Зеленяк О. П.

Практикум программирования на TURBO PASCAL

- Задачи
- Алгоритмы
- Решения

ПРАКТИКУМ

ДМК ПРЕСС издательство

DS торгово-издательский дом DiaSoft

ББК 32.973.2
УДК 681.3. 06(075)
З 58

Зеленяк О.П.
З 58 Практикум программирования на Turbo Pascal. Задачи, алгоритмы и решения.— 3-е изд., испр. и доп. — СПб.: ДиаСофтЮП, М.: ДМК Пресс. — 320 с.

ISBN 5-93772-187-X
ISBN 5-94074-355-2

В книге содержится более 200 задач по программированию в среде Turbo Pascal, снабженных решениями.

В начале каждой главы даётся соответствующий справочный материал. Внутри глав задачи расположены в порядке возрастания трудности. Большое внимание уделено графическим построениям. Последняя глава содержит задачи олимпиад различного уровня по информатике.

Издание предназначено для старшеклассников, студентов, преподавателей.

ББК 32.973.2
УДК 681.3. 06(075)

Рецензенты:

заведующий кафедрой математики Кировоградского государственного педагогического университета, доктор физико-математических наук, профессор Волков Ю.И.

заведующая кафедрой прикладной математики Харьковского государственного политехнического университета, доктор технических наук, профессор Курпа Л.В.

ISBN 5-93772-187-X
ISBN 5-94074-355-2

Оглавление

1. Арифметика вещественных чисел. Вычисления по формулам

♦ Целый тип

В Паскале определены *пять типов (диапазонов значений) для целых чисел*. Областью значений каждого из них является подмножество множества целых чисел.

Тип	Диапазон	Байт
ShortInt – короткое целое со знаком	-128 .. 127	1
Integer – целое со знаком	-32 768 .. 32 767	2
LongInt – длинное целое со знаком	-2 147 483 648 .. ..2 147 483 647	4
Byte – короткое целое без знака	0 .. 255	1
Word – целое без знака	0 .. 65 535	2

В последней колонке таблицы показано, что числу из каждого диапазона отводится фиксированное количество разрядов в памяти ЭВМ. Это позволяет эффективно использовать память и влиять на скорость вычислений. *Все целые типы являются упорядоченными*, поэтому к их значениям применимы две специальные функции Succ(i) и Pred(i), где i – целое, возвращающие следующее и предшествующее i целое число. Определены также две константы MaxInt и MaxLongInt, значениями которых являются числа 32767 и 2147483647, т.е. верхние границы двух соответствующих диапазонов.

Чаще всего целые числа используются в простых арифметических выражениях и выступают в программах в качестве различных счётчиков и значений индексов. Над ними *определены пять основных операций*, результатами которых являются также целые числа:

+	–	сложение;
-	–	вычитание;
*	–	умножение;
div	–	частное от деления;
mod	–	остаток от деления.

Все пять операций – двухместные (бинарные), т.е. применимы к двум аргументам, но операции сложения и вычитания могут использоваться и как одноместные (унарные). При изменении знака числа, например, допустима запись a:= -a.

При делении с точностью до целых получается два результата – частное и остаток. Операция div обозначает целочисленное деление. Знак результата берётся по обычным правилам, а полученный остаток игнорируется. Например, 15 div 2 = 7; 3 div 7 = 0; (-9) div 4 = -2; (-9) div (-4) = 2. Операция mod даёт остаток при делении двух целых чисел. 15 mod 2 = 1; 3 mod 7 = 3; (-9) mod 4 = -1; 9 mod 3 = 0. Операции *, div, mod имеют одинаковый более высокий ранг, а операции + и - также имеют одинаковый, но более низкий ранг.

♦ Вещественный тип

Областью значений вещественного типа является подмножество множества всех вещественных чисел. *Оно не является упорядоченным.* Для представления чисел из широкого диапазона, от очень маленьких до очень больших привычное представление с фиксированной запятой не подходит, т.к. любой вещественный тип в ЭВМ должен иметь ограничение на количество знаков. Если выводятся числа шириной 12 знаков, то, например, число 2 001 000 000 000 555 уже не будет изображено. Поэтому вещественные числа *представляются в ЭВМ в форме с плавающей точкой,* называемой экспоненциальной. Они имеют цифровую часть (мантиссу) и порядок, следующий за знаком e. Приведенное выше число примет вид 2.001e+15. Ясно, что такая форма представления чисел дает возможность выполнять *операции над вещественными числами только приближённо.* Вообще, *нельзя предполагать точное равенство никаких двух чисел с плавающей точкой.* Количество цифр в мантиссе характеризует точность.Чем больше цифр имеет мантисса, тем погрешность меньше.

Погрешность зависит от природы вычислений. Программист должен организовывать вычисления так, чтобы избегать серьёзных ошибок.

Свои особенности имеет использование математического сопроцессора: аппаратные ограничения всегда существуют. Дополнительные трудности возникают также при тестировании программ, содержащих операции над вещественными числами. Следует различать *синтаксические и логические ошибки* и повышать устойчивость программ. Этим важным вопросам следует уделить внимание и для более детального знакомства с ними обратиться к специальной литературе.

Тип	Диапазон	Цифр	Байт
Real – стандартный	2.9e-39 .. 1.7e+38	11-12	6
Single – одинарной точности	1.5e-45 .. 3.4e+38	7-8	4
Double – двойной точности	5.0e-324..1.7e+308	15-16	8
Extended – повышенной точности	3.4e-4932.. 1.1e+4932	19-20	8
Comp – 64-битное целое	-9.2e+18..9.2e+18	19-20	8

Стандартный вещественный тип Real используется без математического сопроцессора, а типы *Single, Double, Extended* относятся к *расширенному набору вещественных типов и рассчитаны на работу с сопроцессором,* применение которого значительно увеличивает точность расчётов и ускоряет их выполнение. Паскаль дает возможность эмулировать работу математического сопроцессора при его отсутствии программным путём с помощью *директив компилятора* {$N+,E+}, которые заключаются в фигурные скобки и имеют отличительный признак $.

