Опыт не требуется

Книги этой серии наглядно свидетельствуют, что новичка можно научить языку и хорошему стилю программирования, не повергая его в тоску и уныние.
– Лу Гринзоу, обозреватель Dr. Dobb's Journal.

Исходные коды находятся на дискете

Сэм А. Аболрус

Программирование на PASCAL

СИМВОЛ

Learn Pascal in Three Days

Third Edition

Sam A. Abolrous

Wordware Publishing, Inc.

Опыт не требуется

Программирование на Pascal

Третье издание

Сэм А. Аболрус

Символ®

Санкт-Петербург — Москва
2003

Серия «Опыт не требуется»

Сэм А. Аболрус

Программирование на Pascal, 3-е издание

Перевод А. Петухова

Главный редактор	*А. Галунов*
Зав. редакцией	*Н. Макарова*
Научный редактор	*В. Озеров*
Редактор	*В. Овчинников*
Корректура	*С. Беляева*
Верстка	*Н. Гриценко*

Аболрус С.

Программирование на Pascal, 3-е издание. – Пер. с англ. – СПб: Символ-Плюс, 2003. – 328 с., ил.

ISBN 5-93286-057-X

Книга Сэма Аболруса «Программирование на Pascal» – третье, обновленное издание одного из лучших руководств по Паскалю, ставшего бестселлером. Она идеально подходит для приобретения базовых знаний о структурном программировании. Легкий стиль изложения и большое количество примеров и упражнений помогут начинающим программистам изучить язык за очень короткий срок. Начав с простейших программ обработки числовых данных и вывода на печать, вы научитесь писать реальные приложения, которые будут выполняться на любой платформе.

Прилагаемая дискета содержит исходный код примеров, ответы к упражнениям и файлы, необходимые для их выполнения.

ISBN 5-93286-057-X
ISBN 1-55622-805-8 (англ)

Издательство «Символ-Плюс». 199034, Санкт-Петербург, 16 линия, 7,
тел. (812) 324-5353, edit@symbol.ru. Лицензия ЛП N 000054 от 25.12.98.
Налоговая льгота – общероссийский классификатор продукции
ОК 005-93, том 2; 953000 – книги и брошюры.
Подписано в печать 15.08.2003. Формат 70x100$^{1}/_{16}$. Печать офсетная.
Объем 20,5 печ. л. Тираж 3000 экз. Заказ N
Отпечатано с диапозитивов в Академической типографии «Наука» РАН
199034, Санкт-Петербург, 9 линия, 12.

Оглавление

Об авторе

Сэм Аболрус (Sam Abolrous), программист с большим опытом дизайна и разработки программного обеспечения, получил степень бакалавра инженера-электрика в университете Александрии, Египет.

Аболрус публиковался в ведущих журналах по программированию, написал более 50 книг в компьютерной области от Кобола до C++, включая «Learn C in Three Days» и «Learn Pascal», опубликованные издательством Wordware Publishing. Он разработал множество программ в области исследования слуха в медицинском центре университета штата Луизиана (Louisiana State University Medical Center, LSUMC). В настоящее время работает программистом в корпорации Microsoft.

Благодарности

Я бы хотел поблагодарить мою дочь Салли Аболрус (Sally Abolrous) за помощь при редактировании этой книги.

Предисловие

На примерах, приведенных в этой книге, можно изучить Паскаль за очень короткий срок. Вы начнете с простейших программ обработки числовых данных и вывода на печать, а закончите созданием реальных приложений с использованием структурного программирования.

Паскаль был разработан в начале 70-х годов Николасом Виртом (Niklaus Wirth), швейцарским программистом, и назван в честь французского математика Блеза Паскаля (Blaise Pascal, 1623–1662). Стандарт языка появился в 1983 году и был утвержден Институтом инженеров по электротехнике и электронике (IEEE) и Американским национальным институтом стандартов (ANSI). С ростом использования микрокомпьютеров язык подвергся изменениям и дополнениям, из которых наиболее популярны UCSD Pascal (разработан Калифорнийским университетом в Сан Диего) и Turbo Pascal фирмы Borland International.

Цель книги – научить писать переносимые, платформо-независимые программы с использованием, главным образом, стандартов IEEE/ANSI. Мы обсудим также новые возможности языка, ссылаясь на источники. Здесь не рассматриваются подробно непереносимые части языка (такие как графика), но зато уделяется внимание мощным возможностям современных реализаций, полезным при обработке данных. Программы, приведенные в книге, компилировались с помощью Turbo Pascal, но вы вправе использовать компилятор по своему выбору. Лишь в редких случаях потребуются небольшие изменения, к которым есть соответствующие комментарии.

Книга содержит ряд специальных обозначений:

Примечания. Содержат дополнительную информацию по сложным темам.

Предупреждения. Предостерегают о возможной опасности совершения ошибки.

Советы. Содержат конкретные рекомендации, облегчающие понимание материала и правильное выполнение примеров.

1

Привет, Паскаль

1–1. Ваша первая программа на Паскале

Программа на Паскале может быть простой, как в примере 1-1. Она выводит на экран фразу «Привет всем».

```
{ ---------------------------- Пример 1-1 ----------------------------- }
PROGRAM FirstProgram(OUTPUT);
BEGIN
   WRITELN('Привет всем')
END.
```

Большая или маленькая, программа на Паскале должна иметь определенную структуру. В примере программа состоит, главным образом, из одного оператора (WRITELN), который и делает всю работу – выводит на экран содержимое в скобках. Оператор помещен в блок, начинающийся с ключевого слова BEGIN и заканчивающийся ключевым словом END. Он называется *основным программным блоком* (или просто *программным блоком*) и обычно представляет основной алгоритм обработки данных.

