



Язык **Pascal**. Основные команды и функции

Автор
Жулин И.И.

ОПЕРАТОР ПРИСВАИВАНИЯ

Присваивание – это запись в участок памяти компьютера, отведенной для значения величины М, тех данных, которые хранятся в другом участке памяти компьютера, где записано значение величины N.

Обозначение: «:=»

переменная := выражение

Механизм работы: вычисляется значение выражения в правой части от знака «:=», результат которого необходимо занести в память. Адрес ячейки, куда будут заноситься данные, определяет переменная, находящаяся слева от знака «:=», т.е. *переменная ← получаемое выражение*.

Примеры: $A:=b+c/2$; $b:=n$; $x:=15$; $y:=y+3$.

Особенностью оператора присваивания является и то, что данные, находящиеся по разные стороны знака «:=» («присвоить»), должны принадлежать одному типу, но целочисленное выражение может быть присвоено вещественной переменной, поскольку подмножество целых переменных входит в область дробных.

ВВОД И ВЫВОД ДАННЫХ В ЯЗЫКЕ ПАСКАЛЬ

Любая программа при вводе исходной информации и выводе результатов взаимодействует с внешними устройствами. Частным случаем обмена данными с внешними устройствами является обмен с *консолью*.

Консоль представляет собой совокупность клавиатуры и экрана монитора.

Стандартным устройством *ввода* является *клавиатура*, а *вывода* – *монитор*.

Ввод данных – это процесс передачи исходных данных от внешнего устройства (клавиатура или файл с данными) в оперативную память.

Вывод данных – это процесс передачи данных после обработки из оперативной памяти на внешнее устройство (экран, файл, принтер).

ВЫВОД ДАННЫХ

write (**список_вывода**); {после вывода последнего элемента списка на экран, курсор останется в той же строке}

writeln (**список_вывода**); {после завершения вывода переводит курсор на следующую строку, если используется без параметров, то курсор просто переводится на следующую строку}

Список_вывода – различные выражения (символьные, числовые, логические, переменные или константы), отделённые друг от друга запятыми.

Примеры: `s := 5 + 10;`

`write ('summa ', '= ', s);` {результат: «*summa = 15*»}

`writeln;` {просто перевод курсора на следующую строку}

ФОРМАТИРОВАННЫЙ ВЫВОД

При использовании форматированного вывода можно установить **количество позиций** на экране, занимаемых выводимой величиной. Обычно такой тип вывода применяется для вещественных чисел.

write (a : m);

write (a : m : n); {для вещественных чисел}

где «a» – то, что мы собираемся вывести (переменная, константа или другое выражение),
«m» – ширина поля для вывода «a» (общее количество знакомест),
«n» – количество позиций для дробной части «a» (если «a» – вещественное число).

Замечание 1: Двоеточие относится к переменной, после которой оно следует.

Пример, a := 1; b := 4; c := a + b;

write (a, ' + ', b, ' = ', c:3); {результат: «1_+_4_=___5», где «_» - обозначает «пробел»}

Замечание 2: Если для вещественных чисел не осуществлять форматирование, то они отобразятся так, как определено для данного компьютера. Если указать только число знакомест без фиксирования дробной части, то вывод будет в экспоненциальной форме (a = 1.280000E+01).

ПЕРВАЯ ПРОГРАММА НА PASCAL

```
program first_progr;    {заголовок программы}  
  
begin    {начало тела программы}  
  
    write('Hello, World!');    {тело программы}  
  
end.    {конец тела программы}
```

Пример простейшей программы на языке Pascal, тело которой состоит всего лишь из одного оператора «**write**».

Результат работы программы – отображение на экране монитора «Hello, World!».

ВВОД ДАННЫХ

read (список_ввода); {после ввода последнего элемента списка на экран, курсор останется в той же строке}

readln (список_ввода); {после завершения ввода переводит курсор на следующую строку, если используется без параметров, то курсор просто переводится на следующую строку}

Список_ввода – переменные(ая), отделённые друг от друга запятыми.

