ВИКТОР ПЕСТРИКОВ
АРТУР МАСЛОБОЕВ

DELPHI

НА ПРИМЕРАХ

ОСНОВНЫЕ КОМПОНЕНТЫ

МЕТОДИКА НАПИСАНИЯ И ОТЛАДКИ ПРОГРАММ

ГРАФИЧЕСКИЕ ВОЗМОЖНОСТИ DELPHI

70 ПРИМЕРОВ С ПОДРОБНЫМИ РЕШЕНИЯМИ

bhv

УДК 681.3.06
ББК 32.973.26-018.2
П28

Пестриков В. М., Маслобоев А. Н.

П28 Delphi на примерах. — СПб.: БХВ-Петербург, 2005. — 496 с.: ил.

ISBN 5-94157-713-3

Изложены основы программирования в среде Delphi, начиная с составления программ в Turbo Pascal 7.0 и Object Pascal. Особое внимание уделено программам для решения задач из области высшей математики. Рассмотрены все этапы создания проекта в Delphi, начиная с разработки интерфейса и заканчивая особенностями работы с уже написанной программой. Приведены готовые проекты, которые должны помочь обучающемуся при выполнении самостоятельных заданий, помещенных в книге. Ко всем приведенным в книге заданиям имеются ответы и решения.

Для начинающих программистов, учащихся и студентов

УДК 681.3.06
ББК 32.973.26-018.2

Группа подготовки издания:

Главный редактор	*Екатерина Кондукова*
Зам. главного редактора	*Игорь Шишигин*
Зав. редакцией	*Григорий Добин*
Редактор	*Екатерина Капалыгина*
Компьютерная верстка	*Ольги Сергиенко*
Корректор	*Зинаида Дмитриева*
Дизайн обложки	*Игоря Цырульникова*
Зав. производством	*Николай Тверских*

Лицензия ИД № 02429 от 24.07.00. Подписано в печать 24.05.05.
Формат 70×100 1/16. Печать офсетная. Усл. печ. л. 39,99.
Тираж 4000 экз. Заказ №
"БХВ-Петербург", 194354, Санкт-Петербург, ул. Есенина, 5Б.

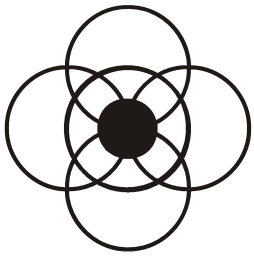
Санитарно-эпидемиологическое заключение на продукцию
№ 77.99.02.953.Д.006421.11.04 от 11.11.2004 г. выдано Федеральной службой
по надзору в сфере защиты прав потребителей и благополучия человека.

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

ISBN 5-94157-713-3

Оглавление

ГЛАВА 1



От алгоритма к программе

Решение задач на персональном компьютере представляет собой диалог между человеком и электронной машиной на языке программирования. Этот диалог будет успешным только тогда, если правильно понята поставленная задача, разработан план ее решения, и она представлена в виде отлаженной программы на выбранном языке программирования.

1.1. Что такое алгоритм?

Ежедневно каждому человеку, как в своей профессиональной деятельности, так и в жизни, приходится решать ряд возникающих перед ним задач. Опыт показывает, что решение проблемы будет происходить быстрее, успешнее и эффективнее, если у человека есть готовый план или рецепт решения этой задачи. Такой план может быть составлен человеком самостоятельно на основе собственных знаний и жизненного опыта, или при условии, что у него есть возможность воспользоваться рецептом решения проблемы, составленным кем-то ранее, и проверенной на практике. В любом случае использование такого плана при четком и неукоснительном соблюдении его положений должно привести, в конечном счете, к решению задачи. Такая последовательность четких однозначных указаний, разработанных для решения поставленной задачи, называется *алгоритмом*.

Одним из наиболее простых примеров алгоритма являются правила перехода улицы пешеходом. Их можно свести к нескольким пунктам.

1. Перед началом перехода улицы посмотреть налево и убедиться, что в непосредственной близости не находится движущийся транспорт.
2. В случае если вблизи пешехода находится движущийся транспорт, подождать, пока он проедет. Если транспорта нет, начать переход улицы и дойти до середины проезжей части.

