УДК 681.3.06(075.8)
ББК 32.973-018
И201

Рецензенты:
профессор Л.Д. Забродин (Московский государственный инженерно-физический институт); кафедра «ЭВМ, комплексы и сети» Московского государственного авиационного института (зав. кафедрой профессор О.М. Брехов)

Иванова Г.С.
И201 Основы программирования: Учебник для вузов. – 4-е изд., стер. – М.: Изд-во МГТУ им. Н.Э. Баумана, 2007. – 416 с.: ил. (Сер. «Информатика в техническом университете».)
ISBN 978-5-7038-3027-7

Изложены основные теоретические положения разработки программного обеспечения с использованием структурного и объектно-ориентированных подходов. Подробно рассмотрены основные приемы решения задач различных классов, в том числе приемы создания и обработки динамических структур данных, без которых невозможно современное программирование. Особое внимание уделено оценке точности получаемых результатов и анализу вычислительной сложности алгоритмов и методов. Большое количество примеров и поясняющих рисунков помогает лучшему усвоению материала.

Содержание учебника соответствует курсу лекций, которые автор читает в МГТУ им. Н.Э. Баумана.

Для студентов вузов, обучающихся по специальностям, связанным с информатикой. Может быть полезен всем изучающим программирование самостоятельно.

УДК 681.3.06(075.8)
ББК 32.973-018

ISBN 978-5-7038-3027-7

Оглавление

ПРЕДИСЛОВИЕ

Преподавание основ программирования в вузах сопряжено с целым рядом проблем. Во-первых, современное программирование – сложная и быстро развивающаяся наука. Если сравнить то, что студент должен знать в этой области сейчас и 20 лет назад, то разница окажется ошеломляющей. В то же время реальные часы, отводимые в программах вузов для изучения основ программирования, практически не изменились. Во-вторых, подготовка студентов, осуществляемая в данной области школой, может варьироваться от полного отсутствия каких-либо знаний по предмету до относительно свободного владения каким-либо языком программирования.

Кроме того, программирование – наука, неразрывно связанная с практикой. Невозможно научиться программировать, не проведя много часов за составлением алгоритмов, написанием и отладкой программ. Причем учебно-практическую работу желательно совмещать с процессом изучения методов разработки программ и освоением особенностей конкретного языка программирования. Следовательно, элементы технологии программирования и алгоритмизации должны изучаться параллельно с языком программирования. Таким образом, один курс как бы включает в себя несколько курсов.

Решение перечисленных проблем потребовало тщательного отбора и структуризации материала, включенного в учебник. Это результат 20-летнего преподавания программирования в МГТУ им. Н.Э. Баумана. Курс, читаемый автором в настоящее время, построен следующим образом.

На лекциях излагаются основы технологии программирования, сведения, необходимые для решения тех или иных задач, и поясняются конкретные языковые средства. Лекции иллюстрируются большим количеством рисунков и примеров (программ и схем алгоритмов), по возможности минимального размера, чтобы конкретные возможности и особенности было легко понять и запомнить.

Семинары посвящаются обсуждению определенных проблем, связанных с решением некоторого класса задач. Как правило, на них анализируются не программы, а алгоритмы или подходы. Например, рассматривается метод пошаговой детализации и его применение для разработки алгорит-

мов, понятия «точность полученных результатов», «вычислительная сложность алгоритма», «емкостная сложность алгоритма» и способы их оценки.

Во время лабораторного практикума студенты самостоятельно под контролем преподавателей разрабатывают программы решения индивидуального набора задач по изучаемым темам. Задание каждому студенту выдается в начале семестра, поэтому он имеет возможность выполнять задания по мере освоения материала, что обеспечивает определенную степень индивидуализации обучения.

Изложение материала курса в учебнике следует той же схеме. Главы содержат необходимые сведения из теории программирования, описание конкретных средств Borland Pascal и особенностей взаимодействия программ с техническими и программными средствами. При этом особое внимание уделяется наиболее важным моментам, без рассмотрения которых дальнейшее изучение программирования практически невозможно. Это, в частности, проблемы создания рекурсивных программ, работа с динамическими структурами данных и объектно-ориентированный подход. Материал проблемных семинаров курса выделен в специальные разделы, названные практикумами. В конце большинства разделов приведены вопросы и задачи для самопроверки.

