J)=S(J)+A(I): NEXT I
J=2: FOR I=1 TO 5: S(J)=S(J)+B(I): NEXT I
J=3: FOR I=1 TO 5: S(J)=S(J)+C(I): NEXT I
J=4: FOR I=1 TO 5: S(J)=S(J)+D(I): NEXT I
J=5: FOR I=1 TO 5: S(J)=S(J)+E(I): NEXT I
J=6: FOR I=1 TO 5: S(J)=S(J)+F(I): NEXT I
J=7: FOR I=1 TO 5: S(J)=S(J)+G(I): NEXT I
```

```
' Максимальное количество за неделю одним сборщиком
MAX=S(1): K=1
FOR J=2 TO 7
  IF S(J) > MAX THEN MAX=S(J): K=J
NEXT J
PRINT "Максимальное количество компьютеров за неделю:"; MAX
PRINT "собрал рабочий №"; K
PRINT
' Среднее количество за каждый день недели
FOR I=1 TO 5
  SR(I)=(A(I)+B(I)+C(I)+D(I)+E(I)+F(I)+G(I))/7
  PRINT "В"; I; "день в среднем собрано"; SR(I); "компьютеров"
NEXT I
PRINT
' Лучший результат за один день, номер сборщика и день недели
RESTORE
READ DAY
MX=DAY: K=1
FOR I=2 TO 35
  READ DAY
  IF DAY>MX THEN MX=DAY: K=I
NEXT I
PRINT "Максимальное количество компьютеров за один день:";
MX
N=K MOD 7
N1=K/7
IF N1<=1 AND N1>0 THEN D$="Понедельник"
IF N1<=2 AND N1>1 THEN D$="Вторник"
IF N1<=3 AND N1>2 THEN D$="Среда"
IF N1<=4 AND N1>3 THEN D$="Четверг"
IF N1<=5 AND N1>4 THEN D$="Пятница"
PRINT "собрал рабочий №"; N
PRINT "Это был день недели "; D$
END
```

Задача 374

```
' Сортировка выбором
CLS : RANDOMIZE TIMER
INPUT "Сколько чисел будет в массиве"; N
```

```
' Объявление, формирование массива и вывод его на экран
DIM S(N)
FOR I=1 TO N
  S(I)=INT(RND(1)*150)+50
  PRINT S(I);
NEXT I
PRINT
' Сортировка с использованием вложенных циклов
FOR I=1 TO N-1
  MIN=S(N)
  K=N
  FOR J=N TO I STEP -1
    IF S(J)<MIN THEN MIN=S(J): K=J
  NEXT J
  SWAP S(K), S(I)
NEXT I
' Вывод отсортированного массива
FOR I=1 TO N
  PRINT S(I);
NEXT I
END
```

Задача 378

```
' Сортировка методом "пузырька"
CLS
DATA Q,W,E,R,T,Y,U,I,O,P
DATA A,S,D,F,G,H,J,K,L,Z
DATA X,C,V,B,N,M
' Объявление, формирование и вывод исходного массива на экран
N=26: DIM ENG$(N)
PRINT "Исходный массив"
FOR I=1 TO N
  READ ENG$(I)
  PRINT ENG$(I); " ";
NEXT I
PRINT
FOR I=1 TO N-1
  FOR J=1 TO N-1
    IF ENG$(J)>ENG$(J+1) THEN SWAP ENG$(J), ENG$(J+1)
```

```
  NEXT J
NEXT I
PRINT
' Вывод отсортированного массива
PRINT "Отсортированный массив"
FOR I=1 TO N
  PRINT ENG$(I); " ";
NEXT I
END
```

Задача 382

```
' Точки в нижней правой четверти
DATA 10, 300, 246, 590, 341, 238, 480, 100, 378, 362
DATA 470, 235, 340, 20, 280, 346, 120, 66, 400, 167
DIM X(10), Y(10)
SCREEN 12
FOR I = 1 TO 10
READ X(I)
NEXT I
FOR I = 1 TO 10
READ Y(I)
NEXT I
K=1
FOR I=1 TO 10
IF X(I)>=320 AND Y(I)>=240 THEN CIRCLE(X(I),(Y(I)),1,4: K=K+1
ELSE CIRCLE(X(I),(Y(I)),1,14
NEXT I
LOCATE 12, 35: PRINT "ТАКИХ ТОЧЕК "; K
END
```

Задача 395

```
'Главная диагональ — нулевая
CLS
DIM A(7,7)
FOR I = 1 TO 7
FOR J = 1 TO 7
IF I=J THEN A(I,J)=1 ELSE A(I,J)=0
NEXT J, I
```

