НАУКА В РЕГИОНЫ

Методические материалы по информатике (циклы)

Фонд развития Физтех-лицея и Физтех-школ

Иннопрактика

2017

Автор
В.В. Мерзляков

Информатика

Корректоры: И.В. Вовченко
Компьютерная верстка: И.В. Вовченко

Введение

Определение: Цикл – это повторение фрагмента алгоритма несколько раз с возвратом в более раннюю точку исполнения алгоритма. Повторяемый фрагмент алгоритма называется телом цикла.

Очевидно, что если возврат в более раннюю точку алгоритма происходит абсолютно всегда, то алгоритм никогда не доходит до своего завершения (выполняется бесконечно долго). Для предотвращения бесконечного повторения нужно поставить точку ветвления: в одной из веток будет возврат, а в другой – выход из цикла и дальнейшее продвижение по алгоритму.

В зависимости от положения точки ветвления выделяются циклы с предусловием (точка ветвления располагается перед телом цикла) и с постусловием (точка ветвления располагается после тела цикла).

Рассмотрим несколько примеров простых исполнителей алгоритмов и реализации их циклов.

Исполнитель 1: *«Редактор»*

Исполнитель *«Редактор»* получает на вход строку цифр и преобразовывает её. *«Редактор»* может выполнять две команды, в обеих командах v и w обозначают цепочки

цифр.

А) заменить (v, w).

Эта команда заменяет в строке первое слева вхождение цепочки v на цепочку w. Например, выполнение команды заменить $(111, 27)$ преобразует строку 05111150 в строку 0527150.

Если в строке нет вхождений цепочки v, то выполнение команды заменить (v, w) не меняет эту строку.

Б) нашлось (v).

Эта команда проверяет, встречается ли цепочка v в строке исполнителя *Редактор*. Если она встречается, то команда возвращает логическое значение «истина», в противном случае возвращает значение «ложь». Строка исполнителя при этом не изменяется. Следующий цикл выполняется, пока условие истинно.

```
ПОКА условие
  последовательность команд
ПОКА
```

Рассмотрим конструкцию

```
ЕСЛИ условие
  ТО команда_1
  ИНАЧЕ команда_2
КОНЕЦ ЕСЛИ
```

В ходе процесса, будет выполняться `команда_1` (если условие истинно) или `команда_2` (если условие ложно).

Рассмотрим примеры задач для этого исполнителя.

Задача 1

Дана программа:

```
ПОКА нашлось (222)
  Заменить (222, 3)
КОНЕЦ ПОКА
```

Сколько будет осуществлено входов в цикл, и каков будет результат работы программы, если изначально строка имеет следующий вид?

- 222222222 (3 входа, результат 333)
- 2222222 (2 входа, результат 332)
- 39829232221 (1 вход, результат 398292331)
- 29891102 (0 входов, результат 29891102)

Задача 2

Напишите программу, которая заменит во входной строке все чётные цифры на единички.

Решение:

В этой задаче нам потребуется написать пять последовательных операторов цикла – по одному на каждую чётную цифру.

```
ПОКА нашлось(0)
  Заменить (0, 1)
КОНЕЦ ПОКА
ПОКА нашлось(2)
  Заменить (2, 1)
КОНЕЦ ПОКА
ПОКА нашлось(4)
  Заменить (4, 1)
```

```
КОНЕЦ ПОКА
ПОКА нашлось(6)
  Заменить (6, 1)
КОНЕЦ ПОКА
ПОКА нашлось(8)
  Заменить (8, 1)
КОНЕЦ ПОКА
```

Задача 3

Какая строка получится в результате применения приведённой ниже программы к строке, состоящей из цифры 1, за которой следуют 80 идущих подряд цифр 8?

Решение:

```
ПОКА нашлось(18) ИЛИ нашлось(288) ИЛИ нашлось(3888)
  ЕСЛИ нашлось(18)
    ТО заменить(18, 2)
    ИНАЧЕ ЕСЛИ нашлось(288)
        ТО заменить(288, 3)
        ИНАЧЕ заменить(3888, 1)
    КОНЕЦ ЕСЛИ
  КОНЕЦ ЕСЛИ
КОНЕЦ ПОКА
```

Промоделируем работу этой программы. У нас происходит постоянное удаление цифр 8 из лево части строки с постоянным изменением первой цифры (между единицей, двойкой и тройкой). Нетрудно подсчитать, что при полном цикле (когда первая цифра вновь становится единичкой) у нас будет «съедено» шесть восьмёрок. Соответственно, можно так суммарно уничтожить 78 восьмё-

рок (число, делящееся на 6). Останется строка 188. Далее можно ещё раз применить команду и останется 28, после чего ни одна проверка не сработает и цикл завершится.