Тип *Comp* содержит 64-битные целые числа от $-2^{63}+1$ до $2^{63}-1$, но в вещественном формате. Компилятор пропустит присваивания a:= 5e-01, a:= a + a для переменной a типа Comp и выдаст результат 0 вместо 1, т.к. значение переменной не является целым числом. В выражениях тип совместим со всеми вещественными и целыми типами. Переменная типа Comp может служить вещественным аргументом функции. Применяется тип для выполнения операций с длинными целыми числами. Цепочка операторов a:= 111111; b:= 999999; Write(a*b:0:0); выведет точный целый результат 111110888889, если переменные a и b типа Comp, что не удастся сделать, если они будут иметь тип LongInt (см. пр.28, пр.40).

Новый знак (/) вводится только для операции деления вещественных чисел, причём результат операции всегда вещественный, даже когда оба операнда целые числа.

♦ Математические функции

В паскале имеется системная библиотека математических процедур и функций, которые ещё называют стандартными или встроенными. Применяя их в выражениях или для конструирования новых функций, необходимо следить за типами аргументов и возвращаемых значений. Выполняя действия с любыми числами, следует помнить основные математические факты:

нельзя делить на нуль;

не существует квадратный корень из отрицательного числа;

не существует логарифм числа, которое меньше или равно нуля.

Функция	Тип аргумента	Тип значения	Математическая запись
Odd (x)	целый	логический	2k+1, k -целое
Pi	–	вещественный	π
Abs (x)	цел / вещ	цел / вещ	$\|x\|$
Sqr (x)	цел / вещ	цел / вещ	x^2
Sqrt (x)	цел / вещ	цел / вещ	$\sqrt{x}$
Sin (x)	вещественный	вещественный	sinx
Cos (x)	вещественный	вещественный	cosx
ArcTan (x)	вещественный	вещественный	arctgx
Exp (x)	вещественный	вещественный	e^x
Ln (x)	вещественный	вещественный	lnx
Int (x)	вещественный	вещественный	[x], x≥0
Frac (x)	вещественный	вещественный	{x}, x≥0
Trunc (x)	вещественный	LongInt	
Round (x)	вещественный	LongInt	
Random	–	[0,1) - вещ	
Random(x)	Word	[0,x) - Word	

Операция возведения в степень в паскале отсутствует. Значение степени $a^x = (e^{\ln a})^x$ даёт выражение Exp(x*Ln(a)), использующее две встроенные функции Exp и Ln (3^7 = Exp(7*Ln(3)); $(-3)^7 = -3^7$ = -Exp(7*Ln(3)), т.к. Ln(-3) не существует).

Функции Int(x) и Frac(x) похожи на математические функции [x] и {x}, обозначающие соответственно целую и дробную части числа, но совпадают с ними только на множестве неотрицательных чисел. Если аргумент отрицательный, то они просто отбрасывают дробную или целую часть. [-5,8] = -6, Int (-5.8) = -5; {-5,8}= 0,2, Frac(-5.8) = -0.8 (пр.18).

Функция Trunc(x) отличается от Frac(x) типом возвращаемого значения. Round(x) – функция округления. Odd(x) возвращает логическое значение true, если аргумент x – нечётное число.

Функции Random, Random(x) и сопутствующая им процедура Randomize предназначены для получения псевдослучайных чисел (стр.54).

Целочисленные значения можно присваивать вещественным переменным, а *присваивание вещественного значения целочисленной переменной является ошибкой программирования.* Вещественные значения в последнем случае преобразовывают в целочисленные с помощью одной из предназначенных для этого встроенных функций: Trunc(x) или Round(x). Операции div и mod не могут употребляться с вещественными числами.

Универсальность ЭВМ, её способность к целенаправленной переработке различных видов информации и объясняют происходящий сейчас стремительный рост информационных технологий. Современные компьютеры умеют многое, но по-прежнему их можно использовать как очень мощные калькуляторы для вычисления арифметических и алгебраических выражений. При записи на языке программирования любое выражение должно быть "вытянуто в одну строчку", а *приоритет операций регулируют скобки.* Рассмотрим несколько примеров записи выражений.

Общематематическая запись	Запись на паскале
$-0{,}123 \cdot 10^{-7}$	- 0.123e-7
$456 \cdot 1997\frac{1999}{2003}$	456 * (1997 + 1999/2003)
$2\pi + e^3$	2 * pi + Exp(3)
$\lvert x\rvert + \sqrt{7} + \sqrt[3]{5}$	Abs(x) + Sqrt(7) + Exp(Ln(5)/3)
$y^2 + 8{,}9^{10}$	Sqr(y) + Exp(10 * Ln(8.9))
Sin13 + Sin13°	Sin(13) + Sin(13* pi/180)
$\log_2 7 + \lg 6 + \ln 3$	Ln(7)/Ln(2) + Ln(6)/Ln(10) + Ln(3)

Пример 1. Вычислить: $\dfrac{(7-6{,}35):6{,}5+9{,}9}{(1{,}2:36+1{,}2:0{,}25-1\frac{5}{16}):7\frac{1}{24}}$

