Государственное бюджетное общеобразовательное учреждение

Средняя общеобразовательная школа №603

Фрунзенского района

Программирование на языке Pascal

Учебно-методическое пособие

Автор: учитель информатики Саблина А.А.

Санкт-Петербург

2013

В учебно-методическом пособии, адресованном прежде всего учащимся школы, рассказывается об основах программирования на языке Паскаль. Теоретический материал излагается так, что его последовательное изучение позволит школьникам освоить данный язык программирования без посторонней помощи. Пособие содержит множество разобранных примеров и задач для самостоятельного изучения. В качестве среды программирования автор предлагает использовать программы Turbo Pascal или PascalABC.

Пособие может быть полезно и учителям информатики, которые используют Паскаль в качестве преподаваемого языка программирования.

СОДЕРЖАНИЕ

Структура программы на языке Pascal. Идентификаторы

Pascal – язык программирования структурного типа, т.е. любая программа, написанная на языке Pascal, имеет определенную *структуру*:

```
Program <имя программы>;
Label <раздел описания меток>;
Const <раздел описания констант>;
Type <раздел описания типов>;
Var <раздел описания переменных>;
Procedure (Function) <раздел описания подпрограмм>;
Begin
  <раздел операторов>
End.
```

Например,

1. Программа выводит на экран слово ***привет***

```
Program prog1;
Begin
  Write('привет')
End.
```

2.

3. Программа увеличивает введенное целое число на 1

```
Program prog2;
Var x:integer;
Begin
  Read(x);
  x:=x+1;
  Write(x)
End.
```

4.

5. Программа не выполняет никаких действий

```
Program prog3;
Begin
End.
```

Все разделы программы отделяются друг от друга знаком «**;**».

Begin и **End** – «***операторные скобки***». Они используются для обозначения начала и конца группы команд, но сами командами не являются. Поэтому точка с запятой после **Begin** и перед **End** не ставится.

Имя программы, а также имена переменных, констант и других используемых в программе объектов называются ***идентификаторами***.

- *Идентификаторы задает сам программист*
- *В качестве идентификатора может использоваться любая* ***последовательность латинских букв, цифр и знака подчеркивания, начинающаяся с буквы***
- *Имена идентификаторов не могут совпадать*
- *Строчные и прописные буквы не различаются*

Определите могут ли быть идентификаторами следующие последовательности символов:

```
1) prog 1  2) prog_1  3) prog1   4) Prog    5) прог1
6) 1_prog  7) prog~1  8) PROG1   9) PrOg    10) p
```

Описание констант и переменных

Чтобы использовать в программе какие-либо переменные или константы, нужно сначала описать их в соответствующем разделе.

Константа – величина, не изменяющая своего значения.
Для описания константы нужно указать ее **имя** и **значение:**

<имя>=<значение>

Например,

```
Const a=3; b=1.145;
```

{*В десятичных дробях ставится не запятая, а точка!*}

Переменная – величина, значение которой может изменяться в процессе выполнения программы. Поэтому заранее указать значение переменной (как для константы) нельзя. Для переменной указывают, какие значения она может принимать (число, целое число, слово, символ и т.д.), т.е. ее тип.
Т.о. для описания переменной нужно указать ее **имя** и **тип:**

<имя>:<тип>

Существуют *стандартные* и *пользовательские* (создаваемые самим программистом) типы.
Некоторые стандартные типы:

Integer – целые числа (-10; 0; 1; 2; …)
Real – вещественные числа (-10; -7,241; 0; 1; 4,25; …)
Boolean – логический тип (**true** и **false**)
Char – символьный тип ('a'; 'б'; '1'; '*' …)
String – строковый тип ('a'; 'б'; '1'; '123'; 'абв'; '*' …)

Например,

```
Var x:integer; Y:char; a:integer; b:integer;
```

Переменные одного типа можно описывать вместе:

```
Var x,a,b:integer; Y:char;
```

1. Каждому из типов поставить в соответствие список номеров всех значений, которые могут быть значениями переменной данного типа:

(1)	1,5	Integer:
(2)	'a'	Real:
(3)	128	Boolean:
(4)	-7	Char:
(5)	true	String:
(6)	'4'	
(7)	'абвгд'	

2. Описать переменные, необходимые для
   a) вычисления значения функции $y=x^2$
   b) пересчета веса из фунтов в килограммы
   c) вычисления площади круга
   d) вычисления стоимости покупки, состоящей из нескольких тетрадей и такого же количества обложек

Операторы ввода и вывода

Вывод данных на экран осуществляется с помощью оператора ***Write*** (или ***Writeln***):

```
Write(<список вывода>)
Writeln(<список вывода>)
```

Элементами списка могут быть константы, переменные, выражения. Указанные данные выводятся на экран в порядке их перечисления в списке. Для вывода на экран определенного текста он указывается *в кавычках*. При использовании оператора ***Writeln*** после вывода на экран осуществляется переход на следующую строку, при использовании оператора ***Write*** перехода на следующую строку не происходит.