Ответ: 28

Исполнитель 2: «Робот»

Система команд исполнителя РОБОТ, «живущего» в прямоугольном лабиринте на клетчатой плоскости:

ВВЕРХ ВНИЗ ВЛЕВО ВПРАВО.

Выполненив указанную команду, РОБОТ перемещается на одну клетку соответственно: вверх ↑, вниз ↓, влево ←, вправо →.

Четыре команды служат для проверки истинности условия отсутствия соответствующей стены у той клетки, где находится РОБОТ: СВЕРХУ СВОБОДНО, СНИЗУ СВОБОДНО, СЛЕВА СВОБОДНО, СПРАВА СВОБОДНО.

Рассмотрим цикл

```
ПОКА <условие> команда
```

Он будет выполняться, пока условие истинно, иначе происходит переход на следующую строку. В следующей конструкции команда выполняется только, если условие истинно. В противном случае ничего не происходит.

```
ЕСЛИ <условие>
  ТО команда
КОНЕЦ ЕСЛИ
```

В конструкциях ПОКА и ЕСЛИ условие может содержать команды проверки, а также слова И, ИЛИ, НЕ.

Если РОБОТ начнет движение в сторону стены, то он разрушится, и выполнение программы прервется.

Также у Робота есть команда ЗАКРАСИТЬ, при которой закрашивается клетка, где Робот находится в настоящий момент.

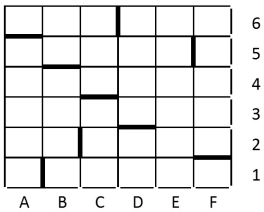
Рассмотрим примеры задач для этого исполнителя.

Задача 4

Сколько клеток приведенного лабиринта соответствуют требованию того, что, выполнив предложенную ниже программу, РОБОТ остановится в той же клетке, с которой он начал движение? По периметру лабиринта стенки.

Решение:

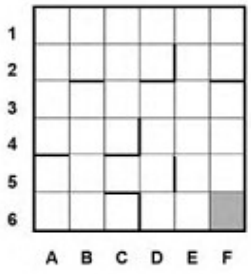
```
НАЧАЛО
ПОКА <сверху свободно> вверх
ПОКА <слева свободно> влево
ПОКА <снизу свободно> вниз
ПОКА <справа свободно> вправо
КОНЕЦ
```



Ответ: 4 (A1, C6, F1, F3).

Задача 5

Сколько клеток лабиринта соответствуют требованию того, что, начав движение в ней и выполнив предложенную программу, РОБОТ уцелеет и остановится в закрашенной клетке (клетка F6)?



Решение:

```
НАЧАЛО
  ПОКА <снизу свободно ИЛИ справа свободно>
    ПОКА <снизу свободно>
        вниз
    КОНЕЦ ПОКА
    ЕСЛИ <справа свободно>
      ТО вправо
    КОНЕЦ ЕСЛИ
  КОНЕЦ ПОКА
КОНЕЦ
```

Ответ: 15 (C5, D3-D6, E1-E6, F3-F6).

Задача 6

На бесконечном поле имеется лестница. Сначала лестница слева направо спускается вниз, затем поднимается вверх. Высота каждой ступени – одна клетка, ширина – две клетки. Робот находится на первой ступеньке лестницы, в левой клетке. Количество ступеней, ведущих вниз, и количество ступеней, ведущих вверх, неизвестно. Напишите для Робота алгоритм, закрашивающий все клетки, расположенные непосредственно над ступенями лестницы. Требуется закрасить только клетки, удовлетворяющие данному условию. Конечное расположение Робота может быть произвольным. Алгоритм должен решать задачу для произвольного размера поля и любого допустимого расположения ступеней внутри прямоугольного поля. При исполнении алгоритма Робот не должен разрушиться; выполнение алгоритма должно завершиться.

