Глава 2. Программирование

§17. Программирование циклических алгоритмов

Цикл с условием

Практическая работа 12. Циклы с условием

Как организовать цикл?

Допустим, мы хотим вывести 5 раз на экран слово «привет». Можно, конечно, записать 5 одинаковых команд:

вывод 'привет', нс	**writeln('привет');**
вывод 'привет', нс	**writeln('привет');**
вывод 'привет', нс	**writeln('привет');**
вывод 'привет', нс	**writeln('привет');**
вывод 'привет', нс	**writeln('привет');**

Но что если нужно будет сделать какие-то действия 1000 или 1000000 раз?

Как организовать цикл?

Простейший цикл, нужный нам в этой задаче, в алгоритмическом языке записывается так:

```
нц 5 раз
  вывод 'привет', нс
кц
```

В Паскале и во многих других языках записать цикл в такой форме нельзя, однако можно легко запрограммировать те же действия немного по-другому.

Как организовать цикл?

Программа выполняется автоматически. И при этом на каждом шаге нужно знать, сколько раз уже выполнен цикл и сколько ещё осталось выполнить. Для этого необходимо использовать ячейку памяти (переменную).

Такая переменная часто называется *счётчиком*.

Как организовать цикл?

Сначала в переменную-счётчик записывают ноль (ни одного шага не сделано), а после каждого шага цикла увеличивают значение на единицу:

```
счётчик:= 0
нц пока счётчик < 5
  вывод 'привет', нс
  счётчик:= счётчик + 1
кц
```

```
count:= 0;
while count < 5 do begin
  writeln('привет');
  count:= count + 1
end;
```

Другой вариант: сразу записать в счётчик нужное количество шагов, и после каждого шага цикла *уменьшать* счётчик на 1. Тогда цикл должен закончиться при нулевом значении счётчика.

Циклы с предусловием

Цикл, в котором проверка условия выполняется при входе *(перед выполнением очередного шага)* называется **циклом с предусловием**, то есть циклом с предварительной проверкой условия.

У них есть два важных свойства:

- условие проверяется при входе в цикл, поэтому цикл не выполнится ни разу, если условие в самом начале ложно;
- как только нарушается условие в заголовке цикла, его работа заканчивается.

Циклы с предусловием

Требуется ввести с клавиатуры натуральное число и найти сумму цифр его десятичной записи. Например, если ввели число 123, программа должна вывести сумму 1+2+3 = 6.

Сначала составим алгоритм решения этой задачи.

Предположим, что число записано в переменной *N*. Нам нужно как-то разбить число на отдельные цифры.

Циклы с предусловием

Остаток от деления числа на 10 равен последней цифре его десятичной записи:

d:= mod(N, 10)	**d:= N mod 10;**

Эту цифру числа нужно добавить к сумме всех цифр, которые мы уже обработали раньше **sum**:

цел sum **sum:= 0**	**var sum: integer;** **...** **sum:= 0;**

В самом начале значение этой переменной равно нулю.

Циклы с предусловием

Для того чтобы добавить к предыдущей сумме новую цифру, нужно заменить значение переменной *sum* на *sum*+*d*, то есть выполнить присваивание

sum:= sum + d	**sum:= sum + d;**

Для того чтобы затем отсечь последнюю цифру числа *N*, разделим *N* на 10 (основание системы счисления):

N:= div(N, 10)	**N:= N div 10;**

Циклы с предусловием

Эти три операции –нужно выполнять несколько раз, пока все цифры не будут обработаны и в переменной *N* не останется ноль:

```
цел N, d, sum
вывод 'Введите число: '
ввод N
sum:= 0
нц пока N <> 0
  d:= mod(N, 10)
  sum:= sum + d
  N:= div(N, 10)
кц
вывод 'Сумма цифр ', sum
```

```
var N, d, sum: integer;
begin
  write('Введите число: ');
  read(N);
  sum:= 0;
  while N <> 0 do begin
    d:= N mod 10;
    sum:= sum + d;
    N:= N div 10;
  end;
  writeln('Сумма цифр ', sum);
end.
```

Циклы с предусловием

Эти три операции –нужно выполнять несколько раз, пока все цифры не будут обработаны и в переменной *N* не останется ноль:

```
цел N, d, sum
вывод 'Введите число: '
ввод N
sum:= 0
нц пока N <> 0
  d:= mod(N, 10)
  sum:= sum + d
  N:= div(N, 10)
кц
вывод 'Сумма цифр ', sum
```

```
var N, d, sum: integer;
begin
  write('Введите число: ');
  read(N);
  sum:= 0;
  while N <> 0 do begin
    d:= N mod 10;
    sum:= sum + d;
    N:= N div 10
  end;
  writeln('Сумма цифр ', sum)
end.
```

Циклы с постусловием

Во многих языках программирования существует **цикл с постусловием**, в котором условие проверяется не до, а *после* завершения очередного шага цикла.

Это полезно в том случае, когда нужно обязательно выполнить цикл хотя бы один раз.

