

# Циклические конструкции

Михаил Марков  
C++ - Разработчик



# Проверка связи



Поставьте “+”, если меня видно и слышно



**Если у вас нет звука:**

- убедитесь, что на вашем устройстве и на колонках включен звук
- обновите страницу вебинара (или закройте страницу и заново присоединитесь к вебинару)
- откройте вебинар в другом браузере
- перезагрузите компьютер (ноутбук) и заново попытайтесь зайти

# Михаил Марков

О спикере:

C++-разработчик, фрилансер

- Разработка алгоритма для релевантной выдачи объявлений.
- Разработка эмуляторов оборудования



# Вспоминаем прошрое занятие

Вопрос: какие операторы **ветвления**  
вы знаете?



# Вспоминаем прошрое занятие

Вопрос: какие операторы **ветвления**  
вы знаете?

**Ответ:** if...else, switch...case



# Вспоминаем прошрое занятие

Вопрос: какие операторы сравнения  
вы знаете?



# Вспоминаем прошрое занятие

Вопрос: какие операторы сравнения  
вы знаете?

Ответ:

- больше:  $>$ ; больше или равно:  $>=$
- меньше:  $<$ ; меньше или равно:  $<=$
- равно:  $==$ ; не равно:  $!=$



# Вспоминаем прошрое занятие

Вопрос: какие логические операторы  
вы знаете?





# Вспоминаем прошлое занятие

Вопрос: какие логические операторы  
вы знаете?

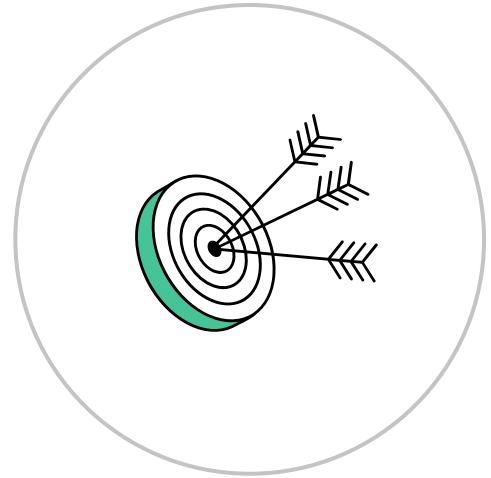
Ответ:

- логическое не: !
- логическое и: &&
- логическое или: ||



# Цели занятия

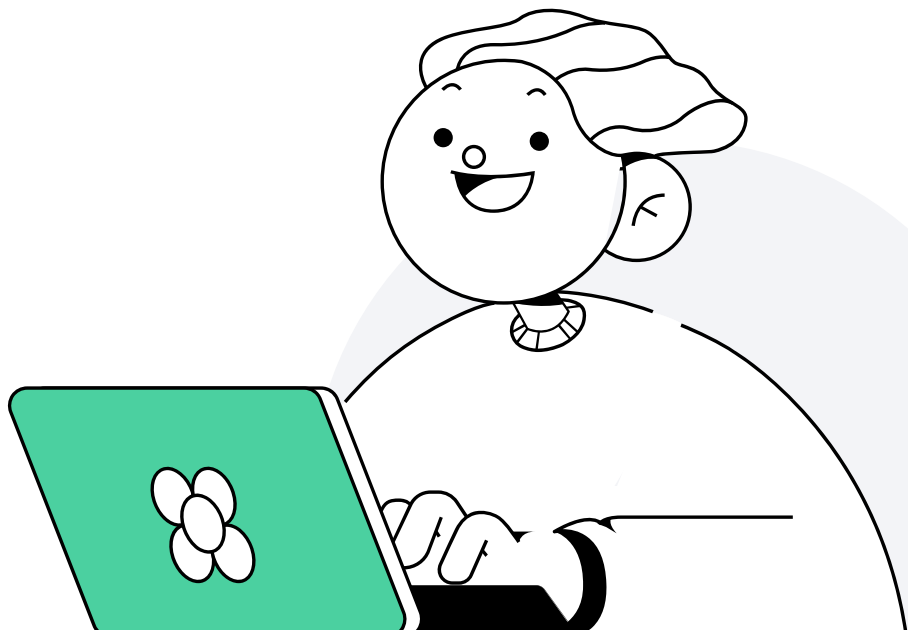
- Научимся решать проблему выполнения большого количества однотипных действий
- Сможем выполнять произвольное количество действий с помощью небольшого количества кода
- Рассмотрим, как создавать циклические конструкции