Решение:

Так как размер ступеней лестницы фиксированный, то решение будет выглядеть следующим образом:

```
закрасить
вправо
закрасить
ПОКА <справа свободно>
  вправо
  вниз
  закрасить
  вправо
  закрасить
КОНЕЦ ПОКА
ПОКА <НЕ справа свободно>
  вверх
  вправо
  закрасить
  вправо
  закрасить
КОНЕЦ ПОКА
```

Исполнитель 3 «Чертёжник»

Исполнитель `Чертёжник` перемещается на координатной плоскости, оставляя след в виде линии. Чертёжник может выполнять команду СМЕСТИТЬСЯ на (a, b) (где a, b – целые числа), перемещающую `Чертёжника` из точки с координатами (x, y) в точку с координатами $(x + a, y + b)$.

Если числа a, b положительные, значение соответствующей координаты увеличивается, если отрицательные – уменьшается. Например, если `Чертёжник` находится в точке с координатами $(4, 2)$, то команда СМЕСТИТЬСЯ на $(2, -3)$ переместит Чертёжника в точку $(6, -1)$.

Следующая запись означает, что последовательность команд `Команда_1 Команда_2 Команда_3` повторится k раз.

```
Повтори k раз
  Команда_1 Команда_2 Команда_3
Конец
```

Задачи

Задача 7

`Чертёжнику` был дан для исполнения следующий алгоритм:

```
Повтори 4 раз
  Сместиться на (3, 0)
  Сместиться на (-2, -1)
  Сместиться на (1, 0)
Конец
```

Какую команду надо выполнить `Чертёжнику`, чтобы вернуться в исходную точку, из которой он начал движение?

Решение:

Если сложить всё перемещение за один шаг цикла, то получится смещение на $(2, -1)$. Соответственно, за 4 шага цикла будет смещение на $(8, -4)$. Значит, для возврата в

исходную точку его надо скомпенсировать и взять те же по модулю числа, но с противоположным знаком.

Ответ: Сместиться на $(-8, 4)$.

Задача 8

Чертёжнику был дан для исполнения следующий алгоритм (буквами n, a, b обозначены неизвестные числа, при этом $n > 1$):

```
НАЧАЛО
  сместиться на (-3,-3)
  ПОВТОРИ n РАЗ
    сместиться на (a,b)
    сместиться на (27,12)
  КОНЕЦ ПОВТОРИ
  сместиться на (-22,-7)
КОНЕЦ
```

Укажите наименьшее возможное значение числа n, для которого найдутся такие чисела a и b, что после выполнения программы Чертёжник возвратится в исходную точку.

Решение:

Сначала подсчитаем суммарное смещение чертёжника вне цикла. Это будет $(-25, -10)$. Соответственно, при выполнении цикла, необходимо получить суммарное смещение $(25, 10)$. Это смещение можно получить за один или более повторов цикла. Очевидно, что количество повторов цикла должно быть общим делителем у чисел 25 и 10. Соответственно, наибольшее количество повторов – это наибольший общий делитель.

Ответ: 5

Задачи для самостоятельного решения

1. Для каждого исполнителя напишите программу, состоящую из оператора цикла, который не выполнится ни разу.

2. Для каждого исполнителя напишите программу, состоящую из оператора цикла, который выполнится ровно 1 раз.

3. Для каждого исполнителя напишите программу, состоящую из оператора цикла, который выполнится 5 раз.

4. Для каждого исполнителя напишите программу, состоящую из оператора цикла, который выполнится бесконечно много раз.

5. Ниже приведена программа для исполнителя Редактор.

```
ПОКА нашлось(722) ИЛИ нашлось(557)
  ЕСЛИ нашлось(722)
    ТО заменить(722, 57)
    ИНАЧЕ заменить(557, 72)
  КОНЕЦ ЕСЛИ
КОНЕЦ ПОКА
КОНЕЦ
```

На вход этой программе подается строка, состоящая из 55 цифр; последняя цифра в строке – цифра 7,

а остальные цифры – пятёрки. Какая строка получится в результате применения программы к этой строке?

6. Напишите программу для исполнителя «Редактор», которая в исходной строке, состоящей исключительно из чётных цифр, заменит все цифры 2 на цифры 6, а все цифры 6 на цифры 2. Например, из строки 626262486 должна получиться строка 262626482.