Данная книга представляет собой четвертое издание учебника. В связи с новой редакцией программ обучения основам программирования в него включены материалы по основам событийного программирования и отличиям Delphi Pascal от Borland Pascal 7.0. Кроме того, изменена графическая нотация, используемая для пояснения основ объектно-ориентированного программирования, что связано с практическим утверждением UML (Unified Modeling Language – Универсальный язык моделирования) в качестве международного стандарта описания объектно-ориентированных разработок.

Автор глубоко признателен кандидату технических наук, доценту Т.Н. Ничушкиной за предоставленные материалы и огромную помощь в подготовке книги, а также рецензентам: заведующему кафедрой «Компьютерные системы и технологии» МИФИ доктору технических наук, профессору Л. Д. Забродину и коллективу кафедры «ЭВМ, комплексы и сети» МАИ во главе с доктором технических наук, профессором О.М. Бреховым за полезные замечания и советы.

Хочется также выразить особую благодарность студентам, принявшим активное участие в обсуждении первого издания учебника, за их советы и замечания, учтенные автором во втором издании данной книги.

ВВЕДЕНИЕ

Язык программирования Паскаль был создан в 1971 г. профессором Цюрихского университета Никлаусом Виртом и предназначался для обучения студентов как основам алгоритмизации и программирования, так и основам конструирования компиляторов. Язык полностью отвечал принципам структурного программирования, сформулированным к тому моменту, имел ярко выраженную блочную структуру и развитое представление данных. Однако, будучи учебным, он имел ограниченные средства реализации ввода-вывода и создания библиотек подпрограмм.

В разные годы было разработано несколько вариантов компиляторов с Паскаля для различных типов ЭВМ. Наибольшее распространение получил Turbo (Borland) Pascal, предложенный фирмой Borland Internation (США). Существовало несколько версий. Последняя версия, предназначенная для создания программного обеспечения «под MS DOS» – версия 7.0, включает:

- интегрированную среду разработки программ, ставшую в некоторой степени прототипом создания аналогичных сред для других языков программирования;
- средства разработки многомодульных программ;
- средства управления экраном в текстовом и графических режимах;
- средства объектно-ориентированного программирования;
- усовершенствованную систему типов данных.

Современным программистам приходится иметь дело с огромным количеством разнообразных языков программирования различных уровней и назначений. Но по-прежнему начинать изучение программирования целесообразно на базе Паскаля, так как при использовании этого языка у будущего программиста быстрее формируется четкое алгоритмическое мышление.

Весомым аргументом в пользу изучения основ программирования именно на базе Паскаля также является существование профессиональной визуальной среды разработки программного обеспечения Delphi, которая использует в качестве базового языка Паскаль. Практика показывает, что переход к разработке программного обеспечения в этой среде после изучения базового курса происходит достаточно безболезненно, хотя и требует некоторых дополнительных знаний.

В настоящее время при разработке сложного программного обеспечения обычно используют одну из двух технологий: *структурное* программирование или *объектно-ориентированное* программирование.

Первая технология для разработки сложных программ рекомендует разбивать (*декомпозировать*) программу на подпрограммы (процедуры), решающие отдельные подзадачи, т. е. базируется на *процедурной декомпозиции*.

Вторая технология использует более сложный подход, при котором в предметной области задачи выделяют отдельно функционирующие элементы. Поведение этих объектов программно моделируется с использованием специальных средств, а затем (уже из готовых объектов) опять же специальным способом собирается сложная программа. Таким образом, в основе второй технологии лежит *объектная декомпозиция*.

Именно объектная технология лежит в основе используемой Delphi библиотеки стандартных компонентов, поэтому переход в эту среду целесообразно осуществлять только после изучения основных принципов объектного подхода, изложенных в данном учебнике.

Кроме объектного подхода для работы в Delphi необходимо иметь представление об основных отличиях Delphi Pascal и визуальных средах, использующих принцип событийного программирования.

Изучение объектной технологии требует наличия базовых знаний, поэтому на первых этапах мы будем придерживаться принципов процедурного программирования.

