ОРГАНИЗАЦИЯ ЦИКЛОВ

Оператор цикла задает повторное выполнение определенных операторов. Для реализации циклов в Паскале предусмотрены три различных структурных оператора: WHILE, REPEAT, FOR. Первые два используются, если число повторений (итераций) заранее не определено, но известно условие завершения цикла. Оператор FOR применяется тогда, когда число повторений тела цикла известно заранее.

Оператор WHILE

Этот оператор является наиболее мощным из всех трех, реализующих циклы. Два других оператора можно выразить с его помощью. Общий этого оператора:

WHILE <Условие> DO <Тело цикла>;

Логическое выражение, стоящее после WHILE, называется *условием возобновления цикла* и должно иметь булевский тип. Оператор, следующий за DO, является *телом цикла*. Он повторяется до тех пор, пока истинно условие возобновления цикла. Как только условие возобновления цикла становится ложным, управление переходит к оператору, стоящему за WHILE. Если условие возобновления не удовлетворяется до начала выполнения цикла, то тело цикла пропускается.

Из указанного описания видно, что оператор WHILE реализует базовую структуру «цикл-пока», так как здесь проверка условия идет до тела цикла. Поэтому оператор WHILE называют *оператором цикла с предусловием*.

П р и м е р . Даны числа A, B (A > 1). Получить все степени числа A, меньшие числа B.

```
program STEPENI;
var A, B, C: real;
begin
   readln (A, B); C := A;
   while C < B do
      begin
        writeln (C);
        C := C*A;
      end;
end.
```

Примечание. Грамотное использование оператора WHILE предполагает умение правильно написать условие возобновления цикла. Здесь надо иметь в виду следующие рекомендации:

1. В условии, как в логическом выражении, должны обязательно фигурировать переменные, изменяющие свои значения в теле цикла.
2. Во избежание зацикливания лучше сначала написать условие прекращения цикла и взять потом в операторе его отрицание.
3. Переменные логического выражения должны получить свои исходные значения до входа в оператор WHILE.

3.2. Оператор REPEAT

Оператор REPEAT называют оператором цикла с постусловием, так как здесь выражение, управляющее повторным выполнением последовательности операторов, помещается после тела цикла:

REPEAT <Тело цикла> UNTIL <Условие>;

Из общего вида оператора видно, что в этом операторе не обязательно использовать для тела цикла операторные скобки. Здесь ключевые слова REPEAT и UNTIL сами играют роль этих скобок.

В этом операторе тело цикла выполняется до тех пор, пока ложно условие, стоящее после UNTIL. Условием выхода из цикла является истинность выражения. Мы видим, что это есть форма «цикла-до».

П р и м е р . Даны числа A, B (A > 1). Получить все степени числа A, меньшие числа B.

```
program STEPENI;
var A, B, C: real;
begin
   readln (A, B); C := A;
   repeat
   writeln (C);
   C := C*A;
   until C >= B;
end.
```

Примечание. Между операторами WHILE и REPEAT существует три основных различия:

1. В операторе REPEAT проверка условия выхода из цикла выполняется в конце, а не в начале цикла, как в операторе WHILE, поэтому в операторе REPEAT тело цикла выполняется хотя бы один раз.

2. В REPEAT выход из цикла осуществляется по истинности условия, а в WHILE – по ложности.

3. В операторе WHILE тело цикла чаще всего имеет форму составного оператора, в операторе REPEAT для организации тела цикла операторные скобки не нужны.

Оператор FOR

Оператор FOR предназначен для организации циклов, когда заранее известно, сколько раз должно повториться тело цикла. Здесь управление числом повторений осуществляется с помощью специальной переменной – параметра цикла (управляющей переменной), которой присваивается возрастающая (убывающая) последовательность значений. Оператор FOR имеет следующий вид:

FOR <Переменная>:= <Выражение 1> TO <Выражение 2> DO;
FOR<Переменная>:=<Выражение1>DOWNTO<Выражение1>DO;

Здесь «Переменная» есть параметр цикла, «Выражение 1» – начальное значение параметра, «Выражение 2» – его конечное значение. В качестве управляющей переменной должна быть переменная, объявленная локальной в блоке, который содержит данный оператор FOR. Управляющая переменная должна иметь ординальный тип. Начальное и конечное значения имеют тип, совместимый с типом параметра цикла.