Замечание: После выполнения оператора **«read»**, компьютер переходит в режим ожидания данных. При вводе данных их разделяют пробелом, табуляцией или переходом на новую строку (Enter). Данные символьного типа не разделяются или разделяются переходом на новую строку. **Типы вводимых значений должны соответствовать типам переменных, указанных в разделе описания переменных.**

Примеры, **read**(a);

read (b, c, d);

readln (m, n);

ПРИМЕР ПРОГРАММЫ НА ВВОД- ВЫВОД ДАННЫХ

```
program summa2;    {заголовок программы}  
  
var    {раздел описания переменных}  
  
    a, b, s: integer;    {объявляем 3 переменные целого типа}  
  
begin    {начало тела программы}  
  
    read (a, b);    {вводим 2 числа: a и b}  
  
    s := a + b;    {сумма чисел a и b}  
  
    writeln ('Сумма чисел a и b равна: s = ', s);    {выводим результат на экран}  
  
    readln    {Ожидание нажатия Enter, после которого программа завершится}  
  
end.    {конец тела программы}
```


ОПИСАНИЕ ДАННЫХ В PASCAL

Общий вид: **var** [список],[переменная]:[тип];

для строковых переменных: **var** <переменная>:**string**[<максимальная длина строки>]

При объявлении, однотипные переменные могут группироваться в список и отделяться друг от друга в этом списке запятыми.

Примеры:

var

r: **real**; {переменная вещественного типа}

n, k: **integer**; {переменные целого типа integer}

i: **byte**; {переменная целого типа byte}

a: **char**; {переменная символьного типа}

s1: **string**[10]; {строковая переменная s1 длиной не более 10 символов}

s2: **string**; {строковая переменная s2 длиной не более 255 символов}

f: **boolean**; {переменная логического типа}

СТАНДАРТНЫЕ МАТЕМАТИЧЕСКИЕ ФУНКЦИИ ЯЗЫКА PASCAL

Функция	Запись на Pascal	Тип аргумента	Тип результата
абсолютное значение аргумента	abs (x)	real, integer	совпадает с типом аргумента
квадрат аргумента	sqr (x)	real, integer	совпадает с типом аргумента
квадратный корень аргумента	sqrt (x)	real, integer	real
косинус аргумента	cos (x)	real, integer	real
синус аргумента	sin (x)	real, integer	real
арктангенс аргумента	arctan (x)	real, integer	real
экспонента аргумента (e^x)	exp (x)	real, integer	real
натуральный логарифм	ln (x)	real, integer	real
целая часть числа	int (x)	real, integer	real
дробная часть числа	frac (x)	real	real
число ПИ (3,141592653)	pi	нет	real
случайное число в интервале [0, 1]	random	нет	real
случайное число в интервале [0, x]	random (x)	integer	integer

ПОРЯДОК ВЫЧИСЛЕНИЙ В ВЫРАЖЕНИЯХ

- 1) вычисляются подвыражения, заключенные в скобки;
- 2) затем выполняются операции с наибольшим приоритетом; обычно используются следующие уровни приоритетов (в порядке убывания):
 - возведение в степень;
 - мультипликативные операции: *****, **/**, **div**, **mod**;
 - унарные операции: **+**, **-**, **abs**, **not**;
 - аддитивные операции: **+**, **-**;
 - операции отношения: **=**, **<>**, **<**, **>**, **<=**, **>=**;
 - логические операции: **and**, **or**, **not**;
- 3) операции с одинаковым приоритетом выполняются слева направо.

Замечание: в Паскале нет стандартной операции или стандартной функции возведения в степень, поэтому используется следующее математическое тождество:

$$x^y = e^{y \ln x}$$

в Паскале существует только стандартная функция вычисления натурального логарифма, поэтому используется следующее математическое тождество:

$$\log_a b = \ln b / \ln a$$

СТАНДАРТНЫЕ МАТЕМАТИЧЕСКИЕ ПРОЦЕДУРЫ

Процедура	Запись на Pascal	Тип аргумента	Тип значения
увеличение "x" на 1 ($x := x + 1;$)	inc (x)	целый	целый
уменьшение "x" на 1 ($x := x - 1;$)	dec (x)	целый	целый
увеличение "x" на n ($x := x + n;$)	inc (x, n)	целый	целый
уменьшение "x" на n ($x := x - n;$)	dec (x, n)	целый	целый