7. Чертёжнику был дан для исполнения следующий алгоритм:

```
Повтори 4 раз
  Сместиться на (-2, 3)
  Сместиться на (0, 2)
  Сместиться на (4, -4)
Конец
```

   Какую команду надо выполнить Чертёжнику, чтобы вернуться в исходную точку, из которой он начал движение?

8. *(Базовый уровень)* На бесконечном поле есть горизонтальная и вертикальная стены. Правый конец горизонтальной стены соединён с верхним концом вертикальной стены. Длины стен неизвестны. В каждой стене есть ровно один проход, точное место прохода и его ширина неизвестны. Робот находится в клетке, расположенной непосредственно под горизонтальной стеной у её левого конца.

   Напишите для Робота алгоритм, закрашивающий все клетки, расположенные непосредственно ниже

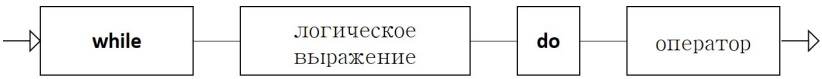
горизонтальной стены и левее вертикальной стены. Проходы должны остаться незакрашенными. Робот должен закрасить только клетки, удовлетворяющие данному условию. При исполнении алгоритма Робот не должен разрушиться, выполнение алгоритма должно завершиться. Конечное расположение Робота может быть произвольным. Алгоритм должен решать задачу для любого допустимого расположения стен и любого расположения и размера проходов внутри стен.

Операторы цикла While и Repeat в языке программирования Паскаль

Перейдём к изучению циклов в языке программирования. В Паскале существует несколько операторов цикла, каждый из которых реализует свою концепцию. Первый оператор цикла реализует алгоритмическую конструкцию цикла с предусловием, то есть условие проверяется до входа в цикл. Рассмотрим его грамматику, алгоритм работы и особенности.

Оператор While

Синтаксическая диаграмма оператора:



Семантика работы:

1. Вычисляется логическое выражение – условие.
2. Если получилось `true`, то переход к пункту 3, иначе выход.
3. Выполняется тело цикла и переход к пункту 1.

Особенности:

1. В теле цикла записывается только 1 оператор, возможно составной (составной оператор представляет собой группу операторов, заключённых в операторные скобки `begin end`).
2. Минимальное число повторов цикла – ноль. Так как условие проверяется до входа в цикл, то можно вообще в него не зайти, если при вычислении логического выражения сразу получится false.
3. Если условие всегда имеет значение true, то происходит зацикливание. Возможные источники зацикливания:
   (a) В условии нет переменных `(while 4=2*2 do)`
   (b) Переменные, о которых зависит условие цикла, никогда не изменяют своё значение при выполнении тела цикла `(while x>0 do y:=y+1;)`
   (c) Осмысленное зацикливание: `while true do...`

Пример: По заданному целому неотрицательному значению n, не применяя формулы, требуется вычислить $s = 1 + 2 + 3 + 4 + ... + n$.

```
readln(n);
s := 0;
i := 0;
while x < n
begin
  i := i + 1;
  s := s + i;
end;
```

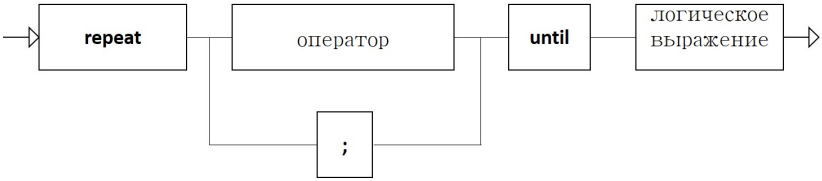
Обращаем внимание на накопительную стратегию вычисления суммы. То есть, на каждом шаге цикла мы добавляем к сумме новое слагаемое (с записью в одну и туже переменную s).

Также помним, что переменную, в которой копится сумма перед началом цикла необходимо обнулить.

Оператор Repeat

Теперь мы познакомимся со вторым оператором цикла в Паскале. Это оператор цикла с постусловием.

Его синтаксическая диаграмма:



Семантика работы:

1. Выполняется тело цикла.
2. Вычисляется логическое выражение – условие.

3. Если получилось `false`, то переход к пункту 1, а если же получилось `true` – то выход.