Часть 1. ОСНОВЫ АЛГОРИТМИЗАЦИИ И ПРОЦЕДУРНОЕ ПРОГРАММИРОВАНИЕ

1. ЭТАПЫ СОЗДАНИЯ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

В процессе разработки программ с использованием процедурного подхода можно выделить следующие этапы:

- *постановка задачи* – определение требований к программному продукту;
- *анализ* – осуществление формальной постановки задачи и определение методов ее решения;
- *проектирование* – разработка структуры программного продукта, выбор структур для хранения данных, построение и оценка алгоритмов подпрограмм и определение особенностей взаимодействия программы с вычислительной средой (другими программами, операционной системой и техническими средствами);
- *реализация* – составление программы на выбранном языке программирования, ее тестирование и отладка;
- *модификация* – выпуск новых версий программного продукта.

1.1. Постановка задачи

Процесс создания нового программного обеспечения начинают с постановки задачи, в процессе которой определяют требования к программному продукту.

Прежде всего устанавливают набор выполняемых функций, а также перечень и характеристики исходных данных. Так, для числовых данных может задаваться точность, для текстовых – возможно, размер текста, способ кодировки и т. п. Затем определяют перечень результатов, их характеристики и способы представления (в виде таблиц, диаграмм, графиков и т. п.). Кроме того, уточняют среду функционирования программного продукта: конкретную комплектацию и параметры технических средств, версию используемой операционной системы и, возможно, версии и параметры другого установ-

2. ПРОСТЕЙШИЕ КОНСТРУКЦИИ ЯЗЫКА

К простейшим конструкциям языка относятся способы представления скалярных данных, конструкции выражений, оператор присваивания и операторы ввода-вывода, без которых не обходится ни одна программа. Однако, прежде чем рассматривать эти конструкции, выясним, что собой представляет язык программирования и каким образом выполняется его описание.

2.1. Синтаксис и семантика языка программирования

Любой язык, в том числе и язык программирования, подчиняется ряду правил. Их принято разделять на правила, определяющие синтаксис языка, и правила, определяющие его семантику.

Синтаксис языка – совокупность правил, определяющих допустимые конструкции (слова, предложения) языка, его *форму*.

Семантика языка – совокупность правил, определяющих смысл синтаксически корректных конструкций языка, его *содержание*.

Языки программирования относятся к группе формальных языков, для которых в отличие от естественных языков однозначно определены синтаксис и семантика. Описание синтаксиса языка включает определение алфавита и правил построения различных конструкций языка из символов алфавита и более простых конструкций. Для этого обычно используют *форму Бэкуса–Наура* (БНФ) или *синтаксические диаграммы*. Описание конструкции в БНФ состоит из символов алфавита языка, названий более простых конструкций и двух специальных знаков:

«**::=**» – читается как «может быть заменено на»,

«|» – читается как «или».

При этом символы алфавита языка, которые часто называют *терминальными символами*, или *терминалами*, записывают в неизменном виде. Названия конструкций языка (*нетерминальные символы*, или *нетерминалы*), определяемых через некоторые другие символы, при записи заключают в угловые скобки («< », « >»).

Например, правила построения конструкции <Целое>, записанные в БНФ, могут выглядеть следующим образом:

3. УПРАВЛЯЮЩИЕ ОПЕРАТОРЫ ЯЗЫКА

Программы, содержащие в разделе операторов только операторы ввода-вывода и операторы присваивания, выполняются последовательно оператор за оператором. Такие программы называют *линейными*, они реализуют линейный процесс вычислений. Для организации разветвленных и циклических процессов вычислений используют *управляющие операторы* языка, определяющие последовательность выполнения операторов программы. В данной главе мы рассмотрим управляющие операторы языка Borland Pascal, к которым относят: оператор условной передачи управления, оператор выбора, операторы организации циклов, а также неструктурные операторы и процедуры передачи управления.

3.1. Оператор условной передачи управления

Оператор условной передачи управления (рис. 3.1) используют для программирования ветвлений, т. е. ситуаций, когда возникает необходимость при выполнении условия реализовывать одни действия, а при нарушении – другие. Условие записывают в виде логического выражения, в зависимости от результата которого осуществляется выбор одной из ветвей: если результат true, то выполняется оператор, следующий за служебным словом then, иначе – оператор, следующий за служебным словом else.

В каждой ветви допускается запись одного оператора (в том числе и другого if) или составного оператора.

Составным оператором в Borland Pascal называют последовательность операторов, заключенную в операторные скобки begin…end. Операторы последовательности отделяют друг от друга точкой с запятой «;». Перед end точку с запятой можно не ставить. Перед else точка с запятой *не ставится никогда*, так как в этом случае запись условного оператора продолжается.