Например,

1. **`Write(a)`** – вывод на экран значения переменной (или константы) a;
2. **`Write('a')`** – вывод на экран буквы **a**;
3. **`Write('a', b, c)`** – вывод на экран буквы **a** и значений переменных (или констант) **b** и **c**;
4. **`Write('a,', b, ',', c)`** – вывод на экран буквы **a** и значений **b** и **c** через запятую;
5. **`Write('a');`** – вывод на экран букв **a** и **b** (в строку)
   `Write('b')`
6. **`Writeln('a');`** – вывод на экран букв **a** и **b** (в столбик)
   `Write('b')`

Значения переменных типа ***Real*** выводятся на экран в ***нормализованном*** виде.
Например, значение ***13,5*** будет выведено как ***1.3500000000E+01***, а число ***7340,1*** – как ***7.3401000000E+03***.
Для того, чтобы задать, в каком виде вывести число на экран, используют **форматный вывод**: **`a:M:N`**, где **M** – общее количество знаков, отводимое под число при выводе на экран, а **N** – количество знаков после запятой.
Например, если **X** – переменная типа ***Real***, то команда **`Write(x:10:3)`** обозначает вывод на экран значения переменной **X** с 3 знаками после запятой (если ***x=13,5***, будет выведено ***13.500***).

Ввод данных с клавиатуры осуществляется с помощью оператора ***Read*** (или ***Readln***):

```
Read(<список переменных>)
Readln(<список переменных>)
```

При выполнении этого оператора компьютер ожидает ввода с клавиатуры значений переменных в том порядке, в каком они указаны в списке. Вводить значения следует через *пробел*, после окончания ввода нажимают клавишу Enter. При использовании оператора *Readln* после ввода значений осуществляется переход на следующую строку, при использовании оператора *Read* перехода на следующую строку не происходит.
С клавиатуры можно вводить только значения переменных.

Например,

1. **`Read(a)`** – ввод значения переменной **a**;
2. ~~**`Read('a')`**~~ – неверная команда, т. к. вводить можно только значения переменных;
3. **`Read(a,b,c)`** – ввод значений переменных **a**, **b**, **c**;

Пример.
Программа выводит на экран введенное число с одним знаком после запятой

```
Program chislo;
Var N:Real;
        Begin
  Write('Введите число: ');
  Readln(N);
  Writeln('N=', N:5:1)
End.
```

Вид экрана после выполнения программы, будет следующим:

```
Введите число: ...
N=...
```

Если введут число 7, вид экрана после выполнения программы, будет следующим:

```
Введите число: 7
N=  7.0
```

1. Написать программу, которая выводила бы на экран строку ***Это программа на языке Pascal***
2. Написать программу, которая выводила бы на экран четверостишье

   Лес, точно терем расписной,
   Лиловый, золотой, багряный,
   Веселой, пестрою стеной
   Стоит над светлою поляной. **(Бунин, «Листопад»)**

3. Написать программу, которая выводила бы на экран введенное целое число
4. Написать программу, которая выводила бы на экран введенное целое число 3 раза в строчку
5. Написать программу, которая выводила бы на экран введенное целое число 3 раза в столбик
6. Написать программу, которая выводила бы на экран введенное число с двумя знаками после запятой

Оператор присваивания

Значения переменных можно изменять в процессе выполнения программы с помощью оператора присваивания. Команда присваивания имеет следующий вид:

<переменная>:=<выражение>

Оператор присваивания выполняется следующим образом:

1. вычисляется значение выражения
2. переменная получает это значение

При этом ***тип выражения*** *(тип выражения определяется типом его значения)* ***должен быть совместим с типом переменной***. Например, числовой переменной нельзя присвоить в качестве значения символ или строку символов и наоборот; переменной символьного типа ***Char*** нельзя присвоить в качестве значения строку символов, а переменной типа ***Integer*** – значение, тип которого ***Real*** и т.д. Но при этом обратное возможно: переменной типа ***String*** можно присвоить значение типа ***Char***, а переменной типа ***Real*** – значение типа ***Integer***.

Выражение может включать в себя **константы**, **переменные**, **знаки операций**, **функции**, **скобки**.

Некоторые операции:

+	–	сложение	(например, значением выражения **7+4** будет **11**)
–	–	вычитание	(например, значением выражения **7–4** будет **3**)
*****	–	умножение	(например, значением выражения **7*4** будет **28**)
/	–	деление	(например, значением выражения **7/4** будет **1.75**)
div	–	деление нацело	(например, значением выражения **7div4** будет **1**)
mod	–	остаток от деления нацело	(например, значением выражения **7mod4** будет **3**)

Некоторые функции:

Abs(x)	–	модуль x ($\|x\|$)
Cos(x)	–	косинус x
Sin(x)	–	синус x
Sqr(x)	–	квадрат x (x^2)
Sqrt(x)	–	корень из x ($\sqrt{x}$)
Random	–	случайное число из отрезка [0, 1]
Random(x)	–	случайное число из отрезка [0, x]

Аргументы функций записываются в скобках!

