

ПАСКАЛЬ для школьников подготовка к ЕГЭ

3-е издание

Теоретическая часть не требует обращения к другим источникам

Дает возможность получить практические навыки выполнения типовых заданий, в том числе части С

Делает акценты на разделах программирования, выносимых на ЕГЭ

Содержит примеры экзаменов 2013 года и заданий ЕГЭ прошлых лет

ИНФОРМАТИКА И
ИНФОРМАЦИОННО-
КОММУНИКАЦИОННЫЕ
ТЕХНОЛОГИИ



Материалы
на www.bhv.ru

УДК 004.438 Pascal(075.3)
ББК 32.973.26–018.1я72
К31

Кашаев, С. М.

К31 Паскаль для школьников. Подготовка к ЕГЭ / С. М. Кашаев,
Л. В. Шерстнева. — 3-е изд., перераб. и доп. — СПб.: БХВ-Петербург, 2014. —
352 с.: ил. — (ИИИКТ)

ISBN 978-5-9775-0934-3

Подробно описаны приемы программирования на Паскале и технология разработки различных алгоритмов программ с акцентом на темы, выносимые на Единый государственный экзамен по информатике и информационно-коммуникационным технологиям. Рассматриваются: описание языка Паскаль, конструкции алгоритмов и блок-схемы, одномерные и двумерные массивы, строки и записи, файлы, численное интегрирование и анализ функций, подпрограммы и функции, работа с данными. По каждому разделу приводится теоретическая информация и типовые задания с подробными пояснениями. По темам, выносимым на прошедшие ЕГЭ, в том числе 2013 года, что отличает третье издание, в соответствующих главах приводятся примеры заданий этих ЕГЭ. Книга может использоваться как при подготовке к ЕГЭ, так и в текущем учебном процессе учащимися и учителями школ и колледжей, а также для самостоятельного изучения языка программирования Паскаль.

На сайте издательства размещены рассматриваемые в книге программы.

Для образовательных учреждений

УДК 004.438 Pascal(075.3)
ББК 32.973.26–018.1я72

Группа подготовки издания:

Главный редактор	<i>Екатерина Кондукова</i>
Зам. главного редактора	<i>Людмила Еремеевская</i>
Зав. редакцией	<i>Екатерина Капалыгина</i>
Редактор	<i>Анна Кузьмина</i>
Компьютерная верстка	<i>Ольги Сергиенко</i>
Корректор	<i>Зинаида Дмитриева</i>
Дизайн серии	<i>Инны Тачиной</i>
Оформление обложки	<i>Марины Дамбиевой</i>

Подписано в печать 29.11.13.

Формат 70×100^{1/16}. Печать офсетная. Усл. печ. л. 28,38.

Тираж 1500 экз. Заказ №

"БХВ-Петербург", 191036, Санкт-Петербург, Гончарная ул., 20.

Первая Академическая типография "Наука"
199034, Санкт-Петербург, 9 линия, 12/28

ISBN 978-5-9775-0934-3

© Кашаев С. М., Шерстнева Л. В., 2014
© Оформление, издательство "БХВ-Петербург", 2014

Оглавление

Введение	9
Краткое содержание книги	10
От авторов книги	11
 Глава 1. Знакомство с языком Паскаль	13
Среда Turbo Pascal.....	14
Программа вывода сообщения на экран.....	16
Вычисления в программе.....	18
Структура программы	21
Типы данных	22
Операции и выражения	26
Арифметические операции	27
Операции отношения.....	29
Приоритет операций.....	30
Ввод данных.....	31
Вывод данных	32
Примеры вычислений в программе	32
Использование блок-схем в разработках.....	34
Использование стандартных функций	35
Действия с данными разных типов	38
Константы	40
Работа с символами	41
Использование логического типа данных	42
Примеры	44
Типовые задачи и задания из ЕГЭ предыдущих лет.....	56
Ответы к задачам и заданиям из ЕГЭ	64
 Глава 2. Условия, выбор и циклы	69
Оператор условия	70
Оператор выбора	75
Оператор цикла <i>for</i>	80
Цикл с предусловием.....	87
Цикл с постусловием	89
Метки.....	90