```
FOR I=1 TO 7
FOR J=1 TO 7
PRINT A(I,J);
NEXT J
PRINT
NEXT I
END
```

ЧАСТЬ 3
Дополнительные возможности

Здесь хотелось бы рассказать о возможностях Бейсика, позволяющих писать более красивые программы, работать с которыми пользователю было бы приятно, и в душе он поминал бы вас хорошими словами.

Экранные режимы: оператор *SCREEN*

Наиболее распространены сейчас видеосистемы VGA, а более всего SVGA. Поэтому рассматривать мы будем только их. Во многих школах, конечно, все еще можно встретить разношерстные HGA, CGA, EGA, но не будем ориентироваться на прошлое.

Основными характеристиками видеосистемы являются ее разрешение, т. е. количество экранных точек (что определяет качество выводимого изображения) и количество отображаемых цветов и оттенков. Для видеосистем VGA и SVGA характеристики перечислены в табл. 3.1.

Таблица 3.1. Характеристики видеосистем

Адаптер	Разрешение	Количество цветов
VGA (Video Graphics Array)	640×480	256
SVGA (Super Video Graphics Array)	До 1024×1024	До 16 миллионов

При запуске Бейсика автоматически загружается текстовый экранный режим `SCREEN 0`. А далее адаптеры VGA и SVGA под-

держивают режимы, представленные в табл. 3.2. Воспользовавшись этой таблицей, вы сможете устанавливать графический режим с помощью оператора SCREEN.

Таблица 3.2. Экранные режимы для адаптеров VGA и SVGA

Номер режима	Разрешение	Количество цветов
0	Текстовый режим	16
1	320×200	4
2	640×200	2
7	320×200	16
8	640×200	16
9	640×350	16
10	640×350	2
11	640×480	2
12	640×480	16
13	320×200	256

Цвет символов и цвет фона: оператор *COLOR*

Оператор COLOR имеет следующий синтаксис:

```
COLOR N1, N2
```

Предназначен он для изменения цвета символов на N1, а цвета фона (т. е. экрана) — на N2. Если в записи оператора опустить параметр N1, то он будет записываться так:

```
COLOR , N2
```

и станет изменять только цвет фона. Например:

```
CLS
SCREEN 9
FOR N2 = 0 TO 15
  COLOR , N2
  SLEEP 1
NEXT N2
```

Запустив данную программу, вы увидите с паузой в 1 секунду все возможные цвета из табл. 3.3, в которые будет окрашен фон.

Если опустить параметр `N2`, то оператор `COLOR` будет менять только цвет символов. Например:

```
SCREEN 9
FOR I = 0 TO 15
  COLOR I
  PRINT I
NEXT I
```

Таблица 3.3. Возможные цвета

Номер	Цвет
0	Черный
1	Синий
2	Зеленый
3	Голубой
4	Красный
5	Фиолетовый
6	Коричневый
7	Светло-серый
8	Темно-серый
9	Светло-синий
10	Светло-зеленый
11	Светло-голубой
12	Светло-красный
13	Васильковый
14	Желтый
15	Ярко-белый

Движущиеся изображения: операторы *GET* и *PUT*

Простой способ "оживить" экран состоит в исполнении операторов GET и PUT. Они позволяют "снять" изображение с части экрана и повторить его в другом месте. Это происходит очень быстро, поскольку исходная фигура не перерисовывается теми графическими операторами, которыми она вначале была создана, а просто дублируются все ее точки.

Представленная ниже программа рисует блоху, которая "скачет" по экрану, т. к. сначала оператор GET сохраняет ее изображение, а затем оператор PUT копирует это изображение в любое заданное место экрана. Ощущение движения возникает из-за того, что перед каждым "прыжком" блохи ее старое изображение стирается.

```
CLS
SCREEN 9
COLOR 9, 1
REM Рисование блохи
DRAW "C2 L20 U20 R20 D20 F10 D10 R2"
DRAW "BM300,175 C2 G10 D10 L2"
DRAW "BM300,155 C2 H10 E10 L2"
DRAW "BM320,155 C2 E10 H10 R2"
DRAW "BM310,155 C2 U4"
DRAW "BM307,151 C2 R10 H6 G6"
PAINT (310, 170), 2, 2
BYTES = INT((55 * 2 + 7) / 8) * 60
DIM BUG(BYTES)
GET (285, 135)-(340, 195), BUG
CLS
PUT (250, 80), BUG: SLEEP 1
PUT (250, 80), BUG: SLEEP 1
PUT (30, 60), BUG: SLEEP 1
PUT (30, 60), BUG: SLEEP 1
PUT (460, 130), BUG: SLEEP 1
PUT (460, 130), BUG: SLEEP 1
PUT (180, 70), BUG: SLEEP 1
PUT (180, 70), BUG: SLEEP 1
PUT (300, 70), BUG: SLEEP 1
END
```

Оператор *GET*

Прежде чем выполнить оператор GET, нужно провести важную подготовительную работу: определить местоположение и размер той области экрана, которую вы хотите снять. Большую помощь в этом вам окажет предварительный набросок на миллиметровой бумаге, где каждый квадратик сопоставляется с точкой экрана.

Оператор GET запоминает в массиве изображение заданной области экрана. Поэтому нужно заранее описать массив оператором DIM. Существует формула, определяющая требуемую длину массива исходя из размеров запоминаемой области и режима работы экрана. Формула для средней разрешающей способности экрана такова:

```
INT ((число точек по горизонтали * 2 + 7) / 8) * число точек
по вертикали
```

Формула для высокой разрешающей способности экрана:

```
INT ((число точек по горизонтали + 7) /8) * число точек
по вертикали
```

В строке программы, содержащей присваивание переменной BYTES, подсчитывается длина массива, необходимая для изображения блохи при средней разрешающей способности. Рисунок занимает примерно 55 позиций по горизонтали и 60 — по вертикали. Результат вычисления определяет длину массива BUG в следующей строке.

Оператор

```
GET (285, 135)-(340, 195), BUG
```

переписывает в массив BUG содержимое прямоугольной области экрана, левый верхний угол которой совпадает с точкой (285, 135), а правый нижний — с точкой (340, 195).

Оператор *PUT*

Теперь уже совсем просто "посадить" блоху в любое место экрана. Это делается с помощью оператора PUT. В строках, следующих за оператором GET программы, содержимое массива BUG раз-

мещается на экране с той позиции, которая указана в операторах PUT.

Так, оператор

```
PUT (250, 80), BUG
```

накладывает содержимое области экрана, снятое с помощью GET, на прямоугольник, левый верхний угол которого находится в точке (250, 80). Эффект наложения таков, что если два одинаковых оператора PUT следуют подряд, то второй нейтрализует действие первого: изображение сначала рисуется, а затем стирается. Таким образом, несколько сдвоенных операторов PUT создают иллюзию движения изображения.

Дополнительный параметр оператора PUT определяет способ наложения рисуемого изображения с тем, что уже есть на экране. Оператор с параметром PSET

```
PUT (250, 80), BUG, PSET
```

игнорирует прежнее изображение в данной области экрана.

Оператор

```
PUT (250, 80), BUG, PRESET
```

действует так же, но рисует негативное изображение, а оператор

```
PUT (250. 80), BUG, XOR
```

действует так же, как оператор без последнего параметра. Два таких оператора, следующих подряд, рисуют и тут же стирают изображение, оставляя экран неизменным.

Оператор

```
PUT (250, 80), BUG, AND
```

оставляет только общую часть нового и старого изображения.

Оператор

```
PUT (250, 80), BUG, OR
```

"добавляет" изображение к тому, что уже есть на экране. Поэкспериментируйте с оператором PUT и постарайтесь использовать предоставляемые им возможности.

Цвет точки экрана: функция *POINT*

Вы можете определить, какой цвет имеет в данный момент любая точка экрана, задав ее координаты в виде аргумента функции `POINT (X,Y)`. Например:

```
SCREEN 9
LINE (0, 0)-(100, 100), 4
LOCATE 10, 1
FOR y = 1 TO 10
  FOR x = 1 TO 10
    PRINT POINT(x, y);
  NEXT x
  PRINT
NEXT y
```

Данная программа рисует линию зеленого цвета по диагонали, начиная с верхнего левого угла, а затем во вложенных циклах поточечно сканирует область от точки с координатами (1, 1) до точки с координатами (10, 10) и выдает на экран массив составляющих ее цветов. Он будет выглядеть следующим образом:

```
2 0 0 0 0 0 0 0 0 0
0 2 0 0 0 0 0 0 0 0
0 0 2 0 0 0 0 0 0 0
0 0 0 2 0 0 0 0 0 0
0 0 0 0 2 0 0 0 0 0
0 0 0 0 0 2 0 0 0 0
0 0 0 0 0 0 2 0 0 0
0 0 0 0 0 0 0 2 0 0
0 0 0 0 0 0 0 0 2 0
0 0 0 0 0 0 0 0 0 2
```

Одиночный звуковой сигнал: оператор *BEEP*

Звуковое оформление можно осуществить с помощью оператора `BEEP`. Это, вероятно, самый простой, но весьма полезный оператор Бейсика. Он вызывает звуковой сигнал из динамика — гудок продолжительностью четверть секунды — и обеспечивает тот же

эффект, что и оператор PRINT CHR$(7). Несколько таких гудков подряд дадут длительный сигнал и привлекут, если требуется, ваше внимание к программе. Попробуйте поэкспериментировать со следующими операторами:

```
FOR B = 1 TO 10
  BEEP
NEXT B
```

Звуковое оформление: оператор *SOUND*

Запустите программу, приведенную ниже, и вы поймете, как работает оператор SOUND. Вы услышите звук той частоты и длительности, которые указаны как параметры оператора. Первый параметр X определяет частоту в диапазоне от 37 до 32 767 Гц, хотя высокие частоты едва ли будут приятны (если вообще доступны) для вашего слуха.