3. Дойдя до середины проезжей части, посмотреть направо для проверки наличия или отсутствия в непосредственной близости движущегося транспорта.
4. В случае если вблизи находится движущийся транспорт, подождать, пока он проедет. При отсутствии транспорта можно завершить переход улицы.

Естественно, что когда человек, проживший достаточно длительное время в большом городе, переходит улицу, он необязательно должен вспоминать все указанные ранее правила, т. к. они уже будут запечатлены у него в мозгу на подсознательном уровне. Но при решении какой-либо более сложной задачи даже на бытовом уровне, особенно на первых порах, без тщательного соблюдения всех инструкций, изложенных "на бумажке", часто не обойтись. Примером такого более сложного алгоритма является инструкция по пользованию банкоматом, с помощью которого в настоящее время получают зарплату сотрудники многих организаций и предприятий. Такая инструкция состоит из следующих пунктов:

1. Вставить кредитную карточку в банкомат определенным образом (как правило, логотипом банка, изображенным на карточке, от себя).
2. Набрать индивидуальный цифровой код владельца карточки, после чего нажать клавишу **Ввод**.
3. Выбрать язык, на котором пользователь дальше будет общаться с банкоматом (как правило, в нашей стране банкоматы запрограммированы на работу с двумя языками: русским и английским), подтвердив выбор нажатием соответствующей клавиши.
4. Выбрать операцию, которую пользователь будет производить с банкоматом, из списка, приведенного на экране банкомата (это может быть получение денег, получение сведений о нескольких последних операциях, произведенных со счетом пользователя, получение информации справочного характера о работе сети банкоматов и т. д.), нажав клавишу, соответствующую выбранной операции.
5. В случае если пользователю необходимо получить деньги со счета, ответить на вопрос, хочет ли он получить также и чек, в котором будет указан размер полученной суммы и остаток денег на счете.
6. Ввести сумму, которую пользователь хочет получить со счета и подтвердить ее правильность нажатием клавиши **Ввод**. В случае если снимаемая пользователем со счета сумма превышает ту сумму, которая находится в данный момент на счете пользователя, ввести другую, меньшую по величине сумму.
7. Подождать несколько секунд, пока банкомат отсчитывает требуемую сумму, а затем, после появления разрешающей надписи на экране банко-

мата, вынуть из банкомата кредитную карточку, а затем забрать деньги и чек.

Такой же набор инструкций может быть с успехом использован для решения задач из различных областей науки и техники. В качестве простого примера приведем алгоритм нахождения среднего арифметического ряда, состоящего из нескольких чисел. Такой алгоритм состоит всего из двух пунктов.

1. Сложить все числа ряда.
2. Разделить полученную сумму на число членов ряда. Полученное частное и будет средним арифметическим.

Все три приведенные примеры алгоритмов относятся к разным сферам человеческой деятельности, однако можно заметить, что у них имеются некоторые общие свойства, которыми должен обладать любой работоспособный алгоритм.

- *Понятность* — тот, кто выполняет алгоритм, должен понимать, как на основе алгоритма и имеющихся исходных данных можно получить искомый результат.
- *Дискретность* — процесс решения задачи можно разбить на несколько этапов (пунктов), каждый из которых представляет собой некоторое законченное действие.
- *Определенность* — каждый пункт алгоритма должен быть сформулирован четко и однозначно и не оставлять места для произвольного толкования.
- *Результативность* — с помощью алгоритма задача должна быть решена за конечное число шагов.
- *Массовость* — алгоритм должен быть пригоден для решения целого класса задач, если такие задачи отличаются друг от друга только различными исходными данными.
- *Правильность* — решение задачи, полученное посредством использования алгоритма, должно соответствовать действительности.

Важной особенностью правильно составленного алгоритма является то, что он может выполняться не только человеком, но и любым устройством, которое в состоянии механически, не размышляя, выполнять указанные в алгоритме действия. Часто человека или подобное устройство называют *исполнителем алгоритма*. В качестве исполнителя, например, может выступать персональный компьютер. При этом, однако, нужно иметь в виду, что алгоритм, по которому работает компьютер, должен быть задан на понятном ему (компьютеру) языке, что требует иной записи алгоритма, нежели та, которая была использована в приведенных примерах. В данных примерах применялась словесная форма записи алгоритма, помимо которой используются еще две формы записи: графическая и в виде программы, написанной на какомлибо из языков программирования.

