

Циклические вычислительные процессы

Наумов Д.А., доц. каф. КТ

Программирование и алгоритмические языки, 2019

Содержание лекции

- 1 Циклические алгоритмы
- 2 Вложенные циклы

Циклические алгоритмы

Вычислительный процесс с многократным повторением однотипных вычислений для различных значений обрабатываемых величин называется **циклическим**, повторяющиеся участки вычислений - **циклами**, изменяющиеся в цикле величины - **переменными цикла**. Для организации циклических алгоритмов необходимо предусмотреть:

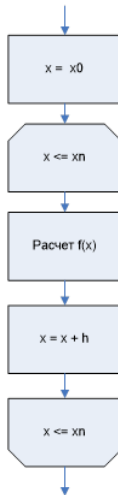
- 1 подготовку цикла: задание начальных значений переменным цикла перед первым его выполнением;
- 2 тело цикла: действия, повторяемые в цикле для различных значений переменных цикла;
- 3 модификацию (изменение) значений переменных цикла перед каждым новым его повторением;
- 4 управление циклом: проверку условия продолжения цикла и переход на начало тела цикла, если условие продолжение выполняется (или выход из цикла, если условие не выполняется).

Блок-схема циклического алгоритма



Табулирование функции - вычисления значений функции f переменной, изменяющейся от x_0 до x_n с постоянным шагом h .

- 1 подготовка - присвоение переменной x начального значения x_0 ;
- 2 тело цикла - вычисление значения функции для очередного значения аргумента и вывод полученного значения;
- 3 переменной цикла является переменная x ;
- 4 модификацией является увеличение значения переменной x на величину шага h ;
- 5 управление цикла - проверка условия ($x \leq x_n$).



Оператор цикла с параметром

Различают циклы:

- с заданным числом повторений;
- с заранее неизвестным числом повторений.

Циклы первого типа называются **циклами со счетчиком** или **циклы с параметром**.

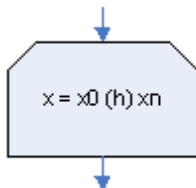
- 1 число повторений такого тела такого цикла подсчитывается с помощью специальной переменной - счетчика, для которой известны начальное и конечное значение, а также шаг ее изменения.
- 2 переменную-счетчик называют параметром цикла.

Оператор цикла с параметром

При схематичном изображении цикла с параметром в блок «Управление циклом 1» записывается информация в следующей форме:

- Параметр = Начальное значение (Шаг изменения) Конечное значение

Для задачи табулирования функции $f(x)$ переменной, изменяющейся в интервале от x_0 до x_n с постоянным шагом h , блок «Управление циклом 1» будет иметь следующий вид:



Оператор цикла с параметром

Для программирования циклов с известным числом повторений следует пользоваться оператором цикла с параметром, который имеет следующий вид:

```
1  for Параметр := Выражение1 to  Выражение2 do
2      Оператор
```

или

```
1  for Параметр := Выражение1 downto  Выражение2 do
2      Оператор
```

- Параметр - переменная порядкового типа
- Выражение1, Выражение2 - выражения того же типа, что и Параметр
- Оператор - оператор языка

- 1 Выражение1 вычисляется до выполнения тела цикла один раз и является блоком подготовки цикла. При помощи данного выражения задается начальное значение переменной цикла.
- 2 Выражение2 вычисляется в цикле перед выполнением тела цикла и является блоком управления цикла.
 - Параметр \leq Выражение 2 для цикла «to»
 - Параметр \geq Выражение 2 для цикла «downto»
- 3 Модификация параметра выполняется после выполнения тела цикла.
 - inc(Параметр) для цикла «to»
 - dec(Параметр) для цикла «downto»



Этапы выполнения цикла for

- 1 вычислить значение выражение1; присвоить значение параметру цикла;
- 2 вычислить выражение2 и условие продолжения цикла;
- 3 если условие продолжения цикла истинно, то перейти к пункту 4, иначе завершить цикл и перейти к оператору, непосредственно следующему за оператором for
- 4 выполнить оператор (тело цикла);
- 5 модифицировать значение параметра цикла;
- 6 перейти к пункту 2.