```
FOR S = 37 TO 3000 STEP 100
  SOUND X, 18
NEXT X
```

Второй параметр 18 задает длительность звучания, измеряемую в тактах часового механизма ("тиках"). Поскольку секунде соответствует 18,2 "тика", в нашем примере звук каждой частоты длится примерно одну секунду.

Обычно оператор SOUND не исполняется до тех пор, пока не "прозвучит" предыдущий оператор. Однако оператор SOUND с нулевой длительностью, сам по себе не вызывающий звука, прерывает работу предыдущего оператора SOUND, даже если тот не отзвучал до конца.

Пример программы, воспроизводящей звуки занятой телефонной линии:

```
' Как работает оператор SOUND
CLS
FOR S = 1 TO 10
  SOUND 25000, 3
```

```
  LOCATE 20, 11: ?" Занято! "
  SOUND 400, 5
NEXT S
```

Первый оператор SOUND воспроизводит звук очень высокой частоты, не воспринимаемой человеческим ухом, и прерывает звук, создаваемый вторым оператором SOUND.

Музыка в Бейсике: оператор *PLAY*

Оператор PLAY позволяет составлять музыкальные фрагменты и проигрывать их в различных сочетаниях в ходе выполнения программы. Как и оператор DRAW, оператор PLAY имеет свой набор команд для определения мелодии. С помощью этих команд вы можете задать темп, октаву, паузы и другие музыкальные характеристики. Команды объединяются в строку и присваиваются строковой переменной. Когда вы захотите услышать мелодию, обратитесь к оператору PLAY с командой

```
"X"+VARPTR(строковая переменная)
```

В представленной ниже программе оператор PLAY используется для последовательного проигрывания нот в семи октавах.

```
' Последовательно проигрывает все ноты в 7 октавах
scale$ = "CDEFGAB"
PLAY "L16"
FOR I = 0 TO 6
  PLAY "O" + STR$(I%)
  PLAY "X" + VARPTR$(scale$)
NEXT I
```

А вот еще один пример — всем известная мелодия "Happy birthday to you!".

```
CLS
PLAY "L8 MF MS O3 GG L4 AG O4 C O3 L2 B"
PLAY "L8 MF MS O3 GG L4 AG O4 D L2 C"
PLAY "L8 MF MS O3 GG L4 O4 GE L8 CC O3 L4 B A"
PLAY "O4 L8 FF L4 ECD L1 C"
END
```

Все команды оператора PLAY перечислены в табл. 3.4. Испытав их, вы убедитесь, что они предоставляют широкие возможности для "музицирования". Таково еще одно замечательное применение двоичного кода.

Таблица 3.4. Команды оператора PLAY

Команда	Действие
Буква от A до G	Исполнение ноты, обозначенной указанной буквой
Буква от A до G, за которой следует знак #, + или –	Исполнение ноты в соответствии с буквой, где знак # или + означают "диез", а знак — соответствует "бемоль"
Lx	Исполнение последующих нот с интервалом 1/x, где x меняется в диапазоне от 1 до 64
Ox	Играть в октаве x. Третья октава начинается со среднего "ми" (нота C). Всего имеется семь октав (с 0 по 6), расположенных между "ми" и "ре" (от C до B)
Px	Пауза длительностью 1/x (x от 0 до 64)
Tx	Задание темпа или четвертных нот, исполняемых в минуту (x может меняться от 32 до 255; по умолчанию равно 120)
Nx	Исполнение ноты x, которая может меняться от 0 до 84. Каждая октава имеет 12 нот, включая диезы и бемоли. 0 означает паузу
MN	Нормаль. Каждая нота звучит 7/8 времени, заданного в команде L
ML	Легато. Каждая нота звучит полный интервал, заданный в команде L
MF	Стаккато. Каждая нота звучит 3/4 времени, заданного в команде L
MS	Установка режима непосредственного исполнения. Каждая нота, заданная в операторах PLAY и SOUND, исполняется только после завершения предыдущей. Исполнение программы приостанавливается до окончания музыки. Этот режим принимается по умолчанию

Таблица 3.4 *(окончание)*

Команда	Действие
MB	Установка режима фонового исполнения. Каждая нота, заданная в операторах PLAY и SOUND, сохраняется в отдельном буфере и исполняется в процессе работы основной программы
. (точка)	Увеличение продолжительности звучания на 50%

Фоновая музыка — весьма эффективное средство, дополняющее визуальное восприятие работы компьютера. Она успокаивающе действует на нетерпеливого клиента, который вынужден ждать, скажем, окончания выполнения программы службы знакомств.

В очередной программе фоновая музыка делает поздравление из ранее рассмотренного примера более теплым. В операторах PLAY команда MF заменена командой MB (фоновое исполнение). В результате программа выводит на экран имя именинника, не дожидаясь окончания музыки. Музыка продолжается в процессе исполнения цикла FOR...NEXT, во время которого на экране пишется имя крокодила Гены.

