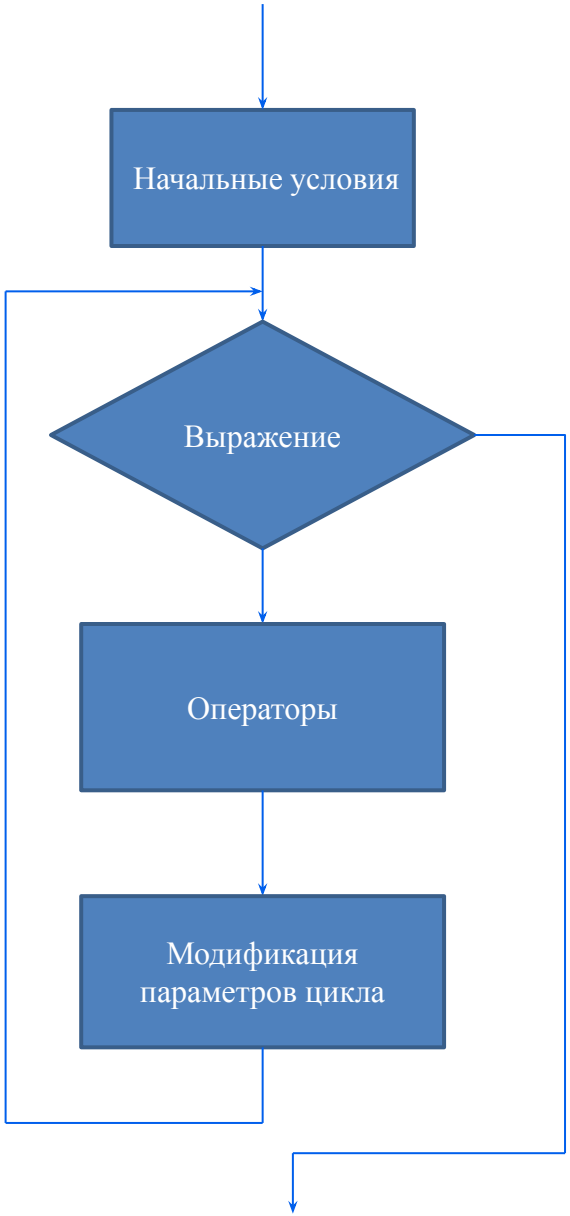
Презентация на тему: Pascal

Выполнил Карпов В.

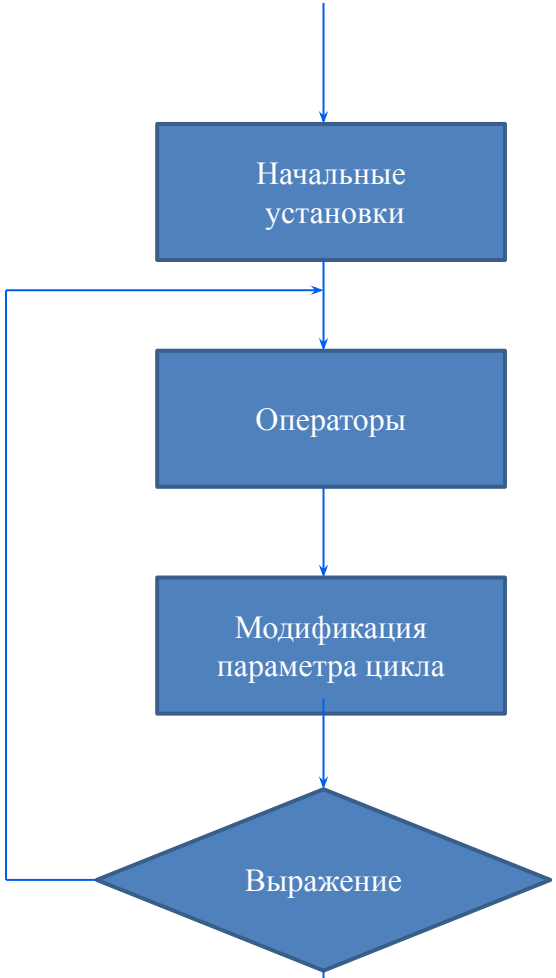
Оглавление

Операторы цикла

Операторы цикла используются для вычислений, повторяющихся многократно. В Паскале имеется три вида циклов: цикл с предусловием *while*,цикл с постусловием *repeat* и цикл с параметром *for.* Каждый из них состоит из определенной последовательности операторов. Блок, ради выполнения которого и организуется цикл, называется *телом цикла*. Один проход цикла называется *итерацией.*



