

*** Операторы
организации циклов.
Простой и составной
операторы**

Лекция 6

В программах, связанных с обработкой данных или другими сложными вычислениями, часто выполняются *циклически повторяющиеся действия*.

Цикл представляет собой последовательность операторов, которая выполняется *неоднократно*.

Если заранее известно количество необходимых повторений, то цикл называется *арифметическим*. Если же количество повторений заранее неизвестно, то говорят об *итерационном* цикле.

В языке программирования Pascal имеется стандартный набор из трех разновидностей цикла :

• *цикл с постусловием (инструкция repeat),*

• *цикл с предусловием (инструкция while)*

• *цикл со счетчиком (инструкция for).*

• подавляющее большинство задач с циклами можно решить используя при этом любой из трех операторов цикла;

• *самым универсальным* из всех операторов цикла считается **while**, поэтому в случае затруднений с выбором можно отдать предпочтение ему;

• цикл **for** обеспечивает удобную запись циклов с *заранее известным числом повторений*;

• при неумелом использовании циклов любого типа возможна ситуация, когда компьютер не сможет нормально закончить цикл (в таком случае говорят, что программа "зациклилась").

* Цикл с параметром (со счетчиком, с заданным числом повторений)

В нем важную роль играет *переменная-параметр*, которая на каждом шаге цикла автоматически изменяет свое значение ровно на *единицу* — поэтому ее и называют *счетчиком*.

Инструкцию `for` можно реализовать двумя способами.

Вариант 1 (с увеличением счетчика):

```
for Счетчик:=НачальноеЗначение to КонечноеЗначение  
do Инструкции ;
```

* Описание выполнения оператора for

Если НачальноеЗначение больше, чем КонечноеЗначение, то инструкции не выполняются ни разу, а значение переменной Счетчик не определено, в противном случае:

1. Переменной Счетчик присваивается НачальноеЗначение.
2. Проверятся, если переменная Счетчик не больше КонечногоЗначения, то выполняются Инструкции, стоящие после служебного слова do, в противном случае переход к п. 4.
3. Переменная Счетчик увеличивает свое значение на единицу (переходит к следующему значению). Переход к п. 2.
4. Цикл завершает свою работу и управление передается оператору, следующему за оператором for.

Строка, содержащая **for...do**, называется *заголовком цикла*, оператор, стоящий после **do**, образует *тело цикла*. Очень часто тело цикла — *составной оператор*.

Инструкции выполняются **[(КонечноеЗначение - НачальноеЗначение) + 1]** число раз.

Это соответствует всем значениям счетчика от начального до конечного включительно.

Например, выполнение цикла

```
for i:=10 to 14 do write(i: 3);
```

выведет на экран последовательность чисел в виде:

```
10 11 12 13 14
```

Вариант 2 (с уменьшением счетчика):

**for Счетчик:=НачальноеЗначение downto
КонечноеЗначение do Инструкции ;**

Инструкции выполняются [(НачальноеЗначение -
КонечноеЗначение) + 1] число раз.

Если НачальноеЗначение меньше, чем
КонечноеЗначение, то инструкции не выполняются ни
разу.

Например, выполнение цикла

```
for i:=14 downto 10 do write(i: 3);
```

выведет на экран последовательность цифр: 14 13 12 11 10.

Если переменная-счетчик имеет символьный тип `char`, то
оператор `for ch: = 'a' to 'e' do write (ch: 2) ;`

выведет на экран : a b c d e

а оператор `for ch:= 'e' downto 'a' do write (ch: 2) ;`

выведет на экран : e d c b a

- оператор `for` используется для организации циклов с *фиксированным*, заранее известным или определяемым во время выполнения программы числом повторений;
- **переменная-счетчик должна быть порядкового типа.** *Использование вещественного типа недопустимо;*
- *начальное и конечное значения параметра цикла могут быть константами, переменными, выражениями и должны принадлежать к одному и тому же типу данных. Начальное и конечное значения параметра цикла нельзя изменять во время выполнения цикла;*
- после нормального завершения оператора `for` значение параметра цикла равно конечному значению (в некоторых компиляторах (Delphi) – на единицу больше).
- *параметр цикла `for` может изменяться (увеличиваться или уменьшаться) каждый раз при выполнении тела цикла только на единицу. Если нужен другой шаг изменения*

* Составной оператор

Составной оператор - это последовательность произвольных операторов программы, заключенная в операторные скобки - зарезервированные слова **begin . . .end.**

Паскаль не накладывает никаких ограничений на характер операторов, входящих в составной оператор. Среди них могут быть и другие составные операторы - допускается произвольная глубина их вложенности.