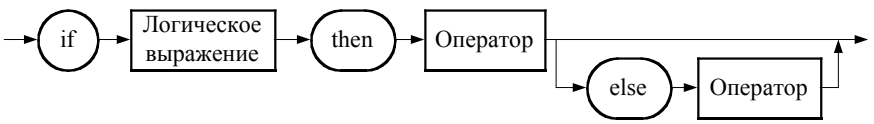

Рис. 3.1. Синтаксическая диаграмма <Оператор условной передачи управления>

4. СТРУКТУРНЫЕ ТИПЫ ДАННЫХ

Достаточно часто возникает необходимость программирования обработки однотипных данных: таблиц, текстов, множеств и т.д. Для их представления используют структурные типы данных. В Borland Pascal определены следующие структурные типы данных:

- *массивы* – для представления однотипных или табличных данных;
- *строки* – для представления символьной (текстовой) информации;
- *множества* – для представления абстрактных математических множеств;
- *записи* – для представления таблиц с данными различных типов.

4.1. Массивы

Массив – это упорядоченная совокупность однотипных данных. Каждому элементу массива соответствует один или несколько *индексов*, определяющих положение элемента в массиве. Индексы образуют упорядоченные последовательности. Синтаксическая диаграмма объявления массива представлена на рис. 4.1.

Тип индекса определяет его допустимые значения. В качестве типа индекса может быть указан любой порядковый тип (boolean, char, integer, перечисляемый тип, а также диапазоны этих типов), кроме типа longint и его производных.

В зависимости от количества типов индексов различают: одномерные, двумерные, трехмерные и n-мерные массивы. Двумерные массивы обычно называют *матрицами*, считая первый индекс – номером строки, а второй – номером столбца.

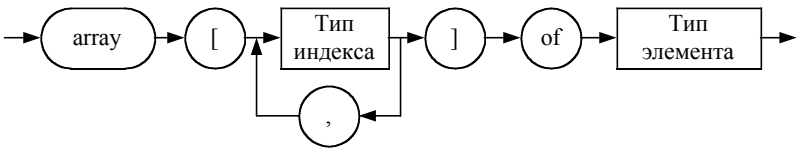

Рис. 4.1. Синтаксическая диаграмма <Объявление массива>

5. МОДУЛЬНОЕ ПРОГРАММИРОВАНИЕ

Большие программы обычно разрабатывают и отлаживают по частям. Целесообразно при этом, чтобы каждая такая часть, называемая *подпрограммой,* была оформлена так, чтобы ее можно было использовать при решении аналогичной подзадачи в той же программе или даже при решении других задач. В Borland Pascal реализованы два типа подпрограмм: процедуры и функции.

5.1. Процедуры и функции

Процедуры и функции представляют собой относительно самостоятельные фрагменты программы, соответствующим образом оформленные и снабженные именем (программные блоки). По правилам Borland Pascal программные блоки – такие же ресурсы, как типы и переменные. Соответственно, они также должны быть описаны перед использованием в разделе описаний программного блока, который их использует (основной программы или вызывающей подпрограммы). Каждый блок имеет такую же структуру, как основная программа, т.е. включает заголовок, раздел описаний и раздел операторов, но заканчивается не точкой, а точкой с запятой (рис. 5.1).

Заголовок блока определяет форму вызова подпрограммы. В разделе описаний блока объявляют внутренние локальные ресурсы блока (переменные, типы, внутренние подпрограммы). Раздел операторов содержит инструкции подпрограммы в операторных скобках begin…end.

Заголовки процедур и функций описываются по-разному. В отличие от процедуры функция всегда возвращает в точку вызова скалярное значение, адрес или строку. Тип возвращаемого результата описывается в заголовке функции (рис. 5.2).

Рис. 5.1. Синтаксическая диаграмма конструкции <Программный блок>

6. ФАЙЛОВАЯ СИСТЕМА. ФАЙЛЫ

Файлом называют именованную последовательность элементов данных (компонент файла), расположенных, как правило, во внешней памяти: на дискетах, винчестере, CD или других устройствах хранения информации, также устройствах ввода-вывода. В файле может храниться текст, программа, числовые данные, графическое изображение и т.д. Для организации работы с файлами программа на Borland Pascal взаимодействует с операционной системой MS DOS.