ФУНКЦИИ ПРЕОБРАЗОВАНИЯ ТИПОВ

Функция	Запись на Pascal	Тип аргумента	Тип результата
ASCII код символа "x", порядковый номер аргумента "x"	ord (x)	char, порядковый	byte, integer, longint
округление числа "x" до целого	round (x)	real	integer, longint
целая часть числа "x"	trunc (x)	real	integer, longint
проверяет аргумент на нечетность, результат <i>true</i> , если <i>аргумент</i> нечетный, <i>false</i> – если <i>четный</i>	odd (x)	integer, longint	boolean
символ ASCII кода, соответствующего значению аргумента "x"	chr (x)	byte	char

ПРИМЕР ПРОГРАММЫ (МАТЕМАТИЧЕСКИЕ ОПЕРАЦИИ)

{Вывод на экран суммы и произведения цифр двухзначного числа}

```
program progr_sum_multi;    {заголовок программы}
var ab, b, s, p, a: integer; {раздел описания переменных (все переменные целого типа)}
begin {начало тела программы}
  readln (ab);    {ввод двузначного числа}
  a := ab div 10; {вывод первой цифры числа}
  b := ab mod 10; {вывод второй цифры числа}
  s := a + b;    {сумма цифр числа}
  p := a * b;    {произведение цифр числа}
  writeln (s);   {вывод суммы}
  writeln (p);   {вывод произведения}
end.    {конец тела программы}
```

Замечание: **div** — целая часть от деления, **mod** — остаток от деления.

Пример: $5 \bmod 2 = 1$
 $5 \operatorname{div} 2 = 2$

ПРИМЕР ПРОГРАММЫ (МАТЕМАТИЧЕСКИЕ ОПЕРАЦИИ)

{Поменять местами первую и третью цифры трехзначного числа}

```
program progr_zamena_cifr;    {заголовок программы}  
var abc, a, b, c, cba: integer; {раздел описания переменных (все переменные целого типа)}  
begin {начало тела программы}  
  readln (abc);    {ввод трехзначного числа}  
  a := abc div 100;    {первая цифра числа}  
  b := (abc div 10) mod 10;    {вторая цифра числа}  
  c := abc mod 10;    {третья цифра числа}  
  cba := c * 100 + b * 10 + a;    {единицы * 100 + единицы * 10 + единицы}  
  writeln (cba);    {вывод результата – измененного числа}  
end.    {конец тела программы}
```

ПРИМЕР ПРОГРАММЫ (МАТЕМАТИЧЕСКИЕ ОПЕРАЦИИ)

{Даны две переменных с разными значениями. Поменять местами значения переменных}

```
program progr_zamena;    {заголовок программы}
var a, b, c: integer;    {раздел описания переменных (все переменные целого типа) – вводим
                        {дополнительную переменную для промежуточного хранения данных}
begin  {начало тела программы}
  readln (a, b);        {вводим 2 числа}
  c := a;               {в дополнительную переменную записываем первое число}
  a := b;               {в переменную, где хранилось первое число, записываем вместо него второе число}
  b := c;               {в переменную, где хранилось второе число, записываем вместо него первое число,
                        {которое мы сохранили в дополнительной переменной}
  writeln (a);          {вывод первого числа}
  writeln (b);          {вывод второго числа}
end.  {конец тела программы}
```


ПРИМЕР ПРОГРАММЫ (МАТЕМАТИЧЕСКИЕ ОПЕРАЦИИ)

{Вывести на экран среднее арифметическое цифр трехзначного числа}

```
program progr_sredn_arifm;  {заголовок программы}  
var  {раздел описания переменных}  
    sr: real;  {объявляем переменную вещественного типа}  
    x: integer;  {объявляем переменную целого типа}  
Begin  {начало тела программы}  
readln (x);  {ввод трехзначного числа}  
sr := ((x div 100) + ((x div 10) mod 10) + (x mod 10)) / 3;  {среднее арифметическое цифр числа}  
writeln (sr);  {вывод результата}  
end.  {конец тела программы}
```

СИМВОЛЬНЫЙ И СТРОКОВЫЙ ТИПЫ ДАННЫХ

Символьный тип данных (`char`) – тип данных, значениями которого являются *одионочные символы*. Данный тип может содержать всего один любой символ (это буквы ['A'...'Z', 'a'...'z'], ['A'...'Я', 'a'...'я'], цифры ['0'...'9'], знаки препинания, специальные символы и другие, в т.ч. «*», «/», «.», «!»). Каждый такой символ занимает *8 бит (1 байт)* памяти, всего существует *256 восьмибитовых символов* из таблицы символов *ASCII* (American Standart Code for Information Interchange – Американский стандартный код для обмена информацией).