Например,

1. **a:=14.2** – переменной ***a*** присваивается значение ***14,2***
2. **k:='слово'** – переменной ***k*** присваивается значение ***'слово'***
3. **x:=x–4** – значение переменной ***x*** уменьшается на ***4***
4. **c:=a+b** – переменной ***c*** присваивается сумма значений переменных (или констант) ***a*** и ***b***
5. **c:=sin(sqr(a+b))** – переменной ***c*** присваивается значение синуса от квадрата суммы переменных (или констант) ***a*** и ***b***
6. **x1:=(–b+sqrt(sqr(b)–4*a*c))/(2*a)**
   x2:=(–b–sqrt(sqr(b)–4*a*c))/(2*a)
   – значения переменных ***x1*** и ***x2*** вычисляются по формуле корней квадратного уравнения

1. Присвоить переменной ***P*** значение ***1,75***
2. Увеличить на единицу значение переменной ***t***
3. Поменять знак переменной ***Summa***
4. Записать в виде команды присваивания

а) $Z = \sqrt{|x| + |y|}$

б) $A = \dfrac{\sin x + \sin y}{\cos x - \cos y}$

в) $Y = \dfrac{1 + \sin\sqrt{x+1}}{\cos(12x - 4)}$

г) $M = 4 \cdot \dfrac{x + y}{x + 1} - \dfrac{xy - 12}{34 + x}$

д) $X = \dfrac{a + \sqrt{a^2 + 4b - ac}}{7c} - a^3c + b^{-2}$

Программы с линейной структурой

В каждой программе такого типа следует:

1. вывести на экран сообщение о том, что следует вводить (одно число, два числа, радиус, длину стороны треугольника и т. д.)
2. ввести нужные данные
3. вычислить значения требуемых величин
4. вывести полученные значения на экран

Примеры

1. Программа вычисляет площадь круга радиуса ***R***

```
Program krug;
Var R,S:real;
Begin
  Write('Введите радиус круга:');
  Readln(R);
  S:=3.14*sqr(R);
  Writeln('S=', S:8:2)
End.
```

2. Программа вычисляет площадь параллелограмма с основанием ***a*** и высотой ***h***

```
Program paral;
Var a,h,S:real;
Begin
  Write('Введите основание и высоту пар-ма:');
  Readln(a,h);
  S:=a*h;
  Writeln('S=', S:8:2)
End.
```

3. Программа вычисляет площадь и периметр равностороннего треугольника со стороной ***a***

```
Program treug;
Var a,P,S:real;
Begin
  Write('Введите сторону треугольника:');
  Readln(a);
  P:=3*a;
  S:=sqrt(3)*sqr(a)/4;
  Writeln('P=', P:10:2, 'S=', S:10:2)
End.
```

4. Программа вычисляет значение выражения $Y = \dfrac{\sin(2x+1)+x^2}{1-\sqrt{|x|+2}}$

```
Program func;
Var x,y:real;
Begin
  Write('Введите значение переменной x:');
  Readln(x);
  y:=(sin(2*x+1)+sqr(x))/(1-sqrt(abs(x)+2));
  Writeln('y=', y:10:2)
End.
```

1. Написать программу для вычисления суммы, разности, произведения и частного 2-х введенных вещественных чисел. Требуемый вид экрана:

```
Введите 2 числа:
a= …
b= …
a+b= …  ;   a-b= …  ;   a*b= …  ;   a/b= …
```

   Значения выводить с 2 знаками после запятой. Считать b≠0.
2. Написать программу для вычисления площади и периметра прямоугольного треугольника по введенным длинам его катетов a и b.
3. Написать программу для вычисления длины окружности и площади круга одного и того же введенного радиуса R.
4. Написать программу для вычисления расстояния между двумя точками по их координатам (x1,y1) и (x2,y2).
5. Написать программу для вычисления значения функции Y(x) по заданному значению переменной X, если

$$Y = \frac{1+\sin\sqrt{x^2+1}}{\cos(12x-4)}$$

Условный оператор

Для осуществления ветвления в программе используется условный оператор.
Он имеет полную и неполную формы:

Полная форма:
If <*условие*> Then <*оператор1*> Else <*оператор2*>

Неполная форма:
If <*условие*> Then <*оператор1*>

{Точка с запятой перед Else не ставится!}

Условный оператор выполняется следующим образом:

1. проверяется *условие*
2. если *условие* выполнено, выполняется *оператор1*
3. если *условие* не выполнено, в полной форме выполняется *оператор2*, а в неполной форме никаких действий не выполняется
4. осуществляется переход к следующей команде

Условие может быть простым или сложным.
Простое условие – сравнение значений двух выражений с помощью знаков сравнения:

<	– меньше	**<=**	– меньше или равно	**=**	– равно
>	– больше	**>=**	– больше или равно	**<>**	– неравно

Например:
1) X>=3; *2) A<>B;* 3) *sqr(c)=sqr(a)+sqr(b)*

Сложное условие – несколько простых условий, соединенных логическими операциями:

Not	– не	**And**	– и	**Or**	– или	**Xor**	– исключающее или

Простые условия, входящие в состав сложного, записываются в скобках.
Например:

1) *(X>3) and (Y<=17) and (Z<>2)* – *выполнено, если выполнены все простые условия*
2) *(X<2) or (X=8) or (X>=12)* – *выполнено, если выполнено хоть одно из простых условий*
3) *(X=7) xor (Y=9) xor (Z=1)* – *выполнено, если выполнено ровно одно из простых условий*

Операторы 1 и 2 могут быть простыми и составными. ***Простой оператор*** – один оператор.
Составной оператор – несколько операторов, заключенных в *операторные скобки* (`Begin … End`).

