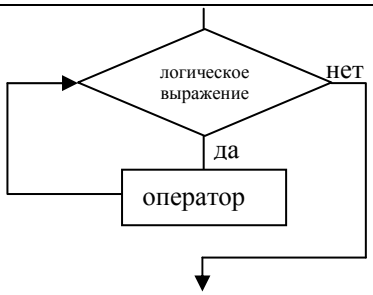
Циклы

Циклом называют такую форму организации действий, при которой одна и та же последовательность действий выполняется несколько раз или ни разу в зависимости от некоторого условия. Саму многократно выполняемую последовательность действий называют телом цикла.
В языке Паскаль есть три оператора цикла.

Цикл с предусловием

Блок-схема	Паскаль
	**While** логическое выражение **Do** оператор

Оператор While используют для выполнения оператора тела цикла до тех пор, пока логическое выражение в заголовке цикла остаётся истинным. Значение этого выражения вычисляется до выполнения оператора. Оператор тела цикла может быть простым или составным. Если логическое выражение изначально ложно, то оператор тела цикла не выполнится ни разу.

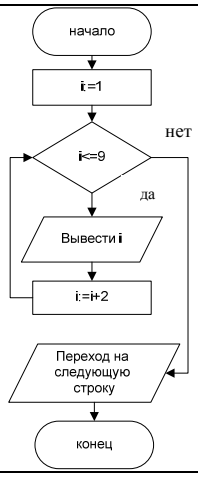
Примеры применения цикла с предусловием

```
While x>=1e-3 Do
x:=x/2
```

В этом примере пока значение переменной *x* больше или равно 10^{-3} значение *x* уменьшается в два раза.

В следующем примере выводятся в одну строку нечётные числа от 1 до 9 включительно, а потом курсор переводится на новую строку.

Блок-схема	Программа

	`Program TestWh;`
	`Var i : Byte;`
	`Begin`
	`  i:=1;`
	`  While i<=9 Do`
	`  Begin`
	`  Write(i, ' ');`
	`  I:=i+2`
	`  End;`
	`  WriteLn`
	`End.`

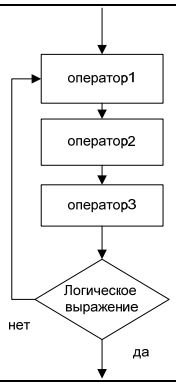
Вызов WriteLn без параметров приводит к переводу курсора на новую строку.

Обратите внимание на недопустимость использования точки с запятой после Do. Если бы мы написали

```
i:=1;
While i<=9 Do;
```

то наша программа просто бы зациклилась: в качестве оператора тела цикла рассматривался бы пустой оператор, обозначаемый просто точкой с запятой, логическое выражение i<=9 было бы всегда истинным, и выполнение оператора цикла продолжалось бы бесконечно долго. Точнее, пока она не будет завершена принудительно, например, с помощью диспетчера задач.

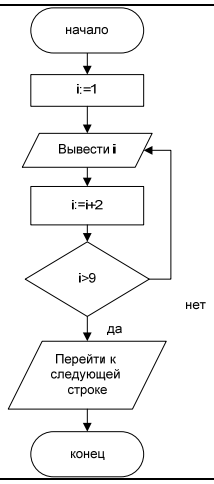
Цикл с постусловием

Блок-схема	Паскаль
	**Repeat** оператор1; оператор2; оператор3 **Until** логическое выражение

Сначала выполняются операторы тела цикла, т.е. операторы, написанные между Repeat и Until. Потом вычисляется значение логического выражения. Если оно равно false, то процесс повторяется. Таким образом, истинность логического выражения означает, что выполнение цикла будет завершено. В отличие от цикла с предусловием, операторы тела цикла выполняются хотя бы по одному разу. В отличие от других операторов языка Паскаль, между Repeat и Until можно написать любое количество операторов, не прибегая к begin и end. У последнего перед Until оператора точка с запятой не пишется.

Ниже приводится пример только что решавшейся задачи вывода на экран нечётных чисел от 1 до 9 включительно, но с использованием оператора цикла с постусловием.