Работа с символьными строками.....	91
Типовые примеры и задания из ЕГЭ.....	93
Подсчет суммы цифр в числе	93
Анализ четности пары чисел.....	95
Построение треугольников из отрезков.....	96
Перевод числа в шестнадцатеричную систему	98
Подсчет по условию	99
Возможность построения прямоугольного треугольника.....	100
Представление слова с учетом падежа.....	100
Формирование таблицы стоимости товаров.....	101
Поиск чисел.....	102
Анализ чисел	103
Графики зависимостей	111
Изменение чисел по условию	112
Типовые задачи и задания из ЕГЭ прошлых лет	113
Ответы к задачам и заданиям из ЕГЭ	121
Глава 3. Одномерные массивы.....	127
Нахождение суммы элементов массива.....	128
Суммирование элементов массива с учетом условия.....	128
Нахождение среднего арифметического	129
Нахождение среднего арифметического при условии	130
Поиск максимального элемента в массиве	131
Поиск индексов в массиве	132
Проверка упорядоченности массива.....	134
Обмен значений массива.....	135
Суммирование соседних элементов массива	137
Подсчет соседних элементов по условию	138
Перенос модулей значений в другой массив.....	140
Подсчет количества максимальных элементов.....	141
Изменение значений элементов массива с заданными свойствами	144
Нахождение индексов элементов с заданными свойствами	144
Удаление из массива определенного элемента	145
Циклическое перемещение элементов массива	147
Заполнение массива случайными числами.....	148
Нахождение суммы группы элементов массива	149
Задания из ЕГЭ прошлых лет	150
Ответы к заданиям из ЕГЭ.....	154
Глава 4. Двумерные массивы	161
Нахождение суммы элементов массива.....	161
Сумма элементов с заданными свойствами	162
Расчет среднего арифметического	163
Поиск минимального элемента	163
Поиск номера строки с минимальной суммой	166
Подсчет числа учащихся.....	167
Определение результата турнира	168
Расчет доходов по отделу	170
Анализ средней зарплаты сотрудников	171

Подсчет элементов по условию	171
Подсчет суммы элементов по условию	172
Нахождение индексов элементов	173
Нахождение уникальных элементов	174
Анализ тестирования	175
Изменение знака элементов	175
Изменение элементов по условию	176
Тур коня на шахматной доске	177
Задания из ЕГЭ прошлых лет	180
Ответы к задачам и заданиям из ЕГЭ	185
Глава 5. Строки и записи	191
Описание символьных строк	191
Действия над строками	192
Работа со строками как с элементами массивов	194
Строковые процедуры и функции	196
Процедуры для вставки и удаления символов	196
Функции для работы со строками	196
Процедуры преобразования типов	198
Примеры программ с обработкой строк	199
Поиск подстроки и удаление подстроки	199
Удаление пар символов при условии	199
Вставка одиночных символов в строку	201
Подсчет количества слов в строке	202
Подсчет количества символов фрагмента в строке	204
Удаление лишних пробелов	206
Вставка слова при условии	206
Нахождение суммы цифр	208
Удаление из строки цифр	209
Замена в строке	209
Использование массивов строк	210
Записи	211
Формирование списка учащихся	212
Анализ оценок учащихся	213
Поиск автомобиля по цене	214
Глава 6. Работа с файлами	217
Текстовые файлы	218
Создание текстового файла	219
Чтение из файла	220
Формирование файла с набором символов	220
Подсчет строк по условию	221
Создание файла на основании другого файла	222
Обмен символов в файле	223
Добавление в файл информации	224
Программа вывода на экран собственного текста	225
Нахождение максимальных чисел в строках	225
Нахождение среднего арифметического	226
Анализ файла	227

Обмен содержимого файлов	227
Разделение файла	228
Добавление в файл информации о количестве символов	229
Типизированные файлы	230
Создание нового типизированного файла	231
Чтение данных из типизированного файла	232
Последовательный доступ к файлу	233
Прямой доступ к файлу	233
Создание массива данных в файле	238
Организация базы данных учащихся	239
Разделение списка учащихся	241
Глава 7. Подпрограммы.....	243
Организация процедур	244
Параметры-массивы	246
Примеры использования процедур	247
Формирование разделяющей линии	247
Передача символа рисования линии	248
Передача переменной для символа линии	249
Процедура анализа четности числа	250
Передача параметров через глобальные переменные	250
Глобальное описание массива	251
Передача массива через ссылку	254
Вычисление факториала	255
Вычисление математических функций	256
Обмен значений переменных	257
Анализ чисел	257
Функции пользователя	258
Вычисление наибольшего значения	259
Вычисление процента	259
Расчет дохода по вкладу	260
Анализ текста	260
Функция поиска минимума в одномерном массиве	261
Функция подсчета соседних элементов массива	262
Функция изменения значений элементов массива	263
Функция вычисления суммы элементов двумерного массива	264
Глава 8. Математические вычисления.....	265
Расчет значений функции	265
Численное интегрирование	266
Решение уравнений	270
Квадратное уравнение	274
Решение неравенства	276
Определение принадлежности множеству	277
Метод Монте-Карло	280
Вычисление площади фигуры	281
Моделирование бросания игрального кубика	283
Статистика подбрасывания монет	284
Задания из ЕГЭ прошлых лет	285

Глава 9. Обработка данных.....	293
Анализ тестирования учащихся.....	293
Отчет по олимпиаде	297
Сертификаты	299
Результаты экзамена.....	305
Полупроходной балл	309
Сортировка.....	314
Сортировка выбором.....	314
Сортировка обменом значений.....	318
Анализ числа учащихся в классах	319
Статистика температуры.....	323
Формирование результатов тестирования в файле.....	326
Формирование в файле отчета об олимпиаде	330
Отчет о результатах экзамена.....	332
Формирование числа из символов	335
Анализ автозаправочных станций.....	342
Приложение. Описание электронного архива.....	345
Список используемой литературы	347
Предметный указатель	349