Примеры:

1. **Программа выводит на экран большее из двух целых чисел**

```
Program max;
Var a,b,m:integer;
Begin
  Write('Введите 2 числа');
  Readln(a,b);
  If a>b then writeln(a, 'большее')
         else writeln(b, 'большее')
End.
```

2. Программа вычисляет значение функции $f = \sqrt{x}$

```
Program func1;
Var x,f:real;
Begin
  Write('x=');
  Readln(x);
  If x>=0 then begin
                 f:=sqrt(x);
                 Writeln('f=',f:7:2)
               end
End.
```

3. Программа вычисляет значение функции

$$f = \begin{cases} -x, x < 2 \\ 0, x = 2 \\ x, x > 2 \end{cases}$$

```
Program func2;
Var x,f:real;
Begin
  Write('x=');
  Readln(x);
  If x<2 then f:=-x
         else if x=2 then f:=0
                     else f:=x;
  Writeln('f=',f:7:2)
End.
```

1. Вывести на экран меньшее из двух заданных чисел
2. Вычислить значение функции f по заданному значению x, если

а) $f = |x|$ б) $f = \begin{cases} \sin x, x \geq 0 \\ \sqrt{|x|}, x < 0 \end{cases}$ в) $f = \begin{cases} 1, x < 5 \\ 2, x = 5 \\ 3, x > 5 \end{cases}$

3. Определить, является ли введенное целое число четным (*число четное, если остаток от деления на 2 равен 0*)
4. Определить, является ли число ***c*** средним арифметическим (средним геометрическим) чисел ***a*** и ***b***
5. Определить, является ли треугольник со сторонами ***a***, ***b*** и ***c*** (где ***c*** – большая сторона) прямоугольным

Программы с условием

1. Программа для заданного числа R выводит на экран ‘R<0’, если R отрицательное, и вычисляет площадь круга радиуса R, если R≥0

```
Program krug;
Var r,S:real;
Begin
  Write('Введите число r: ');
  Readln(r);
  If r<0 Then Writeln('r<0')
         Else
           Begin
             S:=3.14*sqr(r);
             Writeln('площадь круга радиуса r равна ',S:8:1)
           End
End.
```

2. Программа определяет, является ли треугольник со сторонами a, b и c равнобедренным

```
Program treug1;
Var a,b,c:real;
Begin
  Write('Введите стороны треугольника: ');
  Readln(a,b,c);
  If (a=b)or(b=c)or(a=c) Then Writeln('треугольник равнобедренный')
                         Else Writeln('треугольник не равнобедренный')
End.
```

3. Программа по заданным значениям трех углов определяет, может ли существовать треугольник с такими углами, и если может, то определяет вид этого треугольника

```
Program treug2;
Var A,B,C:real;
Begin
  Write('Введите значения трех углов в градусах: ');
  Readln(A,B,C);
  If A+B+C=180 Then
                 If (A<90)and(B<90)and(C<90)
                 Then Writeln('треугольник с такими углами остроугольный')
                 Else
                   If (A=90)or(B=90)or(C=90)
                   Then Writeln('треугольник с такими углами прямоугольный')
                   Else Writeln('треугольник с такими углами тупоугольный')
               Else Writeln('треугольника с такими углами не существует')
End.
```

1. Для заданного числа ***R*** вывести на экран ***‘R<0’***, если ***R*** отрицательное, и вычислить длину окружности радиуса ***R***, если ***R≥0***
2. Определить, является ли треугольник со сторонами ***a***, ***b*** и ***c*** равносторонним
3. Определить, является ли треугольник со сторонами ***a***, ***b*** и ***c*** равнобедренным, но не равносторонним
4. По заданным значениям двух углов определить, может ли существовать треугольник с такими углами, и если может, то определить вид этого треугольника (остроугольный, тупоугольный или прямоугольный)
5. Определить, сколько чисел из заданных чисел ***a***, ***b*** и ***c*** являются
   а) положительными б) отрицательными в) равными 0
6. Решить квадратное уравнение $ax^2+bx+c=0$ для введенных коэффициентов ***a***, ***b*** и ***c*** ($a\neq0$). Вывести на экран значения корней или «***корней нет***».
   Проверить правильность работы программы для уравнений
   - $x^2+2x+1=0$ (1 корень, $x=1$)
   - $2x^2+x-1=0$ (2 корня, $x=-1, x=0.5$)
   - $2x^2+x+1=0$ (нет корней)
7. Решить уравнение $ax^2+bx+c=0$ для произвольных ***a***, ***b*** и ***c*** (любой из коэффициентов может быть равен 0)

Оператор выбора CASE

Оператор выбора позволяет программировать ветвление по многим направлениям. Этот оператор организует переход на одну из нескольких ветвей в зависимости от значения заданного выражения.