```
' Поздравление с днем рождения!
CLS
PLAY "L8 MB MS O3 GG L4 AG O4 C O3 L2 B"
PLAY "L8 MB MS O3 GG L4 AG O4 D L2 C"
PLAY "L8 MB MS O3 GG L4 O4 GE L8 CC O3 L4 B A"
PLAY "O4 L8 FF L4 ECD L1 C"
FOR I = 1 TO 24
  LOCATE I, I
  PRINT "КРОКОДИЛ ГЕНА, С ДНЕМ РОЖДЕНИЯ!"
NEXT I
END
```

В следующей программе параметр X цикла FOR...NEXT внутри цикла используется для обозначения длительности звучания ноты G (соль) в третьей октаве.

```
FOR X = 1 TO 64
  PLAY "L" + STR$(X)
  PLAY "O3G"
NEXT
```

Поскольку аргумент команды `L` является знаменателем дроби, обозначающей длительность ноты, с возрастанием `X` длительность звучания становится все короче. Запустите программу и вы услышите музыку, напоминающую звук скачущего пластмассового шарика.

Приложение

Язык QBasic. Краткий справочник

Здесь мне хочется коротко представить наиболее часто используемые операторы Бейсика, снабдив их краткими комментариями (табл. П1—П10). Полная справка по языку имеется в его оболочке, где вы легко можете найти полные сведения о том или ином операторе с наглядными примерами.

Таблица П1. Числовые функции

Функция	Описание
ABS	Возвращает абсолютное значение (модуль) аргумента
ATN	Арктангенс (в радианах)
CDBL	Переводит числовое выражение в значение с двойной точностью
CINT	Округление
CLNG	Округление числового выражения до длинного (4 байта) целого значения
COS	Косинус
CSNG	Переводит числовое выражение в значение с одинарной точностью
EXP	Экспонента e^x
FIX	Округление выражения с плавающей запятой до его целой части
INT	Возвращает наибольшее целое, меньшее либо равное числовому выражению

Таблица П1 *(окончание)*

Функция	Описание
LOG	Натуральный логарифм числового выражения
RND	Случайное число одинарной точности между 0 и 1
SCN	Возвращает значения знака числового выражения (1, если выражение положительное; 0, если равно 0 и –1, если отрицательно)
SIN	Синус
SQR	Корень квадратный
TAN	Тангенс

Таблица П2. Функции графики и экрана

Функция	Описание
CIRCLE	Рисование окружностей и эллипсов
CLS	Очистка текстовой и графической областей экрана
COLOR	Установка цветов экрана
CSRLIN	Возвращает текущую позицию строки курсора
DRAW	Рисование объектов при помощи набора специальных команд
GET	Сохранение в памяти графической прямоугольной области экрана
LINE	Рисование отрезка прямой линии, прямоугольников со сторонами, параллельными экрану
LOCATE	Перемещение курсора в указанную позицию
PAINT	Закрашивание замкнутого контура
PALETTE	Изменение установок атрибутов цвета
PALETTE USING	Изменение атрибутов цвета пользователем
PCOPY	Копирование одной страницы видеопамяти в другую
PRESET	Рисование точки цветом фона
PRINT	Вывод данных на экран или в файл

Таблица П2 *(окончание)*

Функция	Описание
PRINT USING	Осуществляет форматированный вывод данных на экран или в файл
PSET	Рисование точки
PUT	Вывод на экран образа, сохраненного оператором GET
SCREEN	Установка режима и характеристик экрана
VIEW	Определяет размер и положение области просмотра, в которую может быть выведена графическая информация
VIEW PRINT	Устанавливает на экране границы области просмотра текста
WIDTH	Изменение числа строк и столбцов, видимых на экране
WINDOW	Определяет логическое пространство для текущей графической области просмотра
WRITE	Запись данных на экран или в последовательный файл

Таблица П3. Операторы выбора и перехода

Оператор	Описание
GOTO	Безусловный переход на метку
IF ... THEN ... ELSE	Переход в зависимости от истинности или ложности проверяемого условия
SELECT CASE	Переход в зависимости от значения выражения

Таблица П4. Операторы и функции для работы с файлами

Оператор, функция	Описание