ГЛАВА 1



Знакомство с языком Паскаль

Из языков программирования среди учащихся и педагогов школ наибольшей популярностью пользуется Паскаль. При этом в качестве среды программирования чаще всего выбирается система Turbo Pascal, которая была разработана фирмой Borland и на протяжении последних двух десятков лет остается одним из наиболее удобных средств для изучения программирования. Все рассмотренные в книге теоретические разделы и примеры программирования алгоритмов приводятся именно в этой среде.

Основой функциональности среды Turbo Pascal является возможность создания и выполнения программ на Паскале. При этом для набора текста программ в этой среде имеется собственный текстовый редактор, хотя сами тексты можно создавать и в другом текстовом редакторе (например, в известном приложении Windows — Блокноте). Кроме редактора, основными компонентами среды также являются: *компилятор, компоновщик и отладчик*.

Turbo Pascal представляет собой технологическое средство для разработки программ, где каждая программа реализует тот или иной алгоритм. Понятие алгоритма занимает ключевое место в процессе программирования. *Алгоритм* представляет собой строгую систему правил, определяющую последовательность действий для решения конкретной задачи. Следуя такой системе, в разных ситуациях нужно действовать совершенно одинаково (по единой схеме). Можно сказать, что алгоритмом является такая последовательность арифметических и логических действий, которая позволяет решить поставленную задачу за конечное число шагов. В качестве решаемой задачи может быть, например, нахождение решения уравнения или поиск необходимой информации в файле с данными. Этапы разработки вычислительной программы выглядят следующим образом:

1. *Постановка задачи*, которая выполняется специалистом в предметной области (математика, физика, строительство и т. д.). На этом этапе определяется общий подход к решению задачи. Среди задач, рассматриваемых в книге, мы встретимся с поиском максимального значения в массиве данных, сортировкой чисел, расчетом средних показателей и т. д.

2. *Анализ задачи и моделирование*, которые приводят к построению (или выбору) математической модели. Этот этап очень важен, т. к. в ряде случаев анализ показывает, что поставленную задачу можно решить аналитическими методами.
3. *Построение алгоритма* решения задачи, выполняемое на основании математической модели. В большинстве ситуаций существует несколько способов построения работающего алгоритма. От разработчика всегда требуется найти оптимальную систему правил для решения поставленной задачи.
4. Реализация алгоритма в виде *работающей программы* на одном из языков программирования. Этот этап часто называют *кодированием* — записью алгоритма в виде набора синтаксических конструкций выбранного языка программирования.
5. *Отладка и тестирование* программы. Отладка заключается в устранении программных ошибок. Для поиска ошибок разработчик должен предложить набор тестов, представляющих собой контрольные примеры с характерными параметрами, для которых решение задачи известно. Тестирование позволяет ответить на вопрос — является ли разработанная программа наилучшим решением данной задачи.

Описанная последовательность шагов достаточно понятна, но ее осуществление зависит от сложности поставленной задачи (для простых задач реализация перечисленных шагов достаточно очевидна, а для сложных ситуаций часто требуются длительное время и большие усилия). Фактически все программные разработки, которые вы знаете и с которыми познакомитесь в книге, также вписываются в данную схему. Мы будем выполнять перечисленные этапы, начиная с простых задач. Затем после приобретения определенного опыта в последующих главах разберем темы, которые являются ключевыми как для школьного курса информатики, так и для ЕГЭ по дисциплине "Информатика и информационно-коммуникационные технологии".

Среда Turbo Pascal

Для запуска Turbo Pascal используется один из исполняемых файлов среды, в качестве которых могут выступать `turbo.exe` или `br.exe`. После выполнения любого из них на экране вы увидите элементы интегрированной среды:

- ☐ *главное меню* (строка сверху);
- ☐ *окно редактирования* ("полотно" синего цвета в центре);
- ☐ *описание функциональных клавиш* (строка в нижней части окна).

Несмотря на обширный набор элементов меню, тем не менее основных пользовательских действий не так уж много. Так, в разделе меню **File** (Файл) присутствуют команды **New** (Новый) и **Open** (Открыть). С помощью первой из них можно приступить к созданию нового файла, который будет содержать программу на языке Паскаль, а посредством второй открыть уже имеющийся файл. И в том, и в другом случае мы переходим к работе с текстовым редактором, входящим в систему Turbo Pascal.

Важно, что расширение файла, содержащего программу на языке Паскаль, должно быть PAS, и когда вы приступаете к созданию нового файла, то среда автоматически открывает новое окно с заголовком NONAME00.PAS. Это и есть имя файла, которое система Turbo Pascal определила автоматически. Далее от нас требуется написать программу, решающую ту или иную задачу.

