Циклы C++

Краткое содержание

Циклы повторяют действия, пока выполняется условие, а выбор между `while`, `do...while` и `for` зависит от момента проверки условия и известности числа повторений.

Основы циклов

Ключевые понятия

Цикл — повторяющаяся последовательность действий. Одно выполнение тела цикла называется **итерацией**.

Циклические задачи возникают, когда программа многократно выполняет одни и те же операции. Число повторений может быть:

- заранее известным — обычно используется цикл со счётчиком, например `for`;
- неизвестным и зависящим от условия — используются `while` или `do...while`.

Например, программа может повторять действие, пока идёт дождь, пока пользователь не введёт специальный символ или пока вычисляемое значение не достигнет заданного порога.

Любой цикл должен иметь изменяемый параметр. Если условие зависит от `x`, но `x` внутри цикла не меняется, условие останется истинным и возникнет **бесконечный цикл**.

Предусловие и постусловие

Четыре этапа цикла

1. **Начальная подготовка** — задаются исходные значения, например x_0.
2. **Проверка условия** — определяется, нужно ли выполнять тело цикла.
3. **Тело цикла** — выполняются повторяющиеся операторы.
4. **Модификация параметра** — значение изменяется на шаг Δx, например увеличивается или уменьшается.

Изменение параметра обязательно: без него цикл может никогда не завершиться.

`while` **с предусловием**	`do...while` **с постусловием**
Условие проверяется до тела цикла	Условие проверяется после тела цикла

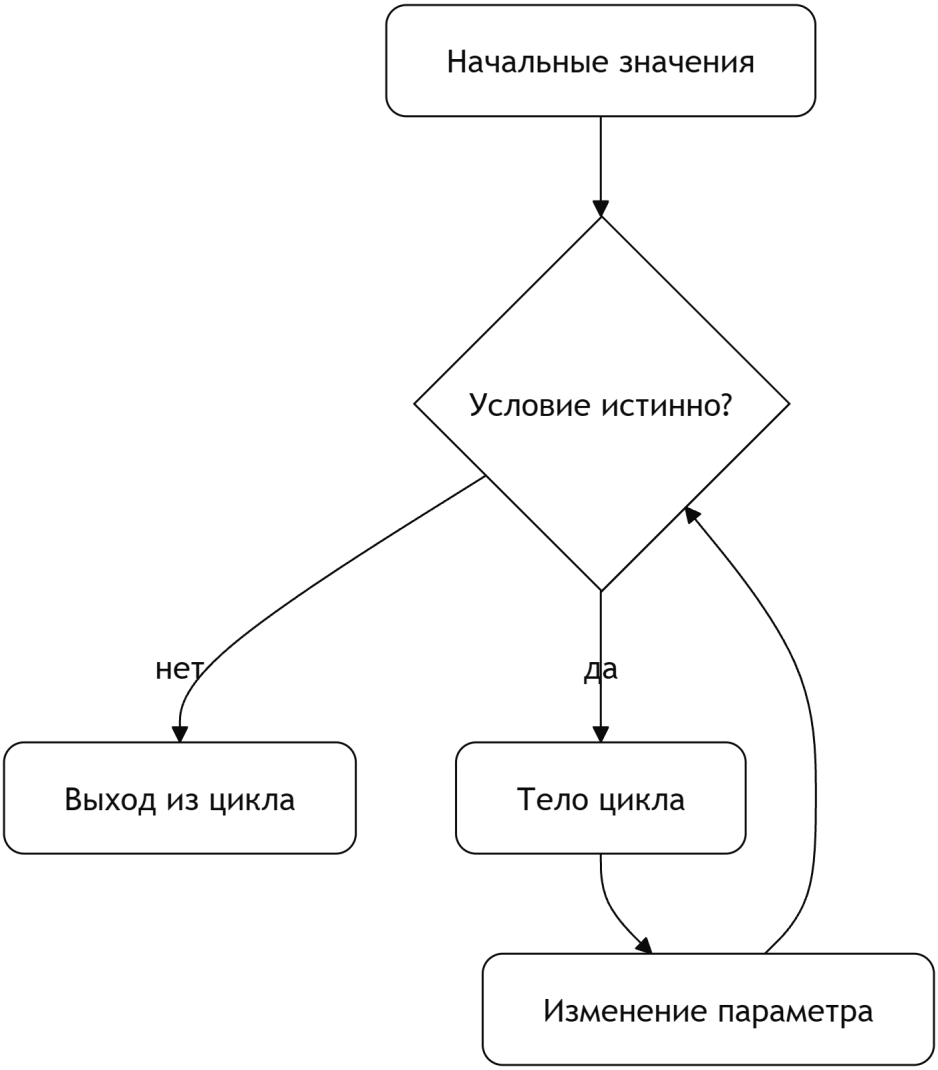
Цикл может не выполниться ни разу	Тело выполняется хотя бы один раз
Подходит, если начальное условие может быть ложным	Подходит, если одно выполнение необходимо всегда

```
x = x0;
while (condition) {
// тело цикла
 x += dx;
}
```

```
x = x0;
do {
// тело цикла
 x += dx;
} while (condition);
```