# План занятия

- 1 Циклы
- 2 Оператор continue
- 3 Оператор break
- 4 Вложенные циклы
- 5 Итоги
- 6 Домашнее задание

\*Нажмите на раздел для перехода



# Циклы

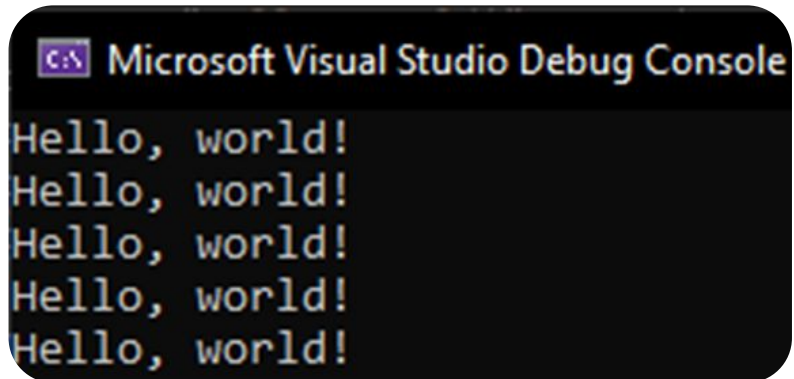


1

# Проблема

Задача: вывести пять раз на консоль фразу “Hello, World!”

```
std::cout << "Hello, world!" << std::endl;  
std::cout << "Hello, world!" << std::endl;  
std::cout << "Hello, world!" << std::endl;  
std::cout << "Hello, world!" << std::endl;  
std::cout << "Hello, world!" << std::endl;
```



C:\> Microsoft Visual Studio Debug Console

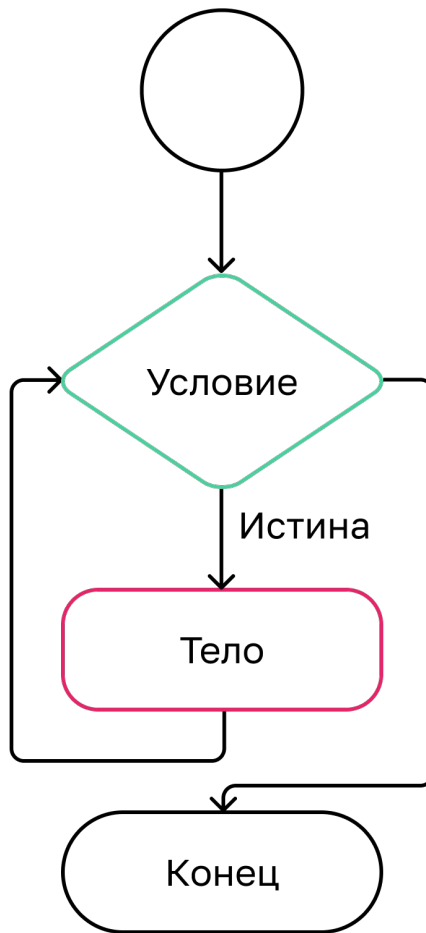
```
Hello, world!  
Hello, world!  
Hello, world!  
Hello, world!  
Hello, world!
```

# Цикл

Цикл — это структура кода, позволяющая выполнить некоторые действия несколько раз.