Примечание

Для языка Паскаль не имеет значения, какие буквы вы используете при наборе текста программы (строчные либо заглавные). Обычно каждый программист сам выбирает свой стиль.

После набора текста программы ее необходимо сохранить на жестком диске. Для этого в меню **File** (Файл) имеется команда **Save File As** (Сохранить как). И от вас далее потребуется указать только имя для сохраняемого файла (практически всегда его выбирают самостоятельно вместо NONAME00.PAS).

Написанную программу на Паскале следует скомпилировать, скомпоновать и запустить на выполнение. В этом случае текст программы на языке Паскаль будет преобразован в последовательность команд, понятных процессору компьютера. Для выполнения всех этих перечисленных шагов за один прием предназначена команда **Run** (Запустить), которая находится в меню **Run** (Пуск).

Как правило, любая программа в процессе своей работы выводит информацию на экран (это могут быть промежуточные и итоговые результаты), после чего управление передается в среду (на экране опять отображается окно текстового редактора с исходной программой). Данную информацию можно увидеть на *экране пользователя*, для перехода в который предназначена комбинация клавиш <Alt>+<F5>. Чтобы вернуться обратно в окно редактора программного кода, необходимо нажать любую клавишу. В принципе, перечисленных сведений вполне достаточно для того, чтобы приступить к практической работе в среде Turbo Pascal и начать непосредственное знакомство с языком программирования. Но предварительно полезно пояснить *процесс компиляции* текста программы (написанной на языке Паскаль или на каком-либо другом языке программирования).

Исходный текст программы на Паскале не может непосредственно исполняться компьютером — вычислительная система не понимает синтаксических конструкций языка программирования. Для получения выполняемой программы исходный текст требуется обязательно *откомпилировать*, т. е. перевести в машинный код. *Машинный код* — это последовательность команд микропроцессора, состоящая из нулей и единиц. Так, первая команда представляет собой один набор двоичных разрядов, следующая команда — другой набор и т. д. Важно заметить, что каждый такой набор нулей и единиц вычислительная система однозначно понимает (в принципе, она *только это* и понимает).

Работу по переводу текста на языке Паскаль в машинный код выполняет специальная программа, называемая *компилятором*. Для компиляции программы в меню **Compile** (Компиляция) следует воспользоваться командой **Compile** (Компилировать). Если компилятор не обнаружит ошибок, то на экране появится окно с сообщением: *Compilation successful: press any key* (Компиляция прошла успешно:

нажмите любую клавишу). Данное окно остается на экране до тех пор, пока вы не нажмете какую-нибудь клавишу. Если обнаружена ошибка, то курсор устанавливается на ошибку в окне редактирования и выдается сообщение об ошибке.

Примечание

На практике удобнее пользоваться командой **Run** (Запустить), т. к. в этом случае ваша программа после компиляции будет автоматически запущена на выполнение (при условии, что в ней нет ошибок).

Программа, которую удалось откомпилировать, не обязательно работает правильно. Она может содержать ошибки, для выявления которых предназначен этап *отладки*. В целом программные ошибки могут быть трех видов:

- ☐ *синтаксические* (ошибки в правилах языка, которые обнаруживает компилятор);
- ☐ *алгоритмические* (ошибки в логике программы, которые при внешне безошибочной работе программы приводят к неверным результатам);
- ☐ *исполнения* (ошибки, возникающие в процессе работы запущенной программы).

Компилятор способен найти только синтаксические ошибки, для выявления же алгоритмических ошибок служит этап *тестирования* программы. Ошибки исполнения часто возникают как результат некорректных действий пользователя, недопустимых операций над данными (например, попытка извлечь квадратный корень из отрицательного числа или деление на ноль).

Программа вывода сообщения на экран

Практическую работу с языком программирования Паскаль начнем с простого примера. Так, наша задача заключается в создании программы, которая выводит на экран сообщение: "Пример первой программы". Подобные программы традиционно рассматриваются в качестве первых при изучении той или иной системы программирования. Необходимый текст программы (точнее один из вариантов, т. к. обеспечить вывод сообщения на экран можно и по-другому), который потребуется набрать в редакторе, представлен в листинге 1.1.

Листинг 1.1. Вывод сообщения на экран

```
program listing_1_1;  
begin  
  write('Пример');  
  write(' первой программы');  
end.
```

Разберем содержание листинга 1.1. Так, первая строка является заголовком программы и начинается со слова **program**, после которого располагается имя программы (`listing_1_1`). Имена программ следует выбирать самостоятельно. Слово **program** является *ключевым* (зарезервированным) в Паскале и используется для определения начала программы.

Примечание

Ключевых слов в Паскале довольно много, и в дальнейшем мы познакомимся и с другими.