Графическая форма записи удобна для более компактного и наглядного представления различных элементов, из которых складывается алгоритм. При графическом представлении алгоритм изображается в виде последовательности связанных между собой блоков, каждый из которых соответствует выполнению одного или нескольких действий и изображается в виде определенной геометрической фигуры. Такой способ представления алгоритма называется *блок-схемой* или *графической схемой*, что более правильно, т. к. соответствует названию, употребляемому в государственном стандарте. Наиболее часто употребляемые блоки, входящие в состав графических схем, приведены на рис. 1.1.

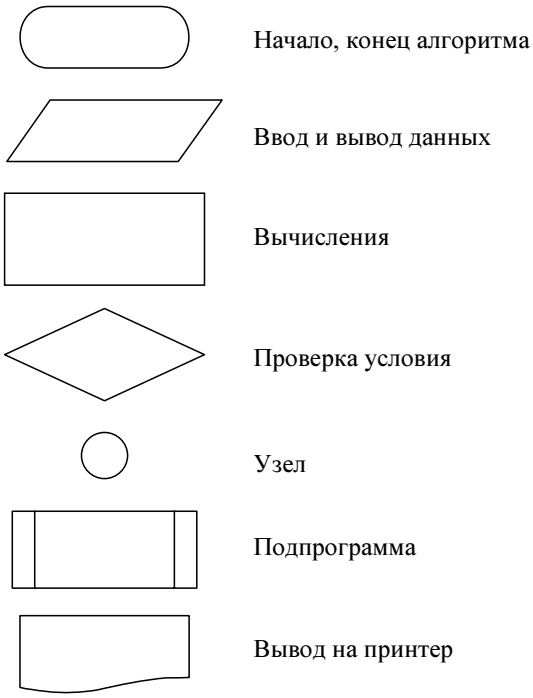


Рис. 1.1. Изображения основных блоков, используемых при составлении графических схем

Поясним назначение этих блоков. Блоки "начало" и "конец" используются для обозначения начала и конца алгоритма. Внутри обозначающей их фигуры пишется слово "начало" или "конец". Ввод и вывод соответственно используются для ввода исходных величин, необходимых для решения задачи, или вывода полученных результатов. Соответствующие величины указываются внутри параллелограмма, обозначающего ввод или вывод. Обработка данных (как правило, это какие-либо вычисления) изображается в виде прямоугольника. Внутри прямоугольника записывается содержание этих вычислений.

Проверка определенного условия изображается в виде ромба. Само условие записывается внутри ромба. В результате проверки условия осуществляется выбор одного из двух возможных путей дальнейшего выполнения алгоритма.

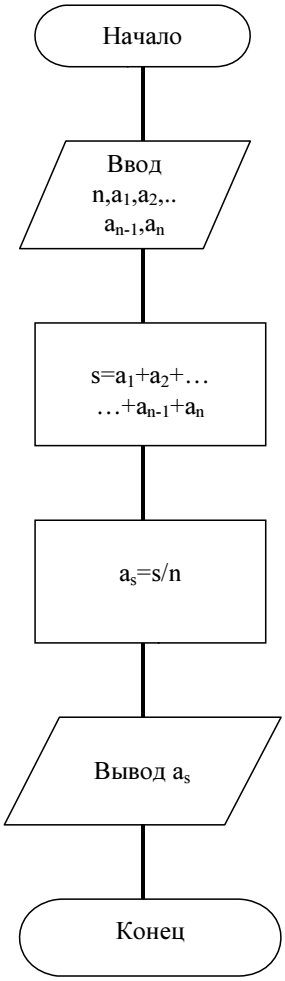


Рис. 1.2. Графическая схема нахождения среднего арифметического

Запишем в виде графической схемы процесс вычисления среднего арифметического (рис. 1.2). После блока, означающего начало алгоритма, находится блок ввода исходных данных, к которым в этом случае относятся n — число членов ряда и значения членов этого ряда a_1, a_2 и т. д. вплоть до последнего члена ряда a_n. Эти блоки, как и все входящие в состав графической схемы, объединяются друг с другом прямыми линиями. Следующий прямоугольный

блок представляет собой процесс вычисления — нахождение величины s, являющейся суммой членов ряда. Следующий блок также относится к разряду блоков обработки информации. На этот раз в нем производится операция деления, которая позволяет найти величину a_s — искомое среднее арифметическое членов ряда. В предпоследнем блоке осуществляется вывод полученного результата. Последний блок представляет собой конец алгоритма, о чем оповещает соответствующая надпись внутри него.

