УДК 004.42:591.6
К78

Рецензент
доцент *А.П. Смирнов*

Крапухина Н.В., Светозарова Г.И.

К78 Информатика. Основы алгоритмизации и программирования: Учеб. пособие. – М.: МИСиС, 2005. – 177 с.

Рассматриваются общие вопросы алгоритмизации и программирования, современные методы разработки программ. Дается описание языка программирования Турбо-Паскаль 7.0. Рассматриваются основные приемы программирования, типовые алгоритмы, базовые численные методы и особенности их программной реализации.

Предназначено для изучения основ алгоритмизации и программирования на языке Турбо-Паскаль студентами I курса, обучающимися по направлению подготовки 230400 «Прикладная математика» специальности 230410 «Прикладная математика», а также специальностей других направлений, и для самостоятельного изучения.

ОГЛАВЛЕНИЕ

ПРЕДИСЛОВИЕ

В настоящее время решение любой инженерной, научной или технической задачи немыслимо без использования компьютера. Современные компьютеры имеют мощное программное обеспечение, включающее пакеты программ (приложения), позволяющие решать задачи из различных областей без трудоемкой разработки программ, обеспечивая таким образом доступ к этой мощной технике и непрофессиональным пользователям.

Однако, чтобы грамотно использовать эти возможности, необходимо иметь понятие об основах программирования, способах представления данных, ограничениях, связанных с особенностями вычислительного процесса, осуществляемого компьютером, вычислительных методов, к которым приходится прибегать, чтобы получить решение задачи, и т.п. Поэтому в курсе «Информатика» перед использованием приложений Windows изучаются основы алгоритмизации и программирования и простейшие численные методы в том минимальном объеме, который позволяет сформировать представление об используемых в этой области подходах, дает возможность грамотно обратиться к специалистам в сложных случаях, когда использование стандартных средств невозможно, или внести изменения в уже разработанные программы, приспосабливая их к своей задаче. Для студентов, обучающихся по специальностям, предполагающим профессиональное владение компьютерными технологиями, изучение этого раздела также является необходимым этапом, а начальное знакомство с численными методами облегчает углубленное изучение этого раздела на старших курсах и, кроме того, является хорошим упражнением по программированию.

При изучении основ алгоритмизации с применением структурного подхода и метода пошаговой детализации формируются навыки, которые с успехом можно использовать и для решения сложных инженерных задач, представляя исходную задачу в виде последовательности более простых задач, что может быть полезным в будущей работе.

В качестве языка программирования используется Турбо-Паскаль (версия 7.0), являющийся современным языком, разработанным с использованием идей структурного программирования и ориентированным на обучение программированию.

Пособие состоит из двух частей: первая (гл. 1–8) посвящена основам алгоритмизации и программирования, а вторая (гл. 9) – вычислительным методам решения типовых задач, часто встречающихся в

инженерной практике, и особенностям их программной реализации. При этом рассматриваются наиболее простые и доступные для понимания методы с тем, чтобы дать представление о подходах к решению реальных инженерных задач, что позволит грамотно пользоваться пакетами прикладных программ или в нестандартных случаях обращаться к специалистам.

Изложение сопровождается примерами программ, демонстрирующих основные средства или приемы программирования. Каждая глава содержит вопросы для самопроверки и задания для самостоятельного выполнения, работа над которыми будет способствовать усвоению изучаемого материала.

Глава 1 дает общее представление о компьютере и основных понятиях программирования (переменная, алгоритм, операторы), типовых структурах алгоритма, современных методах разработки алгоритмов и программ. К главе 1 придется многократно возвращаться на протяжении всего курса, и определенные в ней понятия будут полностью усвоены только после приобретения собственного опыта программирования.

В главе 2 дается описание основных средств языка Турбо-Паскаль.

Глава 3 посвящена разработке циклических и разветвляющихся программ, глава 4 – работе с массивами.

Главы 1–4 формируют базовые понятия программирования и в сочетании с выполнением приведенных заданий позволяют освоить основы программирования.