Составим программу для решения приведенного арифметического примера, записывая его по правилам языка в одну строку и регулируя скобками порядок выполнения действий:

```
{$N+,E+}
BEGIN
       {   числитель    }   {          знаменатель            }
 Write(((7-6.35)/6.5+9.9) / ((1.2/36+1.2/0.25-21/16)/(7+1/24)));
 ReadLn
END.
```

В громоздких примерах используют переменные. Решение может выглядеть так:

```
{$N+,E+}
Var a,b: Double;
BEGIN
  a:= (7-6.35)/6.5+9.9;                    { числитель }
  b:= (1.2/36+1.2/0.25-21/16)/(7+1/24);    {знаменатель}
  Write (a/b); ReadLn
END.
```

Пример 2. Проверить истинность заданных неравенств или равенств:

1) $\sqrt{1+\sqrt{1+\sqrt{1+\sqrt{1}}}} < \frac{1+\sqrt{5}}{2}$; 2) $\pi^e < e^\pi$; 3) $\log_5 6 > \log_6 7$;

4) $tg^6 10° + tg^6 50° + tg^6 70° = 433$; 5) $\arcsin 10 = 3\pi - 10$;

6) $16\cos\frac{360°}{17} = \sqrt{34-2\sqrt{17}} + \sqrt{17} - 1 + 2\sqrt{17+3\sqrt{17} - \sqrt{170+38\sqrt{17}}}$.

1).

```
{$N+,E+}
BEGIN
 Write(Sqrt(1+Sqrt(1+Sqrt(2))),' < ',(1+Sqrt(5))/2); ReadLn
END.
```

2).

```
{$N+,E+}
BEGIN Write( Exp(Exp(1)*Ln(Pi)),' < ',Exp(Pi) ); ReadLn END.
```

3).

```
{$N+,E+}
BEGIN Write( Ln(6)/Ln(5) > Ln(7)/Ln(6)); ReadLn END.
```

4).

```
{$N+,E+}
Var  a,b,c,x,Left,Mult: Extended;
BEGIN
  Mult:=Pi/18;     {10*Pi/180 - 10 градусов}
  x:=  Mult;  a:=Sqr(Sin(x)/Cos(x));
  x:=5*Mult;  b:=Sqr(Sin(x)/Cos(x));
  x:=7*Mult;  c:=Sqr(Sin(x)/Cos(x));
  Left:= a*Sqr(a)+b*Sqr(b)+c*Sqr(c);
  WriteLn('Левая  часть : ', Left );
  WriteLn('Правая часть : ',' 433'); ReadLn
END.
```

5).

```
{$N+,E+}
Var  x, Left, Right: Extended;
BEGIN
  x:=Sin(10);      {промежуточная величина}
  Left:= ArcTan(x/Sqrt((1-Sqr(x))));
```

```
 WriteLn('Левая часть: ', Left);
 Right:= 3*Pi-10;
 WriteLn ('Правая часть: ', Right); ReadLn
END.
```

6).

```
{$N+,E+}
Var  Left,Right,s: Extended;
BEGIN
  Left:= 16*Cos(2*Pi/17);
  WriteLn('Левая  часть : ' , Left);
  s:= Sqrt(17);                    {промежуточная величина}
  Right:= Sqrt(34-2*s)+s-1 + 2*Sqrt(17+3*s-Sqrt(170+38*s));
  WriteLn ('Правая часть : ' , Right); ReadLn
END.
```

В упражнениях 1 – 4 и 6 применяются встроенные функции Sqrt, Exp, Ln, Sin, Cos, Sqr, а в пятом – соотношение $\arcsin x = \operatorname{arctg}\frac{x}{\sqrt{1-x^2}}$, имеющее место, если $x\in(-1,1)$ и встроенная функция ArcTan. Вычислив левые и правые части равенств, обратим внимание на то, что все 14 знаков после запятой у них совпадают. Это говорит о высокой точности вычислений.

Одно из доказательств неравенства $\log_6 7 < \log_5 6$ использует известное неравенство Коши для двух переменных: $\sqrt{ab} \le \frac{a+b}{2}$. Действительно, $\log_6 7 < 1/\log_6 5$, $\log_6 7 \cdot \log_6 5 < 1$, $\sqrt{\log_6 7 \cdot \log_6 5} < 1$. Последнее неравенство истинно, т.к. $\sqrt{\log_6 7 \cdot \log_6 5} < \frac{\log_6 7 + \log_6 5}{2} = \frac{\log_6 35}{2} < 1$. В программировании *неравенство Коши* применяется при оценке эффективности алгоритмов (пр.188).

Пример 3. (№11(а)) Даны x, y, z. Вычислить a и b, если

$$a = \frac{\sqrt{|x-1|} - \sqrt[3]{|y|}}{1 + \frac{x^2}{2} + \frac{y^2}{4}}, \quad b = x(\operatorname{arctg} z + e^{-(x+3)}).$$

```
{$N+,E+}
Var a,b,x,y,z: Extended;
BEGIN
  Write('Введите три числа x,y и z  ');
  ReadLn(x,y,z);
  a:= (Sqrt(Abs(x-1))-Exp(Ln(Abs(y)/3)))/(1+Sqr(x)/2+Sqr(y)/4);
  b:=  x*(ArcTan(z)+Exp(-x-3));
  WriteLn('a = ',a,';  b = ',b); ReadLn
END.
```

В программе используются встроенные функции Abs, Sqr, Sqrt, ArcTan, Exp и Ln.