- Условие определяет, продолжать выполнение цикла или нет
- Тело цикла определяет действия, которые необходимо выполнить на каждом шаге
- Шаг цикла — однократное выполнение тела цикла

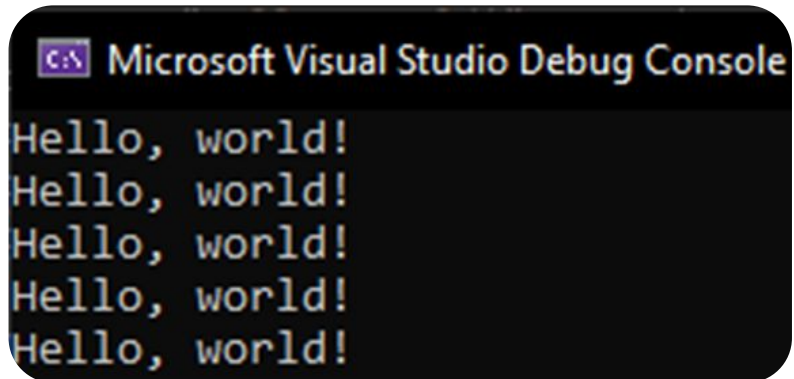
```
while(условие)  
    тело цикла;
```



# Решение

Задача: вывести пять раз на консоль фразу “Hello, World!”

```
int count = 5;
while (count-- > 0)
{
    std::cout << "Hello, world!" << std::endl;
}
```



C:\> Microsoft Visual Studio Debug Console

Hello, world!  
Hello, world!  
Hello, world!  
Hello, world!  
Hello, world!

# Виды циклов

Существуют три основных вида циклов:

- 1 Цикл с предусловием:
  - условие проверяется перед выполнением тела цикла
  - тело цикла может не выполниться ни разу
- 2 Цикл с постусловием:
  - условие проверяется после выполнения тела цикла
  - тело цикла выполнится хотя бы один раз
- 3 Цикл с подсчётом:
  - выполняется определённое количество раз
  - параметр, условие и изменения которого задаются в начале цикла



# Цикл с предусловием

**Применение:** когда вы не знаете заранее, сколько шагов нужно будет выполнить.

**Задача:** при помощи ведра наполнить бочку водой из реки.

## Алгоритм:

- 1 Проверить, заполнена ли бочка
- 2 Сходить с пустым ведром на реку
- 3 Наполнить ведро водой из реки
- 4 Вернуться с полным ведром к бочке
- 5 Вылить воду из ведра в бочку
- 6 Перейти к пункту 1