Комментарии

Рассмотрим первую строку программы:

```
{ ---------------------------- Пример 1-1 ----------------------------- }
```

Это *комментарий*, и он всегда игнорируется компилятором. Можно располагать комментарии в любом месте программы на Паскале между фигурными скобками {} или между символами (* и *), например так:

```
(* Это комментарий *)
```

Заголовок программы

Вторая строка называется *заголовком программы*. Она начинается с ключевого слова PROGRAM, затем следуют пробел и имя программы (FirstProgram). Имя программы определяется пользователем. Определенные пользователем слова называются в Паскале *идентификаторами*. Идентификатор обязательно начинается с буквы и может состоять из любого количества букв и цифр (в Турбо Паскале он может также содержать символ подчеркивания). Для программы можно выбрать любое осмысленное имя, но не рассчитывайте, что пройдут такие имена, как «BEGIN» или «PROGRAM». Это *зарезервированные слова*, которые могут быть использованы только в определенных местах программы. Зарезервированные слова Паскаля собраны в приложении В.

За именем программы в скобках следует слово OUTPUT, которое завершается точкой с запятой:

```
PROGRAM FirstProgram(OUTPUT);
```

Ключевое слово OUTPUT сообщает компилятору, что программа будет выводить данные (например, на экран). Его противоположностью является слово INPUT (обозначающее, например, ввод с клавиатуры). Слова OUTPUT и INPUT называются *файловыми параметрами*. Программа может выполнять и ввод и вывод, и в этом случае параметры принимают вид:

```
PROGRAM FirstProgram(INPUT,OUTPUT);
```

В Турбо Паскале заголовок необязателен. Можно пропустить всю строку и начать программу со слова BEGIN или указать имя программы без параметров, например так:

```
PROGRAM FirstProgram;
```

Синтаксис и соглашения

Наиболее важный элемент синтаксиса – это точка с запятой после заголовка программы (которая используется в качестве разделителя) и точка после слова END, завершающая программу. Общепринято писать ключевые слова Паскаля в верхнем регистре, а слова, определенные пользователем (идентификаторы), – в нижнем регистре, начиная с заглавной буквы. Если имя программы состоит из двух и более слов (как в нашем случае), то каждое слово начинается с заглавной буквы. Таким образом, в программах, написанных на Паскале, можно встретить такие идентификаторы:

Wages

PayRoll

HoursWorkedPerWeek

Это всего лишь соглашение, призванное облегчить чтение программы. Компилятор Паскаля не чувствителен к регистру, поэтому всю программу можно написать и в нижнем (как в примере 1-2) и в верхнем (как в примере 1-3) регистре. Все три программы скомпилируются и будут работать.

```
{ ------------------------------ Пример 1-2 ------------------------------ }
program firstprogram(output);
begin
   writeln('Привет всем')
end.
```

```
{ ------------------------------ Пример 1-3 ------------------------------ }
PROGRAM FIRSTPROGRAM(OUTPUT);
BEGIN
   WRITELN('Привет всем')
END.
```

Все пустые строки, отступы и пробелы (за исключением тех, которые стоят после ключевых слов) необязательны, но такой стиль является хорошей программистской привычкой, позволяющей писать хорошо организованные и легко читаемые программы.

1–2. Вывод текста: WRITELN, WRITE

Как показано в примере 1-4, чтобы вывести на экран несколько строк текста, необходимо написать оператор WRITELN для каждой строки. Не забудьте заключить текстовые строки в кавычки.

Для того чтобы сберечь ваше время и силы, на прилагаемую к книге дискету помещены исходные коды примеров и решения заданий. Ознакомьтесь с файлами Readme.txt или Readme.htm, имеющимися на дискете. В них приведены инструкции по установке файлов на жесткий диск.

```
{ ------------------------------ Пример 1-4 ------------------------------ }
PROGRAM LinesOfText(OUTPUT);
BEGIN
   WRITELN('Привет всем');
   WRITELN('Как вы сегодня?');
   WRITELN('Вы готовы к изучению Паскаля?')
END.
```

Теперь в программе больше одного оператора. Все операторы разделяются точкой с запятой. Это единственный признак, по которому компилятор может определить, где заканчивается оператор, хотя для последнего оператора в программном блоке точку с запятой можно опустить.

