Циклы в Паскале

При решении задач может возникнуть необходимость повторить одни и те же действия несколько или множество раз. В программировании блоки кода, которые требуется повторять не единожды, оборачиваются в специальные конструкции – *циклы*. У циклов выделяют заголовок и тело. Заголовок определяет, до каких пор или сколько раз тело цикла будет выполняться. Тело содержит выражения, которые выполняются, если в заголовке цикла выражение вернуло логическую истину (True, не ноль). После того как достигнута последняя инструкция тела, поток выполнения снова возвращается к заголовку цикла. Снова проверяется условие выполнения цикла. В зависимости от результата тело цикла либо повторяется, либо поток выполнения переходит к следующему выражению после всего цикла.

В языке программирования Паскаль существует три вида циклических конструкций.

Цикл for

Часто цикл for называют циклом со счетчиком. Этот цикл используется, когда число повторений не связано с тем, что происходит в теле цикла. Т.е. количество повторений может быть вычислено заранее (хотя оно не вычисляется).

В заголовке цикла указываются два значения. Первое значение присваивается так называемой переменной-счетчику, от этого значения начинается отсчет количества итераций (повторений). Отсчет идет всегда с шагом равным единице. Второе значение указывает, при каком значении счетчика цикл должен остановиться. Другими словами, количество итераций цикла определяется разностью между вторым и первым значением плюс

единица. В Pascal тело цикла не должно содержать выражений, изменяющих счетчик.

Цикл for существует в двух формах:

```
for счетчик:=значение to конечное_значение do
    тело_цикла;
for счетчик:=значение downto конечное_значение do
    тело_цикла;
```

Счетчик – это переменная любого из перечисляемых типов (целого, булевого, символьного, диапазонного, перечисления). Начальные и конечные значения могут быть представлены не только значениями, но и выражениями, возвращающими совместимые с типом счетчика типы данных. Если между начальным и конечным выражением указано служебное слово **to**, то на каждом шаге цикла значение параметра будет увеличиваться на единицу. Если же указано **downto**, то значение параметра будет уменьшаться на единицу.

Количество итераций цикла **for** известно именно до его выполнения, но не до выполнения всей программы. Так в примере ниже, количество выполнений цикла определяется пользователем. Значение присваивается переменной, а затем используется в заголовке цикла. Но когда оно используется, циклу уже точно известно, сколько раз надо выполниться.

```
var
   i, n: integer;

begin
   write ('Количество знаков: ');
   readln (n);

   for i := 1 to n do
      write ('(*) ');

readln
end.
```

Цикл while

Цикл **while** является циклом с предусловием. В заголовке цикла находится логическое выражение. Если оно возвращает **true**, то тело цикла выполняется, если **false** – то нет.

Когда тело цикла было выполнено, то ход программы снова возвращается в заголовок цикла. Условие выполнения тела снова проверяется (находится значение логического выражения). Тело цикла выполнится столько раз, сколько раз логическое выражение вернет **true**. Поэтому очень важно в теле цикла предусмотреть изменение переменной, фигурирующей в заголовке цикла, таким образом, чтобы когда-нибудь обязательно наступала ситуация **false**. Иначе произойдет так называемое **зацикливание**, одна из самых неприятных ошибок в программировании.

```
var
   i, n: integer;

begin
   write ('Количество знаков: ');
   readln (n);

   i := 1;
   while i <= n do begin
      write ('(*) ');
      i := i + 1
   end;

readln
end.
```

Цикл repeat

Цикл **while** может не выполниться ни разу, если логическое выражение в заголовке сразу вернуло **false**. Однако такая ситуация не всегда может быть приемлемой. Бывает, что тело цикла должно выполниться хотя бы один раз, не зависимо оттого, что вернет логическое выражение. В таком случае используется цикл repeat – цикл с постусловием.

В цикле **repeat** логическое выражение стоит после тела цикла. Причем, в отличие от цикла **while**, здесь всё наоборот: в случае **true** происходит выход из цикла, в случае **false** – его повторение.

```
var
   i, n: integer;

begin
   write ('Количество знаков: ');
   readln (n);

   i := 1;
```

```
   repeat
      write ('(*) ');
      i := i + 1
   until i > n;

readln
end.
```

В примере, даже если n будет равно 0, одна звездочка все равно будет напечатана.
Задачи:

1. Найти сумму чисел от 1 до n.

2. Вводится натуральное число. Найти сумму четных цифр, входящих в его состав.
3. Составить таблицу значений функции y = 5 - x2/2 на отрезке [-5; 5] с шагом 0.5.
4. Вывести на экран все простые числа до заданного числа n.
5. Вывести на экран первых n простых чисел.