# Цикл с предусловием

**Задача:** при помощи ведра наполнить бочку водой из реки.

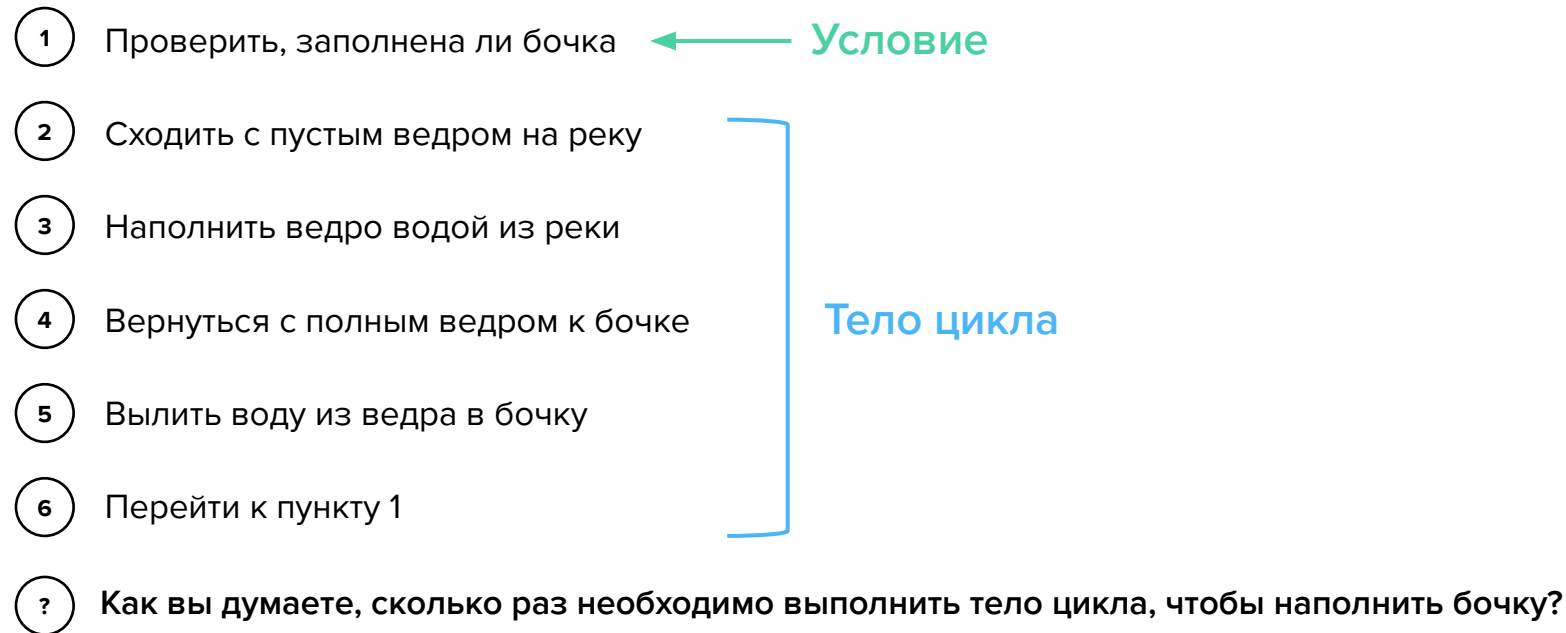
**Алгоритм:**

- 1 Проверить, заполнена ли бочка ← **Условие**
- 2 Сходить с пустым ведром на реку
- 3 Наполнить ведро водой из реки
- 4 Вернуться с полным ведром к бочке
- 5 Вылить воду из ведра в бочку
- 6 Перейти к пункту 1

# Цикл с предусловием

**Задача:** при помощи ведра наполнить бочку водой из реки.

**Алгоритм:**

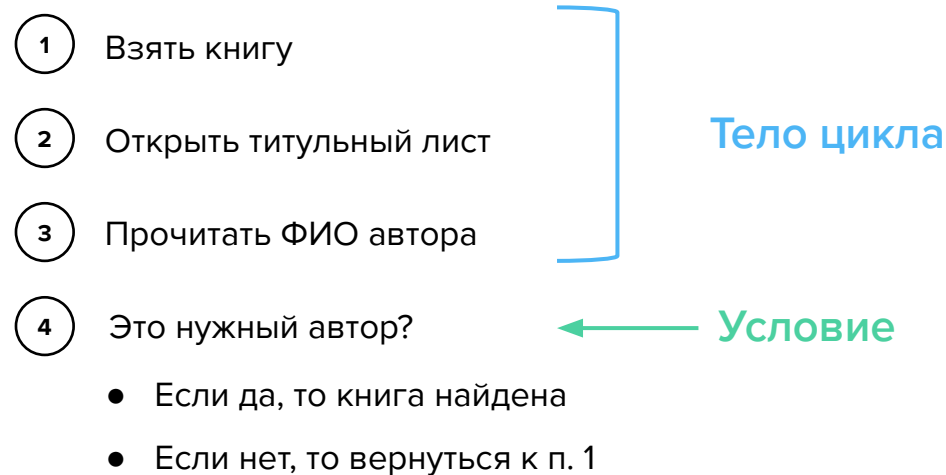
- ① Проверить, заполнена ли бочка ← **Условие**
  - ② Сходить с пустым ведром на реку
  - ③ Наполнить ведро водой из реки
  - ④ Вернуться с полным ведром к бочке
  - ⑤ Вылить воду из ведра в бочку
  - ⑥ Перейти к пункту 1
  - ⑦ Как вы думаете, сколько раз необходимо выполнить тело цикла, чтобы наполнить бочку?
- 

# Цикл с постусловием

**Применение:** когда вы не знаете заранее, сколько шагов нужно будет выполнить, но знаете, что потребуется хотя бы один.