Когда начинает выполняться оператор FOR, начальное и конечное значения определяются один раз, и эти значения сохраняются на протяжении всего выполнения оператора.

Оператор, который содержится в теле цикла, выполняется один раз для каждого значения управляющей переменной в диапазоне между начальным и конечным значениями. Управляющая переменная всегда инициализируется начальным значением. Она принимает все свои значения из диапазона с шагом 1, если TO, и с шагом -1, если DOWNTO.

В случае TO, если начальное значение превышает конечное, тело цикла не выполняется.

Для случая DOWNTO это имеет место, когда начальное значение меньше, чем конечное. Отсюда заключаем, что оператор цикла FOR реализует, как и WHILE, схему цикла «пока» – проверка условия повторения цикла идет до тела цикла.

Примечание.

1. Если тело цикла в этом операторе состоит из более одного оператора, то они все заключаются в операторные скобки (реализуют конструкцию составного оператора).

2. В отличие от школьного алгоритмического языка, оператор FOR нельзя прервать путем присваивания управляющей переменной ее конечного значения. Изменения переменной цикла не влияют на число повторений тела цикла.

3. После выполнения оператора значение управляющей переменной становится неопределенным, если только выполнение оператора FOR не было прервано с помощью оператора перехода.

Рассмотрим примеры использования оператора FOR для организации циклических процессов.

П р и м е р 1. Печать отсчета цифр при старте.

```
program START;
var SEC: integer;
begin
   writeln ('До старта осталось ...');
   for SEC := 10 downto 1 do
   writeln (SEC:4);
   writeln ('ноль');  writeln ('Старт !!')
end.
```

В данном примере управляющая переменная SEC принимает значения типа INTEGER, однако в Паскале она определена как переменная ординального типа и, следовательно, может принимать значения типа CHAR или принадлежать перечислимому типу, как показано в примере 2.

П р и м е р 2 . Подсчет числа часов рабочей недели.

```
program WORKTIME;
        type DAYS = (MO, TU, WE, TH, FR, SA, SU);
        var DEN: DAYS; WT: integer;
        begin
                WT := 0;
                for DEN := MO to SA do
                        if DEN <> SA then WT := WT + 8
                                else WT := WT + 7; writeln (WT);
                                end.
```

Пример: На промежутке от 1 до *M* найти все числа Армстронга. Натуральное число из *n* цифр называется числом Армстронга, если сумма его

цифр, возведенных в степень *n*, равна самому числу. Например, число 153 ($153=1^3+5^3+3^3$).

Решение. После организации ввода данных программа будет содержать цикл с параметром *i* (от 1 до *M*) с двумя вложенными циклами. Первый предназначен для подсчета количества цифр *n*, второй – для вычисления суммы *s* степеней цифр числа *i*. Если числа *i* и *s* равны, то *i* – число Армстронга, его необходимо вывести на экран.

```
PROGRAM Primer_1;
var i,k,s,p,n,M: Integer;
begin
  Write('Введите M '); Readln(M);
  for i:=1 to M do
    begin
      s:=0; k:=i; n:=0;
      while k<>0 do
        begin k:=k div 10; n:=n+1 end;
      k:=i;
      While k<>0 do
        begin p:=k mod 10; k:=k div 10;
            if p<>0 then s:=s+ Round(Exp(n*Ln(p)))
        end;
      if s=i then Writeln(i);
    end;
  Readln;
end.
```

Вопросы для самопроверки:

1. Как записывается и как работает оператор *for*?
2. Для организации каких циклов применим оператор *for*?
3. В чем отличие оператора *while* от оператора *repeat*?
4. Как программируются циклические алгоритмы с явно заданным числом повторений цикла?
5. Напишите пример оператора цикла, который не выполняется ни разу.
6. С какими ограничениями реализована конструкция цикла со счетчиком?
7. Замените оператор "*repeat A until B*" равносильным фрагментом программы с оператором *while*.

Задания

1. Найти все двузначные числа, сумма цифр которых не меняется при умножении числа на 2,3.