Важно заметить, что имя программы не должно содержать пробелов. В ситуациях же, когда пробел напрашивается, используют символ подчеркивания. В приведенном примере мы как раз этим и воспользовались.

Рассматриваемую заголовочную строку можно не включать в программу, т. к. она не является обязательной. Однако для придания программным разработкам определенного стиля заголовки программ с информативными (для программиста) именами рекомендуется включать.

Далее в тексте листинга 1.1 размещается исполняемая часть программы, которая начинается с ключевого слова `begin`. После него следуют выполняемые *операторы* (инструкции) *программы*. Каждый оператор в Паскале должен завершаться точкой с запятой. Это фактически является символом отделения одного оператора от другого. И поэтому операторы можно записывать в одной строке (один оператор может следовать за другим через точку с запятой), что делает текст программы существенно компактнее.

Примечание

Имеются ситуации, когда точку с запятой в конце строки ставить не надо, и они нам встретятся в книге далее.

После слова `begin` в приведенной программе листинга 1.1 располагаются два оператора `write`. Данный оператор позволяет обеспечить вывод информации на экран. Варианты его использования могут быть достаточно сложными. Это зависит от того, что мы выводим (в последующих разделах об этом будет сказано более подробно). В рассматриваемом примере осуществляется вывод текстового сообщения на экран. Важно отметить, что выводимый текст в операторе `write` необходимо заключать в одинарные кавычки — *апострофы*.

Примечание

Оператор `write` фактически представляет собой *стандартную процедуру языка Паскаль*. По-другому такие процедуры называются *встроенными*.

Последняя строка листинга 1.1 является *директивой окончания* программы (`end.`). Таким образом, теперь текст программы стал понятен, и следующий этап заключается в ее выполнении в среде Turbo Pascal.

Для выполнения программы следует воспользоваться одним из перечисленных вариантов:

- ☐ воспользоваться командой **Run** (Запустить) в меню **Run** (Пуск);
- ☐ использовать сочетание клавиш <Ctrl>+<F9>.

Для просмотра результата работы необходимо нажать комбинацию клавиш <Alt>+<F5>. В этом случае на экране мы увидим строку с текстовым сообщением: "Пример первой программы".

В данной программе мы могли бы обойтись одним оператором вывода, в качестве параметра которого следовало включить все сообщение целиком:

```
write('Пример первой программы')
```

Такая редакция не требует завершения единственного оператора программы точкой с запятой (разделять нечего). Однако в листинге 1.1 мы разбили вывод сообщения на два оператора только для того, чтобы пояснить необходимость включения символа разделения операторов.

Таким образом, на рассмотренном примере мы познакомились с первой программой на Паскале и технологией ее выполнения в среде программирования Turbo Pascal.

Вычисления в программе

Как правило, большинство программ связано с организацией вычислительных действий. В качестве простого примера на данную тему рассмотрим реализацию программы, которая должна обеспечить возведение в квадрат значения целого числа, вводимого с клавиатуры. После этого результат вычисления необходимо вывести на экран. Подобная цель достаточно понятна, а набор правил для ее реализации выглядит так: ввод целого числа с клавиатуры, возведение этого числа в квадрат и вывод полученного результата на экран. Для описания алгоритмов, кроме такого словесного описания, используются *блок-схемы*, и с ними мы познакомимся далее в этой главе. Здесь же в связи с максимальной простотой решаемой задачи, кроме словесного перечисления шагов построения необходимого алгоритма для его пояснения больше ничего не требуется.

Примечание

Блок-схема — это графическое изображение алгоритма в виде плоских фигур, соединенных линиями. Блок-схема представляет собой стандартный способ записи алгоритма и существует государственный стандарт (ГОСТ), содержащий перечень правил построения блок-схем.

После формулировки задачи и определенности с алгоритмом перейдем к кодированию. Для этого в текстовом редакторе среды Turbo Pascal следует набрать текст программы (листинг 1.2), которая реализует поставленную задачу.

Листинг 1.2. Вычисление квадрата числа, вводимого с клавиатуры

```
program listing_1_2;  
var  
  N: integer;  
begin  
  read(N);  
  N:=N*N;  
  write(' Квадрат введенного числа равен ', N);  
end.
```

Новое ключевое слово, которое нам здесь встретилось, — это **var**. Оно означает начало области объявления переменных в программе.

Примечание

Сокращение **var** образовано от английского "variable" (переменная).

В тексте данной программы нам также встретился оператор *присваивания* (обозначается парой следующих друг за другом символов: двоеточие и знак равенства). Это наиболее часто используемый оператор, который работает следующим образом: сначала вычисляется выражение, стоящее *справа* от оператора присваивания **:=**, а затем результат записывается в переменную, стоящую *слева* от данного знака. Например, после выполнения оператора

```
K:=K+2;
```

текущее значение переменной **K** увеличится на 2.