**Задача:** найти на книжной полке книгу Айзека Азимова.

**Алгоритм:**

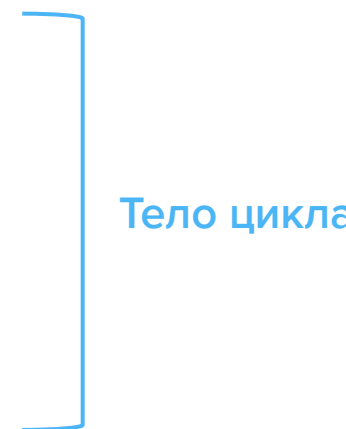


# Цикл с подсчётом

**Применение:** когда точно известно, сколько раз необходимо выполнить действие.

**Задача:** положить в корзину десять яблок.

## Алгоритм:

- ① Установить счётчик яблок в корзине равным 0 ← **Условие**
  - ② Если счётчик меньше десяти
  - ③ Взять яблоко
  - ④ Положить одно яблоко в корзину
  - ⑤ Увеличить счётчик на единицу
  - ⑥ Перейти к п. 2
- 

# Цикл с предусловием

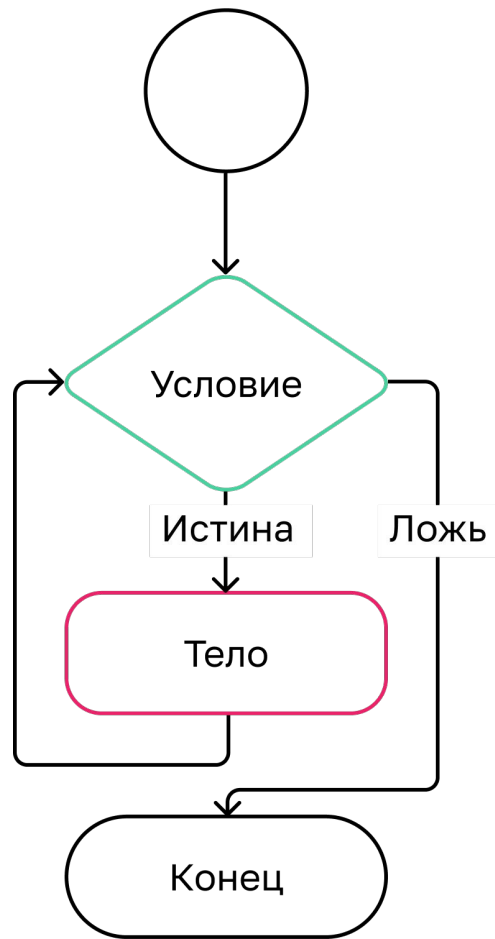


1.1

# Цикл while

Оператор `while` выполняет оператор или блок операторов, пока логическое выражение равно значению `true`. Так как это выражение вычисляется и проверяется перед каждым выполнением цикла, цикл `while` выполняется ноль или несколько раз