Условием может быть неравенство или любое выражение. В C++ нулевое значение трактуется как ложь, ненулевое — как истина. Поэтому `while (1)` или `while (true)` создаёт бесконечный цикл, если внутри нет механизма выхода.

В блок-схеме цикл должен образовывать замкнутый контур: стрелка возвращается к проверке условия.



Если в теле больше одного оператора, используются фигурные скобки. Даже когда скобки формально необязательны для одного оператора, их применение делает границы цикла очевидными.

Анализ циклической задачи

Перед написанием кода нужно определить пять элементов:

1. какие данные заданы первоначально;
2. что именно повторяется;
3. какое условие продолжает или завершает цикл;
4. какой параметр изменяется на каждой итерации;
5. что вычисляется и выводится после завершения цикла.

Среднее арифметическое последовательности

Пользователь вводит целые числа. После каждого числа программа спрашивает, нужно ли продолжать ввод. Требуется найти среднее арифметическое.

Подход:

- `sum = 0` — начальная сумма;
- `count = 0` — количество введённых чисел;
- тело цикла: ввести число, прибавить его к `sum`, увеличить `count` на единицу;
- считать символ продолжения `ch`;
- повторять цикл, если введено `y` или `Y`;
- после цикла вычислить и вывести результат.

Здесь выбирается `do...while`: пользователь должен сначала ввести хотя бы одно число, и только потом программа спрашивает, продолжать ли ввод.

```
int value;
int sum = 0;
int count = 0;
char ch;

do {
 std::cin >> value;
 sum += value;
++count;

 std::cin >> ch;
} while (ch == 'y' || ch == 'Y');

std::cout << static_cast<double>(sum) / count;
```

Среднее вычисляется **после цикла**, потому что только тогда известны итоговые сумма и количество. Выражение внутри цикла пересчитывать не нужно.

Если `sum` и `count` имеют тип `int`, обычное деление будет целочисленным: $5/2 = 2$. Чтобы получить 2.5, хотя бы один операнд нужно привести к вещественному типу:

$$\overline{x} = \frac{S}{N}$$

Частые ошибки

- `=` выполняет присваивание, а `==` сравнивает значения;
- `char` хранит один символ, а для строки используется строковый тип;
- при нескольких операторах тела нужны фигурные скобки;

- после условия `do...while` ставится точка с запятой;
- кавычки нужны для вывода текстового литерала, но не для вывода значения переменной;
- в условии продолжения нужно явно указать все допустимые варианты, например `ch == 'y' || ch == 'Y'`.

Пороговое условие

Задача о тренировочном пробеге

В первый день спортсмен пробегает расстояние x. Каждый следующий день пробег увеличивается на **10%**. Нужно определить номер дня, когда расстояние станет не меньше заданного значения y.

Число дней заранее неизвестно, поэтому используется цикл с условием. Сначала спортсмен уже пробежал первый день:

- `x` — текущая дистанция, обычно `double`;
- `y` — требуемая дистанция;
- `k = 1` — номер текущего дня.

Пока $x < y$, выполняются действия:

- увеличить пробег: $x \gets 1.1x$;
- увеличить номер дня: `k = k + 1`.