Важно отметить, что справа от оператора присваивания может располагаться любое число либо выражение, а слева обязательно должно находиться имя переменной (мы должны указать системе — куда следует записать результат вычисления или просто данное). При этом тип результата выражения справа должен быть таким, чтобы он помещался в переменную слева от оператора присваивания.

Продолжим рассмотрение листинга 1.2. После начала программы располагается строка, где определяется переменная **n**. Дело в том, что данные, с которыми мы работаем, необходимо где-то хранить. И для этого предназначается основная (оперативная) память компьютера. Такая память состоит из набора однотипных ячеек, при этом каждая ячейка имеет свой номер, который называется *адресом*. Чтобы не работать с адресами ячеек напрямую (не запоминать длинные числовые адреса ячеек), предусмотрена возможность присвоения ячейкам символического имени (идентификатора). Во второй строке текста, приведенного в листинге 1.2, мы использовали ключевое слово **var**, которое говорит о том, что после него определяют переменные программы. В нашем случае это переменная **n**, и для нее мы указали тип **integer**, который определяет целочисленное данное. Для переменных указанного типа в памяти отводится 16 бит (16 двоичных разрядов или 2 байта). При этом числа, хранящиеся в переменных типа **integer**, являются знаковыми, и их диапазон составляет от -2^{15} до $2^{15} - 1$ [1, 2].

Примечание

Поскольку любые данные в памяти компьютера хранятся в двоичной системе счисления, то кроме имени переменной обязательно следует присвоить и *тип*, определяющий *диапазон значений*, принимаемых переменной, и *способ ее обработки* вычислительной системой.

Без дополнительной информации о *типе* данных, хранящихся в некоторой ячейке памяти, компьютеру было бы невозможно решить, что именно представляют собой эти данные. Например, 65 может быть просто числом в десятичной системе счисления, а может являться кодом буквы английского алфавита 'A' — все символы

в компьютере представлены *ASCII-кодами* (American Standard Code for Information Interchange, американский стандартный код обмена информацией) [1].

Имена (идентификаторы) переменных следует выбирать самостоятельно, и в рассматриваемой программе мы выбрали в качестве имени единственной переменной идентификатор *N*. Любые имена переменных в языке Паскаль строятся по следующим правилам:

- ❑ имена могут включать латинские буквы, цифры и знак подчеркивания (буквы русского алфавита не должны входить в состав идентификатора);
- ❑ имя состоит из одного слова, а если требуется пробел в имени, то он заменяется подчеркиванием (*St_1* будет правильным вариантом для имени, а *St 1* приведет к ошибке на этапе компиляции текста программы);
- ❑ имя всегда начинается с буквы либо со знака подчеркивания;
- ❑ между двумя именами должен располагаться как минимум один пробел;
- ❑ прописные и строчные буквы в именах не различаются компилятором языка Паскаль (*Z1* и *z1* — это одно и то же имя);
- ❑ имена переменных не могут совпадать с зарезервированными в языке *ключевыми (служебными) словами* (например, нельзя назвать *begin* ни одну переменную в программе);
- ❑ максимальное число символов, распознаваемых системой Turbo Pascal, в имени равняется 63, однако на практике практически всегда используются имена, не превышающие 10—20 символов.

Примечание

В программах, приводимых в книге, мы будем использовать следующий стиль выбора имен переменных: первая буква будет прописной, а последующие — строчными.

Вернемся к листингу 1.2, где после описания переменной располагаются три оператора. Первый из них представляет собой стандартную команду (*read*) ввода данных с клавиатуры. В скобках указано, что введенные данные фиксируются в переменной *N*. После этого располагается оператор (обозначается знаком ***), который реализует умножение — значение переменной умножается на само себя. Фактически это является возведением исходного значения в квадрат. Результат перемножения заносится обратно в переменную *N* (в ячейку памяти, которую мы отвели для переменной).

Заметим, что введенное с клавиатуры значение должно представлять собой целое число (тип *integer*), и в случае ввода набора букв вместо целого числа при выполнении программы произойдет ошибка. Это связано с тем, что операция умножения для строк и символов отсутствует.

После умножения в программе располагается команда вывода информации на экран. При этом через запятую необходимо перечислить данные, которые следует вывести. В данном примере мы выводим на экран текст, который информирует пользователя (текст в команде *write* должен заключаться в одинарные кавычки), а также значение переменной *N*.

Важно заметить, что точку после ключевого слова **end** в конце программы ставить нужно обязательно.

Структура программы

После двух рассмотренных простых практических примеров приведем теперь общую структуру программы на языке Паскаль:

```
program name_prog
{Раздел описаний}
begin
{Исполнительный раздел}
end.
```

В начале программы, как мы уже знаем, располагается необязательный заголовок, состоящий из ключевого слова **program** и имени программы. В заголовке могут также присутствовать параметры, с помощью которых программа взаимодействует с внешним миром. Имена программ позволяют программисту их быстро идентифицировать, поэтому ими желательно не пренебрегать.