Приведенный способ записи является, как уже говорилось, более строгим и лаконичным, чем словесное описание алгоритма, и нередко используется на этапе подготовки к решению задачи на компьютере, но непосредственно в компьютер для получения искомого результата графическую схему вводить нельзя, так же как и словесный рецепт. Единственный способ описания алгоритма, который непосредственно понятен центральному процессору компьютера (центральный процессор — это компьютерное устройство, координирующее работу всех прочих устройств компьютера и выполняющее элементарные арифметические и логические операции) и с которым он может в дальнейшем работать, является *компьютерная программа*, записанная на одном из языков программирования. Об этих понятиях и пойдет речь в следующем разделе книги.

1.2. Программа и языки программирования

Итак, если попытаться дать в самом общем виде определение компьютерной программе, то можно сказать, что *программа* — это алгоритм, понятный компьютеру. Только язык, на котором написаны программы, понятен компьютеру, а при отсутствии программ компьютер из полезного, а во многих случаях незаменимого для современного человека устройства, превращается в абсолютно бесполезный набор микросхем и проводов. Процесс обработки информации, осуществляемый компьютером, требует наличия как *технических средств* (называемых часто англоязычным словосочетанием *hardware* или русскоязычным жаргонным словом "железо"), так и *программного обеспечения* (называемого *software* или "софт"). *Программированием* называется процесс создания компьютерной программы.

Единственным языком, который непосредственно понимает процессор компьютера, является *язык машинных кодов*, который представляет собой последовательность двоичных чисел, изображаемых нулями и единицами. Эта особенность языка машинных кодов объясняется причинами технического характера. Оказалось, что технически гораздо легче построить компьютер на базе элементов, каждый из которых может находиться в одном из двух устойчивых состояний, одно из которых соответствует нулю, а другое единице. Когда в 40-е годы XX века были созданы первые компьютеры, то машинный

код был единственным языком, на котором составлялись компьютерные программы. Естественно, что составление программ на таком языке является для человека крайне скучным и утомительным процессом, чреватым к тому же многочисленными ошибками. Еще одним существенным недостатком машинных кодов является то, что они привязаны к компьютерам определенной модели, конструкции. Созданные для компьютера одной конструкции программы в машинных кодах нельзя переносить без соответствующей переработки на компьютеры другой конструкции.

Уже через несколько лет после появления первых компьютеров для облегчения и упрощения работы программистов был создан *язык ассемблера*, в котором команды машинных кодов заменялись более понятными для человека. Такими командами в основном были сокращенные слова английского языка. Эти более удобные для запоминания команды называют *мнемоническими* или *мнемониками языка ассемблера*. Язык ассемблера до сих пор употребляется для составления программ в тех случаях, когда предъявляются повышенные требования к компактности программ. Программы, написанные на языке ассемблера, требуют минимального объема памяти и времени выполнения. Однако этот язык сохранил ряд существенных недостатков своего предшественника — языка машинных кодов. Это, во-первых, достаточно жесткая привязка языка к компьютерам определенной конструкции. Во-вторых, как и в случае с машинными кодами, язык ассемблера требует очень высокой квалификации программиста, знания не только нюансов программирования, но и знания тонкостей устройства компьютера, особенно в том, что касается архитектуры его центрального процессора. Поэтому язык ассемблера называют машинно-ориентированным языком. По отношению к машинным кодам и языку ассемблера употребляют термин *"языки низкого уровня"*, т. к. они максимально приближены к уровню технических устройств компьютера.

Следующим шагом в развитии программирования стало создание *алгоритмических языков высокого уровня*. Такие языки еще удобнее для понимания, чем команды ассемблера, т. к. они включают в себя обычные слова человеческого языка и общепринятые обозначения математических операций. Команды языков высокого уровня (называемые также операторами) зачастую напоминают фразы, составленные на английском языке. Надо сказать, что именно английский язык, являющийся в настоящее время языком международного общения, своего рода латынью XX—XXI веков, стал основой, на которой было создано подавляющее большинство языков программирования высокого уровня. Кроме того, языки высокого уровня стали машинно-независимыми, т. е. программы, написанные на таких языках, можно без труда переносить с одного компьютера на другой. Первым широко распространенным во всем мире языком программирования высокого уровня стал язык Фортран, созданный в первую очередь для решения различных научно-технических

задач. Название этого языка представляет сокращение английского словосочетания formula translation, что означает "перевод формул", т. е. имеется в виду перевод различных математических формул с обычного человеческого языка на язык, понятный компьютеру. За ним последовали такие языки высокого уровня, как Кобол, Алгол, Паскаль, Бейсик, Си и многие, многие другие.