CLOSE	Закрывает один или несколько файлов или устройств
FIELD	Отводит место под переменные в буфере файлов прямого доступа

Таблица П4 *(окончание)*

Оператор, функция	Описание
FILEATTR	Возвращает информацию об открытом файле
GET	Считывает из файла в буфер прямого доступа или в переменную
INPUT #	Считывает данные из файла
IOCTL	Посылает управляющую строку драйверу устройства
LINE INPUT #	Считывает строку до 255 символов с клавиатуры или из файла
LOCK	Ограничивает или закрывает доступ к файлу при работе в сети
OPEN	Открывает файл или устройство
PRINT #	Записывает данные в файл
PRINT # USING	Записывает отформатированные данные в файл
PUT	Записывает содержимое переменной или буфера прямого доступа в файл
RESET	Закрывает все открытые файлы и устройства
SEEK	Устанавливает позицию файла для последующей записи или считывания
UNLOCK	Снимает ограничения, наложенные последним оператором LOCK
WRITE #	Записывает данные в последовательный файл

Таблица П5. Переменные

Конструкция	Описание
CLEAR	Закрывает все файлы, освобождает буферы файлов, очищает все общие переменные, устанавливает числовые переменные и массивы в ноль, устанавливает строковые переменные в ноль и инициализирует стек. Кроме того, CLEAR может изменять размер стека
CONST	Описывает одну или несколько символьных переменных

Таблица П5 *(окончание)*

Конструкция	Описание
DATA	Указывает значения данных для последующего считывания оператором READ
INPUT	Считывает входные данные с клавиатуры или из файла
LET	Присваивает значение выражения переменной
RANDOMIZE	Инициализирует генератор случайных чисел
READ	Считывает данные, указанные в операторе DATA
RESTORE	Восстанавливает считанные значения в операторе DATA
SWAP	Обменивает значения двух переменных

Таблица П6. Массивы

Конструкция	Описание
DIM	Оператор объявления массива
ERASE	Для статических массивов каждому элементу присваивается ноль. Для строкового — определяются строки нулевой длины. Для динамического — освобождает память, используемую массивом
OPTION BASE	Устанавливает нижнюю границу индекса массива
REDIM	Описывает или изменяет размер динамического массива

Таблица П7. Циклы

Оператор	Описание
DO ... LOOP	Повторяет блок операторов, пока условие верно, или пока оно не станет верным
END	Заканчивает программу, процедуру или блок
FOR ... NEXT	Цикл с параметром, с заранее известным числом повторений
WHILE ... WEND	Выполняет блок операторов, пока указанное условие верно

Таблица П8. Подпрограммы и функции

Оператор	Описание
CALL	Передает управление в процедуру типа SUB
DECLARE	Описывает процедуру типа FUNCTION или SUB
DEF FN	Определяет функцию
FUNCTION	Определяет процедуру FUNCTION
GOSUB	Переходит в подпрограмму и возвращается из нее
ON GOSUB	Выполняет переход к одной из нескольких подпрограмм в зависимости от выражения
RETURN	Возвращает из подпрограммы в основную программу
SUB	Определяет процедуру SUB

Таблица П9. Звуки и музыка

Оператор	Описание
BEEP	Генерирует звуковой сигнал через встроенный динамик
ON PLAY GOSUB	Обращение к подпрограмме, когда число нот в музыкальном буфере меньше определенного числа
PLAY	Воспроизводит музыкальные ноты
SOUND	Генерирует звук через встроенный динамик

Таблица П10. Строковые функции

Функция	Описание
CHR$	Возвращает ASCII-код первого символа в строковом выражении
DATE$	Возвращает текущую системную дату
INSTR	Возвращает позицию первого вхождения подстроки в строку
LCASE$	Переводит все символы строковой переменной в строчные буквы

Таблица П10 *(окончание)*

Функция	Описание
LEFT$	Возвращает указанное число символов слева строки
LEN	Возвращает длину строковой переменной в символах