Пример 4. Определить нормальный вес человека и индекс массы его тела по формулам: $h \cdot t/240$ и m/h^2, где h – рост человека (измеряемый в см в первой формуле и в метрах – во второй); t – длина окружности грудной клетки (в см); m – вес (в кг). Индекс массы тела принят Всемирной организацией здравоохранения и не должен превышать 25 пунктов.

```
Var h,t,m: Real;
BEGIN
  Write('Введите рост (в см) ');  ReadLn(h);
  Write('Введите длину окружности грудной клетки (в см)  ');
  ReadLn(t);
  WriteLn('Нормальный вес    : ', h*t/240 :3:1,' кг');
  Write  ('Введите вес (в кг)  ');   ReadLn(m);
  WriteLn('Индекс массы тела : ', m/Sqr(h/100):2:1  );
  ReadLn
END.
```

Пример 5. (№7) Смешано v1 литров воды температуры t1 с v2 литрами воды температуры t2. Найти объём и температуру образовавшейся смеси.

```
{$N+,E+}
Var v1,v2,t1,t2,v: Extended;

BEGIN
 Write('Введите последовательно величины v1,t1 и v2,t2 ');
 ReadLn(v1,t1,v2,t2);
 v:=v1+v2;
 WriteLn ('Объем смеси равен ', v);
 WriteLn ('Температура смеси равна ', (t1*v1+t2*v2)/v ); ReadLn
END.
```

Пример 6. (№9) Три сопротивления R1, R2, R3 соединены параллельно. Найти сопротивление соединения.

```
{$N+,E+}
Var R1,R2,R3: Extended;
BEGIN
 Write ('Введите величины трёх сопротивлений ');
 ReadLn(R1,R2,R3);
 WriteLn ('Сопротивление соединения равно ',1/(1/R1+1/R2+1/R3));
END.
```

Пример 7. (№17) Найти площадь кольца, внутренний радиус которого равен 20, а внешний – заданному числу r (r >20).

```
{$N+,E+}
Const r1=20;
Var   r: Extended;

BEGIN
 Write  ('Введите внешний радиус r (r>20)  ');
 ReadLn(r);
 WriteLn('Площадь кольца равна  ',Pi*(r-r1)*(r+r1))
END.
```

В примерах 4–7 значения искомых величин находятся по заданным или известным формулам и без запоминания выводятся на экран оператором вывода. Это приемлемо всякий раз, когда они в дальнейшем не используются в программе.

Пример 8. (№25) Треугольник задан координатами своих вершин. Найти периметр и площадь треугольника.

```
{$N+,E+}
Var x1,x2,x3,y1,y2,y3,a,b,c,p: Extended;

BEGIN
 WriteLn('Введите координаты вершин треугольника ');
 Write('1 - ой   ');  ReadLn(x1,y1);
 Write('2 - ой   ');  ReadLn(x2,y2);
 Write('3 - ей   ');  ReadLn(x3,y3);

 a:= Sqrt(Sqr(x2-x1)+Sqr(y2-y1));
 b:= Sqrt(Sqr(x3-x2)+Sqr(y3-y2));
 c:= Sqrt(Sqr(x3-x1)+Sqr(y3-y1));
 p:=a+b+c;  WriteLn('Периметр: ',p);
 p:=p/2;
 WriteLn('Площадь(1):',Sqrt(p*(p-a)*(p-b)*(p-c)));

 WriteLn('Площадь(2):',0.5*Abs((x3-x2)*(y2-y1)-(y3-y2)*(x2-x1)))
END.
```

В программе последовательно вводятся координаты вершин треугольника и по формуле расстояния между двумя точками, заданными координатами, находятся длины его сторон, периметр и, наконец, площадь по формуле Герона.

Интересно, что можно доказать формулу для вычисления площади треугольника, которая использует только координаты всех его вершин (см. последнюю строку вывода в программе).

Пример 9. Бутылка воды стоит 45 копеек. Пустые бутылки сдаются по 20 копеек, и на полученные деньги опять покупается вода. Какое

наибольшее количество бутылок воды можно купить, имея некоторую сумму денег S копеек?

```
Var s: LongInt;

BEGIN
  Write('Введите сумму денег в копейках  S = ');
  ReadLn(s);
  Write((s - 20) div 25, ' бут. можно купить '); ReadLn
END.
```

Пример 10. Определить номера подъезда и этажа по номеру квартиры девятиэтажного дома, считая, что на каждом этаже ровно 4 квартиры, а нумерация квартир начинается с первого подъезда.

```
Var n,x,y: Word;

BEGIN
 Write('Введите номер квартиры ?',#8);
 ReadLn(n);

 x:= (n-1) div 36 + 1;          { номер подъезда }

 n:=  n-(x-1)*36;               {   n∈[1,36]     }
 y:= (n-1) div 4 + 1;           { номер этажа    }

 Write('Подъезд - ',x, ';    этаж - ',y);  ReadLn
END.
```

Последние два примера иллюстрируют применение в формулах оператора целочисленного деления div.

Примечание: С целью сокращения текстов программ:

1) опускается заголовок;

2) модуль Crt подключается не всегда, хотя имеет смысл подключать этот модуль даже тогда, когда его процедуры и функции не применяются;

3) в приведенных выше линейных программах и практически во всех последующих примерах считается, что данные корректны и проверка правильности ввода не осуществляется;

4) в некоторых строках записаны по нескольку операторов, что не способствует читаемости программ; тексты программ нагляднее, если они содержат по одному оператору в строке и если смысловые блоки программы разделены пустыми строками.