Для понимания основ программирования необходимо с самого начала изучения курса иметь практику составления программ и их реализации на компьютере. Это требует знания и учета многих аспектов решения задач с использованием компьютера, а именно: основных понятий программирования, таких, как переменная, присваивание, цикл и т.п., основ алгоритмизации, особенностей представления данных в компьютере, языка и среды программирования (т.е. основных правил работы на компьютере при наборе, редактировании, отладке и выполнении программы), умения интерпретировать полученные результаты и т.д. Изучение этих разделов по отдельности не имеет смысла, так как значение каждого проявляется (и становится понятным) только во взаимодействии с другими. С этим связаны трудности начального освоения программирования, так как продвигаться вперед можно, только развивая параллельно все аспекты. Поэтому при работе с настоящим пособием важно придерживаться

рекомендаций, касающихся последовательности изучения отдельных разделов.

Рекомендуется следующий порядок работы: прочитать гл. 1, прочитать гл. 2 (2.1, 2.2, 2.5, 2.6.1, 2.6.7) и перейти к гл. 3 (3.1), где рассматриваются типовые алгоритмы циклической структуры и приемы разработки циклических программ. Параллельно с изучением 3.1 необходимо решить задачи, предлагаемые для самостоятельного выполнения к гл. 3 (все или часть из них по указанию преподавателя). Если возникают трудности при разработке алгоритмов и программ на Турбо-Паскале, то необходимо вернуться к гл. 1 и соответствующим фрагментам гл. 2, а также использовать типовые алгоритмы циклической структуры, приведенные в гл. 3. Заметим, что для понимания алгоритма, записанного, как правило, в виде программы или фрагмента программы, его нужно выполнить, используя правила, по которым работают операторы Турбо-Паскаля, задавая, если необходимо, простые исходные данные, не требующие больших вычислений.

Порядок действий при реализации на компьютере разработанной программы и правила, которых при этом необходимо придерживаться, содержатся в гл. 8. Содержащаяся в этой главе информация облегчит набор программы, внесение исправлений и т.п. С этой главой нужно ознакомиться перед первым сеансом работы на компьютере и далее обращаться к ней по мере необходимости.

После составления программы ее целесообразно проверить вручную (см. гл. 7). Это значит, что Вы работаете как компьютер, выполняя операторы программы последовательно друг за другом (забыв о задаче, видя перед собой только программу), выписывая значения изменяющихся переменных. Если при реализации на компьютере программа дает не те результаты, которые были получены при ручном выполнении, то это означает, как правило, что Вы неправильно представляете, как работают операторы Турбо-Паскаля. Нужно вернуться к гл. 2 и проверить свои знания.

Далее таким же образом изучить 3.2 и 3.3, выполнив задания для самостоятельного выполнения к этим пунктам.

Глава 4 посвящена работе с массивами (средства Турбо-Паскаля для работы с массивами изложены в 2.3). Усвоив типовые алгоритмы обработки одномерных массивов (4.1), матриц (4.2) и решив задачи для самостоятельного выполнения, можно перейти к решению реальных задач, требующих использования записей и массивов записей (4.3). Перед этим целесообразно более подробно изучить гл. 7 и воспользоваться содержащимися в ней рекомендациями. При решении

реальных задач, связанных с обработкой больших объемов данных и необходимостью документирования результатов работы программы, требуется использование внешних устройств для хранения файлов данных (2.9).

Главы 5 и 6 посвящены более сложным приемам программирования и средствам языка, используемым для реализации принципа программирования сверху вниз (гл. 5), обработке текстовых данных (6.1), использованию графического режима (6.2).

Глава 9 посвящена вопросам решения инженерных задач с использованием компьютера. Здесь наряду с численными методами и особенностями их программной реализации рассматриваются практические аспекты распространения ошибок, достижения необходимой точности и т.п. По каждой теме приводятся описания методов, особенности их программной реализации, а также задания для самостоятельного выполнения, реализация которых требует применения навыков разработки программ, приобретенных при освоении первой части пособия.