MID$	Возвращает указанное число символов из середины строковой переменной, начиная с указанной позиции
RIGHT$	Возвращает указанное число символов справа строки
SPACE$	Возвращает строку пробелов
STRING$	Возвращает строку указанных символов
UCASE$	Переводит все символы строковой переменной в прописные буквы

Сообщения об ошибках и их коды

Коды сообщений об ошибках приведены в табл. П11. Для наиболее распространенных ошибок даются комментарии.

Таблица П11. Сообщения об ошибках

Код	Сообщение об ошибке
1	**NEXT без FOR** Для окончания цикла NEXT нет соответствующего заголовка FOR. Количество FOR и NEXT должно совпадать
2	**Синтаксическая ошибка** Оператор содержит грамматическую ошибку в написании ключевого слова или ошибку в пунктуации, либо есть непарные скобки или другие нарушения правил синтаксиса языка Бейсик
3	**RETURN без GOSUB** Для оператора возврата из подпрограммы RETURN нет соответствующего обращения к подпрограмме GOSUB
4	**Нет данных** В операторе DATA нет данных. Посчитайте количество данных в операторе DATA и количество считываний из него оператором READ. Посмотрите внимательно, не поставили ли вы при перечислении данных в каком-нибудь месте точку вместо запятой

Таблица П11 *(продолжение)*

Код	Сообщение об ошибке
5	**Неверный вызов функции** Возникает чаще всего при попытке извлечения квадратного корня из отрицательного числа или применения графических операторов без включения графического режима SCREEN. Вообще же подобное сообщение возникает при попытке вызова функции с недопустимым параметром
6	**Переполнение** Числовая переменная или строковая константа выходят за пределы допустимого диапазона. Проверьте и измените значение в случае необходимости
7	**Не хватает памяти**
8	**Метка не определена** Оператор GOTO или GOSUB пытаются осуществить переход на несуществующую метку
9	**Индекс вне режима** Сообщение возникает при работе с массивами, когда индекс какого-либо элемента массива превышает его объявленный в операторе DIM размер, а также в том случае, когда массив занимает в памяти объем более 64 Кбайт
10	**Повторяющееся определение**
11	**Деление на ноль** Выражение в знаменателе после подстановки соответствующих переменных и вычислений, видимо, обращается в ноль
12	**Ошибка в режиме управления**
13	**Ошибка ввода**
14	**В строке нет места**
16	**Слишком сложная строковая формула**
17	**Невозможно продолжить**
18	**Функция не определена** Возможно, используемая функция не определена оператором DEF FN, или допущена ошибка при определении или вызове функции
19	**Нет RESUME**
20	**RESUME без ошибки**

Таблица П11 *(продолжение)*

Код	Сообщение об ошибке
24	**Устройство в тайм-ауте**
25	**Ошибка устройства**
26	**FOR без NEXT** Для заголовка цикла FOR нет соответствующего окончания цикла NEXT. Количество FOR и NEXT должно совпадать
27	**Нет бумаги**
29	**WHILE без WEND** Для ключевого слова WHILE нет соответствующего слова WEND
30	**WEND без WHILE** Для ключевого слова WEND нет соответствующего слова WHILE
33	**Повторяющаяся метка** При расстановке меток допущен повтор одной и той же метки в разных местах программы
35	**Подпрограмма не определена** Сообщение возникает при попытке обращения к несуществующей подпрограмме
37	**Ошибка счетчика аргументов**
38	**Массив не определен** Попытка работать с элементами массива, который не был объявлен оператором DIM
40	**Необходима переменная**
50	**Переполнение FIELD**
51	**Внутренняя ошибка** Чаще всего неверная работа компьютера, реже — смысловые ошибки программы, не сразу различимые на первый взгляд. Часто требуется ручная прокрутка программы