Пример: вычисление степени с натуральным показателем

Вычислить степень $y = a^n$ действительного числа a с натуральным показателем n .

$$a^n = \underbrace{a \cdot a \cdot \dots \cdot a}_{n \text{ раз}}$$

Компактно такое произведение может быть записано в следующем виде:

$$a^n = \prod_{i=1}^n a$$

Пример: вычисление степени с натуральным показателем

Для вычисления указанного произведения целесообразно организовать цикл с параметром i , в котором осуществлялось бы последовательное накопление произведения y по следующему правилу:

- до начала цикла $y = 1$,
- на каждом шаге цикла (для $i=1,2,\dots,n$) значение переменной y будет увеличиваться в a раз (то есть $y = y * a$).
- цикл должен быть выполнен n раз (то есть условие цикла $i \leq n$).

```

1 var
2   i, n: integer; a, y : extended;
3 begin
4   //ввод a, n
5   y := 1;
6   for i := 1 to n do
7     y := y * a;
8   //вывод y
9 end.
```

Пример: вычисление суммы чисел

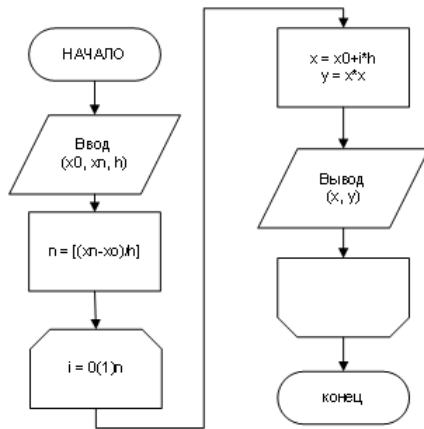
Вычислить сумму нечетных чисел, находящихся в интервале от a до b .

- используем цикл с параметром i от a до b
- в теле цикла следует проверить, что текущее значение параметра цикла нечетным, и если является, то прибавить это значение к переменной s , которая в данной задаче будет использоваться для накопления суммы.
- перед началом цикла значение s должно быть нулевым.

```
1 var
2   a, b, i, s: integer;
3 begin
4   //ввод a, b
5   s := 0;
6   for i := a to b do
7     if i mod 2 = 1 then
8       s := s + i;
9   //вывод s
```

Табулирование функции

Вывести на печать значения функции $y = x^2$ для переменной x , изменяющейся в интервале от x_0 до x_n с постоянным шагом h .



Пример: вычисление суммы чисел

Вывести на печать значения функции $y = x^2$ для переменной ,
изменяющейся в интервале от x_0 до x_n с постоянным шагом h .

```
1 var
2   h, x, x0, xn, y: extended;
3   i, n: integer;
4 begin
5   //ввод x0, xn, h
6   n := round((xn-x0)/h);
7   for i := 0 to n do
8     begin
9       x := x0 + i * h;
10      y := x * x;
11    //вывод x, y
12  end;
13 end.
```

Циклы с неизвестным числом повторений

Достаточно часто приходится сталкиваться с вычислительными процессами, когда число повторений цикла неизвестно, а задано некоторое условие его окончания (или продолжения). Для программной реализации таких вычислительных процессов существует два типа операторов:

- оператор цикла с предусловием
- оператор цикла с постусловием.

1 while Выражение do

2 Оператор

- while, do - ключевые слова,
- Выражение - логическое выражение,
- Оператор - любой оператор языка.

- Выражение задает условие продолжения цикла
- Выражение вычисляется перед каждым выполнением цикла (отсюда и термин - предусловие)
- Если значение Выражения ложно с самого начала, то Оператор не выполнится ни разу.

1 while Выражение do

2 Оператор

Выполнение оператора цикла с предусловием состоит из следующих шагов:

- 1 вычислить Выражение. Если значение выражения истинно, то перейти к пункту 2, иначе завершить цикл и перейти к оператору, непосредственно следующему за оператором while.
- 2 выполнить оператор (тело цикла);
- 3 перейти к пункту 1



Оператор цикла с постусловием

```
1 repeat  
2   Группа операторов  
3 until Выражение
```

- Выражение задает условие завершения цикла
- Выражение вычисляется после каждым выполнения цикла
- Операторы тела цикла выполняются хотя бы один раз.

Выполнение цикла с постусловием:

- 1 выполнить операторы тела цикла;
- 2 вычислить Выражение. Если значение выражения ложно, то перейти к пункту 1, иначе завершить цикл и перейти к оператору, непосредственно следующему за оператором repeat-until.
- 3 перейти к пункту 1



Правило программирования циклов

- 1 перед каждым выполнением цикла условие его продолжения должно быть определено (иметь конкретное значение);
- 2 тело цикла должно содержать хотя бы один оператор, влияющий на условие продолжения цикла, иначе цикл будет продолжаться бесконечно;
- 3 условие продолжения цикла должно в конце концов принять значение `false`;
- 4 условие вычисляется при каждом выполнении цикла и поэтому должно быть насколько можно простым