Говоря о достоинствах языков программирования высокого уровня, надо иметь в виду, что процессор компьютера не может воспринимать эти языки непосредственно. Для того чтобы компьютер мог выполнять программы, написанные на этих языках, ему требуется перевод их в машинные коды. Такой перевод осуществляется с помощью специальных программ, называемых *трансляторами*. Трансляторы делятся на две основные категории. В одних случаях транслятор непосредственно переводит строку за строкой команды языка высокого уровня в машинные коды, которые немедленно выполняются. Такой способ трансляции называется *интерпретацией* и используется, например, для перевода в машинные коды программ, составленных на языке Бейсик. Для большинства других языков высокого уровня, включая язык Паскаль, используется другой способ трансляции, который заключается в том, что программа обрабатывается целиком, а затем на ее основе создается модуль в машинных кодах. Такой способ обработки программ называется *компиляцией*.

Каждый из этих способов трансляции имеет свои достоинства и недостатки. Составление программ на интерпретируемых языках часто бывает легче для программиста, т. к. многие ошибки в программе выявляются уже на стадии ввода ее текста в компьютер. После ввода очередной строки текста программы интерпретатор сразу выдает сообщение об ошибках (естественно, при наличии таковых). Однако за удобство программирования на интерпретируемом языке приходится расплачиваться быстродействием составленных на этом языке программ. Такая программа работает в несколько десятков раз медленнее, чем программа, переведенная в машинные коды посредством компилятора. Последнее обстоятельство, как правило, оказывается более важным, поэтому в большинстве случаев для создания программ, особенно на профессиональном уровне, используются компилирующие языки.

1.3. Этапы работы над программой. Система программирования

Из сказанного в предыдущем разделе понятно, что составление программ даже с помощью современных средств и языков программирования не является неким одномоментным процессом, а включает в себя ряд последовательных этапов.

Перечислим основные этапы:

1. **Постановка задачи.** Для того чтобы правильно составить программу, программисту нужно предварительно решить для самого себя, чего он хочет добиться, составляя программу, достаточно четко и конкретно сформулировать задачу, решаемую с помощью программы, определить, какие исходные данные необходимы для ее решения и какой требуется получить результат.
2. **Нахождение оптимального метода решения задачи и создание соответствующего алгоритма.** Для того чтобы поставленную задачу можно было успешно решить, программист должен вначале хотя бы в общих чертах составить план решения задачи, наметить наиболее надежный и эффективный путь ее решения. Часто бывает полезно перед тем, как начать составлять программу на компьютере, изобразить на бумаге ее алгоритм в виде графической схемы либо изложить в словесном виде основные пункты ее решения.
3. **Запись текста программы на языке программирования.** Этот процесс часто называют также кодированием программы, т. е. происходит перевод алгоритма в код, понятный компьютеру либо непосредственно (в случае языков низкого уровня), либо через переводчика — программу-транслятор (в случае языков высокого уровня). Для того чтобы можно было набирать текст программы, необходимо наличие на компьютере специальной программы — текстового редактора. *Текстовым редактором* называется программа, с помощью которой пользователь может осуществлять операции ввода и редактирования текста. Под редактированием текста понимается его правка, исправление обнаруженных в тексте ошибок. Наличие возможностей правки является обязательным элементом текстового редактора, т. к., во-первых, набор текста практически любой программы, особенно у неопытных пользователей на начальном этапе обучения программированию, не обходится без опечаток. Во-вторых, даже опытный программист часто в ходе составления программы осуществляет оптимизацию программы, т. е. делает уже работающую программу более качественной и эффективно работающей, что требует удаления определенных фрагментов программы и замены их новыми.