2. Найти все трехзначные числа, сумма кубов цифр которых равна данному целому числу.
3. Найти все трехзначные числа, средняя цифра которых равна сумме первой и третьей цифр.
4. Найти все трехзначные числа, которые можно представить разностью между квадратом числа, образованного первыми двумя цифрами и квадратом третьей цифры.
5. Найти все двузначные числа, сумма квадратов цифр которых делится на 17.
6. Найти все трехзначные числа, представимые в виде сумм факториалов своих цифр.
7. Найти двузначное число, обладающее тем свойством, что куб суммы его цифр равен квадрату самого числа.
8. Найти двузначное число, равное утроенному произведению его цифр.
9. В каких двузначных числах удвоенная сумма цифр равна их произведению?
10. Можно ли заданное натуральное число M представить в виде суммы квадратов двух натуральных чисел? Написать программу решения этой задачи.

Вычисление выражений:

Дано натуральное n. Вычислить:

1. $(1+\frac{1}{1^2})+(1+\frac{1}{2^2})+(1+\frac{1}{3^2})+\ldots+(1+\frac{1}{n^2})$;
2. $\frac{1}{\sin 1}+\frac{1}{\sin 1+\sin 2}+\ldots+\frac{1}{\sin 1+\ldots \sin n}$;

Дано действительное число x, натуральное число n. Вычислить:

3. x (x - n)(x - 2 n)(x - 3 n)…(x - n2);
4. $\frac{1}{x}+\frac{1}{x(x+1)}+\ldots+\frac{1}{x(x+1)\ldots(x+n)}$;
5. $\frac{x^1}{1!}+\frac{x^2}{2!}+\frac{x^3}{3!}+\ldots+\frac{x^n}{n!}$;

Дано натуральное n. Вычислить:

6. $\prod_{i=1}^{n}(2+1/i!)$;
7. $\sum_{i=1}^{n}(1+i)/i!$;

Вычислить приближенно значение бесконечной суммы (справа от каждой суммы дается ее точное значение, с которым можно сравнить полученный ответ):

8. $1+\frac{1}{2^2}+\frac{1}{3^2}+\frac{1}{4^2}+\ldots=\frac{\pi^2}{6}$;
9. $\frac{1}{1\cdot 3}+\frac{1}{2\cdot 4}+\frac{1}{3\cdot 5}+\ldots=\frac{3}{4}$;
10. $1+\frac{x^1}{1!}+\frac{x^2}{2!}+\frac{x^3}{3!}+\ldots=e^x$;

Нужное приближение считается полученным, если вычислена сумма нескольких первых слагаемых, и очередное слагаемое оказалось по модулю меньше данного положительного числа e.

11. Даны два целых числа A и B (A < B). Вывести все целые числа, расположенные между данными числами (не включая сами эти числа), в порядке их возрастания, а также количество N этих чисел.
12. Даны два целых числа A и B (A < B). Вывести все целые числа, расположенные между данными числами (включая сами эти числа), в порядке их убывания, а также количество N этих чисел.
13. Дано вещественное число A и целое число N (> 0). Вывести A в степени N: AN = A·A·...·A (числа A перемножаются N раз).
14. Дано вещественное число A и целое число N (> 0). Вывести все целые степени числа A от 1 до N.
15. Дано вещественное число A и целое число N (> 0). Вывести 1 + A + A2 + A3 + ... + AN.
16. Дано вещественное число A и целое число N (> 0). Вывести 1 - A + A2 - A3 + ... + (-1)NAN.
17. Дано целое число N (> 1). Вывести наименьшее целое K, при котором выполняется неравенство 3K > N, и само значение 3K.
18. Дано целое число N (> 1). Вывести наибольшее целое K, при котором выполняется неравенство 3K < N, и само значение 3K.
19. Дано вещественное число A (> 1). Вывести наименьшее из целых чисел N, для которых сумма 1 + 1/2 + ... + 1/N будет больше A, и саму эту сумму.
20. Дано вещественное число A (> 1). Вывести наибольшее из целых чисел N, для которых сумма 1 + 1/2 + ... + 1/N будет меньше A, и саму эту сумму.