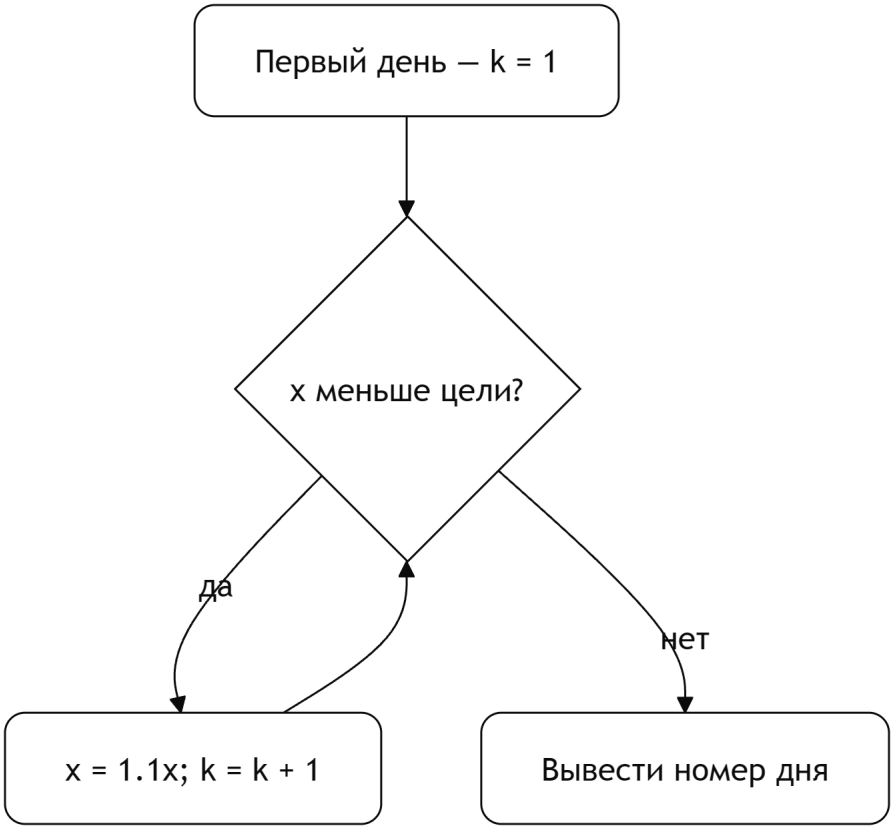
Когда $x \geq y$, цикл завершается, и выводится `k`.

```
int k = 1;

while (x < y) {
 x *= 1.1;
++k;
}

std::cout << k;
```

Нужен именно `while`, потому что начальное значение уже может удовлетворять цели. Если спортсмен в первый день пробежал достаточно, ответ должен быть `1`, а дополнительная итерация не нужна.



Это пример общего правила: если начальное состояние может сразу привести к завершению, выбирается цикл с предусловием.

Делители и сравнение циклов

Поиск всех делителей

Для заданного целого числа `n` можно последовательно перебрать значения от `1` до `n`. Число `i` является делителем, если остаток от деления равен нулю:

```
n % i == 0
```

Простой вариант с `while`:

```
int i = 1;

while (i <= n) {
 if (n % i == 0) {
 std::cout << i << '\\n';
}
++i;
}
```

Тот же алгоритм с `do...while` отличается только положением проверки:

```
int i = 1;

do {
 if (n % i == 0) {
 std::cout << i << '\\n';
}
++i;
} while (i <= n);
```

В обоих вариантах одинаковы ввод данных, начальное значение `i`, проверка делимости и увеличение `i`. Различается только момент проверки условия. Для простого решения перебираются все числа до `n`; более эффективные варианты могут ограничивать перебор квадратным корнем числа.

Проверка программ

Для циклов обязательны тестовые примеры. Полезно проверять граничные случаи:

- начальное значение уже удовлетворяет условию — цикл `while` не выполняется;
- тело должно выполниться хотя бы один раз — нужен `do...while`;
- результат деления не является целым — проверяется приведение типа;
- для поиска делителей проверяется число, у которого есть несколько делителей, включая само число.

Упомянутый справочный ресурс — методичка с перечнем операций и примерами их применения.