Особенности:

1. В отличие от оператора `while` у `repeat` структура такая, что есть явные начало и конец оператора, поэтому в теле цикла операторов может быть сколько угодно. Составной оператор использовать не требуется.

2. В отличие от оператора `while` тело цикла `repeat` всегда выполняется хотя бы 1 раз, поскольку условие проверяется после выполнения тела. Поэтому оператор `repeat` является менее общим чем `while` и, по сути, может быть отнесён к разряду «синтаксического сахара», то есть, можно обойтись и без него, но с ним иногда приятнее.

3. По аналогии с `while`, если в условии цикла всегда будет `false`, то происходит зацикливание. Здесь никаких отличий нет.

Рассмотрим несколько примеров задач на использование операторов цикла. Эти задачи демонстрируют несколько стандартных алгоритмов и стратегий, которые необходимо усвоить всем для дальнейшего изучения темы.

Задача 9

Выведите все точные квадраты натуральных чисел, не превосходящие данного числа.

Решение:

В этой задаче нам просто нужна переменная, которая будет на каждом шаге цикла принимать значение

натуральных чисел, квадраты которых мы хотим вывести. Начальное значение для неё возьмём единичку, а выход из цикла происходит, когда квадрат очередного натурального числа окажется больше заданного параметра. Приведём код программы.

```
var i,n:integer;
begin
  readln(n);
  i:=1;
  while i*i<=n do begin
    writeln(i*i);
    i:=i+1;
  end;
end.
```

Задача 10

Дано целое число, не меньшее 2. Выведите его наименьший натуральный делитель, отличный от 1.

Решение:

Для решения этой задачи нам необходимо перебирать натуральные числа, начиная с двух и проверять каждое из них, не является ли оно делителем исходного числа. Процесс завершается, когда делитель найден. Очевидно, что процесс завершится всегда, поскольку в худшем случае число разделится само на себя. Приведём код программы.

```
var i,n:integer;
begin
  readln(n);
  i:=2;
  while n mod i <> 0 do
    i:=i+1;
  end;
  writeln (i);
end.
```

Задача 11

В первый день спортсмен пробежал x километров, а затем он каждый день увеличивал пробег на 10% от предыдущего значения. По данному числу y определите номер дня, на который пробег спортсмена составит не менее y километров. Программа получает на вход действительные числа x и y. Программа должна вывести одно натуральное число.

Решение:

В этой задаче нам нужно реализовать постепенное увеличение пробега. То есть, на каждом шаге цикла мы будем сохранять значение пробега в соответствующий день в одной переменной, а номер этого дня – в другой. Завершение, когда значение первой переменной станет не меньшим чем у. Приведём код программы. Все переменные, отвечающие за километры, имеют тип REAL (из условия).

```
var x,y:real; i:integer;
begin
  readln(x,y);
  i:=1;
  while x < y do begin
    x:=x/100*10+x;
    i:=i+1;
  end;
  writeln(i);
end.
```

Задача 12

Дано натуральное число N. Вычислите его сумму цифр.

Решение:

Для решения этой задачи на каждом шаге цикла нужно изменять наше число: при помощи операции mod можно выделить последнюю цифру из числа и прибавить её к сумме, а затем её надо выбросить из числа при помощи операции div. Делить нужно, естественно, на 10. Критерий завершения – когда число станет равным нулю, ибо это будет означать, что мы уже рассмотрели все цифры и поделили на 10 однозначное число (по свойствам операции целочисленного деления известно, что при делении меньшего числа на большее получается ноль). Приведём код программы.

```
var a,n,s:integer;
begin
  readln(n);
  s:=0;
  while n>0 do begin
    a:=n mod 10;
    s:=s+a;
    n:=n div 10;
  end;
  writeln(s);
end.
```

Задача 13

Дано натуральное число N. Необходимо вывести наименьшую и наибольшую цифры данного числа через пробел.