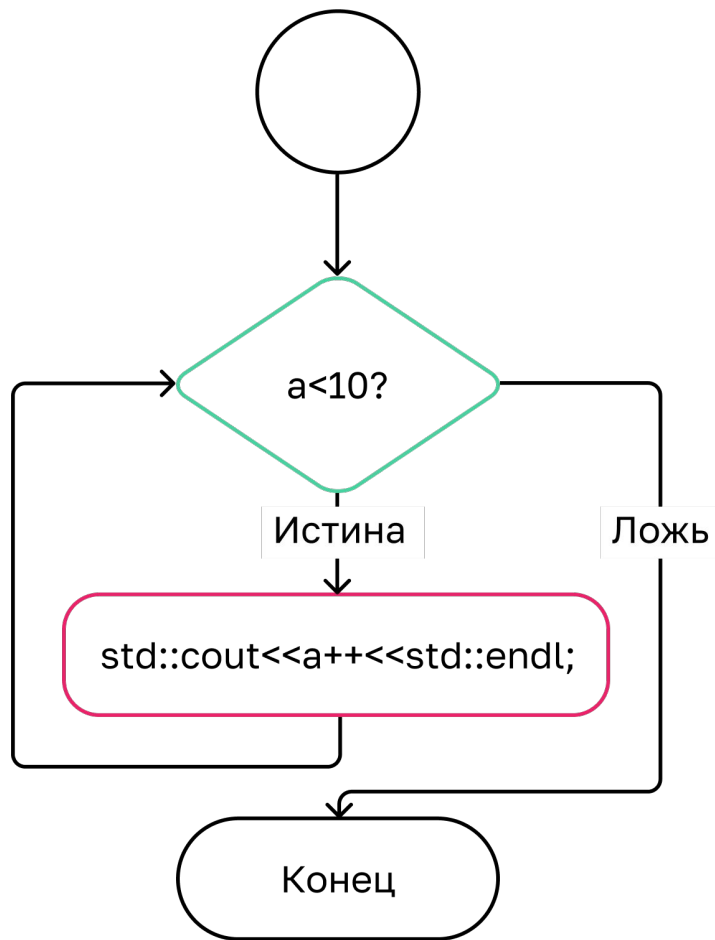
```
while (условие)
{
    тело цикла;
}
```



# Цикл while

Оператор `while` выполняет оператор или блок операторов, пока логическое выражение равно значению `true`. Так как это выражение вычисляется и проверяется перед каждым выполнением цикла, цикл `while` выполняется ноль или несколько раз

```
int a = 0;
while (a < 10)
{
    std::cout << a++<< std::endl;
}
```





# Пример цикла

Для чисел, квадрат которых меньше 100, вывести на экран число и его квадрат.

```
int a = 1;
while (a * a < 100)
{
    std::cout << a << " " << a * a << std::endl;
    a++;
}
```

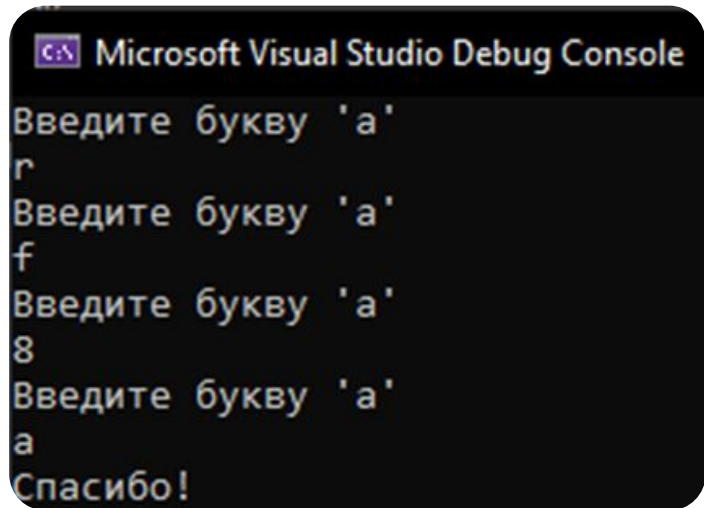
A screenshot of the Microsoft Visual Studio Debug Console. The title bar at the top reads "Microsoft Visual Studio Debug Console". The console displays the output of the program, showing the first nine iterations of the while loop. Each line contains a number followed by a space and then its square. The numbers are 1 through 9, and their squares are 1, 4, 9, 16, 25, 36, 49, 64, and 81 respectively. The text is displayed in a light blue font on a dark background.

```
C:\> Microsoft Visual Studio Debug Console
1 1
2 4
3 9
4 16
5 25
6 36
7 49
8 64
9 81
```