1. ВВЕДЕНИЕ В ПРОГРАММИРОВАНИЕ

1.1. Понятие о компьютере. Языки программирования

Компьютеры умеют выполнять лишь ограниченный набор простых действий, причем выполняют эти действия последовательно друг за другом, автоматически переходя от одного к другому. Последовательность действий, необходимых для решения конкретной задачи, записанная в необходимой для компьютера форме, представляет собой *машинную программу*, или программу на машинном языке. Отдельное действие в программе имеет вид *машинной команды*, которая в закодированном виде (в виде последовательности нулей и единиц) содержит наименование выполняемой операции (код операции), адреса операндов (величин, участвующих в операции) и адрес, по которому следует поместить результат (адресная часть). Наиболее просто представить себе компьютер, команда которого содержит в адресной части три адреса (два, по которым расположены операнды, и третий – для размещения результата). Однако существуют также двухадресные, одноадресные машины.

Машинная программа является очень детальной, и форма ее представления неудобна для человека. (Вспомним, что любая информация в компьютере представляется последовательностью нулей и единиц.)

Пользователи ЭВМ в настоящее время пишут программы в более привычном и естественном для человека виде, используя так называемые языки программирования – алгоритмические языки. *Языком программирования* называется система обозначений и правил, позволяющая записать программу решения задачи в виде последовательного текста в удобном для человека виде. Преобразование программы, составленной на языке программирования (исходной программы или исходного модуля), в машинную программу (объектную программу или объектный модуль) осуществляется самим компьютером при помощи программы-*транслятора* (компилятора), входящей в состав программного обеспечения компьютера.

1.2. Этапы решения задачи. Метод пошаговой детализации

Для решения задачи с помощью компьютера в первую очередь необходимо до конца осмыслить ее, т.е. понять, какие результаты

требуется получить и какие для этого имеются исходные данные. Затем программа может быть укрупненно представлена в виде трех последовательных этапов: ввод исходных данных; преобразование исходных данных в результат; вывод результата.

Далее необходимо установить взаимосвязь между исходными данными и результатом. Часто (особенно при решении технических задач) эту взаимосвязь выражают математически при помощи каких-либо уравнений, соотношений, ограничений и т.п. В некоторых случаях связь между исходными данными и результатом не может быть выражена математически и приходится ограничиваться четкой словесной формулировкой. (Это относится, например, ко многим задачам обработки текстов.)

После этого в соответствии с математической (или словесной) формулировкой задачи необходимо представить процесс преобразования исходных данных в результат в виде последовательности более простых этапов. Иногда (для простых задач) последовательность этапов очевидна. В большинстве случаев для этого необходимо использовать специальные методы (например, для задач вычислительного характера используют численные методы).

При этом некоторые этапы могут представлять собой менее сложную, но самостоятельную задачу, для которой необходимо повторить указанную процедуру, т.е. представить решение в виде последовательности более простых этапов. Этот метод, называемый *методом пошаговой детализации,* является основой разработки программ. Детализация заканчивается, когда каждый отдельный этап может быть записан на выбранном языке программирования или представляет собой известную задачу, для которой уже имеется готовая программа. Полученная последовательность этапов, записанная на языке программирования, и представляет собой программу, которая далее может быть выполнена на компьютере.

Таким образом, подготовка к решению задачи на компьютере включает два крупных этапа: *формализация задачи*, в которую входит выделение исходных данных и результата, а также ее математическая (или словесная) формулировка, т.е. установление в каком-либо виде взаимосвязи между входными и выходными данными, и *алгоритмизация задачи*, т.е. представление ее в виде последовательности этапов, куда входит выбор метода решения и планирование последовательности этапов с использованием метода пошаговой детализации. Результатом этого этапа является разработка алгоритма решения задачи.

Математическая постановка (или, в более общем случае, формализация) задачи – необходимый и очень важный этап, от которого больше всего зависит результат решения задачи на компьютере. Его выполнение требует досконального знания предмета, способности к абстрактному мышлению, владения математическим аппаратом, а также некоторого опыта в решении задач на компьютере.

После того, как задача сформулирована, необходимо выбрать метод решения, который позволил бы свести её к последовательности простых этапов.