Формат:

```
Case K of
  A1: <оператор 1>;
  A2: <оператор 2>;
  …
  An: <оператор n>
  Else <оператор n+1>
End
```

K – *выражение простого типа (Integer, Char, Boolean);*
A_1, …, A_n – *значение, несколько значений или диапазон значений того же типа;*
все операторы могут быть простыми или составными;
Else может отсутствовать;
«;» перед Else не ставится

Оператор Case выполняется следующим образом:

1. вычисляется значение выражения K;
2. полученное значение сравнивается со значениями A_1, …, A_n;
3. если значение выражения K совпало с одним из значений (или попало в один из диапазонов) A_1, …, A_n, то выполняется соответствующий оператор;
   иначе выполняется оператор, следующий за Else (или ничего не происходит, если Else отсутствует)

Например,

1. **программа по номеру дня недели выводит его название**

```
Program days;
Var x:integer;
Begin
  Write('Введите № дня недели:');
  Readln(x);
  Case x of
    1: write('Понедельник');
    2: write('Вторник');
    3: write('Среда');
    4: write('Четверг');
    5: write('Пятница');
    6: write('Суббота');
    7: write('Воскресенье')
    Else write('такого нет')
  End
End.
```

2. программа по номеру месяца выводит количество дней в этом месяце *(если для разных значений следует выполнить одни и те же действия, то значения можно перечислять через запятую)*

```
Program month;
Var m:integer;
Begin
  Write('Введите № месяца: ');
  Readln(m);
  Case m of
    1,3,5,7,8,10,12: write('31');
    4,6,9,11: write('30');
    2: write('28 или 29')
  End
End.
```

(Else можно не использовать, при вводе числа не от 1 до 12 ничего не происходит)

3. программа вычисляет для целых значений X значение функции

$$Y = \begin{cases} x + 1, \text{ если } -3 \le x \le 0 \\ x - 1, \text{ если } 0 < x \le 5 \\ \text{не задана, если } x < -3 \text{ или } x > 5 \end{cases}$$

(Если значения идут подряд, их можно записывать как диапазон, указывая верхнюю и нижнюю границу, например,

- *в диапазон* `3..7` *входят* `3,4,5,6,7`;
- *в диапазон* `-2..1` *входят* `-2,-1,0,1`;
- *в диапазон* `'b'..'e'` *входят* `'b','c','d','e'` *и т. д.)*

```
Program func;
Var x,y:integer;
Begin
  Write('Введите x');
  Readln(x);
  Case x of
    -3..0: begin
             y:=x+1;
             Writeln('Y=', y)
           end;
     1..5: begin
             y:=x+1;
             Writeln('Y=', y)
           End
     Else Writeln('функция не задана')
  End;
End.
```

1. Для введенной цифры от 0 до 9 вывести ее название на английском языке;
2. По последней цифре числа определить последнюю цифру его квадрата;
3. Элементы равностороннего треугольника: **1** - сторона **a**, **2** - площадь **S**, **3** - высота **h**, **4** - радиус вписанной окружности **r**, **5** - радиус описанной окружности **R**;
   По заданному номеру и значению элемента вычислить значения всех остальных элементов.

Операторы цикла

Операторы цикла используются для реализации многократного выполнения одной или нескольких команд. Различают 3 типа циклов:

- ***цикл с параметром***;
- ***цикл с предусловием***;
- ***цикл с постусловием***.

Все 3 типа реализованы операторами языка Pascal.

1. Цикл с параметром

Цикл с параметром используется тогда, когда заранее известно количество повторений и шаг постоянен. Этот тип цикла реализуется в языке Pascal оператором ***For***, который имеет 2 варианта записи:

1) **For <*параметр цикла*>:=<*начальное значение*> to <*конечное значение*> do <*тело цикла*>**

2) **For <*параметр цикла*>:=<*начальное значение*> downto <*конечное значение*> do <*тело цикла*>**

Параметр цикла – переменная целого типа*;
н*ачальное и конечное значения* – выражения того же типа;
тело цикла – простой или составной оператор.

- Цикл повторяется, пока значение параметра лежит в интервале между начальным и конечным значениями
- При каждом повторении цикла значение параметра автоматически изменяется: в первом варианте увеличивается, а во втором уменьшается на 1.
- Значение параметра цикла можно использовать, но *нельзя изменять* внутри цикла.