После компиляции программа выведет на экран:

```
Привет всем.
Как вы сегодня?
Вы готовы к изучению Паскаля?
```

Оператор WRITELN выводит строку текста и переходит на новую строку (перевод строки и возврат каретки). Для того чтобы вывести две строки в одной, надо применить оператор WRITE, как показано в следующей программе:

```
{ ------------------------------ Пример 1-5 ------------------------------ }
PROGRAM TwoLines(OUTPUT);
BEGIN
   WRITE('Привет всем. ');
   WRITELN('Как вы сегодня?);
   WRITELN('Вы готовы к изучению Паскаля?')
END.
```

Вывод программы:

```
Привет всем. Как вы сегодня?
Вы готовы к изучению Паскаля?
```

Как видите, второе предложение выводится в той же строке, что и первое, поскольку для вывода первой строки использовался оператор WRITE. Это единственное различие между операторами вывода WRITE и WRITELN. Для вывода пустой строки напишите оператор:

```
WRITELN;
```

Задание 1–1

Напишите программу на Паскале, которая выводит на экран следующий текст:

```
Wordware Publishing, Inc.
-------------------------
2320 Los Rios Boulevard
Plano, Texas 75074
```

1–3. Обработка чисел

Любой программе проще всего работать с числами. Оператор WRITELN (или WRITE) можно использовать и для вывода чисел, и для вычисления выражений. Арифметические выражения конструируются с помощью *арифметических операторов*:

- \+ сложение
- – вычитание
- * умножение
- / деление

Взгляните на примеры:

```
WRITELN(123);
WRITELN(1.23 * 4);
```

В первом примере выводится число, заключенное в скобки. Во втором примере выводится результат умножения двух чисел. Заметьте, что для численных значений, в отличие от текстовых строк, кавычки не используются.

Применив запятую в качестве разделителя, можно в одном операторе WRITELN вывести и числа и текст:

```
WRITELN('Результат=', 125 * 1.75);
```

Следующая программа вычисляет два выражения (умножение и деление) и отображает результат, предваряя его соответствующим текстом.

```
{ ------------------------------ Пример 1-6 ------------------------------}
PROGRAM CrunchNumbers(OUTPUT);
BEGIN
   WRITELN('Я легко могу обрабатывать числа.');
   WRITELN('Это умножение 50x4:',50*4);
   WRITELN('..а это деление 2400/8:',2400/8)
END.
```

Вывод этой программы:

```
Я легко могу обрабатывать числа.
Это умножение 50x4:200
..а это деление 2400/8: 3.0000000000E+02
```

Умножение выполнилось с ожидаемым результатом. Два операнда (50 и 4) были целыми числами, и результат (200) тоже целое число. Однако формат результата деления требует пояснений.

Целые и вещественные числа

Деление, выполненное оператором /, называется *вещественным делением* и всегда дает в результате вещественное число. Вещественные числа могут быть записаны в форме с фиксированной точкой, например 300.0, или в экспоненциальной форме (scientific), например 3.0E+02, но в Паскале по умолчанию вещественные числа выводятся всегда в экспоненциальной форме. Число в экспоненциальной форме состоит из двух частей, разделенных буквой «E» (или «e»). Левая часть называется *мантиссой* и представляет собой число со знаком, а правая называется *экспонентой*. Экспонента – это степень десяти, которая определяет положение десятичной точки. Так, в этом примере число:

3.0000000000E+02 – это то же самое, что и 3×10^2.

Это же число в форме с фиксированной точкой будет выглядеть так:

300.0

Если перед экспонентой стоит минус:

3.124E–2

то десятичная точка сдвигается в данном случае на две позиции влево. Тогда число будет таким:

0.03124

В случае отрицательного числа минус размещается перед мантиссой:

–0.0124E–02

Если число положительное, то знак перед мантиссой не ставится:

1.23E02

Оператор деления (/) называется *оператором вещественного деления*, так как независимо от типа операндов результат всегда вещественный.

Для *целочисленного деления* надо применять оператор DIV, например:

```
WRITELN(2400 DIV 8);
```

В результате получим 300.

При целочисленном делении дробные части отбрасываются, например:

```
WRITELN(9 DIV 4);    результат выполнения равен 2.
```

Другой важный оператор предназначен для получения остатка целочисленного деления (деление по модулю), например:

```
WRITELN(9 MOD 4);    результат выполнения равен 1,
WRITELN(3 MOD 4);    результат выполнения равен 3.
```

Операторы DIV и MOD принимают только целые операнды и выдают целый результат.

Остальные операторы (+, – и *) выдадут вещественный результат, если хотя бы один из операндов будет вещественным.

Задание 1–2

Вычислите следующие выражения и напечатайте результат в виде целого числа (если результат целочисленный) или в виде вещественного числа (если результат вещественный):

A. 144 / 12 B. 144 DIV 12

C. 17 MOD 5 D. 3 MOD 5

E. 3e+02 + 3 F. 345E–01 – 1

Вычисление арифметических выражений

При создании более сложных арифметических выражений необходимо следить за приоритетом входящих в выражение операторов. Взгляните на эти два выражения:

```
2 + 10 / 2
(2 + 10) / 2
```

Оба выражения состоят из одних и тех же чисел и операторов, но первое равно 7, а второе – 6. Дело в том, что в первом выражении деление предшествует сложению, а во втором для изменения порядка вычислений используются скобки, при этом первым вычисляется выражение в скобках. В основном арифметические операторы Паскаля имеют два уровня приоритета: *высокий* и *низкий*.

Операторы + и – имеют низкий приоритет, все остальные – высокий. Если два оператора в выражении имеют одинаковый приоритет, они выполняются слева направо. Рассмотрим пример:

```
5 + 3 * 2 - 6 DIV 2
```

Первой должна быть выполнена операция умножения:

```
5 + 6 - 6 DIV 2
```

Вторая операция высшего приоритета – деление:

```
5 + 6 - 3
```

Оставшиеся две операции имеют равный приоритет. Они выполняются слева направо, давая в результате:

```
8
```

Когда для изменения порядка вычислений используются скобки, они образуют подвыражение, которое вычисляется первым. Рассмотрим тот же пример с вложенными скобками:

```
((5 + 3) * 2 - 6) DIV 2
```

Это выражение вычисляется по шагам:

```
(8 * 2 - 6) DIV 2
(16 - 6) DIV 2
10 DIV 2
5
```

Приоритеты и свойства арифметических операторов приведены в табл. 1-1.