2. Разветвления

♦ Логический тип

Во всех программах предыдущей главы операторы выполнялись друг за другом в порядке их следования. В определённых точках программ этой главы управление (разветвление) будет зависеть от того, истинно или ложно некоторое логическое выражение. *Для хранения логических выражений существует тип данных* **Boolean (булевский)**. Это название происходит от имени математика Джорджа Буля. Поскольку логическое выражение может иметь только два разных значения, то диапазон этого типа состоит из двух значений: ***false (0 – ложно) и true (1 – истинно)***, причём значение false по определению меньше значения true (0<1). Слова false и true – логические константы языка. С помощью трёх основных логических операций **not, and, or** (не, и, или) строят мощные *логические выражения – составные условия.* Их старшинство следующее: not, and, or, а действие показано в *таблице истинности.*

not a	**a**	**b**	a **and** b	a **or** b	a **xor** b
0	**1**	**1**	1	1	0
	1	**0**	0	1	1
1	**0**	**1**	0	1	1
	0	**0**	0	0	0

Условие not a соблюдается, если не соблюдается a, и, наоборот. Условие a and b соблюдается, если соблюдаются вместе a и b; условие a or b соблюдается, если соблюдается хотя бы одно из условий a или b; xor – ещё одна логическая операция (исключающее или), применяемая реже.

Выражение a or b and c аналогично арифметическому выражению a+b*c, поэтому or и and называют ещё логическими сложением и умножением. Операции or и and – бинарные, а операция not – унарная.

При записи условий используются следующие *отношения между величинами*: =, <>, >, <, >=, <=, in (принадлежность элемента множеству). В паскале подвыражение, содержащее отношение, заключают в круглые скобки. Операции сравнения дают результат булевского типа, поэтому его можно присваивать булевской переменной, например, Increase := a>b. Такой оператор присваивания называется *логическим.*

♦ Символьный тип

Символьный (*литерный*) тип **Char** – тип данных, состоящий из одного символа (буквы, знака, кода). Наиболее широко употребляется *набор символов, называемый Американский стандартный код для обмена информацией* (American Standard Code for Information Interchange) и обозначаемый аббревиатурой ASCII. Он содержит две группы символов: изображаемые и управляющие, но каждому символу соответствует свой код.

Взаимно обратные стандартные функции *Chr* и *Ord* возвращают соответственно символ по ASCII-коду и наоборот. Chr(65)= 'A', Ord ('A') = 65.

В паскале символы чаще всего представляются в одинарных кавычках, но возможно и представление символа его кодом с помощью функции Chr или префикса #. Например, 'A' = Chr(65) = #65.

Операций над значениями литерного типа, которые бы давали значение этого же типа, нет. Символы можно сравнивать друг с другом или присваивать. При сравнении считается, что символы равны, если равны их ASCII-коды; и один символ больше другого, если его код больше. Например, 'a' > 'A', потому что 97 > 65. Функции Succ и Pred подходят и для последовательного перебора символов. Каждый символ можно рассматривать как элемент множества Set of Char и, следовательно, применять к нему операцию in.

♦ Строковый тип

Значение строкового типа **String** – последовательность символов. Количество символов в строке от 0 до 255. Строка, как и символ, заключается в одинарные кавычки. *К отдельным символам строки, совместимыми по типу со значением Char, можно обращаться по номеру (индексу)* данного символа в строке. Для определения данных строкового типа используется идентификатор String, за которым следует заключенное в квадратные скобки значение максимально допустимой длины строки данного типа. Если это значение не указывается, то длина строки по умолчанию считается равной 255 байт, а при попытке записи в переменную строку больше символов, чем объявлено в описании, «лишняя» часть отсекается. Строковые данные используются в программах и в качестве констант. Перед использованием строки необходимо инициализировать пустыми или определёнными значениями. Пустая строка обозначается двумя одинарными кавычками подряд.

Над строковыми данными допустимы *операции сцепления и отношения*. Операция сцепления (+) применяется для сцепления нескольких

строк в одну и имеет более высокий приоритет, чем операции отношения. Результат выполнения операций отношения всегда имеет булевский тип. Для обработки строковых данных в паскале существует целая библиотека стандартных процедур и функций, которые будут рассмотрены в главе 8.

Пример 11. (№34) Даны три действительных числа x, y, z. Получить Max(x,y,z).

```
Var  x,y,z,max: Real;

BEGIN
  Write('Введите три действительные числа  ');
  ReadLn(x,y,z);

  if x>y then max:=x else max:=y;  {большее из двух чисел}
  if z>max then max:=z;

  WriteLn('Max = ', max); ReadLn
END.
```

После ввода трёх чисел поиск наибольшего осуществим с помощью *последовательных ветвлений в полной и краткой форме*. Сначала определим большее из двух чисел, после чего результат, значение переменной max, сравним с оставшимся третьим числом. В операторе вывода WriteLn 'Max = ' – это строковая константа, а max – искомая величина.

Если значения переменных x,y,z не надо сохранять, то большее из трёх чисел можно определить и без использования промежуточной переменной max, например, так: ... if x<y then x:=y; if x<z then x:=z; ... В этом случае результатом будет значение переменной x, которое, вообще говоря, не совпадает с начальным значением этой переменной. Очевидно, что для нахождения меньшего из трёх чисел достаточно изменить только знаки неравенств на противоположные.

Пример 12. В каждый подарочный набор входят 1 ручка, 2 линейки и 4 тетради. Имеется a линеек, b тетрадей, c ручек. Сколько всего получится подарочных наборов?

```
Var a,b,c: LongInt;

BEGIN
  Write('Введите количества линеек, тетрадей и ручек ');
  ReadLn(a,b,c);

  a:= a div 2;
  b:= b div 4;
  if a>b then a:=b;
```

```
  if a>c then a:=c;

  Write ('Всего наборов: ', a); ReadLn;
END.
```

Выполнив целочисленные деления a:= a div 2 и b:= b div 4, определим сколько наборов может получиться из имеющихся количеств линеек и тетрадей. После этого искомое число наборов найдём как наименьшее из трёх чисел a, b и c.