6.1. Файловая система MS DOS

Как сказано выше, каждый файл обязательно имеет имя. Имена файлов в MS DOS подчиняются определенным правилам:

- имя файла должно содержать не менее одного и не более восьми символов;
- имя файла может иметь расширение, которое отделяется от имени точкой и содержит не более трех символов;
- для записи имен и расширений могут использоваться строчные и прописные буквы латинского алфавита a-z, A-Z, арабские цифры и некоторые специальные символы, например, символ подчеркивания «_» или знак доллара «$»;
- в качестве имен запрещается использовать некоторые буквенные сочетания, которые зарезервированы операционной системой для обозначения устройств, например: PRN, CON, NUL, COM1, COM2, AUX, LPT1, LPT2, LPT3.

В операционных системах типа Windows некоторые из этих правил отменяются, например, имя файла может содержать больше восьми символов и включать символы русского алфавита. Однако при работе с файлами из Borland Pascal лучше придерживаться правил MS DOS.

Независимо от используемой операционной системы имена обычно составляют так, чтобы они указывали на содержимое файла. Расширение обычно определяет тип хранящихся данных.

7. ПРОГРАММИРОВАНИЕ С ИСПОЛЬЗОВАНИЕМ ДИНАМИЧЕСКОЙ ПАМЯТИ

До настоящего момента мы имели дело с переменными, которые размещаются в памяти согласно вполне определенным правилам. Так, память под глобальные переменные программы выделяется в процессе компиляции, и эти переменные существуют в течение всего времени работы программы. Для локальных переменных, описанных в подпрограмме, память отводится при вызове подпрограммы, при выходе из нее эта память освобождается, а сами переменные прекращают свое существование. Иными словами, распределение памяти во всех случаях осуществляется полностью автоматически. Переменные, память под которые выделяется описанным образом, называют *статическими*. Под эту категорию подпадают все переменные, объявленные в области описаний программных блоков. Однако Borland Pascal предоставляет возможность создавать новые переменные во время работы программы, сообразуясь с потребностями решаемой задачи, и уничтожать их, когда надобность в них отпадает.

Переменные, созданием и уничтожением которых может явно управлять программист, называют *динамическими*. Для более полного понимания механизма работы с динамическими переменными следует сначала разобраться в механизме адресации оперативной памяти MS DOS.

7.1. Указатели и операции над ними

Наименьшей адресуемой единицей памяти персонального компьютера, построенного на базе микропроцессоров фирмы Intel и их аналогов, является байт. Таким образом, память представляет собой последовательность нумерованных байтов. Для обращения к конкретному байту необходимо знать его номер, который называют его *физическим адресом*.

Память принято делить на слова, двойные слова и параграфы. Слово имеет длину 2 байта, двойное слово – 4 байта, а параграф – 16 байт.

При работе с памятью используется адресация по схеме «база + смещение» (рис. 7.1). При этом адрес конкретного байта M определяется как адрес некоторого заданного байта $A_б$ (*адрес базы*) + расстояние до требуемого байта $A_{см}$ (*смещение*).

8. УПРАВЛЕНИЕ ТЕХНИЧЕСКИМИ СРЕДСТВАМИ И ВЗАИМОДЕЙСТВИЕ С MS DOS

Управление техническими и многими программными средствами в операционной системе MS DOS осуществляется через систему прерываний.

Прерывание – это событие в системе, которое требует специальной обработки. К таким событиям относятся: требования обработки от внешних устройств, например, часов или устройств ввода-вывода, и требования программ на выполнение некоторых действий, например, операций ввода-вывода. Каждое прерывание имеет свой номер. Необходимая обработка осуществляется специальными программами – *обработчиками прерываний*, большая часть которых входит в состав BIOS (базовой системы ввода-вывода) или MS DOS.

В Borland Pascal для взаимодействия с техническими средствами и MS DOS существует набор специальных процедур и функций, которые входят в состав модулей Crt, Graph и Dos.

8.1. Управление экраном в текстовом режиме

Выше рассматривались программы, которые выводили результаты на экран в так называемом *консольном режиме*. В этом режиме вывод на экран происходит построчно, доступ возможен только к последней выводимой строке. По мере заполнения экрана осуществляется его «прокрутка», при которой строки перемещаются по экрану вверх, причем верхние строки безвозвратно теряются, а внизу появляются новые строки. В таком режиме программист почти не может управлять формой представления выводимой информации.