Символьные константы заключаются в апострофы ('.', '*', '7', 's'). Также символьную константу можно записать с помощью символа – «решетки» (#185 — выведет символ '№', стоящий под номером 185 в таблице ASCII).

Строка в Паскале (`string`) – упорядоченная последовательность символов. Количество символов в строке называется ее *длиной*. *Длина строки* в Паскале может лежать в диапазоне *от 0 до 255*. Каждый символ строковой величины занимает *1 байт* памяти и имеет числовой код в соответствии с таблицей кодов ASCII.

Строковая константа Паскаля – последовательность символов, заключенная в апострофы ('строковая константа', '253').

Два следующих друг за другом апострофа (") обозначают *пустую строку*, т.е. *строку с нулевой длиной*.

ФУНКЦИИ СИМВОЛЬНОГО ТИПА

Функция	Запись на Pascal	Тип аргумента	Тип результата
возвращает символ, соответствующий ASCII-коду числа "x"	chr (x)	byte	char
возвращает число, соответствующее символу "x" в ASCII-таблице	ord (x)	char	byte
преобразует символы из строчных букв в прописные, но распространяется только на литеры латинского алфавита (русские буквы просто игнорируются)	upcase (x)	char	char
возвращает символ, который предшествует в ASCII-таблице символу "x"	pred (x)	char	char
возвращает символ, который следует в ASCII-таблице за символом "x"	succ (x)	char	char

ПРИМЕРЫ С ИСПОЛЬЗОВАНИЕМ ФУНКЦИЙ «ORD» И «CHR»

```
program progr_ord;    {заголовок программы}  
var x: char;         {описание переменных (x - символьный тип)}  
begin               {начало программы}  
    readln (x);      {считывание переменной}  
    writeln (ord (x)); {вывод номера в таблице ASCII}  
end.                {конец программы}
```

```
program progr_chr;   {заголовок программы}  
var x: integer;      {описание переменных (x - целочисленный тип)}  
begin               {начало программы}  
    readln (x);      {считывание переменной}  
    writeln (chr (x)); {вывод символа по номеру в таблице ASCII}  
end.                {конец программы}
```

ПРИМЕРЫ С ИСПОЛЬЗОВАНИЕМ ФУНКЦИЙ «PRED(SUCC)» И «UPCASE»

```
program progr_pred_succ;    {заголовок программы}  
var x: char;    {описание переменных (x - символьный тип)}  
begin    {начало программы}  
    readln (x);    {считывание переменной}  
    writeln (pred (x));    {вывод предыдущего символа в таблице ASCII}  
    writeln (succ (x));    {вывод следующего символа в таблице ASCII}  
end.    {конец программы}
```

```
program progr_upcase;    {заголовок программы}  
var x: char;    {описание переменных (x - символьный тип)}  
begin    {начало программы}  
    readln (x);    {считывание переменной}  
    writeln (upcase (x));    {вывод английской буквы верхнего регистра}  
end.    {конец программы}
```

ОПЕРАЦИИ СО СТРОКАМИ

- 1) **Операция слияния (сцепления, конкатенации)** применяется для соединения нескольких строк в одну, обозначается знаком «+». Операция слияния применима для любых строковых выражений (константы, переменные).

Пример: Выражение: 'Turbo' + ' Pascal ' + '7.0'

Результат: 'Turbo Pascal 7.0'

- 2) **Операции отношения** позволяют сравнивать строки на отношение равенства (=), неравенства (<>), больше (>), меньше (<), больше или равно (>=), меньше или равно (<=). В результате сравнения двух строк получается логическое значение (**true** или **false**). **Сравнение строк** производится **слева направо посимвольно** до первого несовпадающего символа, большей считается та строка, в которой первый несовпадающий символ имеет больший код в таблице кодировки. Если строки имеют различную длину, но в общей части символы совпадают, считается, что короткая строка меньше. **Строки равны**, если они имеют равную длину и соответствующие символы совпадают.

Примеры: 'строка' <> 'строки' (верно, т.к. не совпадают последние символы);

'Аbc' < 'abc' (верно, т.к. код символа 'А' равен 65, а код 'a' – 97);

'год' > 'век' (верно, т.к. буква 'г' в алфавите стоит после буквы 'в');

'Иванов' = 'Иванов И.И.' (неверно, т.к. длина второй строки больше).