   Текст программы, уже набранный и откорректированный, находится до определенного момента в оперативной памяти компьютера. Если программа не является "одноразовой", необходимой для получения только один раз некоторых результатов (а такими, как правило, бывают только программы, написанные для тренировки, в учебных целях), то ее следует сохранить на жестком диске компьютера в виде отдельного файла. *Файлом* называется область на диске компьютера, имеющая свое собственное имя и служащая для хранения программ и данных. В противном случае

(если не сохранить программу в файле) сразу после выключения компьютера содержимое его оперативной памяти очистится, а вместе с ним пропадет и набранный пользователем текст программы.

4. **Проверка программы.** Данный этап начинается с того, что программа запускается на трансляцию. При этом программа-транслятор проверяет исходный текст программы (называемый также *исходным кодом*) на правильность с точки зрения орфографии и синтаксиса того языка программирования, на котором составлен текст данной программы. При обнаружении подобных ошибок транслятор выдает об этом соответствующее сообщение. В этом случае программа заведомо не будет работать, поэтому в случае выдачи сообщений об ошибках пользователь должен внести в программу соответствующие коррективы, а затем снова запустить ее на трансляцию. В том случае, если в тексте программы транслятором не было обнаружено явных ошибок, на основе исходного кода программы транслятор создает перевод текста программы в машинные коды, называемый *объектным модулем*.

   Однако объектный модуль еще не является готовой к выполнению на компьютере программой. Дело в том, что практически ни одна современная компьютерная программа не обходится без использования некоторых вспомогательных программ, разработанных ранее другими программистами. Такие вспомогательные программы называются *стандартными подпрограммами* и объединяются в специальные *библиотеки*. Подключение же этих библиотек производится с помощью специальной программы, называемой *редактором связей*. После подключения необходимых стандартных подпрограмм и создается готовая к выполнению программа, называемая *исполняемым модулем*.

5. **Отладка программы.** Нередко встречается ситуация, когда пользователь составил программу, в которой отсутствуют грамматические ошибки, но тем не менее программа не выдает желаемого результата, либо выдает результат в неправильном виде. Такая ситуация говорит о том, что в программе имеются семантические (смысловые) ошибки. Для исправления таких ошибок требуется не просто формальное знание правил языка программирования, как на предыдущем этапе, а наличие четкой логики мышления, знания основных принципов программирования и наиболее эффективных методов выявления ошибок, определенный опыт в составлении компьютерных программ. Однако и эти ошибки при наличии у пользователя терпения и желания довести работу до логического конца являются в конечном счете устранимыми, и тогда итогом работы над программой становится работоспособный окончательный вариант программы.

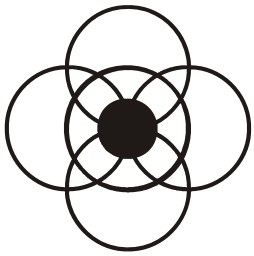
Если мы охватим мысленно все описанные этапы работы над программой, то станет ясно, что на первых двух этапах основным инструментом для работы

над программой является мозг программиста, позволяющий должным образом поставить задачу и найти способ ее решения. Однако на последующих этапах разработки программы имеется много рутинной работы, которую в значительной своей части способен выполнять не человек, а компьютер. Поэтому на основе опыта программирования на языках высокого уровня возникла идея дополнить транслятор рядом вспомогательных программ, облегчающих и упрощающих в значительной степени процесс создания и отладки программ, интегрировав эти программные средства в единый программный комплекс. Такой комплекс называется *интегрированной системой* или *средой программирования*. Данный комплекс должен включать в себя несколько основных элементов.

- *Встроенный текстовый редактор* — использование такого редактора гораздо удобнее, нежели подключение какого-либо внешнего. Ему можно, в частности, поручить такую задачу, как отмена одного или нескольких последних действий пользователя в том случае, если они были ошибочными.
- *Компилятор* — это основное ядро системы программирования. Компилятор может создавать как исполняемые модули в рамках самой системы программирования, так и модули, которые, будучи созданы в данной системе, могут работать независимо от нее.
- *Редактор связей* подключает требуемые стандартные библиотеки.
- *Программа-отладчик* выдает сообщения об обнаруженных ошибках, причем сообщает не только о самом факте ошибки, но и указывает характер сделанной ошибки, а в большинстве случаев также называет номер строки, в которой ошибка была обнаружена, или автоматически устанавливает на нее курсор.
- *Справочная система* содержит различную информацию как об основных структурах языка и правилах их использования с примерами программ, так и необходимую информацию о самой системе программирования.