В программе можно использовать *комментарии* — пояснительный текст. Текст комментария должен быть окружен либо фигурными скобками (`{...}`), либо следующими парами символов: круглая скобка и звездочка. Например:

```
(*      текст комментария      *)
```

Комментарий можно записать в любом месте программы, где разрешен пробел. Символами комментария удобно пользоваться при отладке программ для временного исключения группы операторов. В этом случае фрагмент, заключенный в символы комментария, воспринимается компилятором как комментарий и, следовательно, не будет выполнен.

В разделе описаний необходимо объявить все встречающиеся в программе данные. Это касается переменных, констант и т. д. Важно, чтобы все описания объектов программы были сделаны до того, как они будут использоваться. В разделе описаний можно выделить несколько подразделов:

- ☐ *раздел описания констант* (в отличие от переменных константы не изменяются в операторах исполнительного раздела);
- ☐ *раздел описания переменных*;
- ☐ *раздел описания меток* (перед любым оператором можно поставить метку, которая позволит выполнить переход на него с помощью оператора **goto** из любого места программы);
- ☐ *раздел объявления типов* (кроме уже имеющихся типов данных в системе можно определить и свои пользовательские типы);
- ☐ *раздел для подключения стандартных и пользовательских библиотечных модулей*;

□ *раздел описания процедур и функций* (это фрагменты программного кода, которые вызываются как единое целое).

В исполнительном разделе расположены инструкции (операторы), которые определяют алгоритм обработки данных. Важно, что этот раздел обязательно должен начинаться с ключевого слова `begin`, а завершаться ключевым словом `end`. (с точкой в конце).

Операторы в программах на Паскале не привязаны к конкретной позиции строки. Разрешается в одной строке размещать несколько операторов, разделяя их точкой с запятой.

Типы данных

Для решения любой задачи с использованием программирования производятся действия над определенными данными. Эти данные могут иметь разные типы. *Тип данных* определяет множество значений, которые могут принимать объекты программы (константы и переменные). Также тип определяет и совокупность действий, которые можно выполнять над данными конкретного типа. Например, с числами можно выполнять большой набор действий (сложение, вычитание, умножение, деление и т. д.), а строки можно только складывать.

Все типы данных, которые используются в системе Turbo Pascal, можно разделить на группы:

- скалярные (или простые);
- структурированные (составные).

Скалярные типы в свою очередь подразделяются на *стандартные* и *пользовательские*. Стандартные типы заложены в систему Turbo Pascal, и к ним относятся: целочисленные, вещественные, символьные, логические (булевы) и указатели. Категория структурированных типов (строки, массивы, множества, записи и файлы) рассмотрена в последующих главах.

Тип данных очень важен при выделении памяти под переменные. Это связано с тем, что каждому типу данных соответствует определенное число байтов памяти компьютера.

Рассмотрим, что представляют собой стандартные типы данных в системе Turbo Pascal. В примере листинга 1.2 мы уже встретились с одним из типов данных.

При организации вычислений программисты наиболее часто используют целочисленные типы данных, перечень которых представлен в табл. 1.1. Здесь указаны минимальные и максимальные значения для каждого типа данных, а также количество отводимых для них байтов в памяти компьютера. Действия над целыми числами указанных типов определены лишь тогда, когда исходные данные (операнды) и результат лежат в заданном интервале. В противном случае возникает *переполнение* [1]. Приведем пример описания нескольких переменных целочисленного типа:

```
var
```

```
  A: integer;
```

```
  B: longint;
```

```
  C: byte;
```

Как известно [1, 2], числа в вычислительной системе представлены в двоичной системе счисления и на каждое число в оперативной памяти отводится определенное количество разрядов (битов). В этом плане прокомментируем используемые в Паскале целые числовые типы данных. Так, наименьшее число разрядов (8) имеют типы `shortint` и `byte`. В случае использования `shortint` минимальное число равно -128 , а максимальное 127 . Такой тип данных позволяет работать только с короткими знаковыми числами. Если же воспользоваться типом `byte`, то нам предоставляется возможность работы только с беззнаковыми числами (но максимальное значение в этом случае будет больше).

Для вычислений наиболее часто используется тип `integer` (под данное отводится 16 двоичных разрядов), в котором минимальное число равно -2^{15} , а самое большое положительное равняется $2^{15} - 1$. Если требуется отразить большие целые числа, то следует воспользоваться типом `longint`. В этом случае числа могут принимать значения в интервале от -2^{31} до $2^{31} - 1$.

Не все числа, с которыми приходится работать, являются целыми. В большинстве вычислений используются дробные числа. С точки зрения математики таких чисел бесконечно много. Что же касается вычислительной системы, то их также много, но общее количество в этом случае конечно.

Дробные числа, кроме целой части, включают еще и дробную составляющую, отделяемую от целой разделителем (в качестве разделителя используется точка). Такие числа принято называть *вещественными*. В Паскале существует несколько вещественных типов для числовых данных.