Решение:

Алгоритм решения этой задачи похож на алгоритм решения предыдущей. Нужно на каждом шаге цикла выделять очередную цифру, а потом делить всё число на десять. Однако на этот раз нам нужно ещё научиться искать максимум и минимум. Для этого необходимо завести две переменных max и min, в которых будут храниться текущие значения максимума и минимума. Затем каждая цифра сравнивается с текущим значением максимума и, в случае если она больше, максимум принимает значение данной цифры. Аналогично с минимумом. В качестве начального значения для максимума нужно взять какое-то число, которое меньше любой цифры, например число минус один. В качестве начального значения для минимума нужно взять какое-то число, которое больше любой

цифры, например число десять. Приведём код программы.

```
var a,n,min,max:integer;
begin
  readln(n);
  max:=-1;
  min:=10;
  while n>0 do begin
    a:=n mod 10;
    if a>max then max:=a;
    if a<min then min:=a;
    n:=n div 10;
  end;
  write(min,' ',max);
end.
```

В заключение отметим, что при желании можно код каждой программы переписать через цикл `repeat`. Попробуйте проделать это самостоятельно.

Задачи для самостоятельного решения

Звёздочкой помечены задачи повышенного уровня сложности.

1. По данному числу N распечатайте все целые степени двойки, не превосходящие N, в порядке возрастания. Операцией возведения в степень пользоваться нельзя.

2. Вклад в банке составляет x рублей. Ежегодно он увеличивается на p процентов, после чего дробная часть копеек отбрасывается. Каждый год сумма вклада становится больше. Определите, через сколько лет вклад составит не менее y рублей. Программа получает на вход три натуральных числа: x, p, y. Программа должна вывести одно целое число.

3. Дано натуральное число N. Необходимо вывести количество нулей среди всех цифр данного числа. Ведущие нули не учитывать.

4. Дано натуральное число N. Необходимо вывести цифры данного числа в обратном порядке.

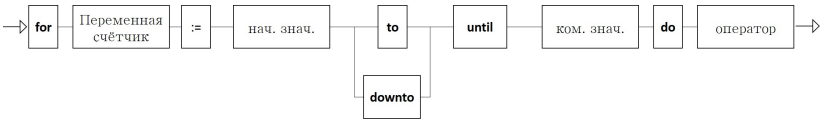
5. * Дано натуральное число N. Выведите слово YES, если число N является точной степенью двойки, или слово NO в противном случае. Операцией возведения в степень пользоваться нельзя.

6. * Исполнитель "Раздвоитель" преобразует натуральные числа. У него есть две команды: «Вычесть 1» и «Разделить на 2», первая команда уменьшает число на 1, вторая команда уменьшает число в два раза, если оно чётное, иначе происходит ошибка. Дано два натуральных числа A и B (A>B). Напишите программу, которая сгенерит и выведет на экран алгоритм для Раздвоителя, который преобразует число A в число B и при этом содержит минимальное число команд. Команды алгоритма нужно выводить по одной в строке, первая команда обозначается, как -1, вторая команда как : 2.

Оператор цикла For. Обработка последовательностей

Оператор цикла `for` – это оператор цикла, для которого заранее известно количество повторений. Это количество определяется при помощи счётчика или параметра цикла.

Синтаксическая диаграмма оператора For:



Семантика работы:

1. Вычисляются начальное и конечное значения (там могут быть выражения) и записываются в служебные ячейки памяти, к которым нет доступа.

2. Счётчику присваивается начальное значение.

3. Значение счётчика сравнивается с конечным. Для то, если оно больше конечного то выход, иначе пе-

реход к пункту 3. Для DOWNTO, если оно меньше конечного, то выход, иначе переход к пункту 3.

4. Выполняется тело цикла.

5. Для TO, счётчик принимает следующее значение в типе (+1 для целых чисел). Для DOWNTO, счётчик принимает предыдущее значение в типе (−1 для целых чисел).

6. Переход к пункту 2.

Особенности:

1. Точка ветвления, из которой возможен выход, находится до тела цикла, поэтому можно сказать, что оператор FOR – это частный случай оператора цикла с предусловием. Соответственно, минимальное число выполнений тела цикла – ноль.

2. В теле цикла должен быть только 1 оператор, возможно составной.

3. Цикл FOR никогда не зацикливается, а его число повторений можно вычислить по формулам. Для TO: «конечное значение-начальное значение+1». Для DOWNTO: «начальное значение-конечное значение+1»

4. Типы счётчика, начального и конечного, значений должны совпадать. При этом, тип счётчика должен быть порядковым. Из изученных ранее, порядковыми типами являются целые числа и логика.

5. Начальное и конечное значение вычисляются 1 раз при входе в цикл и больше не перевычисляются. То есть, цикл FOR J:=J TO J DO... всегда будет работать ровно 1 раз.