Фактически, весь раздел операторов, обрамленный словами **begin...end**, представляет собой один составной оператор.

* Пустой оператор

Поскольку зарезервированное слово `end` является закрывающей операторной скобкой, оно одновременно указывает и конец предыдущего поэтому ставить перед ним символ “;” необязательно. Наличие точки с запятой перед `end` означает, что между последним оператором и операторной скобкой `end` располагается *пустой оператор*.

Пустой оператор не содержит никаких действий, просто в программу добавляется лишняя точка с запятой. В основном пустой оператор используется для передачи управления в конец составного оператора.

Пример. Пользователь вводит целое положительное число. Определить сумму последовательно расположенных натуральных чисел от 1 до введенного числа.

```
var a,s,i:integer;  
begin  
writeln('Введите число');  
readln(a);  
s:=0;  
for i:=1 to a do  
    s:=s+i;  
writeln('Сумма равна', s);  
readln;  
end.
```

Вычислите сумму $1 + \frac{1}{2^2} + \frac{1}{3^2} + \dots + \frac{1}{n^2}$.

n вводит

```
var a,i:integer; s,k:real;
begin
writeln('Введите число');
readln(a);
s:=0;
for i:=1 to a do
begin
k:=1/(sqr(i));
s:=s+k;
end;
writeln('Сумма равна', s:8:3);
readln;
end.
```

* Операторы завершения цикла

Для всех операторов цикла выход из цикла осуществляется как вследствие естественного окончания оператора цикла, так и с помощью операторов перехода и выхода.

В Паскале определены стандартные процедуры:

- Break
- Continue

Процедура Break выполняет безусловный выход из цикла.

Процедура Continue обеспечивает переход к началу новой итерации цикла.

Пример. Пользователь вводит натуральное положительное число. Определить первое следующее за ним число, кратное 11.

```
var a,i:integer;
```

```
begin
```

```
writeln('Введите число');
```

```
readln(a);
```

```
for i:=a+1 to a+11 do
```

```
    if i mod 11=0 then break;
```

```
writeln(i);
```

```
readln;
```

```
end.
```

Пример. Найти сумму квадратов всех натуральных чисел от 1 до введенного пользователем числа, исключая числа, кратные 3.

```
var a,i,s:integer;
begin
writeln('Введите число ');
readln(a);
s:=0;
for i:=1 to a do
begin
if i mod 3=0 then continue;
s:=s+i*i;
end;
writeln(s);
readln;
end.
```

* Вложенные циклы

В теле любого оператора цикла могут находиться другие операторы цикла. При этом цикл, содержащий в себе другой, называют *внешним*, а цикл, находящийся в теле первого, — *внутренним (вложенным)*. Правила организации внешнего и внутреннего циклов такие же, как и для простого цикла.

Обратите внимание — при программировании вложенных циклов необходимо соблюдать следующее дополнительное условие: *все операторы внутреннего цикла должны полностью располагаться в теле внешнего цикла.*

При вложении циклов внутренний цикл выполняется *полностью* от начального до конечного значения параметра при *неизменном* значении параметра *внешнего* цикла. Затем значение параметра *внешнего* цикла изменяется на *единицу*, и опять от начала и до конца выполняется *вложенный* цикл. И так до тех пор, пока значение параметра внешнего цикла не станет больше конечного значения, определенного в операторе `for` внешнего цикла.

Рассмотрим задачу вывода на экран таблицы умножения, решение которой предполагает использование вложенных циклов. С использованием цикла **for** соответствующий фрагмент программы имеет вид:

```
Var
```

```
i, j : byte;
```

```
begin
```

```
for i:=1 to 9 do
```

```
begin
```

```
  for j:=1 to 9 do write(i, '*', j, ' = ', i*j:2, '  ');
```

```
  writeln; {перевод строки – во внешнем цикле}
```

```
end;
```

```
end.
```

* Домашнее задание

1. Составить опорный конспект лекции по теме «Операторы организации циклов. Простой и составной операторы» на основе презентации.
2. Программирование на языке Pascal. Рапаков Г. Г., Ржеуцкая С. Ю. СПб.: БХВ-Петербург, 2004, с.132-138
3. Составить программы:
 - Составьте программу вычисления суммы всех двузначных чисел.
 - Найдите сумму членов ряда $S = 1 + 1/2 + 1/4 + 1/8 + 1/(2^n)$, n вводит пользователь.