Пример 13. (№35) Даны три действительных числа x, y, z. Вычислить Max(x+y+z, xyz).

```
{$N+,E+}
Var  x,y,z,s,p: Double;

BEGIN
 Write('Введите три действительные числа ');
 ReadLn(x,y,z);
 s:= x+y+z;
 p:= x*y*z;

 if s>p then
        Write('сумма больше: ',s,' >',p )
        else
        if s<p then
               Write('произведение больше: ',p,' >',s)
               else
               Write('сумма и произведение равны: ',s,' =',p)
END.
```

В этом примере на самом деле три различных варианта, потому что кроме неравенств возможно равенство суммы и произведения трёх чисел. Например, 1 + 2 + 3 = 1 · 2 · 3. Используем промежуточные переменные s, p и *вложенные ветвления*: альтернативная ветвь содержит ветвление в свою очередь.

Пример 14. (№47) Даны действительные положительные числа x, y, z. Выяснить, существует ли треугольник с длинами сторон x, y, z и, если треугольник существует, то определить его вид по углам.

```
Var  x,y,z,t,max: Real;

BEGIN
   Write('Введите длины сторон треугольника  ');
   ReadLn(x,y,z);
   if x>y then max:=x else max:=y;
   if z>max then max:=z;
```

```
   Write('Треугольник ');
   if 2*max<x+y+z then
                  begin
                  t:= sqr(x)+sqr(y)+sqr(z)-2*sqr(max);
                  if t>0 then Write('остроугольный!');
                  if t=0 then Write('прямоугольный!');
                  if t<0 then Write( 'тупоугольный!')
                  end
                  else
                  Write('не существует! ',#7);
END.
```

Треугольник существует, если его наибольшая сторона меньше суммы двух других сторон. После ввода длин сторон большую из них определим как в пр.11. Но значение какой переменной x, y или z будет хранить промежуточная переменная max, и с суммой какой пары сторон её сравнивать? В этом случае условие существования треугольника запишем так: max+max < x+y+z. Действительно, если, например, y – наибольшая сторона, то имеем: max < x+z.

Аналогичное выражение $t = x^2+y^2+z^2-2max^2$ можно рассмотреть и для записи условий, определяющих вид треугольника по углам. Пусть y – наибольшая сторона треугольника. Тогда $t = x^2+z^2-max^2$, а по следствию из теоремы косинусов $\cos Y=\frac{x^2+z^2-y^2}{2xz}$. Знак числителя в правой части этого равенства совпадает со знаком косинуса наибольшего угла треугольника. Следовательно, если числитель: а) положителен, то треугольник остроугольный; б) равен нулю, то треугольник прямоугольный; с) отрицателен, то треугольник тупоугольный.

Обратите внимание, на необходимость использования в программе *составного оператора* – последовательности операторов, заключённых в операторные скобки, т.е. зарезервированные слова *begin ... end*. Паскаль допускает произвольную глубину вложенности таких операторов.

Рассмотрим фрагмент еще одного варианта решения этой задачи:

```
                  . . .
if x<y then begin r:=x; x:=y; y:=r end;
if x<z then begin r:=x; x:=z; z:=r end;
Write('Треугольник ');
if x<y+z then
         begin
         t:= sqr(y)+sqr(z)-sqr(x);
         if t>0 then Write('остроугольный!');
                  . . .
```

Решение основано на *попарном обмене значений переменных*. Такая операция применяется часто, поэтому рассмотрим её подробнее. С помощью трёх команд присваивания **r:=x; x:=y; y:=r** меняются местами значения переменных x и y. Действительно, промежуточная

переменная r предварительно запоминает “старое” значение x, переменная x получает своё “новое” значение y, и, наконец, переменной y присваивается значение x. В фрагменте после двух серий обменов переменной x присваивается значение наибольшей стороны треугольника, а переменным y и z значения двух других его сторон. Запись необходимых для решения условий упрощается.

Пример 15. (№57) Дано действительное число x. Вычислить f(x), если

$$f(x) = \begin{cases} 0 & \text{при } x \le 0, \\ x^2 - x & \text{при } 0 < x \le 1, \\ x^2 - \mathrm{Sin}(\pi x^2) & \text{при } x > 1. \end{cases}$$

```
{$N+,E+}
Var x,y: Double;

BEGIN
 Write('Введите любое действительное число   x = ?',#8);
 ReadLn(x);
 if x<=0 then
         y:=0
         else
         if x>1 then y:= Sqr(x)-Sin(Pi* Sqr(x))
                else y:= x*(x-1);
 WriteLn('f(',x,' ) = ',y);  ReadLn
END.
```

Нахождение значений кусочно заданных функций, т.е. функций заданных различными аналитическими выражениями при различных значениях аргумента, – это типичный выбор, при котором, как правило, количество формул совпадает с количеством вариантов выбора. Используем вложенные ветвления. При соблюдении условия $x \le 0$ переменной y будет присвоено значение 0 и выбор закончится. В противном случае, необходимо выбрать один из двух оставшихся вариантов: $0 < x \le 1$, $x > 1$. Вывод осуществим в виде f(1) = 0, где 1 и 0 значения заданной и искомой величин x и y, а 'f(' и ') = ' – строковые константы.

Пример 16. Преобразование Цагира.