# Пример: цикл пользовательского ввода

Запросить у пользователя ввести целое число. При некорректном вводе запросить повторный ввод

```
std::string s;  
std::cout << "Введите букву 'a'" << std::endl;  
std::cin >> s;  
while (s != "a")  
{  
    std::cout << "Введите букву 'a'" << std::endl;  
    std::cin >> s;  
}  
std::cout << "Спасибо!" << std::endl;
```



Напишем программу  
с помощью while,  
которая инвертирует  
целое число

123 —————> 321

Готовый пример кода

```
const log = require('log');
let embed;

function transform(t) {
  // Promise.resolve to promise
  return transform(t);
}

function removeLinkHeader(prev) {
  return prev.then(() => {
    $(':header').each(() => {
      const children = $(header).children();
      if ($(children).length) {
        $(header).text(children);
        $(children).remove();
      }
    });
    return header;
  });
}
```

# Цикл с постусловием

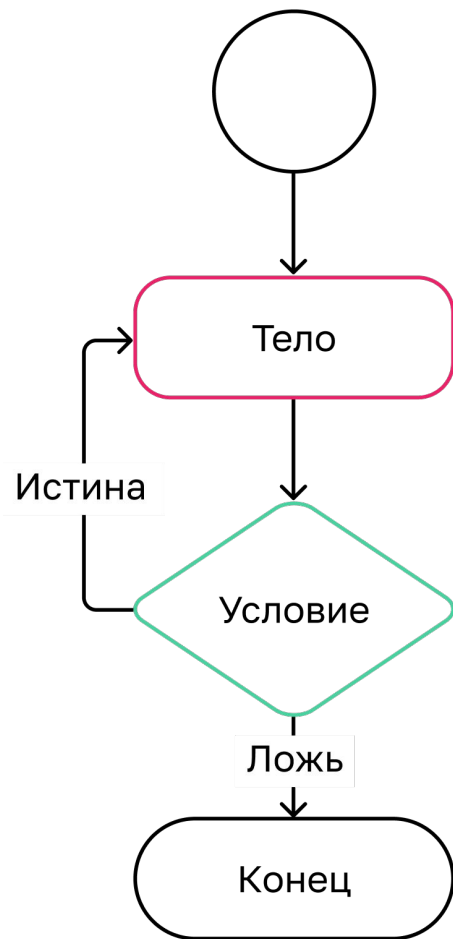


1.2

# Цикл do...while

Оператор **do** выполняет оператор или блок операторов, пока определённое логическое выражение равно значению **true**. Так как это выражение вычисляется и проверяется после каждого выполнения цикла, цикл **do...while** выполняется один или несколько раз

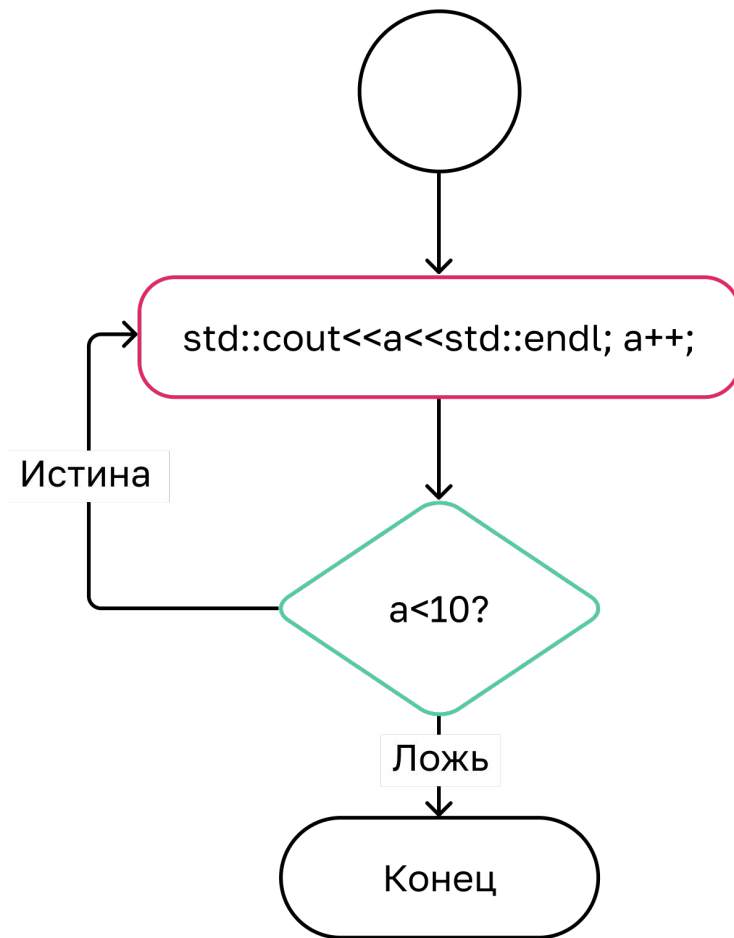
```
do
{
    тело цикла
} while (условие);
```