6. В теле цикла нельзя изменять значение счётчика. Имейте в виду, что если у вас старый компилятор (Borland Pascal или Turbo Pascal), то это не синтаксическое правило языка, а рекомендация от мастера. Если у вас нормальный компилятор, то это синтаксическое правило языка.

7. После завершения цикла FOR значение счётчика считается неопределённым.

Пример 1
Вычислим сумму целых чисел от 1 до N.

```
s:=0;
for j:=1 to N do s:=s+j;
```

Выполнение любого оператора цикла можно завершить досрочно, если использовать процедуру `break`. На данном этапе она особенно актуальна, если надо досрочно прервать цикл `for`. Выполнение этой процедуры передаст управление на оператор, который следует за прерываемым оператором цикла.

Пример 2
Вводится натуральное число x ($2 \leq x \leq 30000$). Выведите наименьший делитель числа x, отличный от 1.

Мы уже разбирали алгоритм решения этой задачи. Продемонстрируем его с использованием цикла `for`.

```
var a,i: integer;
begin
  readln(a);
  for i := 2 to a do
    if a mod i = 0 then
    begin
      writeln(i);
      break;
    end;
  end;
end.
```

Обработка последовательностей

Перейдём к изучению задач из класса «Обработка последовательностей». В данном классе формулировка всех задач начинается со слов: «Вводится последовательность чего-нибудь. . . ». Для простоты будем считать, что это чисел – с ними мы умеем работать, но в общем случае это необязательно так. Далее нам нужно оценить какое-то свойство данной последовательности. Например, сколько в ней нулей, является ли она возрастающей (каждое следующее число больше предыдущего) и т.д. Причём, для оценки данного свойства, сохранение всей последовательности в памяти не требуется.

Общий алгоритм решения задач этого класса таков: мы считываем очередной элемент последовательности, обрабатываем его так, как поставлено в задаче, и после про него забываем. Для реализации данного алгоритма нам придётся поместить оператор ввода внутрь цикла.

Рассмотрим стандартные шаблоны решения задач данного класса:

Тип 1

Количество элементов в последовательности известно заранее (N).

```
readln(N);
for j:=1 to N do begin
  read(a);
  Содержательна обработка
end;
```

Обращаем внимание, что внутрь оператора цикла обязательно помещается оператор `read`, а не `readln`. Дело в том, что, как правило, числа будут вводиться в строчку и оператор ввода будет на каждом шаге забирать из буфера по 1 числу. Оператор `readln` при этом ещё удаляет из буфера всё лишнее до конца строки, поэтому числа прочитаны не будут. Количество элементов последовательности, как правило, задаётся отдельным числом в первой строке, поэтому его логичнее читать оператором readln. Рассмотрим примеры задач данного класса.

Задача 14

Вводится натуральное число N, а затем N натуральных чисел, сумму которых необходимо вычислить.

Решение:

```
var i,N,a,s:integer;
begin
  readln(N);
    s:=0;
  for i:=1 to N do begin
    read(a);
    s:=s+a;
  end;
```

```
  writeln(s)
end.
```

Задача 15

Вводится натуральное число N, а затем N целых чисел. Выведите YES, если среди введенных чисел есть хотя бы один ноль, или NO в противном случае.

Решение:

```
var n,i,k,a:integer;
begin
  s:=0;
  readln(n);
  for i:=1 to n do begin
    read(a);
    if a=0 then k:=k+1;
  end;
if k<>0
  then writeln('YES')
  else writeln('NO')
end.
```

Теперь рассмотрим второй тип задач из класса «Обработка последовательностей», когда количество элементов неизвестно заранее, но известен признак конца.

Признак конца – это служебный элемент, который в саму последовательность не входит. При обработке числовых последовательностей чаще всего признаком конца является ноль (но может быть и любое другое число).

Наиболее распространённая ошибка – обработать признак конца как содержательный элемент (например, при расчёте среднего арифметического элементов последовательности). Поэтому порядок действий должен быть таким:

1. Считывание
2. Проверка на признак конца
3. Содержательная обработка

За 3 пунктом алгоритм замыкается в цикл. Выход из цикла – на втором пункте, в случае положительного ответа. Соответственно, шаблон алгоритма должен отражать данную логику действий.