Знаки + и – используются в качестве *унарных операторов* (для обозначения положительных и отрицательных чисел). Унарные операторы имеют тот же низкий приоритет, что и бинарные операторы + и –. Если бинарный оператор предшествует унарному, например 5*–4, то унарный оператор вместе с числом необходимо заключить в скобки: 5*(–4). Компилятор может принять первую форму записи, но лучше ее не использовать.[1]

[1] Современные компиляторы языка Паскаль не требуют при совершении арифметических действий заключать отрицательные числа в скобки. – *Примеч. науч. ред.*

Таблица 1-1. Арифметические операторы

Оператор	Арифметическая операция	Операнды	Результат	Приоритет
+	Сложение	REAL/INTEGER	REAL/INTEGER	Низкий
–	Вычитание	REAL/INTEGER	REAL/INTEGER	Низкий
*	Умножение	REAL/INTEGER	REAL/INTEGER	Высокий
/	Вещественное деление	REAL/INTEGER	REAL	Высокий
DIV	Целочисленное деление	INTEGER	INTEGER	Высокий

Задание 1–3

Вычислите следующие выражения и напечатайте результат в виде целого числа (если результат целочисленный) или в виде вещественного числа (если результат вещественный):

A. 15 – 15 DIV 15
B. 22 + 10 / 2
C. (22 + 10) / 2
D. 50 * 10 – 4 MOD 3 * 5 + 80

1–4. Переменные

Данные хранятся в памяти по определенным адресам. Однако программисты ссылаются на них с помощью переменных. При использовании переменных в программе им отводится определенное место в памяти. Значением переменной в действительности является содержимое этого участка памяти. При обработке данных программой содержимое любого участка памяти может измениться, меняя, таким образом, значение соответствующей переменной. Имена (идентификаторы) назначаются переменным в соответствии с упомянутыми ранее правилами.

Объявление переменных

Перед тем как использовать переменные в программе на Паскале, их имена и типы необходимо объявить в специальном разделе программы, называемом *объявлением*.

Он начинается с ключевого слова VAR, как показано в примере:

```
VAR
   a :INTEGER;
   x :REAL;
```

Переменная `a` имеет тип INTEGER, то есть она может принимать только целые значения, такие как 4, 556, 32145. Переменная `x` объявляет-

ся с типом REAL и может содержать только вещественные числа, такие как 3.14, 44.567, 3.5E+02.

Для того чтобы объявить более одной переменной того же типа, каждую из них можно поместить в отдельную строку:

```
VAR
   a :INTEGER;
   b :INTEGER;
   c :INTEGER;
   x :REAL;
   y :REAL;
```

Можно также объявить все однотипные переменные списком, например:

```
VAR
   a, b, c    :INTEGER;
   x, y       :REAL;
```

Ключевые слова INTEGER и REAL определяются как *стандартные идентификаторы*, которые предопределены в Паскале. Программист может переопределить стандартные идентификаторы, но это крайне не рекомендуется.[1] Стандартные идентификаторы перечислены в приложении B.

В следующей программе объявляются три переменные: целочисленные a и b и вещественная x. Содержимое каждой их них выводится на экран с помощью оператора WRITELN:

```
{ ------------------------------ Пример 1-7 ------------------------------ }
PROGRAM Variables(OUTPUT);
{ Объявление переменных }
VAR
   a, b :INTEGER;
   x :REAL;
{ Программный блок }
BEGIN
   WRITELN('Значение a=',a);
   WRITELN('Значение b=',b);
   WRITELN('Значение x=',x)
END.
```

Вывод программы может быть таким:

```
Значение a=0
Значение b=631
Значение x=2.7216107254E-26
```

Заметьте, что значения переменных a и b отображаются как целые, а значение x выводится в вещественном формате. Однако этот вывод чисел бесполезен, поскольку никаких значений переменным присвоено

[1] Не все компиляторы позволяют это сделать. – *Примеч. науч. ред.*

не было. Пока вы не присвоите переменным значения, они будут содержать то, что находилось в соответствующей области памяти ранее.[1]

Оператор присваивания

Для занесения значения в переменную можно использовать *оператор присваивания* (:=), например:

```
a := 55;
x := 1.5;
y := 2.3E+02;
```

Не используйте следующую форму записи вещественного числа:

.1234

Правильно записанное вещественное число в Паскале должно иметь цифру слева от десятичной точки, например:

0.1234

Точно так же число:

123.