1.3. Понятие алгоритма

Алгоритмом называется четкое описание последовательности действий, которые необходимо выполнить для решения задачи. Так как решение практически любой задачи требует получения результата по заданным исходным данным, то можно сказать, что алгоритм описывает последовательный процесс преобразования исходных данных в результат [1, 2].

Разработать алгоритм решения задачи означает разбить задачу на последовательно выполняемые шаги (этапы), причем результаты выполнения предыдущих этапов могут использоваться при выполнении последующих. При этом должно быть четко указано как содержание каждого этапа, так и порядок выполнения этапов. Отдельный этап (шаг) алгоритма либо представляет собой другую, более простую задачу, алгоритм решения которой разработан ранее, либо должен быть достаточно простым и понятным без дополнительных пояснений.

Если алгоритм разработан, то его выполнение можно поручить любому исполнителю, не знакомому с решаемой задачей, в том числе и компьютеру, и, точно следуя правилам алгоритма и выполняя последовательно указанные в нем действия, этот человек (или другой исполнитель) получит ее решение.

Алгоритм обладает следующими основными свойствами, раскрывающими его определение.

1. *Дискретность*: алгоритм должен представлять процесс решения задачи как последовательное выполнение простых (или ранее определённых) шагов (этапов). При этом для выполнения каждого шага (этапа) алгоритма требуется некоторый конечный отрезок времени, т.е. преобразование исходных данных в результат осуществляется во времени дискретно.

2. *Определенность* (или детерминированность): каждое правило алгоритма должно быть четким и однозначным. Благодаря этому

свойству выполнение алгоритма носит механический характер и не требует никаких дополнительных указаний или сведений о решаемой задаче.

3. *Результативность* (или конечность): алгоритм должен приводить к решению задачи за конечное число шагов.

4. *Массовость*: алгоритм решения задачи разрабатывается в общем виде, т.е. он должен быть применим для некоторого класса задач, различающихся лишь исходными данными. При этом исходные данные могут выбираться из некоторой области, которая называется областью применимости алгоритма. (В отдельных случаях исходные данные могут отсутствовать.)

Чтобы разработать алгоритм, нужно хорошо представить себе ход решения задачи. При этом полезно решить задачу самому (на бумаге) для каких-либо наборов данных, не требующих громоздких вычислений, запоминая выполняемые действия, чтобы далее эти действия формализовать, т.е. записать в виде последовательности четких правил. Понятия алгоритма и программы разграничены не очень четко. Обычно *программой* называют окончательный вариант алгоритма решения задачи, ориентированный на конкретного исполнителя.

Разработка алгоритма является содержанием этапа алгоритмизации. В широком смысле алгоритмизация включает и выбор метода решения задачи, а также формы представления исходной информации с учетом специфики компьютера. Следует отметить, что при создании программ основные трудности связаны именно с разработкой алгоритмов и основное внимание в настоящем пособии уделяется именно этому вопросу.

1.4. Основные понятия программирования. Переменная. Присваивание. Условный и безусловный переход

Решение любой задачи на компьютере предполагает наличие программы и данных, которые этой программой должны быть обработаны. В соответствии с этим перед нами встают две задачи: как описать порядок действий, необходимых для решения задачи, и как представить данные в памяти компьютера.

Самой простой структурой данных является переменная (простая переменная). *Переменную* можно представить себе как ящик с биркой, на которой написано имя. (В памяти компьютера это – ячейка памяти, имеющая определенный адрес.) Обращение к переменной

осуществляется по имени, являющемуся символическим адресом переменной в программе, заменяющим ее физический адрес. В переменной может одновременно храниться одно значение. При занесении в переменную другого значения старое значение пропадает.

Действия записываются с помощью операторов. Существует три основных оператора, с помощью различных сочетаний которых можно в принципе описать порядок решения любых задач. Это – операторы присваивания, условный оператор и оператор передачи управления.