СТАНДАРТНЫЕ ФУНКЦИИ ДЛЯ РАБОТЫ СО СТРОКАМИ

- 1) **copy** (**s**, **poz**, **n**) – выделяет из строки **s** подстроку длиной **n** символов, начиная с позиции **poz**. Если **poz** больше длины строки, то результатом будет пустая строка. Здесь **s** – любое строковое выражение, **poz**, **n** – целочисленные выражения.

Пример: **s** := 'строка символов';
 s1 := **copy** (**s**,3,3); {результат: **s1** = 'рок'}

- 2) **concat** (**s1**, **s2**,..., **sn**) – выполняет слияние строк **s1**, **s2**,...,**sn** в одну строку в том порядке, в каком они указаны в списке параметров (функция **concat** выполняет те же действия, что и операция конкатенации).

Пример: **s** := **concat** ('AA', 'XX', 'Y'); {результат: **s** = 'AAXXY'}

СТАНДАРТНЫЕ ФУНКЦИИ ДЛЯ РАБОТЫ СО СТРОКАМИ

3) **length** (**s**) – определяет текущую длину в символах строкового выражения **s**. Результат – значение целого типа.

Пример: `n := length ('123-(a+b)*c');` {результат: n=11}

4) **pos** (**subs**, **s**) – определяет позицию первого вхождения подстроки **subs** в строку **s**. Результат – целое число, равное номеру позиции, где находится первый символ искомой подстроки. Если вхождение подстроки не обнаружено, то результат функции будет равен 0.

Пример: `s := 'Turbo Pascal';`

`n1 := pos ('Pascal', s);` {результат: n1 = 7}

`n2 := pos ('pascal', s);` {результат: n2 = 0 ('pascal' и 'Pascal' – это разные строки)}

СТАНДАРТНЫЕ ПРОЦЕДУРЫ ДЛЯ РАБОТЫ СО СТРОКАМИ

- 1) **delete** (**s**, **poz**, **n**) – удаляет из строки **s**, начиная с позиции **poz**, подстроку из **n** символов. Здесь **s** – строковая переменная (т.к. только с именем переменной связана область памяти, куда будет помещен результат выполнения процедуры); **poz**, **n** – любые целочисленные выражения. Если значение **poz** больше, чем размер строки, ничего не удаляется.

Пример: **s** := 'abcdefg';
 delete (**s**, 2, 3); {результат: **s** = 'aefg'}

- 2) **insert** (**subs**, **s**, **poz**) – вставляет в строку **s**, начиная с позиции **poz**, подстроку **subs**. Здесь **subs** – любое строковое выражение, **s** – строковая переменная (именно ей будет присвоен результат выполнения процедуры), **poz** – целочисленное выражение.

Пример: **s** := 'ade';
 insert ('bc', **s**, 2); {результат: **s** = 'abcde'}

ПРОЦЕДУРЫ ПРЕОБРАЗОВАНИЯ ТИПОВ

- 1) **str** (**x**, **s**) – преобразует число **x** в строковый формат. Здесь **x** – любое числовое выражение, **s** – строковая переменная. В процедуре есть возможность задавать формат числа **x**. Например, **str** (**x**: 8: 3, **s**), где 8 – общее число знаков в числе **x**, а 3 – число знаков после запятой. Если в формате указано недостаточное для вывода количество разрядов, поле вывода расширяется автоматически до нужной длины. Удобно использовать процедуру **str** для вставки числовых данных в какой-либо текст, т. к. *операция конкатенации* и процедура **insert** могут работать только со строковыми данными.

Пример: **str** (3456, s); {результат: s = '3456'}

- 2) **val** (**s**, **x**, **kod**) – преобразует строку символов **s** в число **x**. Здесь **s** – строковое выражение, **x** – числовая переменная (именно туда будет помещен результат), **kod** – целочисленная переменная (типа *integer*), которая равна номеру позиции в строке **s**, начиная с которой произошла ошибка преобразования, если преобразование прошло без ошибок, то переменная **kod** равна 0.

Пример: **val** ('12.34', x, kod); {если x – вещественного типа, результат: x = 12.34, kod = 0}
val ('12.34', x, kod); {если y – целого типа, результат: y = 12, kod = 3}

ЛОГИЧЕСКИЙ ТИП ДАННЫХ

Логический тип – это тот тип, который возвращает переменной либо ответ «да» (правда, истина), либо ответ «нет» (ложь).