может быть отвергнуто некоторыми компиляторами. Лучше использовать разрешенную форму записи:

123.0

В следующей программе объявляются два целых числа в разделе объявлений, а затем в программном блоке им присваиваются целые значения. После этого для вычисления и вывода результатов различных арифметических операций над этими числами применяется оператор WRITELN.

```
{ ------------------------------ Пример 1-8 ------------------------------ }
PROGRAM Arithmetic(OUTPUT);
{ Объявление переменных }
VAR
   a, b :INTEGER;
{ Программный блок }
BEGIN
   a := 25;
   b := 2;
   WRITELN('a=',a);
   WRITELN('b=',b);
   WRITELN('a+b=',a+b);
   WRITELN('a-b=',a-b);
   WRITELN('a*b=',a*b);
   WRITELN('a/b=',a/b);
   WRITELN('a div b=',a DIV b);   { используется только с целыми }
```

[1] Современные компиляторы могут инициализировать переменные начальными данными, но полагаться на действия компилятора не стоит. – *Примеч. науч. ред.*

```
   WRITELN('a mod b=',a MOD b)    { используется только с целыми }
END.
```

Вывод этой программы:

```
a=25
b=2
a+b=27
a-b=23
a*b=50
a/b= 1.2500000000E+01       ----> Вещественное деление
a div b=12                  ----> Целочисленное деление
a mod b=1                   ----> Остаток целочисленного деления
```

Можно присвоить одной переменной другую:

```
x := y;
```

В этом случае значение переменной `y` копируется в переменную `x`. Можно также переменным присваивать арифметические выражения, например:

```
z := a + b - 2;
GrossPay := PayRate * HoursWorked;
```

В этих операторах вычисляется выражение справа от оператора присваивания и запоминается в переменной, стоящей слева от оператора присваивания (`z` или `GrossPay`).

Задание 1–4

Напишите программу на Паскале, которая выполняет следующее:

A. Присвоить значение 2 переменной `a` и значение 9 переменной `b`.

B. Вывести на экран значения выражений:

a+b DIV 2

(a+b) DIV 2

1–5. Именованные константы

Значения данных (во многих языках, включая Паскаль) называются *константами*, поскольку они не изменяются в процессе выполнения программы. В Паскале имеется два типа констант:

- Литеральные константы
- Именованные константы

Литеральные константы[1] – это значения данных, такие как числа или текстовые строки, а *именованные константы* – это «переменные-

[1] Иногда их называют типизированными константами. – *Примеч. науч. ред.*

константы». Именованная константа отличается от переменной тем, что ее значение не изменяется в процессе выполнения программы. Подобно переменным, именованной константе дается имя, и она должна быть объявлена в разделе объявлений. В действительности раздел объявлений состоит из двух частей, CONST и VAR; секция CONST предшествует секции VAR.[1] Предположим, вы хотите использовать в вычислениях значение 3.14159 (численная константа, известная как π) много раз.

Было бы удобно дать ей имя и использовать его в коде. Именованные константы можно объявлять следующим образом:

```
CONST
   Pi = 3.14159;
   ThisYear = 1992;
   Department= 'OtoRhinoLaryngology';
```

Некоторые константы в Паскале предопределены как стандартные идентификаторы. Одна из полезных именованных констант – это MAXINT, которая представляет максимально возможное целое. Ее значение зависит от используемого компьютера.[2] Значение MAXINT для вашей машины можно выяснить с помощью оператора WRITELN:

```
WRITELN(MAXINT);
```

Как правило, это 32,767 (два байта).

В следующей программе именованная константа Pi используется для вычисления длины окружности.

```
{ ------------------------------ Пример 1-9 ---------------------------- }
PROGRAM Constants(OUTPUT);
{ Объявление констант}
CONST
   Pi = 3.14159;
{ Объявление переменных }
VAR
   Radius, Perimeter :REAL;
{ Программный блок }
BEGIN
   Radius := 4.9;
   Perimeter := 2 * Pi * Radius;
   WRITELN('Длина окружности=', Perimeter)
END.
```

Вывод этой программы:

```
Длина окружности= 3.0787582000E+01
```

---

[1] Современные компиляторы допускают любой порядок расположения секций. – *Примеч. науч. ред.*

[2] Или компилятора. – *Примеч. науч. ред.*

В Delphi или Турбо Паскале не надо переопределять константу Pi, поскольку она предопределена как стандартный идентификатор.

1–6. Преобразование типов: ROUND, TRUNC

Можно присвоить целое значение переменной вещественного типа, но не наоборот. Причина в том, что для целого числа отводится меньше памяти, чем для вещественного. Если бы это было разрешено, то данные, перемещаемые в область памяти недостаточного размера, могли бы потеряться или испортиться. Однако есть возможность выполнить преобразование при помощи следующих двух функций:

ROUND(*n*) округляет *n* до ближайшего целого

TRUNC(*n*) отсекает дробную часть *n*

где *n* – вещественная переменная или выражение.

Рассмотрим примеры:

`ROUND(8.4)` возвращает 8
`ROUND(8.5)` возвращает 9
`TRUNC(8.4)` возвращает 8
`TRUNC(8.5)` возвращает 8

Как видно из примеров, эти две функции могут возвращать разные значения для одного и того же аргумента.