В текстовом режиме программист имеет доступ ко всему экрану. Экран при этом поделен на строки и столбцы. На пересечении строки и столбца находится область, в которою возможен вывод одного знака. Такие области получили название *знакоместо*. Обычно программа на Borland Pascal использует тот же текстовый режим, что и MS DOS, т.е. режим, при котором на экране выделяется 25 строк и 80 столбцов.

Часть 2. ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ ПРОГРАММИРОВАНИЕ

9. ОСНОВНЫЕ ТЕОРЕТИЧЕСКИЕ ПОЛОЖЕНИЯ

Считается, что технология процедурного программирования применима, если размер программы не превышает 100 тыс. операторов. Программы, используемые в настоящее время, существенно длиннее. Поэтому современное программирование в основном базируется на технологии, позволившей снять это ограничение и получившей название «объектно-ориентированное программирование» (ООП). Именно ООП лежит в основе таких современных сред создания программного обеспечения «под Windows», как Delphi, Visual C++, C++ Builder.

В теории программирования ООП определяется как технология создания сложного программного обеспечения, основанная на представлении программы в виде совокупности *объектов*, каждый из которых является экземпляром определенного типа (*класса*), а классы образуют иерархию с *наследованием* свойств [2].

Как следует из определения, ООП в отличие от процедурного программирования, которое рассматривалось в первой части учебника, базируется не на процедурной, а на объектной декомпозиции предметной области программы.

9.1. Объектная декомпозиция

Объектной декомпозицией называют процесс представления предметной области задачи в виде совокупности функциональных элементов (*объектов*), обменивающихся в процессе выполнения программы входными воздействиями (*сообщениями*).

Каждый выделяемый объект предметной области отвечает за выполнение некоторых действий, зависящих от полученных сообщений и параметров самого объекта.

Совокупность значений параметров объекта называют его *состоянием*, а совокупность реакций на получаемые сообщения – *поведением*.

10. КЛАССЫ И ОБЪЕКТЫ В BORLAND PASCAL

Объектная модель, реализованная в Borland Pascal, по современным меркам является упрощенной, но она позволяет изучить основные приемы объектно-ориентированного программирования и оценить его достоинства и недостатки.

В настоящей главе рассмотрены средства, используемые для объявления классов и объектов, и принципы создания «универсальных» классов.

10.1. Объявление класса. Поля и методы

С точки зрения синтаксиса, класс представляет собой структурный тип данных, в котором помимо полей разрешается описывать *прототипы* (заголовки) процедур и функций, работающих с этими полями данных. По форме описание класса напоминает запись.

Как уже упоминалось ранее, процедуры и функции, заголовки которых описаны в классе, получили название *методов*.

Описание типа класс выполняется следующим образом:

```
Type <имя класса> = object
                      <описание полей класса>
                      <прототипы методов>
                    end; ...
```

Тела методов класса описываются после объявления класса. Причем в заголовке метода можно не повторять списка параметров, но перед именем метода необходимо указать имя класса, отделив его от имени метода точкой:

```
Procedure <имя класса>.<имя метода>;
 <локальные ресурсы процедуры>
   Begin
       <тело процедуры>
   End; ...
```

или

11. ИЕРАРХИИ КЛАССОВ

Построение классов с использованием уже существующих – одно из основных достоинств ООП. Как уже упоминалось выше, в профессиональных средах программирования существуют мощные библиотеки классов, на базе которых строятся классы для решения конкретной задачи. При этом используют средства языка, реализующие основные отношения между классами: наследование (простое и полиморфное), композицию и наполнение.

11.1. Наследование

Наследованием называют конструирование новых более сложных производных классов из уже имеющихся базовых посредством добавления полей и методов.

При наследовании объекты класса-потомка получают возможность использования («наследуют») поля и методы класса-родителя, что позволяет повторно не определять этих компонентов класса.

При описании класса-потомка указывают класс-родитель и дополнительные, определенные только для класса-потомка, поля и методы:

Type <имя класса-потомка> = ***object(***<имя класса-родителя>***)***
<описание дополнительных полей и методов класса>
end; ...

Пример 11.1. Разработать класс для реализации объекта Трехмерная комната, который должен реагировать на запрос о площади и объеме комнаты (рис. 11.1).