52	**Плохое имя файла/плохой номер** Имя файла не соответствует требованиям DOS
53	**Файл не найден** При попытке обращения к файлу указано неправильное его имя или путь к нему
54	**Плохой режим файла**

Таблица П11 *(продолжение)*

Код	Сообщение об ошибке
55	**Файл уже открыт** Попытка повторного открытия файла или удаления открытого файла
56	**Оператор FIELD активен**
57	**Ошибка в/вв устройства** Ошибка устройства ввода/вывода, с которой не справляется DOS. Попробуйте посмотреть, все ли в порядке с аппаратной частью, т. е. внешними устройствами компьютера
58	**Файл уже существует** Попытка сохранить файл под именем уже существующего на диске файла
59	**Неверная длина записи**
61	**Диск заполнен** Диск, на который производится запись файла, не имеет достаточно места для этого. Надо освободить дисковое пространство, удалив что-нибудь менее важное
62	**Ошибка: введен конец файла**
63	**Неверный номер записи**
64	**Плохое имя файла** Имя файла не соответствует требованиям DOS
67	**Слишком много файлов**
68	**Устройство недоступно** В дисководе нет диска или он испорчен
69	**Переполнение буфера коммуникации** Попытка копирования в буфер слишком большого объема информации
70	**Нет разрешения**
71	**Ошибка формата диска** В дисководе нет диска или он испорчен
72	**Ошибка диска** В дисководе нет диска или он испорчен
73	**Недоступная возможность**

Таблица П11 *(окончание)*

Код	Сообщение об ошибке
74	**Переименование через диски**
75	**Ошибка доступа к пути/файлу**
76	**Путь не найден** При попытке обращения к файлу указано неправильное его имя или путь к нему

Коды ASCII

Стандартные и расширенные ASCII-коды опубликованы уже в сотнях книг, посвященных программированию. Более того, вы уже можете написать программу, выводящую их на экран. К тому же эти коды есть в справочной системе нашего Бейсика. Но все равно хочется почему-то иметь их перед глазами.

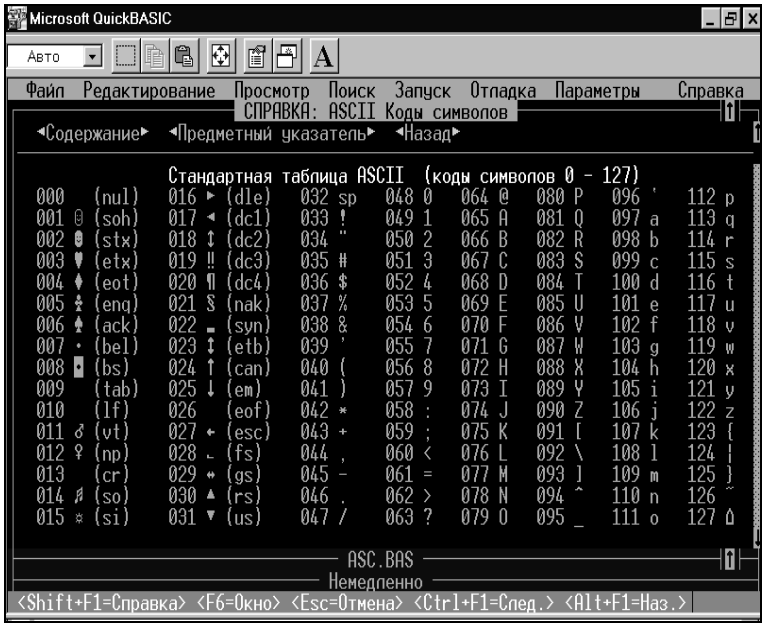


Рис. П1. Стандартные коды (0—127)

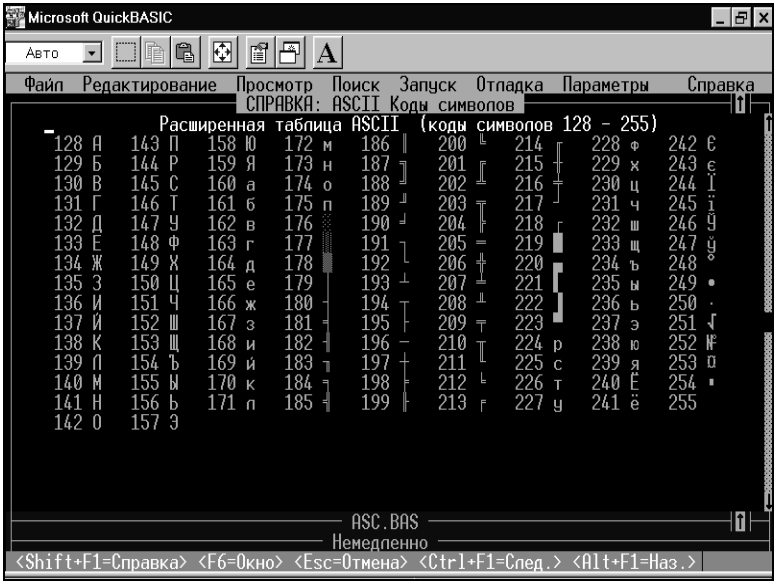


Рис. П2. Расширенные коды (128—255)

Недолго думая, я взял их из вышеупомянутой системы при помощи клавиши <Print Screen>, и вот — пользуйтесь, кому интересно (рис. П1, П2).

Список дополнительной литературы

1. Абрамов С. А., Зима Е. В. Начала информатики. — М.: Наука, 1989. — 256 с.
2. Арсак Ж. Программирование игр и головоломок. — М.: Наука, 1990. — 221 с.
3. Гусева В. А. Учимся информатике: задачи и методы их решения. — М.: ДИАЛОГ-МИФИ, 1999. — 320 с.
4. Задачи и упражнения по программированию / Под ред. А. Я. Савельева. — М.: Высшая школа, 1989. — 110 с.
5. Кушниренко А. Г. Основы информатики и вычислительной техники. — М.: Просвещение, 1990. — 223 с.
6. Мельникова О. И., Бонюшкина А. Ю. Начала программирования на языке QBasic. — М.: ЭКОМ, 2000. — 303 с.
7. Моррилл Г. Бейсик для ПК ИБМ. — М.: Финансы и статистика, 1993. — 207 с.
8. Персональный компьютер в играх и задачах / Под ред. Н. Н. Моисеева. — М.: Наука, 1988. — 191 с.
9. Пильщиков В. Н. Сборник задач по языку Паскаль. — М.: Наука, 1989. — 154 с.
10. Практикум по программированию / Под ред. А. Я. Савельева. — М.: Высшая школа, 1993. — 205 с.
11. Пярнпуу А. А. Программирование на современных алгоритмических языках. — М.: Наука, 1990. — 380 с.

12. Сборник задач по базовой компьютерной подготовке / Под общей ред. И. Н. Котаровой. — М.: МЭИ, 1998. — 177 с.
13. Филичев С. Ф. Информатика — это просто! — М.: ЭКОМ, 1999. — 343 с.
14. Чернов Б. И. Программирование на алгоритмических языках Бейсик, Фортран, Паскаль. — М.: Просвещение, 1991. — 190 с.
15. Элементы информатики / Под ред. Л. Н. Королева. — М.: Наука, 1988. — 318 с.