а



б

Структурные схемы операторов цикла

Начальные установки служат для того, чтобы до входа в цикл задать значения переменных, которые в нем используются. *Проверка условия продолжения цикла* выполняется на каждой итерации либо до тела цикла(тогда говорят о цикле *с предусловием*, а),либо после тела цикла (цикл *с постусловием*, б).Разница между ними состоит в том, что тело цикла с постусловием всегда выполняется хотя бы один раз, после чего проверяется, надо ли его выполнять еще раз. Проверка необходимости выполнения цикла с предусловием делается до тела цикла, поэтому возможно, что он не выполнится ни разу.

Параметром цикла называется переменная, которая используется при проверке условия цикла и принудительно изменяется на каждой итерации, причем, как правило, на одну и ту же величину. Если параметр цикла целочисленный,он называется *счетчиком цикла*. Количество повторений такого цикла можно определить заранее. Параметр есть не у всякого цикла. В так называемом *итеративном* цикле условие продолжения содержит переменные, значения которых изменяются в цикле по рекуррентным формулам. Цикл завершается, если условие его продолжения не выполняется. Возможно принудительное завершение как текущей итерации, так и цикла в целом. Для этого служат процедуры *break, continue* и оператор. Передавать управление извне внутрь цикла не рекомендуется, потому что при этом могут не выполниться начальные установки.

Цикл с предусловием while

Формат оператора прост:

while выражение do оператор

Выражение должно быть логического типа. Например, это может быть операция отношения или просто логическая переменная. Если результат вычисления выражения равен true, выполняется расположенный после служебного слова do простой или составной оператор(составной оператор заключается между begin и end).Эти действия повторяются до того момента, пока результатом выражения не станет значение false. После окончания цикла управление передается на следующий за ним оператор.

Внимание!

Если в теле цикла требуется выполнить более одного оператора, необходимо заключить их в блок с помощью ключевых слов begin и end.

Пример. Программа, печатающая таблицу значений функции

$$y=\begin{cases} t, & x<0 \\ tx, & 0\le x<10 \\ 2tx, & x\ge 10 \end{cases}$$

для аргумента, изменяющегося в заданных пределах с заданным шагом.

Опишем алгоритм в словесной форме:

1. Ввести исходные данные.
2. Взять первое значение аргумента.
3. Определить, какому из интервалов оно принадлежит.
4. Вычислить значение функции по соответствующей формуле.
5. Вывести строку таблицы.
6. Перейти к следующему значению аргумента.
7. Если оно не превышает конечное значение, повторить шаги 3-6, иначе закончить.

Шаги 3-6 повторяются многократно, поэтому для их выполнения надо организовать цикл. Назовем необходимые переменные так: начальное значение аргумента-xk, шаг изменения аргумента-dx, параметр-t.Все величины вещественные. Программа выводит таблицу, состоящую из двух столбцов: значений аргумента и соответствующих им значений функции.

```
program tab1 fun;
var  Xn, Xk, dX, t, x, y: real;
begin
writeln (‘Введите  Xn, Xk, dX, t’);
readln (Xn, Xk, dX, t);
writeln(‘----------------------’);
writeln(‘ |      X          |         Y          | ‘);
writeln(‘----------------------’);
x:= Xn;                                          { Начальные установки }
while x <= Xk do begin                           { Заголовок цикла }
If x<0 then y:=t;
If (x>=0) and (x<10) then y:=t*x;
If x>=10 then y:=2*t;
writeln(‘ |‘,  x:9:2,’          | ‘, y:9:2,’      | ‘);
x:= x+dX;                                        { Модификация параметра цикла }
end;
writeln(‘------------------------’);
end.
```

Цикл с постусловием repeat

Тело цикла с постусловием заключено между служебными словами repeat и until,поэтому заключать его в блок не требуется.

repeat тело цикла until выражение

В отличие от цикла while этот цикл будет выполняться, пока логическое выражение после слова until ложно. Как только результат выражения станет истинным, произойдет выход из цикла. Вычисление выражения выполняется в конце каждой итерации цикла.