Циклы с постусловием

Пользователь должен ввести с клавиатуры положительное число. Для того чтобы защитить программу от неверных входных данных, можно использовать цикл с постусловием:

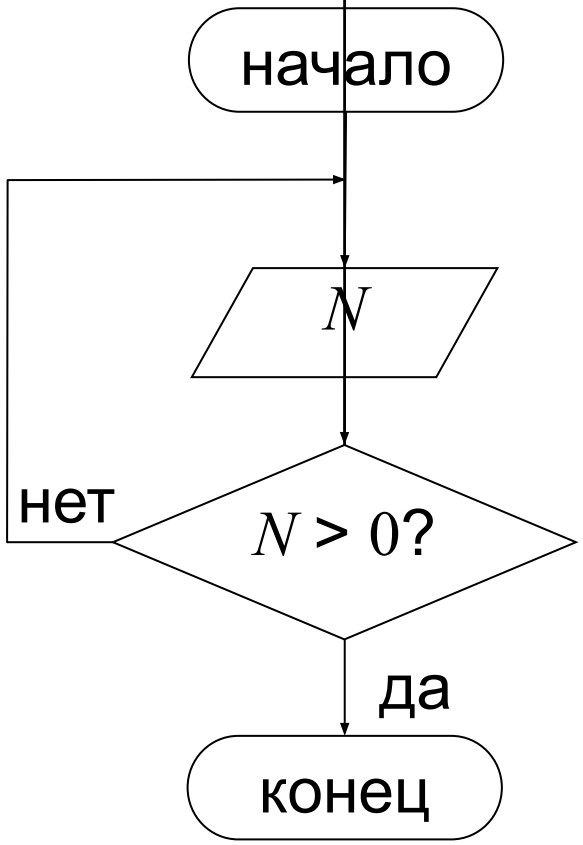
```
нц
  вывод 'Введите N>0: '
  ввод N
кц при N > 0
```

```
repeat
  write('Введите N>0: ');
  read(N);
until N > 0;
```

Этот цикл закончится тогда, когда станет *истинным* условие *N* **> 0** в последней строке, то есть тогда, когда пользователь введет допустимое значение.

Циклы с постусловием

На блок-схеме этого алгоритма видно, что проверка условия выполняется после завершения очередного шага цикла.



- при входе в цикл условие не проверяется, поэтому цикл всегда выполняется хотя бы один раз;
- в последней строке указывают условие окончания цикла.

Цикл по переменной

Практическая работа 16. Цикл по переменной

Циклы по переменной

В информатике важную роль играют степени числа 2 (2, 4, 8, 16 и т.д.).

Давайте выведем на экран все степени двойки от 2^1 до 2^{10}. Для решения этой задачи мы уже можем написать такую программу, использующую цикл с условием:

```
 k:=1
N:=2
нц пока  k <= 10
 вывод N, нс
 N:=N*2
 k:=k+1
кц
```

```
 k:=1;
N:=2;
while  k <= 10  do begin
 writeln(N);
 N:=N*2;
 k:=k+1
end;
```

Циклы по переменной

Переменная **k** используется трижды!

Чтобы собрать все действия с ней в один оператор, во многие языки программирования введен особый вид цикла – *цикл по переменной*.

В заголовке этого цикла задается начальное и конечное значение этой переменной, а шаг её изменения по умолчанию равен 1:

```
N:=2
нц для  k от 1 до 10
  вывод N, нс
  N:=N*2
кц
```

```
N:=2;
for  k:=1 to 10  do begin
  writeln(N)
  N:=N*2;
end;
```

Циклы по переменной

Найдём сумму чисел от 1 до 1000.

Для накопления суммы будем использовать переменную (назовём её **sum**).

В цикле другая переменная (скажем, **i**) изменяется от 1 до 1000, и на каждом шаге этого цикла к сумме добавляется очередное число:

```
цел sum, i
sum:=0
нц для i от 1 до 1000
  sum:=sum+i
кц
```

```
var sum, i: integer;
...
sum:=0;
for i:=1 to 1000 do
  sum:=sum+i;
```

Циклы по переменной

С каждым шагом цикла переменная цикла может не только увеличиваться, но и уменьшаться на 1.

Для этого в алгоритмическом языке добавляется параметр **шаг**, а в Паскале ключевое слово **to** заменяется на **downto** («вниз до…»).

Следующая программа печатает квадраты натуральных чисел от 10 до 1 в порядке убывания:

```
нц для k от 10 до 1 шаг -1
  вывод k*k, нс
кц
```

```
for k:=10 downto 1 do
  writeln(k*k);
```

Выводы:

- С помощью циклов в программе можно выполнять повторяющиеся действия.
- Различают два вида циклов: циклы с условием и циклы по переменной.
- Цикл с условием выполняется до тех пор, пока некоторое условие (условие продолжения работы цикла) не станет ложным. Если это условие никогда не станет ложным, программа зацикливается.
- В некоторых языках программирования есть циклы, в запись которых содержит не условие продолжения, а условие выхода из цикла.
- Проверка условия может происходить перед выполнением очередного шага цикла (в циклах с предусловием) или после него (в циклах с постусловием). Цикл с предусловием может не выполняться ни разу, а цикл с постусловием всегда выполняется хотя бы один раз.
- Цикл по переменной применяют тогда, тогда количество шагов цикла заранее известно или может быть вычислено до начала цикла. В заголовке цикла по переменной указывают начальное значение, конечное значение и шаг изменения переменной цикла.
- Модифицированный алгоритм Евклида для вычисления НОД двух натуральных чисел: заменять бóльшее из чисел на остаток от деления бóльшего на меньшее, пока этот остаток не станет равен нулю. Тогда второе число и есть их НОД.

Интеллект-карта