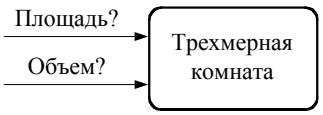

Рис. 11.1. Объект Трехмерная комната

В параграфе 10.3 уже был определен класс TRoom в модуле Room, который содержал поля для хранения длины и ширины ком-

12. РАЗРАБОТКА БИБЛИОТЕКИ ИНТЕРФЕЙСНЫХ ЭЛЕМЕНТОВ

Создание пользовательских интерфейсов – одна из наиболее трудоемких задач программирования, и для ее решения успешно применяют средства ООП. Современный программист, работающий в одной из профессиональных сред программирования, при написании программы с развитым интерфейсом обычно использует большие и достаточно сложные библиотеки классов для реализации интерфейсных элементов. Прежде чем изучать подобные библиотеки, желательно ознакомиться с общими принципами их построения. С этой целью попытаемся разработать упрощенный вариант такой библиотеки.

12.1. Анализ реальной программы и определение основных интерфейсных элементов

Создание библиотеки универсальных интерфейсных элементов начнем с того, что на примере конкретной задачи выясним, какие элементы целесообразно в такую библиотеку включать.

Пример 12.1. Разработать программу «Записная книжка», которая должна осуществлять: создание новой книжки (файла), добавление записей (фамилии, имени и телефона), поиск записей по фамилии и/или имени.

По сути дела данная программа должна обеспечивать удобный интерфейс для хранения и поиска информации в некотором файле. Разработку начинаем с уточнения интерфейса.

Главное меню программы в соответствии с условием задачи должно вызывать основные функции для работы с «Записной книжкой» (рис. 12.1). Выбор функции будем осуществлять клавишами горизонтального перемещения курсора, а ее вызов – нажатием клавиши Enter. Для выхода из программы предусмотрим специальный пункт меню, но будем осуществлять завершение программы или возврат из функций и по нажатию клавиши Esc.

Диаграмма переходов состояний интерфейса, отражающая процесс работы пользователя с программой, приведена на рис. 12.2.

СПИСОК ЛИТЕРАТУРЫ

1. *Бадд Т.* Объектно-ориентированное программирование в действии. СПб.: Питер, 1997.

2. *Буч Г.* Объектно-ориентированный анализ и проектирование с примерами приложений на C++. М.: Бином; СПб.: Невский диалект, 1998.

3. *Вирт Н.* Алгоритмы и структуры данных. М.: Мир, 1989.

4. *Дал У., Дейкстра Э., Хоор К.* Структурное программирование. М.: Мир, 1975.

5. *Иванова Г.С., Ничушкина Т.Н., Пугачев Е.К.* Объектно-ориентированное программирование. М.: Изд-во МГТУ им. Н.Э. Баумана, 2001.

6. *Кормен Т., Лейзерсон Ч., Ривест Р.* Алгоритмы: построение и анализ. М.: МЦНМО, 2000.

7. *Майерс Г.* Искусство тестирования программ. М.: Мир, 1982.

8. *Фаронов В.В.* Турбо-Паскаль. Основы Турбо-Паскаля. М.: «МВТУ – ФЕСТО ДИДАКТИК», 1992.

9. *Хьюз Дж., Мичтом Дж.* Структурный подход к программированию. М.: Мир, 1980.

Предметный указатель

Учебное издание

ИНФОРМАТИКА В ТЕХНИЧЕСКОМ УНИВЕРСИТЕТЕ

Галина Сергеевна Иванова

ОСНОВЫ ПРОГРАММИРОВАНИЯ

Редактор *Н.Е. Овчеренко*
Художники *С.С. Водчиц, Н.Г. Столярова*
Корректоры *Л.И. Малютина, О.Ю. Соколова*
Компьютерная верстка *Б.А. Иванов*

Оригинал-макет подготовлен в Издательстве
МГТУ им. Н.Э. Баумана

Санитарно-эпидемиологическое заключение
№ 77.99.02.953.Д.008880.09.06 от 29.09.2006 г.

Подписано в печать 19.05.2007. Формат 70x100/16. Печать офсетная. Бумага офсетная.
Гарнитура «Таймс». Печ. л. 26. Усл. печ. л. 33,8. Уч.-изд. л. 33,25. Тираж 2000 экз.
Заказ

Издательство МГТУ им. Н.Э. Баумана
105005, Москва, 2-я Бауманская, 5

Отпечатано с оригинал-макета в ГУП ППП
«Типография «Наука»
121099, Москва, Шубинский пер., 6