В следующей программе эти функции используются для получения округленного и усеченного значения вещественной переменной `Perimeter`.

```
{ ------------------------------ Пример 1-10 ---------------------------- }
PROGRAM Functions1(OUTPUT);
{ Объявление констант }
CONST
  Pi = 3.14159;
{ Объявление переменных }
VAR
  Perimeter, Radius                    :REAL;
  RoundedPerimeter, TruncatedPerimeter :INTEGER;
{ Программный блок }
BEGIN
  Radius := 4.9;
  Perimeter := 2*Pi*Radius;
  RoundedPerimeter := ROUND(Perimeter);
  TruncatedPerimeter := TRUNC(Perimeter);
  WRITELN('Длина окружности=', Perimeter);
  WRITELN('Длина окружности (округленная)=', RoundedPerimeter);
  WRITELN('Длина окружности (усеченная)=', TruncatedPerimeter)
END.
```

Вывод:

```
Длина окружности= 3.0772000000E+01 ----> Исходное значение
Длина окружности (округленная)=31  ----> Округленный результат
Длина окружности (усеченная)=30    ----> Усеченный результат
```

1–7. Чтение с клавиатуры: READLN, READ

Предыдущая программа вычисляла длину окружности данного радиуса, значение которого было «зашито» в коде программы. Более полезной была бы программа, которая получала бы значение от пользователя, производила вычисления и выдавала результат. Для того чтобы программа делала паузу и ожидала пользовательский ввод, можно применить оператор READLN или READ. Оператор READLN предназначен для чтения значений в одну или несколько переменных. Его основная форма:

READLN(список переменных);

Для чтения значения с клавиатуры в переменную x можно применить оператор:

```
READLN(x);
```

Чтобы прочитать значения в три переменные x, y и z, напишите оператор:

```
READLN(x, y, z);
```

Значения, вводимые более чем в одну переменную (например, x, y и z), должны разделяться хотя бы одним пробелом или нажатием клавиши <Enter>.

Замените оператор присваивания в предыдущей программе на оператор READLN:

```
READLN(Radius);
```

После запуска программа будет ждать, пока вы не введете число и не нажмете <Enter>. Затем она произведет вычисления и выведет результат на экран. К сожалению, оператор READLN не отображает информацию, вводимую пользователем. Это можно сделать с помощью оператора WRITE (или WRITELN):

```
WRITE('Пожалуйста, введите радиус:');
```

Измененная программа:

```
{ ------------------------------ Пример 1-11 ----------------------------- }
PROGRAM KeyboardInput(OUTPUT);
{ Объявление констант }
CONST
  Pi = 3.14159;
{ Объявление переменных }
VAR
```

```
  Perimeter, Radius                    :REAL;
  RoundedPerimeter, TruncatedPerimeter :INTEGER;
{ Программный блок }
BEGIN
  WRITE(' Пожалуйста, введите радиус:');
  READLN(Radius);
  Perimeter := 2*Pi*Radius;
  RoundedPerimeter := ROUND(Perimeter);
  TruncatedPerimeter := TRUNC(Perimeter);
  WRITELN('Длина окружности=', Perimeter);
  WRITELN('Длина окружности (округленная)=', RoundedPerimeter);
  WRITELN('Длина окружности (усеченная)=', TruncatedPerimeter)
END.
```

Выполнение теста дает следующий результат:

```
Пожалуйста, введите радиус:4.9 ----> Напечатайте число и нажмите ENTER
Длина окружности= 3.0787582000E+01
Длина окружности (округленная)=31
Длина окружности (усеченная)=30
```

На этом этапе вы можете использовать или READ, или READLN, поскольку разница между ними для наших приложений пока не имеет значения.

1–8. Форматирование вывода

Возможно, вы подумали, что экспоненциальная форма – не лучший формат для вывода, особенно для бизнеса или финансовых расчетов. Вы правы. Экспоненциальная нотация хороша только для очень больших или очень маленьких чисел, когда степень десяти представляет порядок величины числа.

Для того чтобы отобразить число в форме с фиксированной точкой, используйте такое *описание формата*:

```
WRITELN(Wages :6:2);
```

Формат «:6:2» определяет поле длиной 6 символов, включая два десятичных знака. Таким образом, если переменная `Wages` равна 45.5, то она отобразится так:

```
B45.50
```

где буква «B» обозначает пробел. Если количество цифр в выводимом числе меньше ширины поля вывода, результат будет сдвинут вправо. Если число не помещается в поле вывода, поле будет увеличено и на экран выведется число целиком. Можно добавить символ (например, символ доллара) слева от числа:

```
WRITELN('$',Wages :6:2);
```

Тогда будет выведено:

```
$ 45.50
```

Используя меньшую ширину поля, можно сдвинуть число влево, к символу доллара:

```
WRITELN('$',Wages :0:2);
```

В результате будет выведено:

```
$45.50
```

Этим методом можно форматировать любой тип данных, за исключением целых чисел и текстовых строк, когда ширина поля определяется без десятичных знаков.

В следующей программе различные типы данных форматируются применительно к определенным полям вывода:

```
{ ------------------------------ Пример 1-12 ----------------------------- }
PROGRAM Format(OUTPUT);
{ Объявление переменных }
VAR
  a :INTEGER;
  b :REAL;
{ Программный блок }
BEGIN
  b := 1.2e+02;
  a := 320;
  WRITELN('Я текстовая строка, начинающаяся с позиции 1.');
  WRITELN('Теперь я сдвинута к правому концу поля.':50);
  WRITELN(Я неформатированное целое:', a);
  WRITELN('Я целое, записанное в поле шириной 6 символов:', a:6);
  WRITELN('Я количество денег, записанное в поле шириной 8 символов:$',b:8:2);
  WRITELN('Я количество денег, сдвинутое влево:$',b:0:2)
END.
```

Вывод:

```
Я текстовая строка, начинающаяся с позиции 1.
    Теперь я сдвинута к правому концу поля.
Я неформатированное целое:320
Я целое, записанное в поле шириной 6 символов:   320
Я количество денег, записанное в поле шириной 8 символов:$  120.00
Я количество денег, сдвинутое влево:$120.00
```

Если вывести одни числа без текста, то будет:

```
320
   320
$  120.00
$120.00
```

Задание 1–5

Напишите программу, вычисляющую зарплату сотрудников по формуле:

```
Wages := HoursWorked * PayRate;
```

Введите значения переменных `HoursWorked` и `PayRate` с клавиатуры, а значение `Wages` выведите на экран в нотации с фиксированной точкой, поместив в начало символ доллара.