Тройке натуральных чисел (x, y, z) сопоставить новую тройку натуральных чисел (x_1, y_1, z_1) по следующему правилу:

x_1= x+2z,	y_1= z,	z_1= y-x-z,	если x < y-z
x_1= 2y-x,	y_1= y,	z_1= x-y+z,	если y-z ≤ x ≤2y
x_1= x-2y,	y_1= x-y+z,	z_1= y	в остальных случаях

В журнале “Квант” (№10-91), в статье “Теорема Ферма-Эйлера о двух квадратах” приведены три доказательства замечательной теоремы о представлении простого числа в виде суммы двух квадратов. Одно из доказательств принадлежит современному математику Д.Цагиру. “Оно совершенно потрясло меня: это какое-то чудо, когда результат получается как бы из ничего”, – пишет автор. Заинтересовавшиеся доказательством, могут найти его в указанной статье, а мы рассмотрим программу-иллюстрацию преобразования как ещё один пример реализации выбора.

```
Var  x,y,z,x1,y1,z1: LongInt;
                 s: String;

BEGIN
 Write('Введите три натуральных числа  '); ReadLn(x,y,z);

 if x<y-z then
          begin x1:=x+2*z; y1:=z; z1:=y-x-z end
          else
          if x<=2*y then
                    begin x1:=2*y-x; y1:=y; z1:=x-y+z end
                    else
                    begin x1:=x-2*y; y1:=x-y+z; z1:=y end;

  WriteLn(x:10,y:10,z:10,'-->':10,x1:10,y1:10,z1:10);
  WriteLn(x*x+4*y*z:36,' = ', x1*x1+4*y1*z1);
  FillChar(s,80,#205); WriteLn(s); ReadLn
END.
```

Исполняя программу, обратите внимание на два интересных свойства линейного преобразования Цагира: 1) *сохранение формы* x^2+4yz ($x^2+4yz = x_1^2+4y_1z_1$); 2) *инволютивность* (дважды примененное оно возвращает нас назад). Например, из тройки (1,11,111) получается тройка (21,11,101) и наоборот, причем $1^2 + 4\cdot11\cdot111 = 21^2 + 4\cdot11\cdot101$.

Пример 17. Главная государственная налоговая инспекция Украины определила следующую шкалу ставок месячного подоходного налога с доходов граждан: до 17 гривен – не облагается; более 17 гривен - 85 гривен – 10% суммы дохода, превышающего размер одного не облагаемого налогом минимума; более 85 гривен - 170 гривен – 6 гривен 80 коп. плюс 15% от суммы, превышающей 85 гривен; более 170 гривен - 1020 гривен – 19 гривен 55 коп. плюс 20% от суммы, превышающей 170 гривен; *более* 1020 гривен - 1700 гривен – 189 гривен 55 коп. плюс 30% от суммы, превышающей 1020 гривен; более 1700 гривен – 393 гривни 55 коп. плюс 40% от суммы, превышающей 1700 гривен.

Требуется определить величину месячного подоходного налога по заданной сумме месячного дохода.

```
Var  x,tax: Real;

BEGIN
 Write('Введите сумму месячного дохода в формате грн.коп  ?',#8);
 ReadLn(x); WriteLn;

 if x<=17 then
          Write('Не облагается ':120)
          else
          begin
          if x<=85 then tax:=(x-17)*0.1 else
           if x<=170 then tax:=(x-85)*0.15+6.80 else
            if x<=1020 then tax:=(x-170)*0.2+19.55 else
             if x<=1700 then tax:=(x-1020)*0.3+189.55 else
                             tax:=(x-1700)*0.4+393.55;
          WriteLn('Сумма налога     : ':35, tax   :6:2,' грн');
          Write  ('Заработная плата : ':115,x-tax :6:2,' грн')
          end
END.
```

Используем полную форму условного оператора. В одной его ветви отделим вариант, когда введенная сумма x не облагается налогом, а в другой - выполним расчёт искомой суммы tax месячного налога по одной из формул указанной шкалы и выведем результат. Один из пяти промежутков шкалы выберем с помощью вложенных ветвлений. По правилам паскаля считается, что *каждый символ else соответствует первому предшествующему ему символу then*.

Пример 18. (№61) Дано действительное число x. Получить целую и дробную части числа x.

```
{$N+,E+}
Var x,y: Double;

BEGIN
 Write('Введите любое действительное число  x = ?',#8);
 ReadLn(x);
 if (x>0) or (Frac(x)=0) then y:=Int(x) else y:=Int(x)-1;
 WriteLn ('[', x ,'] = ', y    :1:0 );    { целая часть }
 WriteLn ('{', x ,'} = ', x-y  :1:15);    {дробная часть}
END.
```

Целая часть числа x – это наибольшее целое, не превосходящее x; так [5]=5; [-5]=-5; [5,8]=5,8; [-5,8]=-6. *Дробная часть числа* – это разность между самим числом и его целой частью; так {5}=5-5=0, {5,8}=5,8-5=0,8; {-5,8}= -5,8-(-6)= 0,2. Если введенное число положительное или целое, то значение искомой величины совпадает со значением встроенной функции Int, в противном случае определим его по формуле Int(x)-1. Имеем *логическое выражение (составное условие), состоящее из двух отношений*

x>0, Frac(x)=0. Оно истинно, если истинно хотя бы одно из входящих в него отношений: неравенство или равенство, поэтому применим логическую операцию or. Простые условия заключим в круглые скобки.

Пример 19. (№71) Дано действительное число a. Вычислить f (a), где функция f – это периодическая функция с периодом 1,5, совпадающая на отрезке [0;1,5] с функцией x^3-2,25x.

```
{$N+,E+}
Var a,x: Double;

BEGIN
 Write('Введите любое действительное число  a = ?',#8);
 ReadLn(a);

 x:= a-Int(a/1.5)*1.5;
 if  x<0 then x:=x+1.5;

 Write('f(',a,' ) = ', x*(x*x-2.25)); ReadLn
END.
```

Из условия примера и определения периодической функции следует, что для вычисления f(a) необходимо вычислить $f(x) = x^3 - 2{,}25x$, где x = = a-1,5n и $x \in [0;1{,}5]$, а множитель n равен математической целой части частного a/1.5 (см. пример18). Можно воспользоваться и стандартной функцией Int, корректируя полученный результат: if x<0 then x:= x+1.5.