Оператор присваивания имеет следующий вид:

переменная := *выражение*;

где знак := обозначает операцию присваивания (такое обозначение используется в языке Турбо-Паскаль), точкой с запятой заканчивается любой оператор в Турбо-Паскале, *выражение* аналогично знакомому из математики алгебраическому выражению (здесь речь идет о числовом выражении), задающему правило получения значения, а *переменная* – имя переменной, в которую будет помещен результат. (В гл. 2 даются более корректные определения введенных понятий.)

Оператор присваивания выполняется следующим образом: сначала вычисляется выражение, стоящее справа, а затем результат помещается в переменную, имя (т.е. символический адрес) которой указано слева от знака операции присваивания.

Например:

`x:= 5;` Выражение состоит из одной константы. В переменную `x` (т.е. в ячейку памяти с символическим адресом `x`) засылается значение 5.

`y:= x+1;` Выражение представляет собой сумму переменной и константы. Если x = 5, то `y` будет присвоено значение 6.

`y:= y+1;` Если перед этой операцией y имело значение 6, то после ее выполнения `y` будет равно 7.

`x:= y;` Если `y` имеет значение 7, то после выполнения этой операции `x` будет иметь то же значение 7 (старое значение пропадает), при этом значение y не изменяется.

Условный оператор в самом простом случае имеет вид

`if` *условие* `then` *оператор*;

Выполняется он следующим образом: если *условие* удовлетворяется, то выполняется *оператор*, если *условие* не удовлетворяется, то *оператор* не выполняется (просто осуществляется переход к следующей по порядку операции).

Например,

```
if x > 5 then x:= x - 1;
```

Если перед выполнением этого оператора переменная `x` имеет, например, значение 7, то условие (`x > 5`) удовлетворяется, оператор после `then` выполняется и значение `x` уменьшается на 1 (т.е. станет равно 6). Если `x` будет иметь, например, значение 3, то оператор после `then` не будет выполняться и `x` сохранит старое значение.

Оператором передачи управления является оператор перехода `goto`, позволяющий изменить порядок выполнения операторов и перейти при необходимости в любую точку программы, указываемую при помощи метки (см. 2.6.3). Он имеет вид

`goto` *метка*

(В программах в явном виде используется крайне редко, см. 1.5.)

С помощью перечисленных операторов может быть описано решение любой задачи (допускающей решение на компьютере). Все остальные операторы являются только их расширением и (или) комбинацией. Более полно операторы языка Турбо-Паскаль рассматриваются в гл. 2.

1.5. Типовые структуры алгоритмов. Современные технологии программирования. Структурный подход к разработке программ

Решение любой задачи на компьютере может быть описано при помощи ограниченного набора простых операций (см. выше). Эти операции при разработке алгоритма группируются, образуя типичные последовательности действий, называемых *основными структурами алгоритма*. Существует несколько типовых структур, рекомендуемых при использовании *структурного подхода* к разработке алгоритмов и программ [1 – 3]. Используя их различные сочетания, можно получить все многообразие алгоритмов и программ. К таким структурам относятся:

– *следование* – размещение отдельных этапов друг за другом;

– *цикл* – многократное выполнение одной и той же последовательности действий. Если количество повторений задано заранее, то говорят о *цикле по счетчику*. Если количество повторений неизвестно, но задано условие окончания цикла, то говорят о *цикле по условию* (см. 3.1);

– *разветвление* – выбор одного из двух вариантов вычислительного процесса после проверки условия (см. 3.2);

– *обход* – частный случай разветвления, когда одна из ветвей не содержит никаких действий;

– *множественный выбор* – выбор одного из многих вариантов вычислительного процесса (обычно в зависимости от какой-либо целой величины).

Перечисленные последовательности называются *типовыми структурами алгоритма*. Каждая из них имеет один вход и один выход, и их можно объединять в любой последовательности, создавая программы ясной и прозрачной структуры.