Пример. Табулирование функции. Цикл while

Вывести на печать значения функции $y = x^2$ для от x_0 до x_n с шагом h .

```
1 var 2 h, x, x0, xn, y: extended; 3 begin 4 //ввод x0, xn, h
5 x := x0; 6 while x <= xn do 7 begin 8 y := x * x; 9 x := x +
10 //вывод x, y 11 end; 12 end.
```

Пример. Табулирование функции. Цикл repeat-until

Вывести на печать значения функции $y = x^2$ для от x_0 до x_n с шагом h .

```
1 var 2 h, x, x0, xn, y: extended; 3 begin 4 //ввод x0, xn, h
5 x := x0; 6 repeat 7 y := x * x; 8 x := x + h; 9 //вывод x, y
10 until x > xn; 11 end.
```

Пример.

Определить, встречается ли в десятичной записи числа n цифра d .

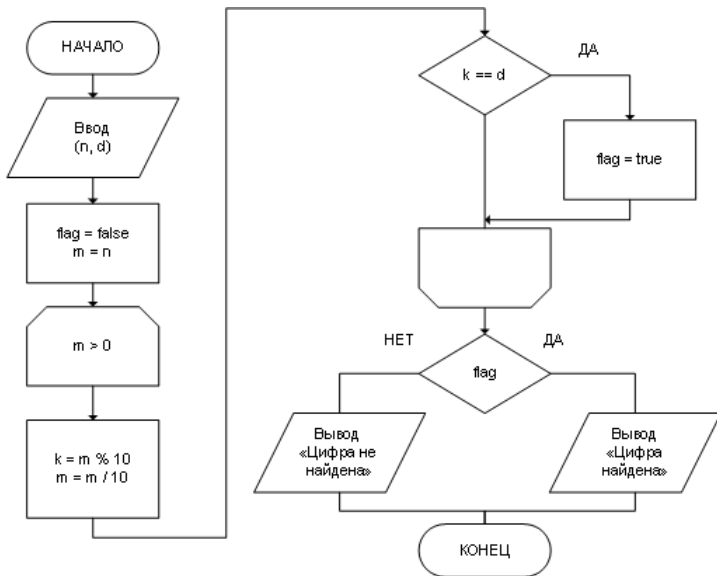
- нам необходима переменная (зададим ей идентификатор `flag`), которая будет использоваться как признак того, встретилась ли цифра d при поиске среди цифр десятичной записи числа n .
- до выполнения основной части алгоритма переменной `flag` следует присвоить значение `false` (так как очевидно, что до того, как мы приступили к перебору цифр числа n , цифру d мы еще нашли)
- необходимо осуществить циклический перебор всех цифр числа d . С точки зрения программирования перебор проще осуществлять, начиная с крайней правой цифры числа.
- для того, чтобы получить значение крайней правой цифры числа d , следует вычислить остаток от деления числа d на 10. Затем следует отбросить крайнюю цифру числа (разделив число на 10) и повторять данный процесс до тех пор, пока не будет отброшена последняя цифра (то есть, пока число d не станет равным нулю);

Пример.

Определить, встречается ли в десятичной записи числа n цифра d .

- так как в предыдущей операции производятся математические действия, приводящие к изменению исходных данных (числа, в десятичной записи которого осуществляется поиск заданной цифры), то следует использовать вспомогательную переменную m (присвоив ей после ввода исходных данных значение n), над которой и производить требуемые вычисления.
- после выполнения перебора всех цифр числа переменная $flag$ будет содержать значение $true$, если цифра d хотя бы один раз встретилась в десятичной записи числа n , и значение $false$ в противном случае. В зависимости от значения переменной $flag$ следует вывести текстовое сообщение о результатах поиска.

Схема алгоритма




```
1 var
2   d, k, n, m: integer;
3   flag: boolean;
4 begin
5   //ввод n;
6   //ввод d;
7   m := n;
8   flag := false;
9   while m > 0 do begin
10     k = m mod 10;
11     m = m div 10;
12     if k = d then
13       flag := true;
14   end;
15   if flag then
16     writeln('число ', n, ' содержит цифру ', d)
17   else
18     writeln('число ', n, ' не содержит цифру ', d);
```

```
1 var
2   d, k, n, m: integer;
3   flag: boolean;
4 begin
5   //ввод n;
6   //ввод d;
7   m := n;
8   flag := (n=0) and (d=0);
9   while not flag and (m > 0) do begin
10     k = m mod 10;
11     m = m div 10;
12     if k = d then
13       flag := true;
14   end;
15   if flag then
16     writeln('число ', n, ' содержит цифру ', d)
17   else
18     writeln('число ', n, ' не содержит цифру ', d);
```

Вложенные циклы

Если телом цикла является циклическая структура, то такие циклы называются вложенными или сложными.