Тип 2

```
read(a);
while a<>0 do begin
  содержательная обработка;
  read(a);
end;
```

В теле цикла оператор ввода ставится последним, чтобы следующим действием за вводом было вычисление условия цикла (то есть, проверка на признак конца).

Задача 16

Программа получает на вход последовательность целых неотрицательных чисел. Ноль – признак конца. Вывести количество членов последовательности (не считая завершающего числа 0).

Решение:

```
var a,k:integer;
begin
  read(a);
  k:=0;
  while a<>0 do begin
    k:=k+1;
    read(a);
  end;
  writeln(k);
end.
```

Задача 17

Вводится последовательность целых чисел. Ноль – признак конца. Вывести, сколько элементов данной последовательности больше (строго) предыдущего элемента.

Решение:

При решении этой задачи нам нужно будет сохранять в отдельную переменную значение предыдущего числа перед вводом нового. Приведём код программы.

```
var a,pr,k:integer;
begin
  read(a);
  pr:=a;
  k:=0;
  while a<>0 do begin
    if (a > pr) then k:=k+1;
    pr:=a;
    read(a);
```

```
  end;
writeln(k);
end.
```

Задача 18

Вводится последовательность натуральных чисел. Ноль – признак конца. Вывести значение максимального элемента.

Решение:

```
var a,m:integer;
begin
  read(a);
  m:=a;
  while a<>0 do begin
    if (a > m) then m:=a;
    read(a);
  end;
  writeln(m);
end.
```

При решении задачи поиска максимума для нас пока остался вопрос корректной инициализации. В общем случае есть два способа.

Первый вариант – в качестве начального значения максимума брать «минус бесконечность» (для минимума – плюс бесконечность). Под «минус бесконечностью» в данном случае понимается некое число, которое гарантированно меньше чем любой элемент, который может нам встретиться. Инициализация бесконечностью допустима,

если мы заранее знаем диапазон элементов последовательности. Это её недостаток, зато она пишется в один оператор присваивания.

Второй вариант – в качестве начального значения брать первый подходящий элемент последовательности. В случае глобального максимума берётся просто первый элемент последовательности. Сложнее в ситуации, когда первый элемент может оказаться неподходящим.

Пример 3

Найти максимальный чётный элемент последовательности в предположении, что он существует.

```
var a,m:integer;
begin
  read(a);
  while (a<>0)and(a mod 2 <> 0) do read(a);
  m:=a;
  while a<>0 do begin
    if (a mod 2 = 0)and(a>m) then m:=a;
    read(a);
  end;
  writeln(m);
end.
```

Задачи для самостоятельного решения

Звёздочкой помечены задачи повышенного уровня сложности.

1. Подсчитайте количество натуральных делителей числа x (включая 1 и само число; x$\leq$ 30000).

2. Программа получает на вход последовательность целых неотрицательных чисел. Ноль – признак конца. Вывести среднее арифметическое членов последовательности (не считая завершающего числа 0).

3. Вводится последовательность целых чисел. Ноль – признак конца. Вывести, сколько раз в данной последовательности встречается максимальный элемент.

4. * Дана последовательность целых чисел, не превосходящих по модулю 1000. Ноль – признак конца. Найдите максимальное произведение двух элементов этой последовательности.

5. * Дана последовательность целых чисел, не превосходящих по модулю 1000. Ноль – признак конца. Найдите минимальное произведение двух элементов этой последовательности.

6. * Дана последовательность натуральных чисел. Ноль – признак конца. Определите, какое наибольшее число подряд идущих элементов этой последовательности равны друг другу.

7. * Элемент последовательности называется локальным максимумом, если он строго больше предыдущего и последующего элемента последовательности. Первый и последний элемент последовательности не являются локальными максимумами. Дана последовательность натуральных чисел, признаком конца которой является число 0. Определите количество строгих локальных максимумов в этой последовательности.

Оглавление

Введение **5**
«Редактор» 5
«Робот» 9
«Чертёжник» 12
Задачи 13
Задачи для самостоятельного решения 15

Операторы цикла While и Repeat **19**
Оператор While 19
Оператор Repeat 21
Задачи для самостоятельного решения 27

Оператор цикла For **29**
Обработка последовательностей 32
Задачи для самостоятельного решения 38