Использование только перечисленных структур лежит в основе так называемого *структурного подхода к программированию,* или структурного программирования. При разработке алгоритма (и программы) полезно использовать метод пошаговой детализации (см. выше), при котором первоначально продумывается и фиксируется общая структура алгоритма (программы) без детальной проработки отдельных его частей (используются лишь основные структуры алгоритма). Далее детализируются отдельные блоки, требующие дополнительной проработки после предыдущего шага (с использованием также только основных структур алгоритма). Таким образом, на каждом шаге разработки уточняется реализация фрагмента алгоритма (программы), т.е. на каждом шаге мы имеем дело с более простой задачей. Полностью закончив детализацию всех блоков, мы получим решение всей задачи в целом. Описанный метод пошаговой детализации называется также *программированием сверху вниз* [1]. При этом окончательный вариант программы должен быть написан в полном соответствии с правилами используемого языка программирования, промежуточные варианты допускают произвольное обозначение отдельных этапов, требующих дальнейшей детализации.

Современные алгоритмические языки содержат средства для реализации типовых структур алгоритмов, что позволяет при составлении программ оставаться в рамках структурного подхода и получать в результате правильно написанные программы, готовые к выполнению.

Разработка языка Паскаль осуществлена с использованием идей структурного программирования. В нем имеются специальные конструкции, позволяющие представить каждую из перечисленных последовательностей действий (типовых структур алгоритма) в виде одного оператора. Собственно автор идей структурного программи-

рования швейцарский специалист в области вычислительной техники Н. Вирт является и автором первого варианта языка Паскаль, разработанного в 1970 г. и названного в честь великого французского математика. Этот язык в дальнейшем изменялся в соответствии с новыми возможностями вычислительной техники, но основные идеи остались в неприкосновенности. Этот язык первоначально предназначался для обучения программированию, но оказался настолько удачным, что стал одним из основных инструментов прикладного и системного программирования при решении задач вычислительного и информационно-логического характера. Язык отличается одновременно стройностью, строгостью и простотой, способствует написанию эффективных и надежных программ, а также обеспечивает высокую эффективность их компьютерной реализации.

Данное пособие ориентировано на наиболее популярную версию Турбо-Паскаль 7.0. Излагаются только основные средства языка, использование которых необходимо для разработки и реализации рассматриваемых в пособии алгоритмов. Более полное описание языка см. в [4 – 9].

Вопросы для самопроверки

1. Что такое машинная программа?
2. Какова роль языков программирования?
3. Математическая постановка задачи.
4. Алгоритмизация задачи. Метод пошаговой детализации.
5. Понятие алгоритма. Основные свойства алгоритма.
6. Основные понятия программирования: переменная, присваивание, типовые действия алгоритма, операторы.
7. Типовые структуры алгоритма.
8. Структурный подход к разработке алгоритмов. Метод пошаговой детализации.

2. ОСНОВЫ ЯЗЫКА ТУРБО-ПАСКАЛЬ

2.1. Вводные понятия. Структура программы

Первый оператор – заголовок программы:

`program` *имя*; (не является обязательным).

Далее следует раздел описаний, где должны быть объявлены используемые в программе переменные (в соответствии с этим под них выделяются ячейки памяти), метки и т.д.

После раздела описаний следует тело программы, т.е. последовательность операторов программы, начинающаяся словом `begin` и заканчивающаяся словом `end`, после которого следует точка.

Таким образом, программа должна иметь следующую структуру:

```
program имя;
раздел описаний;
begin
раздел операторов
end.
```

Для именования объектов программы, а также самой программы, подпрограмм и т.п. используются идентификаторы.

Идентификатор может включать буквы латинского алфавита (строчные и прописные буквы в идентификаторах неразличимы), цифры и символ подчеркивания. Первым символом обязательно должна быть буква. Количество символов может быть произвольным, значащими являются первые 63 символа. В качестве идентификаторов нельзя использовать так называемые зарезервированные слова. *Зарезервированными* называются служебные слова, используемые в определении операторов и имеющие в программе фиксированный смысл, которые можно использовать только по их прямому назначению (например, `begin`, `end`), а также имена директив компилятора.

2.2. Переменные. Типы данных

Одним из основных объектов программы является *переменная*. Под переменную выделяется ячейка памяти, в которую можно поместить какое-либо значение (*присвоить* переменной значение). В Турбо-Паскале присваивание осуществляется при помощи *оператора присваивания*, например,