2. Цикл с предусловием (цикл – *пока*)

Цикл с предусловием – наиболее универсальная циклическая структура (с его помощью можно представить любой цикл). В языке Pascal этот вид цикла реализован оператором ***While***:

```
While <условие> do <тело цикла>
```

Оператор ***While*** выполняется следующим образом:

1. проверяется условие
2. если условие выполнено, выполняется тело цикла и происходит переход к п. 1
3. если условие не выполнено, то происходит переход к следующей команде (выход из цикла)

- Цикл повторяется, пока условие *выполнено*
- Если изначально условие не выполнено, то тело цикла не выполняется ни разу
- Оператор ***While*** может зациклиться: если изначально условие выполнено, а в теле цикла его значение не изменяется, то выхода из цикла не происходит

3. Цикл с постусловием (цикл – *до*)

Цикл с постусловием реализован в языке Pascal оператором *Repeat*:

```
Repeat <тело цикла> until <условие>
```

Оператор ***Repeat*** выполняется следующим образом:

1. выполняется тело цикла
2. проверяется условие
3. если условие не выполнено, то происходит переход к п. 1
4. если условие выполнено, то происходит переход к следующей команде (выход из цикла)

- Цикл повторяется, пока условие *не выполнено* (до момента, когда условие выполниться)
- Тело цикла выполняется по крайней мере 1 раз
- Оператор ***Repeat*** может зациклиться: если изначально условие не выполнено, а в теле цикла его значение не изменяется, то выхода из цикла не происходит
- Если тело цикла ***Repeat*** – составной оператор, то операторные скобки не обязательны (тело цикла в этом операторе всегда ограничено ключевыми словами ***Repeat*** и ***Until***)

---

* Параметром цикла For может быть переменная любого простого порядкового типа (не обязательно Integer). При выполнении цикла значение параметра изменяется на следующее или предыдущее значение в данном типе

Цикл с параметром

Примеры:

1. Программа выводит на экран числа от 1 до 7

```
Program prog1;
var i:integer;
begin
  for i:=1 to 7 do writeln(i);
end.
```

2. Программа выводит на экран числа от 7 до 1

```
Program prog2;
var i:integer;
begin
  for i:=7 downto 1 do writeln(i);
end.
```

3. Программа 5 раз вводит значение переменной ***a*** и выводит на экран число на 1 больше

```
Program prog3;
var i,a:integer;
begin
  for i:=1 to 5 do
    begin
      write('a',i,'=');
      readln(a);
     a:=a+1;
     writeln('a', i, '+1=', a)
    end
end.
```

4. Программа 10 раз увеличивает введенное число на 1 и выводит полученные значения на экран

```
Program prog4;
var a,x:integer;
begin
  write('x=');
  readln(x);
  for a:=1 to 10 do
    begin
      x:=x+1;
      writeln('x+',a,'=',x);
    end
end.
```

5. Программа вычисляет S=1+2+…+N

```
Program sum1;
var i,n,s:integer;
begin
  write('n=');
  readln(n);
  s:=0;
  for i:=1 to n do s:=s+i;
  writeln('s=',s);
end.
```

6. Программа вычисляет $S = \dfrac{1}{1+2+\cdots+N}$

```
Program prog6;
var i,n:integer; s:real;
begin
  write('n=');
  readln(n);
  s:=0;
  for i:=1 to n do s:=s+i;
  s:=1/s;
  writeln('s=',s);
end.
```

1. Написать программу, которая выводила бы на экран:

а) 5 первых четных чисел **б) введенное целое число 7 раз** **в) N первых четных чисел** **г) N первых нечетных чисел**

2. Задано натуральное число N. Вычислить:

а) $S = 1 \times 2 \times 3 \times \ldots \times N$

б) $S = 1 - \dfrac{1}{2} + \dfrac{1}{4} - \dfrac{1}{8} + \ldots + (-1)^N \cdot \dfrac{1}{2^N}$

в) $S = \sin 1 + \sin 2 + \ldots + \sin N$

г) $S = \dfrac{1}{\sin 1} + \dfrac{1}{\sin 1 + \sin 2} + \ldots + \dfrac{1}{\sin 1 + \sin 2 + \ldots + \sin N}$

д) $S = \dfrac{\cos 1}{\sin 1} + \dfrac{\cos 1 + \cos 2}{\sin 1 + \sin 2} + \cdots + \dfrac{\cos 1 + \cos 2 + \ldots + \cos N}{\sin 1 + \sin 2 + \ldots + \sin N}$

3. Вычислить значение функции *F(x)* на отрезке [*a,b*] с интервалом *t*, если

а) $F(x) = x^3 + 5x^2 - 7x + 14$, *a=0, b=10, t=1*

б) $F(x) = \sqrt{|x+1|}$, *a=5, b=8, t=0,5*

в) $F(x) = \dfrac{x+1}{x^2+4}$, *a=0, b=24, t=4*

4*. *Выполнить задания 1-3, используя различные виды циклов*

Символьные и строковые данные

Символьная переменная имеет тип ***Char***. ***Символьная константа*** – символ в апострофах.
Каждому символу соответствует свой уникальный ***код*** (кодировочная таблица ASCII) – целое число от ***0*** до ***255***.
Символы упорядочены в соответствии с кодами:

- `'A'<'B'<'C'< …`
- `'0'<'1'<'2'< …`
- `'a'<'b'<'c'< …`
- `'Z'<'a'` (коды прописных латинских букв меньше кодов строчных)

С символами связаны 2 функции:

1. **`CHR(N)`** – символ, код которого равен **N** (N – число от 0 до 255)
2. **`ORD(C)`** – код символа, являющегося значением переменной **C** (C – символьная переменная)

Строка – упорядоченная последовательность символов. Количество символов в строке называется ее ***длиной***. Длина строки может лежать в диапазоне от ***0*** до ***255***.
Строковая переменная имеет тип ***String***. При описании строковой переменной можно указать максимальную длину строки:

```
Var s:string[20]; a:string[5]; b:string;
```

Если длина не указана, подразумевается, что она равна максимальной величине – ***255***.
Строковая константа – последовательность символов, заключенных в апострофы. Два следующих друг за другом апострофа (’’) обозначают ***пустую строку*** (строку с длиной 0).