Заключение

В этой главе вы познакомились с наиболее важными инструментами программирования на Паскале.

1. Теперь вы хорошо знаете, что такое:
   - Заголовок программы
   - Раздел объявлений
   - Секция CONST
   - Секция VAR
   - Тело программы между BEGIN и END
2. Вы изучили два важных типа данных, INTEGER и REAL, и научились строить и вычислять арифметические выражения с обоими типами данных.
3. Вы знаете арифметические операторы Паскаля, их свойства и порядок старшинства.

   \+ – * / DIV MOD
4. Вы знаете, как объявлять переменные обоих типов, как именовать их, как присваивать им значения (с помощью оператора присваивания (:=) или путем ввода значения с клавиатуры) и как отображать их на экране.
5. Вы научились применять следующие функции преобразования для усечения и округления вещественных выражений:

   TRUNC(*n*) отсекает дробную часть *n*

   ROUND(*n*) округляет *n* до ближайшего целого
6. Вы знаете, как объявлять именованные константы и использовать их в программе.
7. Во время первого путешествия по Паскалю вы изучили следующие операторы вывода для отображения и переменных и численных или строковых констант:

   WRITELN

   WRITE

а также операторы ввода для чтения значений с клавиатуры:

READLN
READ

8. И наконец, вы научились форматировать численный или строковый вывод, получая результат в желаемой форме.

Упражнения

1. В чем различие между литеральной и именованной константами?
2. В чем различие между именованной константой и переменной?
3. Напишите объявления переменных, используя подходящий тип данных, для следующих величин:
   a. Цена автомобиля в долларах и центах.
   b. Площадь поверхности цилиндра.
   c. Количество учащихся в классе.
4. Напишите объявления констант для следующих объектов:
   a. Имя компании (придумайте сами).
   b. Коэффициент преобразования миль в километры.
5. Напишите выражения для подсчета следующих величин:
   a. Площадь круга при известном радиусе.
   b. Общий балл ученика за три класса.
   c. Цена покупки, включая налог, равный 8%.
6. Вычислите следующие выражения:
   a. 10 + 5 * 2 – 6 / 2
   b. (10 + 5) * 2 – 6 / 2
   c. (10 + 5 * 2 – 6) / 2
   d. ((10 + 5) * 2 – 6) / 2
7. Напишите программу на Паскале, выводящую на экран ваше имя, домашний адрес и адрес электронной почты в отдельных строчках.
8. Вычислите следующие выражения:
   a. 1.0/2.0
   b. 1.0/2
   c. 1/2
   d. 1 DIV 2
   e. 1 MOD 2
   f. 10 / 3
   g. ROUND(10/3)

Ответы

6. a. 17.0 b. 27.0 c. 7.0 d. 12.0
8. a. 0.50 b. 0.50 c. 0.50 d. 0 e. 1 f. 3.33 g.3

2

Элементы языка

2–1. Стандартные типы данных и функции

Данные, с которыми работает любая программа, могут состоять из целых или вещественных чисел или из текстовых строк, но каждый тип данных хранится и обрабатывается по-разному. Паскаль поддерживает следующие типы данных (называемые также простыми или *скалярными* типами):

INTEGER
REAL
CHAR
BOOLEAN

Вы уже применяли типы INTEGER и REAL как для констант, так и для переменных. Вы также использовали арифметические операторы с переменными и константами для построения арифметических выражений и познакомились с некоторыми функциями, такими как ROUND и TRUNC. Эта глава представляет полную картину числовых типов данных и соответствующих функций и выражений. Она также вводит тип данных CHAR для представления единичного символа и тип BOOLEN для представления логических значений. Рассмотрение символьного типа включает в себя обзор представления строк в стандартном Паскале, а также их представление в современных реализациях, таких как Turbo Pascal и UCSD Pascal (использующих тип STRING).[1]

[1] Современные реализации, такие как Delphi, поддерживают гораздо больше типов данных. К моменту написания книги Turbo Pascal был самой последней реализацией. – *Примеч. науч. ред.*

2–2. Числовые типы данных

Интервал чисел, которые могут быть представлены как целые (или как вещественные), зависит от реализации. Для типа INTEGER он определяется следующими пределами:

MAXINT	максимальное положительное целое
–(MAXINT+1)	максимальное отрицательное целое

Повторим, значение MAXINT зависит от реализации.

Как правило, для вещественных чисел выделяется большее количество байт, чем для целых, но они имеют ограниченную точность. Дроби 0.333333 и 0.666666, независимо от количества цифр, представляющих число, никогда не будут обладать такой точностью, как точные значения $^1/_3$ и $^2/_3$. По этой причине не рекомендуется проверять два вещественных числа на равенство. Вместо этого лучше сравнивать их разность с некоторым конечным малым значением. В Турбо Паскале существуют дополнительные числовые типы, которые вводятся в следующем разделе.