Пример 20. (№76) Поле шахматной доски определяется парой натуральных чисел, каждое из которых не превосходит восьми: первое число – номер вертикали (при счёте слева направо), второе – номер горизонтали (при счёте снизу вверх). Даны натуральные числа k, l, m, n, каждое из которых не превосходит восьми. Требуется :

а) Выяснить, являются ли поля (k,l) и (m,n) одного цвета.

б) На поле (k,l) расположен ферзь. Угрожает ли он полю (m,n)?

в) Аналогично б), но ферзь заменяется на коня.

```
{$R+}
Var  k,l,m,n,h,v: 0..8;
     z: String[10];

BEGIN
  Write('Введите координаты первого поля  ');  ReadLn(k,l);
  Write('Введите координаты второго поля  ');  ReadLn(m,n);
  if Odd(k+l)=Odd(m+n) then z:='одного' else z:='различного';
  WriteLn('Поля ',z,' цвета');
```

```
 h:= Abs(k-m);  v:= Abs(l-n);
 if (h=v) or (h*v=0) then z:='' else z:='не';
 WriteLn('Ферзь ',z, ' угрожает');

 if ((h=1)and(v=2)) or ((h=2)and(v=1)) then z:='' else z:='не';

 WriteLn('Конь ',z, ' угрожает');  ReadLn
END.
```

Известно, что поля (k,l) и (m,n) одного цвета, если числа k+l и m+n одинаковой четности. Следовательно, равенство двух значений логической функции Odd с аргументами k+l и m+n гарантирует выполнение этого требования. Два других условия легко вытекают из соответствующих шахматных правил. Для их записи вспомогательным переменным h и v присвоим модули разностей номеров двух полей шахматной доски по горизонтали и по вертикали. Тогда в составном условии (h=v) or (h*v=0) первое отношение обозначает, что поля на одной диагонали, а второе – на одной вертикали или горизонтали. Условие ((h=1) and (v=2)) or ((h=2) and (v=1)) исчерпывает все возможные варианты ходов коня.

Заметим, что все используемые в программе числовые величины принимают целые значения от 0 до 8, т.е. их значения пранадлежат ограниченному типу, диапазону. *Диапазон должен быть подмножеством одного из базовых перечислимых типов.* За переменными, объявленными подобным образом, усиливается контроль при выполнении программы, что позволяет избегать ошибок. Директива {$R+} включает режим проверки границ диапазонов. Она замедляет работу программы и увеличивает её размер, поэтому обычно используется лишь при отладке. В случае нарушения границ диапазона выводится сообщение: *Range check error*.

Пример 21. (№46) Даны действительные числа x и y. Если x и y отрицательны, то каждое значение заменить его модулем; если отрицательно только одно из них, то оба значения увеличить на 0,5; если оба значения неотрицательны и ни одно из них не принадлежит отрезку [0,5;2], то оба значения уменьшить в 10 раз; в остальных случаях x и y оставить без изменения.

```
{$N+,E+}
Var x,y: Double;
BEGIN
 Write ('Введите два действительных числа   '); ReadLn(x,y);
 if (x<0)and(y<0) then
                  begin x:=-x; y:=-y end
                  else
                  if (x<0)or(y<0) then
                                  begin x:=x+0.5; y:=y+0.5 end
```

```
                              else
                              if Not((x>=0.5)and(x<=2)) and
                                 Not((y>=0.5)and(y<=2)) then
                                 begin x:=x/10;y:=y/10 end;

 WriteLn('После замены:  x=',x,';  y=',y); ReadLn
END.
```

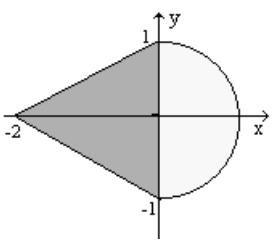
В приведенном решении все отношения логических выражений записаны по тексту условия примера. Рассмотрим одну из частей последнего составного условия: Not ((x>=0.5) and (x<=2)), содержащую отрицание логического умножения. Существуют два полезных правила, которые носят название *законов де Моргана* и позволяют упрощать подобные выражения. В нотации паскаля их можно выразить так:

Not (a and b) = Not a or Not b;
Not (a or b) = Not a and Not b.

Применяя первое из этих правил, получим, что условие Not ((x>=0.5) and (x<=2)) равносильно условию (x<0,5) or (x>2), которое значительно проще. Аналогично поступим со второй частью Not((y>=0.5)and (y<=2)), после чего последнее ветвление может выглядеть так:
if ((x<0,5) or (x>2)) and ((y<0,5) or (y>2)) then begin x:=x/10; y:=y/10; end;

Пример 22. (№59)

Даны действительные числа x и y. Определить, принадлежит ли точка с координатами (x, y) выделенной на рисунке части плоскости.




```
{$N+,E+}
Var Left,Right: Boolean;
    x,y       : Double;
    Z         : String[2];

BEGIN
 Write('Введите координаты точки '); ReadLn(x,y);

 Left := (x<=0) and (y>=-0.5*x-1) and (y<=0.5*x+1);
 Right:= (x> 0) and (Sqr(x)+Sqr(y)<=1);

 if Left or Right then z:='' else z:='не';
 Write(Left,'Точка ',z,' принадлежит фигуре',Right); ReadLn
END.
```