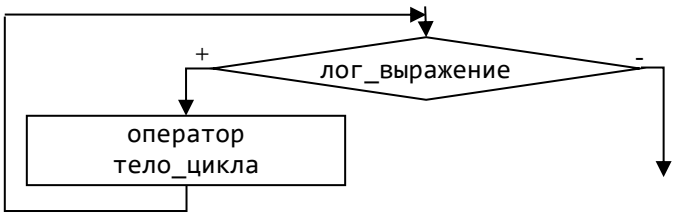
Лабораторная работа 03. Циклы

Циклические операторы

I. Цикл с предусловием




```
while (лог_выражение){
 оператор_тело_цикла;
}
```

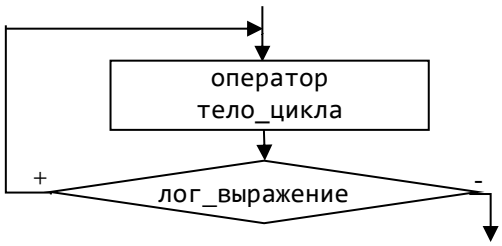
Правила синтаксиса:

1. логич_выражение в круглых скобках
2. если в *операторе_тело_цикла* несколько действий, то они оформляются как блок в { }

Порядок выполнения

1. Вычисляется значение *логич_выражения*
2. Если это значение истинно,
   то выполняется *оператор_тело_цикла* и возврат на п.1
   иначе цикл завершает свою работу

2. Цикл с постусловием




```
do {
 оператор_тело_цикла;
} while (лог_выражение);
```

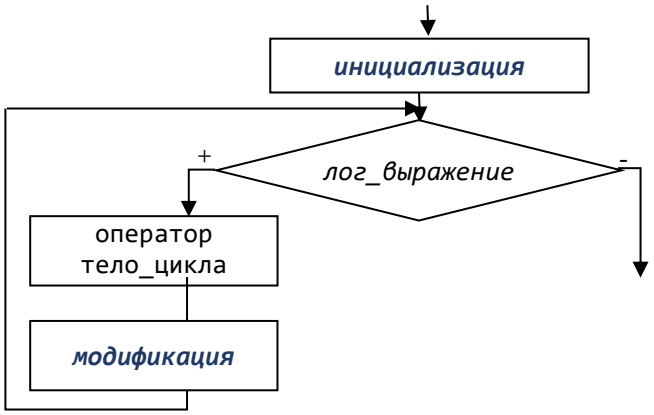
Правила синтаксиса:

1. логич_выражение в круглых скобках
2. если в *операторе_тело_цикла* несколько действий, то они оформляются как блок в { }

Порядок выполнения

1. Выполняется *оператор_тело_цикла*
2. Вычисляется значение *логич_выражения*
3. Если это значение истинно, то выполняется возврат на п.1
   иначе цикл завершает свою работу

3. Параметризированный цикл




```
for (инициализация; лог_выражение; модификация)
   {
      оператор_тело_цикла;
   }
```

Порядок выполнения

1. Выполняется ***инициализация***
2. Вычисляется значение *логич_выражения*
3. Если это значение истинно, то выполняется
   1) *оператор_тело_цикла*
   2) *модификация*
   3) переход к п.2

   иначе цикл завершается

Правила синтаксиса:

1. (***инициализация***; ***лог_выражение***; ***модификация)***
   1) в круглых скобках
   2) любой из них может быть пропущен
   3) разделитель - ; ставится всегда
2. Переменные, объявленные в ***инициализации*** доступны только внутри цикла
3. если в *операторе_тело_цикла* несколько действий, то они оформляются как блок в { }

Операторы прерывания цикла break и continue

Операторы **break и continue** позволяют менять стандартное поведение циклов: заканчивать их выполнение раньше или пропускать некоторые итерации (одно выполнение тела цикла называется итерацией).

Оператор break

У оператора **break** есть два применения:

- завершить текущий цикл любого типа (for, while, do-while) и передать управление на следующий за циклом оператор;
- прекратить выполнение блока в switch;

Оператор break часто применяется, когда цикл не может быть выполнен по какой-либо причине, например, когда приложение обнаруживает ошибку.

Оператор continue

Оператор **continue** позволяет остановить текущую итерацию цикла и начать новую.
Этот оператор можно использовать внутри любого цикла:

- внутри циклов while и do-while он возвращается непосредственно к условию цикла;
- внутри for цикла он сначала вычисляет выражение приращения, а затем возвращается к условию.

Он используется, если ясно, что в текущей итерации цикла делать больше нечего.

Примеры:

```
// вывести все нечетные числа от 101 до 199
for (int x = 101; x < 200; x += 2) {
    printf("%x ", x);
}
```

```
// вывести все нечетные числа от 101 до 199
// пропускать (не выводить) числа, заканчивающиеся на 7
for (int x = 101; x < 200; x += 2) {
    if (x % 10 == 7) continue;
    printf("%x ", x);
}
```

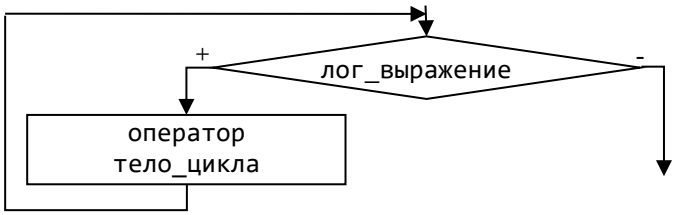
```
// выводить нечетные числа, начиния от 101 и не превышая 199
// на первом же числе, заканчивающемся на 7 - остановиться
for (int x = 101; x < 200; x += 2) {
    if (x % 10 == 7)  break;
    printf("%x ", x);
}
```

Задание 1. Ознакомьтесь с синтаксисом циклических операторов на примере вычисления n!