Числовые типы в Турбо Паскале

В Турбо Паскале есть два дополнительных целых типа (наряду с INTEGER). Они показаны в табл. 2-1 вместе с необходимым для них объемом памяти и пределами значений, которые могут храниться в каждом типе.

В одном байте можно хранить либо SHORTINT, либо BYTE. В действительности BYTE – это беззнаковый SHORTINT, то есть он может хранить только положительные числа. Как видно из таблицы, если знак не используется, то максимальные значения типа удваиваются. Это верно и для INTEGER, и для WORD, поскольку WORD – это положительное целое с удвоенным максимальным значением.

Таблица 2-1. Целые типы Турбо Паскаля

Тип данных	Размер (в байтах)	Интервал значений
SHORTINT	1	от –128 до +127
BYTE	1	от 0 до 255
INTEGER	2	от –32 768 до +32 767
WORD	2	от 0 до 65 535

LONGINT – это наибольшее целое, которое может существовать в Турбо Паскале. Можно проверить его величину, выведя на экран значение константы MAXLONGINT:

```
WRITELN(MAXLONGINT);
```

Заметим, что отрицательный предел любого знакового типа на единицу больше положительного предела (то есть +127 и −128), поскольку ноль считается положительным числом.

Запятые в больших числах используются здесь только для улучшения читаемости. Их никогда не будет в выводе программы, и они не будут восприниматься как часть литеральной константы. Так, число 2,147,483,647 должно быть представлено в виде 2147483647.

Как показано в табл. 2-2, в Турбо Паскале существует также дополнительный вещественный тип (наряду с типом REAL). Для описания точности числа как максимального количества точных цифр в таблицу для вещественных чисел добавлена новая колонка.

Таблица 2-2. Вещественные типы Турбо Паскаля

Тип данных	Размер (в байтах)	Точность (вплоть до)	Интервал значений
SINGLE	4	7 цифр	от 0.71E−45 до 3.4E+38
REAL	6	11 цифр	от 2.94E−39 до 1.7E+38
DOUBLE	8	15 цифр	от 4.94E−324 до 1.79E+308
EXTENDED	10	19 цифр	от 3.3E−4932 до 1.18E+4932
COMP	8	только целые	±9.2E+18

Посмотрев на интервал значений типа SINGLE, вы обнаружите, что он довольно близок к интервалу типа REAL, особенно в области очень больших чисел. Главное отличие состоит в экономичности хранения чисел типа SINGLE (4 байт по сравнению с 6 байт), но достигается это за счет потери точности (7 цифр по сравнению с 11). Без математического сопроцессора вещественные типы, за исключением REAL, недоступны. Тип COMP в действительности принадлежит к множеству целых типов, так как не поддерживает дробных чисел, но обычно упоминается среди вещественных типов, поскольку требует наличия сопроцессора.

2–3. Стандартные арифметические функции

Паскаль включает в себя большое количество предопределенных функций, которые могут быть использованы в выражениях вместе с константами и переменными. В табл. 2-3 приведены стандартные арифметические функции, разделенные на три группы:

- Функции преобразования
- Тригонометрические функции
- Смешанные функции

Любая функция работает с параметром, заключенным в скобки. Параметр – это выражение определенного типа (заметим, что выражением может быть одиночная переменная или константа). Перед тем как использовать функцию, необходимо определить тип параметра и тип возвращаемого функцией значения (который называется также типом функции). Функции преобразования, например, принимают вещественный параметр, а возвращают результат целого типа. Другие функции работают с параметрами целого типа или вещественного, а возвращают значения различных типов. Тип возвращаемого значения важен, когда значение функции присваивается переменной.

Таблица 2-3. Стандартные арифметические функции

Функция	Формат возвращаемого значения	Тип параметра	Тип результата
Функции преобразования			
ROUND(x)	x округляется до ближайшего целого	REAL	INTEGER
TRUNC(x)	Дробная часть x отсекается	REAL	INTEGER
[a]Тригонометрические функции			
ARCTAN(x)	Арктангенс x	REAL/INTEGER	REAL
COS(x)	Косинус x	REAL/INTEGER	REAL
SIN(x)	Синус x	REAL/INTEGER	REAL
Смешанные функции			
ABS(x)	Абсолютное значение x	REAL/INTEGER	REAL/INTEGER
EXP(x)	Экспоненциальная функция x (e^x)	REAL/INTEGER	REAL
LN(x)	Натуральный логарифм x	REAL/INTEGER	REAL
SQR(x)	x в квадрате (x^2)	REAL/INTEGER	REAL/INTEGER
SQRT(x)	Корень квадратный из x (^x)	REAL/INTEGER	REAL

[a.] Все углы должны быть в радианах.

Взгляните на примеры:

```
SQR(3)=9
SQR(2.5)=6.25
SQRT(9)=3.00
ABS (-28.55)=28.55
LN(EXP(1))=1.00
ARCTAN (1)=45 градусов
```

Заметьте, что тип результата, возвращаемого функцией SQR, тот же, что и тип параметра, а функция SQRT возвращает значение вещественного типа независимо от типа параметра. Заметим также, что в ка-