Как уже было сказано в предисловии, подобная система программирования была разработана для языка Паскаль и получила название Turbo Pascal. Приставка Turbo означает "ускорение" и объясняется тем, что компиляция программы в данной системе происходила быстрее, чем в других аналогичных системах. Данная система позволяет успешно создавать достаточно сложные и в то же время эффективно работающие программы как в текстовом режиме работы компьютера, так и в графическом.

ГЛАВА 2



Подготовка к программированию на Паскале

Когда выбран язык для программирования поставленной задачи, возникает резонный вопрос, где взять этот программный продукт? Можно, например, систему программирования Turbo Pascal 7.0 купить на оптическом диске в компьютерном магазине, а можно, подключившись к Интернету, бесплатно ее "скачать" с соответствующего сайта. После того как программный продукт загружен и установлен, можно приступать к решению задач, предварительно изучив основы составления программ на этом языке.

2.1. Где взять Turbo Pascal

У человека, естественно, который ранее не занимался программированием, но хочет приступить к изучению языка программирования, возникает законный вопрос: "Откуда взять этот самый язык программирования?"

Для того чтобы ответить на данный вопрос, необходимо, прежде всего, сказать, что язык программирования неотъемлем от системы программирования. *Система программирования* представляет собой набор взаимосвязанных файлов, который может храниться либо на каком-то из дисковых носителей информации (компакт-диск или дискета), либо находиться на каком-то сайте в Интернете. Задача пользователя заключается в том, чтобы перенести этот набор файлов на жесткий диск (винчестер своего компьютера). Рассмотрим возможные варианты действий применительно к системе программирования Turbo Pascal 7.0.

Система программирования Turbo Pascal 7.0 невелика по объему (она занимает на диске около 2,5 Мбайт), поэтому, как правило, она записывается на имеющиеся в продаже компакт-диски не сама по себе, а вместе с другими программными комплексами. Часто на таких дисках можно обнаружить так называемые Rip-версии программных продуктов (в том числе и Turbo Pascal).

Rip-версия представляет собой сделанную пиратским способом копию того или иного программного продукта. При этом компьютерные пираты нещадно удаляют ненужные на их взгляд файлы, содержащиеся в программном комплексе (приставка Rip представляет собой не что иное, как сокращение английской фразы rest in peace, что в переводе означает "покойся с миром"). Делается это с очевидной целью — на одном и том же компакт-диске разместить как можно больше различных программ. Однако такая "забота" о пользователе может быть чревата тем, что программный комплекс будет неработоспособен. Возможен и другой вариант: в принципе программа будет запускаться и работать, но из-за отсутствия некоторых файлов пользователь сможет воспользоваться далеко не всеми ее возможностями.

Поэтому авторы книги рекомендуют начинающим пользователям переписать Turbo Pascal с одного из сайтов, где имеется нормальная, работоспособная версия данного продукта. Таким сайтом является, например, **Turbo Pascal**, расположенный по адресу **http://borlpasc.narod.ru**. Этот сайт, помимо самого Turbo Pascal, содержит еще много другой интересной и полезной для начинающего программиста информации, но подробнее об этом мы расскажем в другой главе данной книги, посвященной языку Turbo Pascal во Всемирной паутине. Сейчас же нас интересует в первую очередь процесс загрузки Turbo Pascal, о чем мы и расскажем далее.

Процедура загрузки Turbo Pascal рассматривается исходя из того, что у пользователя на компьютере установлена операционная система Windows и браузер Internet Explorer. Итак, набрав в адресной строке вашего браузера адрес **http://borlpasc.narod.ru** и нажав клавишу <Enter>, вы оказываетесь на главной странице сайта, которая показана на рис. 2.1.