Внешний цикл

цикл, содержащий в себе другой цикл.

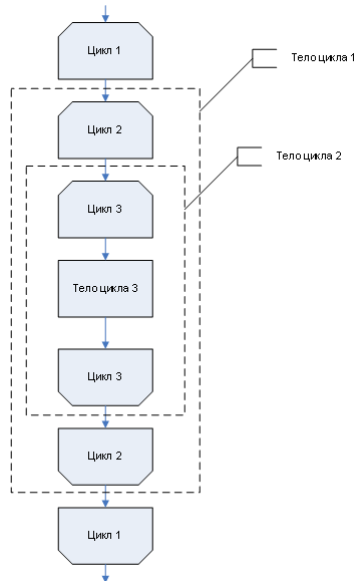
Внутренний цикл

цикл, содержащийся в теле другого цикла.

- Внутренний и внешний циклы могут быть любыми из трех рассмотренных видов: циклами с параметром, циклами с предусловием, циклами с постусловием.
- Правила организации как внешнего, так и внутреннего циклов такие же, как и для простого цикла.
- При построении вложенных циклов необходимо соблюдать следующее дополнительное условие: все операторы внутреннего цикла должны полностью лежать в теле внешнего цикла.

Сложные циклы условно разбиваются на уровни вложенности. Структура вложенных циклов (рис):

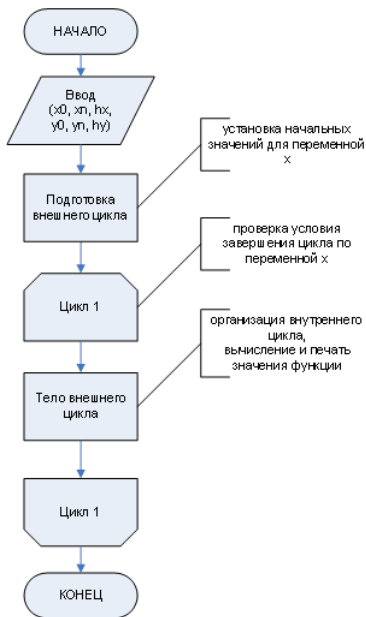
- ❶ цикл 1 имеет уровень 0;
 - ❷ внутренний цикл 2 имеет уровень 1;
 - ❸ внутренний цикл 3 имеет уровень 2.
- Вначале все свои значения изменит параметр цикла наивысшего уровня вложенности при фиксированных (начальных) значениях параметров циклов с меньшим уровнем.
 - Затем изменяется на один шаг значение цикла предыдущего уровня, и снова выполняется самый внутренний цикл.
 - Так происходит до тех пор, пока параметры циклов всех уровней не примут все требуемые значения

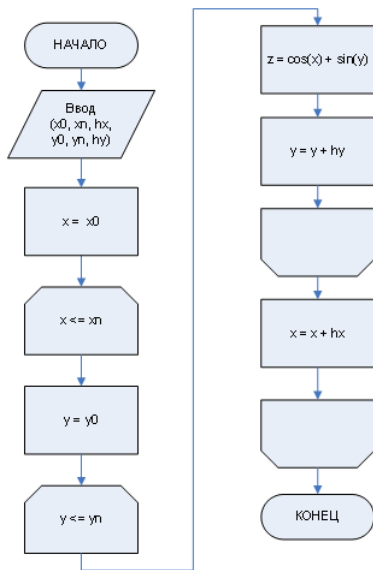


Пример

Составьте алгоритм и программу вычисления значений функции $z = \cos(x) + \sin(y)$ для переменных $x = x_0(hx)x_n$; $y = y_0(hy)y_n$. Для определения значений функции z для всех различных пар (x, y) необходимо процесс вычислений организовать следующим образом.