Этот вид цикла применяется в тех случаях, когда тело цикла необходимо обязательно выполнить хотя бы один раз: например, если в цикле вводятся данные и выполняется их проверка. Если же такой необходимости нет, предпочтительнее пользоваться циклом с предусловием.

Пример. Программа, вычисляющая квадратный корень вещественного аргумента X с заданной точностью eps по итерационной формуле:

$$y_n = \frac{1}{2}(y_{n-1} + \frac{x}{y_{n-1}})$$

где y_{n-1} -предыдущее приближение к корню (в начале вычислений выбирается произвольно),y -последующее приближение. Процесс вычислений прекращается, когда приближения станут отличаться друг от друга по абсолютной величине менее, чем на величину заданной точности.

```
program square root;
var  X, eps,                                  { аргумент и точность
      Yp, Y: real;          { предыдущее и последующее приближение
begin
     repeat
          writeln (‘Введите аргумент и точность (больше нуля):’);
          readln (X, eps);
     until (X>0) and (eps>0);
 Y:=1;
 repeat
          Yp:=Y;
          Y:=(Yp+X/Yp)/2;
    until abs(Y-Yp)< eps;
    writeln (‘Корень из ‘, X:6:3,  ‘с точностью ‘ , eps:7:5, ‘равен ‘ ,Y:9:5);
end.
```

Цикл с параметром for

Этот оператор применяется, если требуется выполнить тело цикла заранее заданное количество раз. Параметр порядкового типа на каждом проходе цикла автоматически либо увеличивается, либо уменьшается на единицу.

for параметр := выражение_1 to выражение_2 do оператор

for параметр := выражение_2 downto выражение_1 do оператор

Выражения должны быть совместимы с переменной цикла по присваиванию, оператор- простым или составным.

Пример 1. Программа выводит на экран в столбик числа от 1 до 10.

```
var i : integer;
begin
    for i :=1 to 10 do writeln(i)
end.
```

Цикл будет выполнен 10 раз, на каждом проходе счетчик цикла переменная i увеличивается на 1.

Пример 2. Программа выводит на экран в столбик числа от 10 до 1 и подсчитывает их сумму.

```
var I, sum: integer;
begin
    sum:= 0;
    for i:=10 downto 1 do begin
      writeln(i); inc(sum, i);
    end;
    writeln('Сумма чисел:',sum);
end.
```

В это цикле переменная i автоматически уменьшается на 1.

Пример 3. Программа выводит на экран символы от ‘a’ до ‘z’.

```
var ch: char;
begin
    for ch:= ‘a’ to ‘z’ do write(ch:2);
end.
```

Здесь счетчик цикла ch символьного типа поочередно принимает значение каждого символа от ‘a’ до ‘z’.

Внимание!

Если в теле цикла требуется выполнить более одного оператора, необходимо заключить их в блок с помощью ключевых слов begin и end.

Выражения, определяющие начальное и конечное значения счетчика, вычисляются один раз до входа в цикл. Цикл for реализован в Паскале как цикл с предусловием, то есть его можно представить в виде эквивалентного оператора while. Это означает, что если условие продолжения цикла не выполняется при первом же значении счетчика, тело цикла не будет пройдено ни разу. Так, цикл из первого примера можно записать в виде:

```
i:= 1;
while i<=10 do begin
      writeln(i);
      inc(i)
end.
```

После нормального завершения цикла значение счетчика не определено. Фактически оно равно первому значению, для которого выполняется условие выхода из цикла, но использовать это в программах не рекомендуется.