К каждому символу в строке можно обращаться по отдельности. Имя ***элемента строки*** состоит из имени строки с индексом, заключенным в квадратные скобки. ***Индекс*** – выражение целого типа с положительным значением. Например, **S[2]** – второй символ строки ***S***; **N[i]** – ***i***-й символ строки ***N***; **K[n+1]** – символ строки ***K*** с номером ***n+1***. **Первый символ строки имеет номер 1. Значение индекса не должно превышать максимальное количество символов в строке.**

Типы ***Char*** и ***String*** совместимы.

Операции над строками

1. ***Сравнение строк***
   Строки можно сравнивать, используя знаки сравнения =, <, >, <=, >=, <>. Строки сравниваются посимвольно слева направо до первого несовпадающего символа. Большей считается та строка, в которой первый несовпадающий символ имеет больший код. Если строки имеют разную длину, но в общей части символы совпадают, то большей считается более длинная строка. Строки равны только в том случае, если они имеют одинаковую длину и все символы совпадают.
   Например,
   - **`'abcd'< 'am'`**, т.к. ’b’<’m’
   - **`'abcd'>'ab'`**, т.к. первые символы совпадают, а строка ’abcd’ длиннее
   - **`'abcd'='abcd'`**, т.к. строки совпадают посимвольно,
     но строки **`'abcd'`** и **`'a bcd'`** не равны, т. к. одна из них содержит пробел, а другая нет

2. ***Сцепление (конкатенация) строк***
   Обозначается знаком «+». Применяется для соединения нескольких строк в одну.
   Например, в результате сцепления ***`'мама '+'мыла '+'раму'`*** получится строка ***`'мама мыла раму'`***

   Сцепление строк выполняется и с помощью функции ***Concat***.
   `Concat(S1, S2, …, Sn)` выполняет сцепление строк ***S1***, ***S2***, …, ***Sn*** в одну строку.

3. ***Определение длины строки***
   `Lenght(S)` – текущая длина строки ***S***. Результатом является целое число.
   Например,
   - значение ***`Length('abcd')`*** равно ***4***
   - значение ***`Length('мама '+'мыла '+'раму')`*** равно ***14***
   - значение ***`Length('')`*** равно ***0***

4. ***Поиск подстроки в строке***
   `Pos(S1,S2)` – первое появление строки ***S1*** в строке ***S2***. Результатом является целое число, равное номеру

позиции, где находится первый символ подстроки ***S1***. Если строка ***S1*** не входит в строку ***S2***, то результатом будет ***0***.

Например,

- значение ***Pos('bc', 'abcd')*** равно ***2***
- значением ***Pos('bc', 'bcbc')*** равно ***1***
- значением ***Pos('bc', 'acbd')*** равно ***0***

5. ***Выделение подстроки из строки***
   Copy(S,P,L) – подстрока строки ***S***, начинающаяся с символа с номером ***P*** и с количеством символов ***L***. Результатом является подстрока, при этом сама строка S не изменяется.
   Например,
   - значением ***Copy('холодильник',1,5)*** будет строка ***'холод'***
   - значением ***Copy('холодильник',6,2)*** будет строка ***'ил'***

6. ***Удаление символов из строки***
   Delete(S,P,N) – удаление из строки ***S N*** символов, начиная с символа с номером ***P***. Если ***P+N*** больше длины строки, то удаляются все символы в строке, начиная с символа с номером ***P*** и до конца строки.
   Например,
   - если ***S*** – это строка ***'холод'***, то после выполнения ***Delete(S,3,2)*** значением ***S*** станет ***'ход'***
   - если ***S*** – это строка ***'бумага'***, то после выполнения ***Delete(S,4,10)*** значением ***S*** станет ***'бум'***

7. ***Вставка символов в строку***
   Insert(S1,S2,P) – вставляет строку ***S1*** в строку ***S2***, начиная с позиции ***P***.
   Например,
   - если ***S*** – это строка ***'ход'***, то после выполнения ***Insert('ло',S,3)*** значением ***S*** станет ***'холод'***
   - если ***S*** – это строка ***'ум'***, то после выполнения ***Insert('ш',S,1)*** значением ***S*** станет ***'шум'***

8. ***Преобразование числа в строку***
   Str(N,S) – преобразует арифметическое выражение, заданное в переменной ***N***, в строку ***S***.
   Например,
   - после выполнения команды ***Str(192,S)*** значением переменной ***S*** будет строка ***'192'***