- Вначале при фиксированном значении одного из аргументов, например, при $x = x_0$ вычислить значения z для всех заданных y : $y_0, y_0 + hy, y_0 + 2hy, \dots, y_n$.
- Затем, изменив значение x на $x + hx$, вновь перейти к полному циклу изменений переменной y .





```
1 var
2   x, x0, xn, hx,
3   y, y0, yn, hy, z: extended;
4 begin
5   //ввод исходных данных x0, xn, hx, y0, yn, hy
6   x := x0; // подготовка внешнего цикла
7   while x <= xn do begin
8     // подготовка внутреннего цикла
9     y := y0;
10    while y <= yn do begin
11      // вычисление значения функции
12      z := cos(x) + sin(y);
13      // вывод x, y, z
14      // модификация переменной внутр. цикла
15      y := y + hy;
16    end;
17  end;
18  x := x + hx;
19 end;
```


Составьте алгоритм и программу вычисления значения выражения:

$$s = \sum_{i=1}^m \prod_{j=1}^n x^{i-j}$$

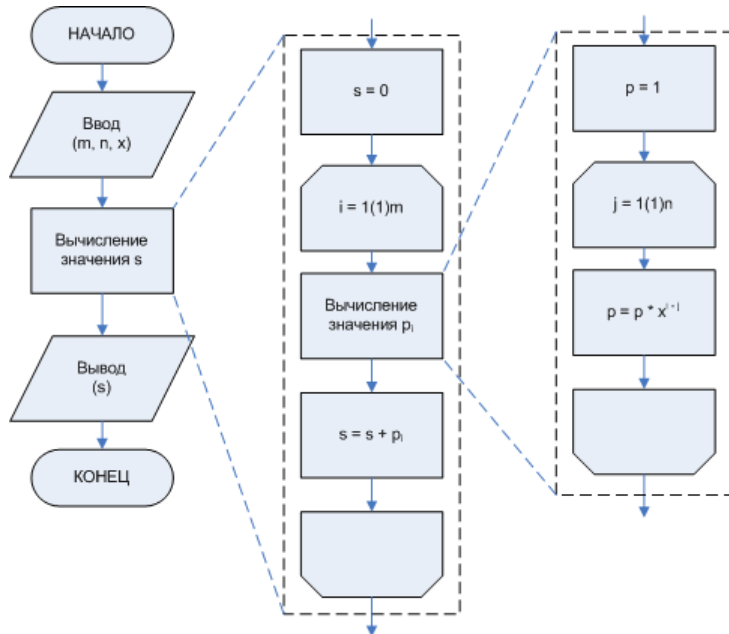
s - сумма m слагаемых, каждое слагаемое - произведение из n множителей $x^{(i-j)}$.

Внутренний цикл будет вычислять значение отдельного слагаемого по формуле:

$$p_i = \prod_{j=1}^n x^{i-j}$$

а внешний цикл - вычислять значение суммы m слагаемых по формуле:

$$s = \sum_{i=1}^m p_i$$



```
1 var i, j, m, n: integer;
2   p, s, x: extended;
3 begin
4   //ВВОД m, n, x
5   s := 0;
6   for i := 1 to m do begin
7     p := 1;
8     for j := 1 to n do
9       p := p * exp(ln(x) * (i - j));
10      s := s + p;
11   end;
12   //ВЫВОД s
13 end.
```

Пример

Определить количество «счастливых билетиков».

Билет является «счастливым», если в его номере сумма первых трех цифр равна сумме последних трех цифр. Минимальный номер билета – 000 001, максимальный номер билета – 999 999.

Вариант 1

- перебрать все возможные номера билетов от 000 001 до 999 999;
- отделить в номере первые три цифры от последних трех;
- сравнить сумму первых трех чисел с суммой последних трех чисел, и если суммы совпали, увеличить значение счетчика счастливых билетов.

```
1 var  a1, a2, a3, a4, a5, a6: byte;
2      g, s: longint;
2 begin
3   s = 0;
4   for g := 1 to 999999 do begin
5     a1 := g mod 10;
6     a2 := g div 10 mod 10;
7     a3 := g div 100 mod 10;
8     a4 := g div 1000 mod 10;
9     a5 := g div 10000 mod 10;
10    a6 := g div 100000 mod 10;
11    if a1 + a2 + a3 = a4 + a5 + a6 then s := s + 1;
12  end;
13  writeln('Всего существует ', s, ' счастливых билетов');
14 end.
```

Пример

Вариант 2

- организовать внешний цикл – перебор возможных значений цифр первой группы от 000 до 999;
- организовать внутренний цикл – перебор возможных значений цифр второй группы от 000 до 999;
- сравнить сумму цифр в числе первой группы с суммой цифр в числе второй группы, и если суммы совпали, увеличить значение счетчика счастливых билетов;
- учесть, что билета с номером 000 000 не существует.

Пример

Вариант 3

- организовать шесть вложенных циклов – каждый цикл для перебора значений от 0 до 9, соответствующий одной из цифр шестизначного номера билета;
- сравнить сумму первых трех чисел с суммой вторых трех чисел, и если суммы совпали, увеличить значение счетчика счастливых билетов;
- учесть, что билета с номером 000 000 не существует.