Примечание

Вещественные типы данных занимают от 4-х до 10-ти байтов.

Для записи вещественных чисел в языке Паскаль можно использовать *обычный* и *экспоненциальный* форматы записи. Приведем в качестве примера несколько вещественных чисел в знакомом обычном формате: 4.5; -9.778; 90.678.

При использовании экспоненциального формата в записи числа выделяются две составляющие: мантисса и порядок. В этом случае число представляется в виде мантиссы, умноженной на 10 в определенной степени (степень соответствует порядку). Приведем пример нескольких чисел в таком формате:

□ $4700 = 4.7 \cdot 10^3$, что в указанном формате выглядит 4.7000000E+03;

□ $-25000 = -2.5 \cdot 10^4$ или в указанном формате -2.5000000E+04;

□ $-0.0005 = -5 \cdot 10^{-4}$ или -5.0000000E-04.

Подчеркнем, что в языке Паскаль при записи вещественных чисел основание 10 заменяется буквой E, после которой размещается значение порядка. Если вы уви-

дите подобные числа в результате выполненных вычислений, то не удивляйтесь. В табл. 1.2 приведены варианты вещественных типов данных в системе Turbo Pascal.

Таблица 1.1. Целочисленные типы данных в Turbo Pascal

Тип	Диапазон значений	Размер в байтах
byte	0...255	1
word	0...65 535	2
integer	−32 768...32 767	2
shortint	−128...127	1
longint	−2 147 483 648...2 147 483 647	4

Таблица 1.2. Вещественные типы данных в Turbo Pascal

Тип	Диапазон значений	Мантисса	Размер в байтах
real	2.9E−39...1.7E38	11—12	6
single	1.5E−45...3.4E38	7—8	4
double	5.0E−324...1.7E308	15—16	8
extended	3.4E−49321...1E4932	19—20	10

Приведем пример описания переменных вещественного типа в программе на языке Паскаль:

```
var K: real;
    W: double;
```

После такого определения уже можно в области исполнительных операторов программы работать с переменными *K* и *W*. Например, выполнить следующее присвоение значений:

```
K:=99.5;    W:=9.157;
```

Поскольку размер памяти, отводимой под мантиссу и порядок, ограничен, то вещественные числа всегда представляются в памяти компьютера с некоторой погрешностью. Например, простейшая вещественная дробь $\frac{2}{3}$ дает в десятичном представлении 0,666666..., и независимо от размера памяти, выделяемой для числа, невозможно хранить все цифры ее дробной части.

Вещественные числа в вычислительной системе представлены приближенно, и арифметические действия над ними также выполняются приближенно.

Для представления одиночных символов (букв, цифр и ряда других символов) в Паскале предназначен *символьный* тип данных. Для каждой переменной данного

типа в памяти отводится 1 байт, а для описания используется ключевое слово `char`. Приведем пример описания набора переменных символьного типа:

```
var
    x1, x2, x3: char;
```

После этого с ними можно работать, в частности, присвоить созданным переменным значения. Для этого следует использовать одинарные кавычки (необходимо поместить символ в одинарные кавычки):

```
x1:= 'A';
x2:= '1';
x3:= '+';
```

В результате в переменную `x1` будет занесен код английской буквы `A`. Как уже говорилось, каждому символу соответствует свой код, который занимает 8 бит или 1 байт. И вместо одинарных кавычек при обозначении символа можно использовать знак `#`, за которым следует поместить код символа. Например, можно написать `#65` вместо английской буквы `A`. Однако числовой код вместо указания самого символа разумно использовать только для специальных (служебных) символов, которые не имеют экранного представления. Например:

- ☐ `#07` — подача короткого звукового сигнала;
- ☐ `#08` — смещение курсора на одну позицию назад;
- ☐ `#10` — горизонтальная табуляция;
- ☐ `#13` — возврат каретки или перевод строки (соответствует нажатию клавиши `<Enter>`);
- ☐ `#26` — конец файла (соответствует комбинации клавиш `<Ctrl>+<Z>`);
- ☐ `#32` — пробел.

Кроме уже упомянутых типов данных, в Паскале используется *логический* (другое название — булевый тип). Ключевое слово для указания этого типа — `boolean`. Данные этого типа могут принимать только два возможных значения: `true` (истина) и `false` (ложь). Приведем пример описания переменных такого типа в программе на языке Паскаль:

```
var
    A1, A2, A3: boolean;
```

Дополнительно к стандартным типам в языке Паскаль можно использовать пользовательские типы данных. К ним относятся:

- ☐ интервальный (ограниченный) тип или диапазон;
- ☐ перечислимый тип.

В целом можно сказать, что данные типы используются для ограничения количества значений, принимаемых переменными. Это весьма полезно для исключения возможных ошибок при вводе или присвоении значений.

Интервальный тип данных позволяет задать две константы, которые определяют границы изменения переменных данного типа. Значение первой константы должно