9. ***Преобразование строки в число***
   Val(S,N,P) – преобразует значение, заданное в строке ***S***, к числовому виду, если это возможно, и присваивает это числовое значение переменной ***N***. Если преобразование выполнено, то значение переменной ***P*** станет ***0***, если нет, то ***P*** станет равным номеру позиции первого ошибочного символа строки ***S***.
   Например,
   - если ***S*** – строка ***'345'***, то после выполнения команды ***Val(S,N,P)*** значением переменной ***N*** будет число ***345***, а значением переменной ***P – 0***
   - если ***S*** – строка ***'34k5'***, то после выполнения команды ***Val(S,N,P)*** значение переменной ***N*** не изменится, а значением переменной ***P*** будет ***3***

1. Сформировать строку из трех заданных и определить длину полученной строки
2. Вводят символ. Вывести на экран строку из 3 таких символов
3. Вводят символ. Вывести на экран строку из N таких символов
4. Определить, есть ли во введенной строке буква Z
5. Определить, является ли строка X подстрокой строки Y
6. Удалить из введенной строки все символы, кроме первого и последнего
7. Вывести на экран введенную строку без первого и последнего символа (2 способа)
8. Определить количество букв Z во введенной строке
9. Во введенной строке заменить первую букву S на значок $
10. Во введенной строке заменить все буквы S на значок $
11. Из введенной строки удалить все символы, кроме цифр
12. Вводят строку. Определить, число ввели или нет
13. Определить первую цифру введенного целого числа
14. Строка содержит один вопросительный знак. Определить количество символов до и после него

15*.Вывести на экран 3 введенные слова в алфавитном порядке

16. Дана строка, в которой один раз встречаются скобки.
    Вывести на экран все символы, расположенные внутри этих скобок
17. Дана строка, в которой один раз встречаются скобки.
    Удалить все символы, расположенные внутри этих скобок, вместе со скобками
18. Удалить из введенной строки все пробелы
19. Во введенной строке после каждого символа вставить пробел
20. Строка содержит предложение, заканчивающееся точкой. Определить, сколько в нем слов
21. Определить, сколько различных символов входит в строку

Массивы

Типы данных могут быть скалярными или структурными. С переменной скалярного типа в каждый момент времени может быть связано только одно значение. С переменной структурного типа в каждый момент времени может быть связана группа значений одного или разных типов. К структурным типам данных относится массив.

Массив – упорядоченный набор однотипных значений.

Описание массива:

```
<имя>:Array[<тип индекса>] of <тип>
```

Тип индекса – последовательность диапазонов целых значений, перечисленных через запятую (1..10, -3..4, 2..7). Количество диапазонов называется *размерностью массива*. Массив размерности 1 называется *одномерным массивом* (*вектором*), размерности 2 – *двумерным массивом* (*матрицей*).

Например,

```
Var a:array[1..5] of integer; b:array[1..3, 1..7] of char;
```

(*Вектор a состоит из 5 элементов целого типа, матрица b состоит из 3 строк и 7 столбцов элементов символьного типа*).

К каждому элементу массива можно обращаться по отдельности. *Имя элемента массива* состоит из имени массива и индексов, заключенных в квадратные скобки. При этом количество индексов должно быть таким же как в описании массива, и их значения не должны выходить за границы диапазонов.

Т.о. элемент вектора определяется своим номером, а элемент матрицы номером строки и номером столбца, на пересечении которых он расположен.

Например,

- *a*[*3*] – третий элемент вектора *a*;
- *b*[*2,5*] – элемент матрицы *b*, расположенный на пересечении второй строки и пятого столбца.

Ввод, вывод и обработка массива производится поэлементно. Обычно для этого используется цикл.

Одномерные массивы

Ввод и вывод элементов массива

Программа вводит элементы одномерного целочисленного массива с количеством элементов до 20, а затем выводит их на экран в строку

```
Program massive;
Var M:array[1..20] of integer; i,N:integer;
Begin
  Writeln('Введите количество элементов массива (не больше 20):');
  Readln(N);
  Writeln('Введите элементы массива:');
  For i:=1 to N do
    Begin
      Write('M[',i,']=');
      Readln(M[i]);
    End;
  Writeln('Элементы массива:');
  For i:=1 to N do Write(M[i],' ');
  Writeln;
End.
```

1. Вывести на экран массив символов из 7 элементов в столбик (в строку через пробел)
2. Выяснить знак первого элемента числового массива
3. Увеличить все элементы целочисленного массива вдвое
4. Найти сумму (произведение) элементов числового массива
5. Найти сумму четных (нечетных, кратных данному числу K) элементов массива натуральных чисел
6. В целочисленном массиве есть нулевые элементы. Создать новый массив из номеров этих элементов
7. Вывести на экран элементы числового массива, большие (меньшие) заданного числа K
8. Массив не содержит нулевых элементов. Посчитать произведение положительных и произведение отрицательных элементов массива. Определить, какое из произведений больше по модулю
9. Посчитать количество положительных, отрицательных и нулевых элементов массива
10. Определить максимальный и минимальный элементы числового массива и их номера
11. Поменять местами наибольший и наименьший элементы числового массива
12. Массив состоит из 0 и 1. Поставить в начало все 0, а затем 1
13. Упорядочить элементы целочисленного массива по возрастанию (убыванию)
14. Удалить из целочисленного массива нулевые элементы
15. Удалить из массива повторяющиеся элементы