Задача: Ввести целое число n. Вывести значение функции n-факториал: n! = 1 · 2 · 3 ·…· n.

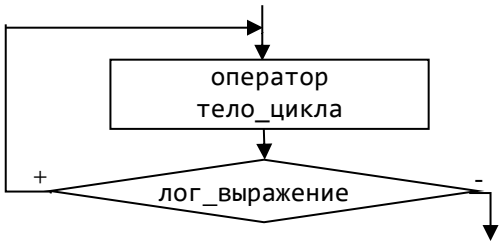
```c
// Решение циклом while
#include <stdio.h>
int main()
{
    int k = 2, p = 1, n;
    printf("Введите n = ");
    scanf("%d", &n);

    while (k <= n)
    {
        p *= k;
        k++;
    }
    printf("n! = %d\n", p);
    return 0;
}
```



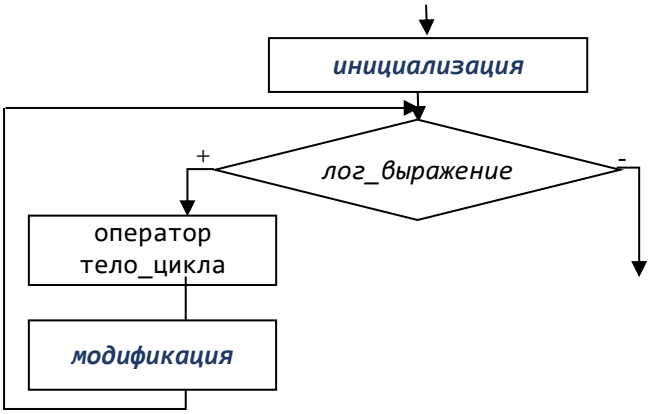

```c
// Решение циклом do..while
#include <stdio.h>
int main()
{
    int k = 1, p = 1, n;
    printf("Введите n = ");
    scanf("%d", &n);

    do
    {
        p *= k;
        k++;
    } while (k <= n);
    printf("n! = %d\n", p);
    return 0;
}
```




```c
// Решение циклом for
#include <stdio.h>
int main()
{
    int p = 1, n;
    printf("Введите n = ");
    scanf("%d", &n);

    for (int k = 2; k <= n; k++) {
        p *= k;
    }
    printf("n! = %d\n", p);
    return 0;
}
```



Самостоятельно напишите и протестируйте программы на языке C++
(**массивы в решении не использовать**)

1. Для 7-ми введенных целых чисел определить и вывести
   a) количество отрицательных чисел
   b) сумму двузначных чисел
   c) наименьшее из всех введенных чисел

2. Задача табуляции функции.
   a) циклом **while** вывести на экран все значения от 0° до 100° с шагом 10°
   для температуры в градусах Цельсия tC
   и их эквиваленты в градусах Фаренгейта: $t_F = \frac{9}{5} t_C + 32$

   b) циклом **do..while** вывести значения функции $y=\ln(x+1)\cdot\sin(x)$ в диапазоне от 0 до 5 с шагом 0.5

   c) циклом **for** вывести таблицу значений функции $y = \cos(x)$ в диапазоне от 0 до 2π с шагом $\pi/6$
   При тестировании обратить внимание на значения на границах диапазона табулирования

   d) любыми циклами в диапазоне [-1.5, 1.5] с шагом 0.25 вывести на экран значения функции

$$y = \begin{cases} 1+\sqrt{|\cos(x)|} & x > 1 \\ x+1 & -0.5 \le x \le 1 \\ 1-x^2 & x < -0.5 \end{cases}$$

3. Дано целое число n. Вычислить и вывести сумму $S = \frac{1}{2} + \frac{1}{4} + \frac{1}{6} + \frac{1}{8} + \ldots + \frac{1}{2n}$

4. Дано целое число n и вещественное число x. Вычислить и вывести значение выражения

   a) $\underbrace{cos(x + cos(x + cos(x + \ldots)))}_{\text{n раз}}$

   b) $\underbrace{\sqrt{x + \sqrt{x + \sqrt{x + \cdots \sqrt{x}}}}}_{\text{n раз}}$

5. Дано малое положительное число (например $\varepsilon = 0.001$). Реализовать алгоритм приближенного вычисления бесконечной суммы. Нужное приближение считать полученным, если вычислена сумма нескольких слагаемых, и модуль следующего слагаемого меньше данного положительного числа.
$1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \cdots$ $\left(\approx \frac{\pi}{4}\right)$ ответ для тестирования

6. Вводить целые числа в диалоге с пользователем до тех пор, пока он не откажется от ввода (хотя бы одно число он должен обязательно ввести). Вывести общее количество введенных чисел и количество среди них четных чисел. *Для проверки четности использовать побитовые операции*

7. Найти периметр n-угольника, вершины которого имеют соответственно координаты $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$. Число n и координаты вводятся пользователем

8. Написать программу вывода на экран текстового изображения шахматной доски (белые клетки можно обозначить, например, пробелом или символом ‘o’, а черные – символом '*'). *Реализовать возможность вывода доски произвольного, задаваемого пользователем размера n x n клеток (n – четное число).*

9. Вводится последовательность, состоящая из натуральных чисел; ввод завершается числом 0. Определить количество элементов этой последовательности, которые равны ее наибольшему элементу.