Оператор выбора (case)

Оператор выбора выполняет один из содержащихся в нем операторов. Какой оператор будет выбран, определяется значением селектора. Оператор выбора имеет следующую структуру:

```
Case селектор of
метка 1: Оператор 1;
метка 2: Оператор 2;
……..
метка N: Оператор N;
Else Оператор_без_метки;
End;
```

Все метки- метка 1,метка 2,…, метка N- являются константами того же типа, что и селектор. Исполнение оператора выбора начинается с вычисления значения селектора. Выполняется тот оператор, значение метки которого совпадает с вычисленным значением селектора. Две одинаковые метки использовать нельзя. Если же вычисленное значение не совпадает ни с одной из меток, то выполняется оператор, расположенный после слова Else. В качестве селектора можно использовать выражение только порядкового типа. Если оператор должен выполняться при нескольких значениях меток, их можно перечислить через запятую или в виде диапазона.

Пример. Программа, определяющая, какое число из диапазона от 2 до 5 введено-четное или нечетное. Для чисел, выходящих за границы диапазона, выводится диагностическое сообщение.

```
program pr1;
var n: ShortInt;
begin
   write (‘Введите число в диапазоне от 2 до 5’);
   readln (n);
   Case n Of
      3, 5: writeln (‘Вы ввели нечетное число’);
      2, 4: writeln (‘Вы ввели четное число’);
      Else writeln (‘Число за пределами диапазона’);
    End;
End.
```

Если диагностическое сообщение выводить не требуется, то строку, начинающуюся со слова Else, можно опустить. Если по какой-либо ветви требуется записать не один, а несколько операторов, они заключаются в блок с помощью ключевых слов begin и end.

Пример 2. Программа, вычисляющая по заданному значению аргумента x значение функции y, определяющейся следующим образом:

$$y=\begin{cases} 0; & x<-5 \\ x+5; & -5 \le x<0 \\ 5-x; & 0 \le x<4 \\ 1; & x \ge 4 \end{cases}$$

причем x является целым числом и принимает значение в диапазоне от -200 до 200.

```
program pr_2;
var n: Integer;
begin
      write (‘Введите число в диапазоне от -200 до 200’);
      readln (n);
      Case n Of
        -200 … -6: writeln (0);
        -5 … -1: writeln (x+5);
         0 …3:    writeln (5-x);
         4 … 200: writeln (1);
       Else writeln (‘Число за пределами диапазона’);
     End;
End.
```

В приведенной программе вместо того, чтобы перечислять все последовательные значения, указан диапазон их изменения. В метках можно сочетать перечисление значений и диапазон. Такой способ задания значения меток также является допустимым в операторе выбора, если значения не повторяются в разных метках.

Параметрами процедур вывода могут быть выражения. В этом случае вычисляется значение выражения и выводится полученный результат.

Оператор метки и безусловного перехода

Метка помечает какое-либо место в тексте программы. Метками могут быть числа от 0 до 9999 или идентификаторы, которые в этом случае уже нельзя использовать для каких-либо иных нужд. Все метки должны быть описаны в специальном разделе Label:

Label <список_всех_меток_через_запятую>;

Любая метка может встретиться в тексте программы только один раз. Используются метки только операторами безусловного перехода goto:

goto <метка>;

Это означает, что сразу после оператора goto будет выполнен не следующий за ним оператор (как это происходит в обычном случае), а тот оператор, который помечен соответствующей меткой.

Пример. Программа, определяющая максимальное число из вводимой последовательности целых чисел. Количество вводимых чисел заранее не известно. Анализ последовательности заканчивается после ввода отрицательного числа.

```
program pr_1;
Label Inp;
var n, max: Integer;
begin
    max:=-1;
    Inp: write (‘Введите число’);
    readln (n);
    If n>max then max:=n;
    If n >= 0 then GoTo Inp;
    writeln (‘Максимум =‘, max);
End.
```

В первой строке программы объявлена метка Inp. Затем объявлены две переменные n и max, первая из которых предназначена для записи текущего вводимого значения, вторая –для записи текущего максимального значения. Первый оператор программы присваивает переменной max начальное значение, равное -1. Следующие четыре строки программы выполняются до тех пор, пока не будет введено отрицательное число. Последняя строка предназначена для вывода полученного максимального значения.