Примеры циклов пересчет

Оператор безусловного перехода

- Этот оператор позволяет перейти без проверки условия либо на один из предыдущих операторов, либо на один из последующих, т.е. изменить порядок выполнения команд.
- Общий вид оператора: **goto n**;
- Где n – целое число, не более чем из четырех цифр, называемое меткой.
- Метка появляется в программе три раза:
  1. В описательной части в разделе Label;
  2. В операторе **goto**;
  3. Перед оператором, на который осуществляется безусловный переход, в этом случае метка от оператора отделяется двоеточием.

Организация циклов с помощью операторов условного и безусловного переходов

- Пусть требуется вычислить наибольший общий делитель двух натуральных чисел A и **B**.
- Воспользуемся алгоритмом Евклида: будем уменьшать каждый раз большее из чисел на величину меньшего до тех пор, пока оба числа не станут равны.

Исходные данные	Первый шаг	Второй шаг	Третий шаг	НОД (А и В)=5
А=25	А=10	А=10	А=5	
В=15	В=15	В=5	В=5	

```
Program E5;
   label 1, 2;
    var  a, b: integer;
begin
    write (‘vvedite dva naturalnih chisla ‘);
    readln (a,b);
    1.  if  a=b then goto 2;
       if  a>b then a:=a-b else b:=b-a;
       goto 1;
    2. Write (‘NOD =‘,a);
        readln
        end.
```

Пример 1. Вычисление p=n! (n факториал)

- По определению n!=1*2*3*…*n.
- Используя программу вычисления степени, вычислим p как произведение чисел от 1 до n, т.е. p каждый раз умножается не на одно и то же число, а на значение переменной цикла.

```
Program E9;
    var  p, i , n: integer;
begin
    write (‘vvedite zeloe n= ‘);
    readln (n);
    p:=1;
    for i:=1 to n do
    p:=p*i;
    write (n,’!=‘,p);
    readln
end.
```

Пример 2. Составление таблицы значений функций y=sin x.

- Пусть требуется составить таблицу значений функций на отрезке [0;3.14] с шагом 0,1.
- Чтобы не определять количество повторений вычислений, можно воспользоваться циклом пока.
- Используя вывод вещественных чисел с фиксированной точкой, определим что количество цифр после запятой в значении функций будет равно 5.
- Тогда все число, учитывая область значений синуса, займет семь позиций (числа положительные, значит, добавится позиция для десятичной точки и целой части числа).

```
Program E10;
    var  x,y: real;
Begin
   x:=0;
    writeln ('x':10,'sin x':10);
    while x<=3.14 do
      begin
        y:=sin(x);
        writeln (x:10,' ',y:7:5);
        x:=x+0.1;
        end;
    readln
end.
```

- При каждом выполнении цикла будет сначала проверяться условие (x<=3.14), затем вычисляться значение функции, печататься аргумент x (для него отведено десять позиций, из них одна, для цифры дробной части) и, через три пробела, - значение функции.
- Наконец, для следующего шага цикла вычисляется новое значение аргумента (x увеличится на 0,1).
- Цикл пока позволяет изменять переменную цикла как угодно, увеличивая ее или уменьшая на любое число.

Пример 3. Суммирование чисел.

- При суммировании, как и при умножении нескольких чисел, необходимо накапливать результат в некоторой ячейке памяти, каждый раз считывая из этой ячейки предыдущее значение суммы и увеличивая его на очередное слагаемое.
- Пусть известно, что будет складываться **n** чисел.
- В этом случае надо **n** раз выполнить действие **s:=s+a**; здесь **a** – очередное число, вводимое с клавиатуры.
- Для первого выполнения этого оператора присваивания надо из ячейки с именем **s** взять такое число, которое не повлияло бы на результат сложения.
- Следовательно, прежде чем начать выполнять цикл, надо поместить в эту ячейку (или, что то же самое, присвоить переменной **s**) число нуль.

```
Program E11;
    var  a, s: real; i, n: integer;
begin
    write (‘vvedite kolichestvo slagaemih n= ‘);
    readln (n);
    s:=0;
       begin
         write(I,’-oe chislo =‘);
         readln (a);
         s:=s+a;
       end;
    write (‘summa s=‘,s);
    readln
end.
```

Если количество чисел неизвестно, то можно задать число – ограничитель, например нуль. В таком случае используется цикл **while** или **repeat**.

```
S:=0;
readln (a);
while a<>0 do
begin
   s:=s+a;
   readln (a);
end;
```

```
S:=0;
repeat
   readln (a);
   s:=s+a;
until a=0;
```

- Оператор цикла обратный пересчет работает аналогично оператору цикла прямого пересчета, только переменная цикла не возрастает с каждым шагом на единицу, а на единицу убывает.
- Оператор имеет вид:

  For i:=n2 downto n1 do оператор;
- Для этого оператора должно также выполняться условие n2>=n1.

При использовании в программе операторов цикла необходимо соблюдать следующие правила:

- Внутри цикла может находиться другой цикл, но необходимо, чтобы циклы имели разные переменные и внутренний цикл полностью находился в теле внешнего цикла;
- Нельзя передавать управление в тело цикла, минуя заголовок (это значит, что метка и оператор **goto** с этой меткой должны находиться в теле цикла);
- Если требуется обойти группу операторов в теле цикла и продолжить цикл, т.е. есть выполнить его следующий шаг, то надо передать управление на замыкающий цикл **end**;
- Можно досрочно выйти из цикла, или используя оператор goto, или изменив параметр условия в операторах **while** и **repeat** так, чтобы цикл больше не выполнялся.

- Проверить работу следующих программ:
  - E5
  - E9
  - E10
  - E11

Домашние задачи

- Напишите программы вычисления сумм:
  - сорока слагаемых вида **n-i**, где **i**=1,2,3,…,40, а **n** – данное число;
  - n – слагаемых вида **x+i**, где x– данное число, а i меняется от 1 до **n**.