Блок-схема	Программа

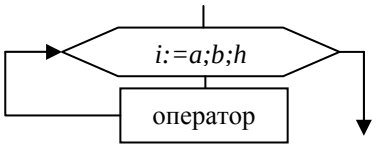
	`Program TestWh2;`
	`Var i : Byte;`
	`Begin`
	`  i:=1;`
	`  Repeat`
	`  Write(i, ' ');`
	`  i:=i+2`
	`  Until i>9;`
	`  WriteLn`
	`End.`

Цикл с параметром

Этот вид оператора цикла применяется, когда заранее известно количество повторений выполнения операторов тела цикла. Исполнением оператора управляет переменная одного из порядковых типов, называемая параметром цикла (в примере ниже это переменная *i*). Задаются её начальное и конечное значения (*a* и *b*). Цикл выполняется следующим образом.

1. Вычисляется конечное значение параметра цикла.
2. Параметр цикла получает начальное значение.
3. Сравнивается значение параметра с конечным значением. Если значение параметра находится между начальным и конечным значением включительно, то выполняется оператор тела цикла. (Это оператор, стоящий после Do. Он может быть простым или составным.) В противном случае выполнение оператора завершается.
4. Значение параметра цикла увеличивается или уменьшается на 1, в зависимости от того, какая форма оператора: с To или DownTo — используется.
5. Переход к п. 3.

Блок-схема	Паскаль


Здесь *i* — параметр цикла, *a* — начальное значение параметра, *b* — конечное значение, *h* — шаг изменения параметра.
	При *h* = 1
	For *i*:=*a* **To** *b* **Do**
	оператор
	
	При *h* = –1
	For *i*:=*a* **DownTo** *b* **Do**
	оператор
	
	Переменная *i* (параметр цикла) не может быть вещественного типа.

Запрещено изменять значение параметра цикла операторами тела цикла. Например, следующий код сгенерирует ошибку.

```
For i:=1 To N Do
Begin
  Write(i);
  i:=i*2      (* Этого делать нельзя! *)
End
```

А это пример правильного использования оператора For:

```
For k:=1 To 9 Do
WriteLn('5 * ',k, ' = ',5*k)
```

В этом примере выводилась таблица умножения на 5.

Использование break

Зарезервированное слово *break* используется для выхода из цикла. Если *break* выполняется в теле цикла, исполнение цикла сразу же прекращается и работа программы продолжается с оператора, следующего за телом цикла. *Break* можно использовать с любым из трёх операторов цикла.

Использование continue

Зарезервированное слово *continue* использование для перехода к следующей итерации цикла. После выполнения *continue* все операторы тела цикла, следующие после *continue*, пропускаются, и выполнение тела цикла начинается с первого оператора. Если этот оператор встречается в цикле *For*, то после выполнения *continue* перед началом следующей итерации изменяется значение параметра цикла.

Следующий пример иллюстрирует применение оператора continue.

```
Program Sample;
Var i   : ShortInt;
    N   : Integer;
    x   : Real;
```

```
Begin
  Write('N=');
  Read(N);
  For i:=-10 To 10 Do
  Begin
    If i=0 Then
    continue;
    x:=N/i;
    WriteLn(N,' /',i:3,'=',x:7:3)
  End
End.
```

Здесь введённое целое число N последовательно делится на все целые числа от -10 до 10 включительно, исключая ноль. Результат округляется до тысячных, т.е. трёх знаков после запятой. Для вывода делителя используется 3 знакоместа.

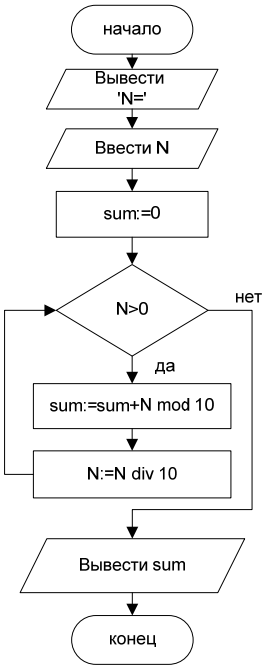
Примеры решения задач

Задача 1.