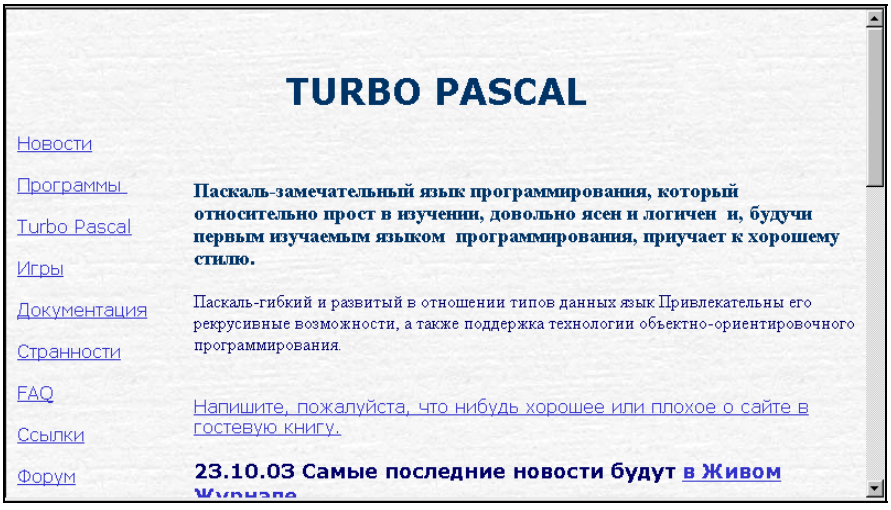


Рис. 2.1. Главная страница сайта **borlpasc.narod.ru**

Далее на главной странице нужно щелкнуть мышью гиперссылку, которая так и называется **Turbo Pascal**. В результате вы оказываетесь в разделе сайта, где содержатся различные версии языка Паскаль (рис. 2.2).

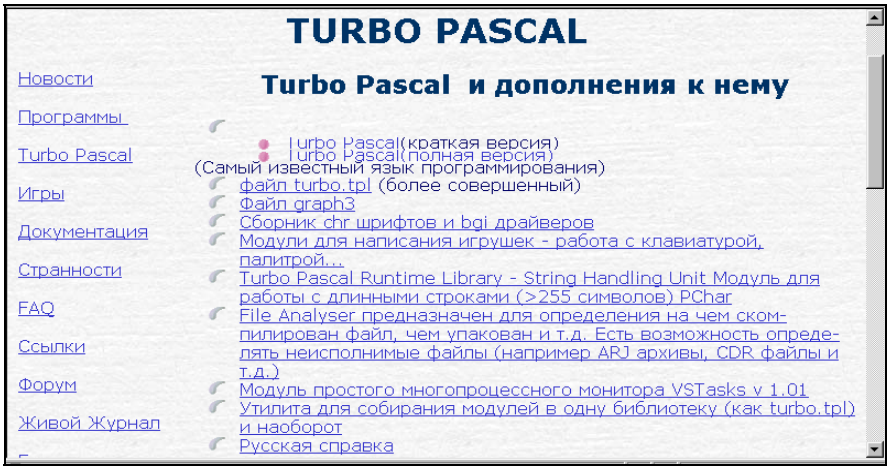


Рис. 2.2. Страница сайта **borlpasc.narod.ru**, содержащая различные версии языка Паскаль

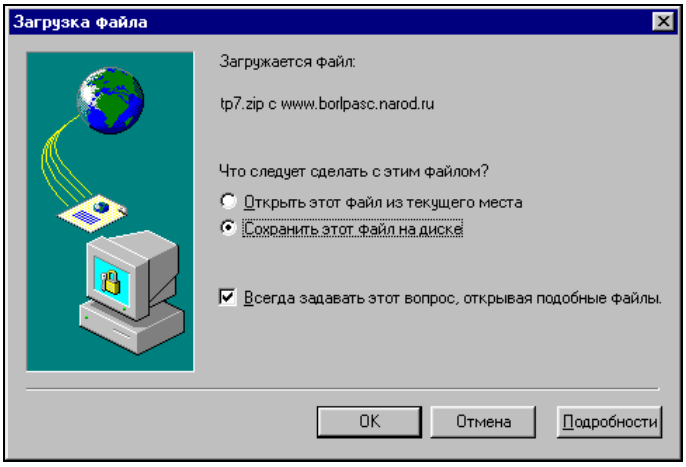


Рис. 2.3. Окно загрузки файла tp7.zip

Для наших целей лучше всего подойдет полная версия языка Turbo Pascal, поэтому щелкаем соответствующую гиперссылку. После щелчка на гиперссылке открывается диалоговое окно загрузки файла, содержащего систему Turbo Pascal (рис. 2.3). Этот файл называется tp7.zip и представляет собой архив, в котором в запакованном виде хранятся файлы системы программи-