В языке программирования *Pascal* такой тип называется **Boolean**, возвращать он может только два значения: **True** (истина) или **False** (ложь).

Значение типа **boolean** занимает в памяти *1 байт*.

При применении логического типа **Boolean** в *Pascal* могут использоваться следующие операции отношения: **<** (меньше), **<=** (меньше или равно), **=** (равно), **>** (больше), **>=** (больше или равно), **<>** (не равно)

Замечание: **False < True**.

Примеры: $5 > 2$ – true

$7 <= 0$ – false

$(5 > 2)$ **and** $(7 <= 0)$ – false

$(5 > 2)$ **or** $(7 <= 0)$ – true

ЛОГИЧЕСКИЙ ТИП ДАННЫХ

Помимо типа **Boolean**, в *Turbo Pascal* версии **7.0** добавлены еще три логических типа данных: **ByteBool**, **WordBool** и **LongBool**.

Название логического типа данных	Значению False соответствует	Значению True соответствует	Размер памяти для логического типа данных
Boolean	Число 0	1	1 байт
ByteBool	Число 0	Любое число, отличное от 0	1 байт
WordBool	Число 0 в обоих байтах		2 байта
LongBool	Число 0 во всех байтах		4 байта

Новые логические типы данных были введены для обеспечения совместимости разрабатываемых программ с *Windows*, в которой значению **False** соответствует **0**, а значению **True** – любое, отличное от **0**, число.

ЛОГИЧЕСКИЕ ОПЕРАЦИИ

x	y	not x	x and y	x or y	x xor y
false	false	true	false	false	false
false	true	true	false	true	true
true	false	false	false	true	true
true	true	false	true	true	false

and – конъюнкция, логическое умножение (И)

or – дизъюнкция, логическое сложение (ИЛИ)

not – инверсия, логическое отрицание (НЕ)

xor – исключающее или

Замечание: Логические операции допустимы только по отношению к *операндам* (константам, переменным, функциям) логического типа (**boolean**).

ПРИОРИТЕТ ЛОГИЧЕСКИХ ОПЕРАЦИЙ

- 1) выражения в скобках
- 2) логическое отрицание (**not**)
- 3) логическое умножение (**and**)
- 4) логическое сложение (**or**), исключающее или (**xor**)
- 5) операции отношения (**<**, **<=**, **=**, **>**, **>=**, **<>**)

К логическим выражениям также можно применять функции **ord**, **succ**, **pred**, процедуры **inc** и **dec**.

ПРИМЕР ПРОГРАММЫ С ИСПОЛЬЗОВАНИЕМ ПЕРЕМЕННЫХ ЛОГИЧЕСКОГО ТИПА

```
program progr_bool;    {заголовок программы}
var    {раздел описания переменных}
  test: boolean;    {объявляем переменную логического типа}
  a, b: integer;    {объявляем 2 переменные целого типа}
begin    {начало тела программы}
  writeln ('Введите a и b');    {выводим «подсказку» на экран}
  readln (a, b);    {вводим 2 числа: a и b}
  test := a > b;    {сравниваем 2 числа и результат записываем в переменную «test»}
  writeln ('a > b – ', test);    {выводим результат на экран}
end.    {конец тела программы}
```

Пусть введены числа: 5 и 16, тогда $test := 5 > 16$; результат на экране: «a > b – False».

ЗАКЛЮЧЕНИЕ

Мы рассмотрели основные команды и функции языка *Pascal* для различных типов данных. *Паскаль* – является одним из самых удобных языков для изучения основ профессионального программирования. Программа на языке *Pascal* (*Паскаль*) имеет блочную структуру.

В языке *Pascal* (*Паскаль*) есть еще один «интересный» оператор: *он не выполняет никакого действия*, это – *пустой оператор* (обозначается знаком ";"). Предусмотрен и оператор останова, который прерывает работу программы (в Паскале это оператор **halt**).

По ходу изложения материала было рассмотрено несколько примеров программ с использованием различных функций и различных типов данных. Каждая строчка кода дополнена комментарием.

Помимо рассмотренных команд и функций, Паскаль имеет множество других конструкций, необходимых для создания программ различной сложности.