Дано натуральное десятичное число N, не превышающее двух миллиардов. Найти сумму его цифр.

Решение.

Для хранения суммы цифр заводим переменную sum. Чтобы найти последнюю цифру числа, достаточно вычислить остаток от его деления на 10. Если же найти неполное частное от деления на 10, то получим число без последней цифры. Сумму следует накапливать до тех пор, пока исходное число не превратится в ноль. Ниже приведена блок-схема алгоритма.



Для правильного выбора типов данных определим диапазон допустимых значений переменных. Заметим, что в этой задаче $N \in [0; 2 \cdot 10^6]$. Ноль сюда включен потому, что хотя исходное значение N не может быть меньше единицы (N по условию задачи — натуральное число), конечное значение N при выходе из цикла равно нулю. Кроме того, число N целое. Значит, для его хранения подойдёт переменная типа Integer или LongInt, поскольку в Delphi это одно и то же. Если вы будете писать на FreePascal, не забудьте в первой строке написать {$mode Delphi}, или использовать тип LongInt. Сумма цифр семизначного числа не может превышать 9*7=63. В нашем случае получится даже меньше, поскольку первая цифра будет не больше двух. Следовательно, для переменной sum выбираем тип Byte или ShortInt: оба они однобайтовые, а диапазон изменения включает отрезок [0;63].

Получаем программу на FreePascal.

```
{$mode delphi}
Program SumDigit;
Var sum : Byte;
    N   : Integer;
Begin
  Write('N=');
  Read(N);
  sum:=0;
  While N>0 Do
  Begin
    sum := sum + N mod 10;
    N := N div 10
  End;
  WriteLn(sum)
End.
```

Верхняя строка этой программы в Delphi не нужна.

Задача 2

Найти все делители натурального числа *N*, не превышающего 4 миллиардов.

Решение

Можно было перебирать все натуральные i числа от 1 до N и находить остаток от деления N на i. Если он равен нулю, то i — искомый делитель. Однако, такое решение будет работать слишком медленно. Для его ускорения можно заметить, что на промежутке $(\frac{N}{2}, N]$ существует только один делитель числа N: само N. Действительно, при делении числа самого на себя получается единица, при делении числа на его половину получается 2, а между единицей и двойкой натуральных чисел нет. Но и этот алгоритм можно улучшить.
Учтём, что умножение подчиняется переместительному закону. Следовательно, если i является делителем числа N, то $\frac{N}{i}$ — тоже делитель N. Делители находятся парами. Причём, если i увеличивать, начиная с 1, то сначала $i < \frac{N}{i}$. После $i = \sqrt{N}$ начинается повторение тех же пар чисел, только теперь $i > \frac{N}{i}$. Само число $\sqrt{N}$, если оно натуральное, тоже будет делителем N, но пары у него не будет. Этот алгоритм лежит в основе следующей программы на FreePascal.

```
{$mode delphi}
Program Divisor;
Var i, N, last : LongWord;
    sq         : Boolean;
Begin
  Write('N=');
  Read(N);
  last := Trunc(sqrt(N)); (* Обрезаем дробную часть корня из N *)
  sq := last*last = N; (* Является ли квадратный корень целым числом? *)
  If sq Then
    last:=last-1;
  For i:=1 To last Do
   If N mod i=0 Then
    Write(i,' ',N div i,' '); (*Делители выводятся парами *)
  If sq Then
    Write(N div (last+1)); (* У этого делителя пары нет *)
  WriteLn
End.
```

Задачи для самостоятельного решения

Напишите программы решения задач.

1. Дано натуральное число N. Найти количество цифр в его десятичной записи. Лидирующие нули не учитывать. В записи числа может содержаться до 18 цифр.
2. Дано действительное число x и целое неотрицательное n. Вычислите x^n.
3. С клавиатуры последовательно вводится N целых чисел. Определить их среднее арифметическое. Все числа, включая N, по модулю не превосходят 30000.