# Цикл do...while

Оператор `do` выполняет оператор или блок операторов, пока определённое логическое выражение равно значению `true`. Так как это выражение вычисляется и проверяется после каждого выполнения цикла, цикл `do...while` выполняется один или несколько раз

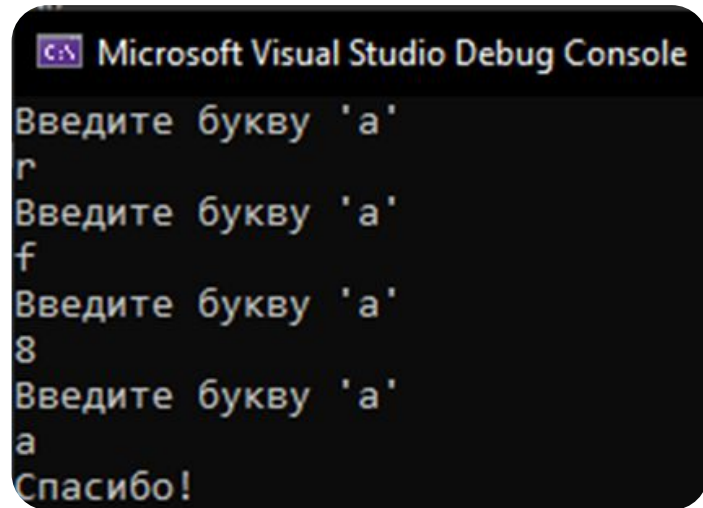
```
int a = 0;
do
{
    std::cout << a << std::endl;
    a++;
} while (a < 10);
```



# Пример цикла пользовательского ввода

Попросить пользователя ввести целое число. При некорректном вводе запросить повторный ввод

```
std::string s;  
do  
{  
    std::cout << "Введите букву 'a'" << std::endl;  
    std::cin >> s;  
} while (s != "a");  
std::cout << "Спасибо!" << std::endl;
```



# Сравнение while и do...while

## while

```
std::string s;  
std::cout << "Введите букву 'a'" << std::endl;  
std::cin >> s;  
while (s != "a")  
{  
    std::cout << "Введите букву 'a'" << std::endl;  
    std::cin >> s;  
}  
std::cout << "Спасибо!" << std::endl;
```

## do...while

```
std::string s;  
do  
{  
    std::cout << "Введите букву 'a'" << std::endl;  
    std::cin >> s;  
} while (s != "a");  
std::cout << "Спасибо!" << std::endl;
```



Напишем программу  
с помощью do...while  
и проверим,  
что пользователь ввёл  
корректный номер  
месяца

Готовый пример кода

```
const log = require('log');
let embed;

function transform($, transform) {
  // Promise.resolve to promise
  return transform($);
}

function removeLinkHeader($, prev) {
  return prev.then(() => {
    $(':header').each((i, header) => {
      const children = $(header).children();
      if ($(children).length) {
        $(header).text(children);
        $(children).remove();
      }
    });
    return header;
  });
}
```

# Цикл с подсчётом



1.3

# Цикл for

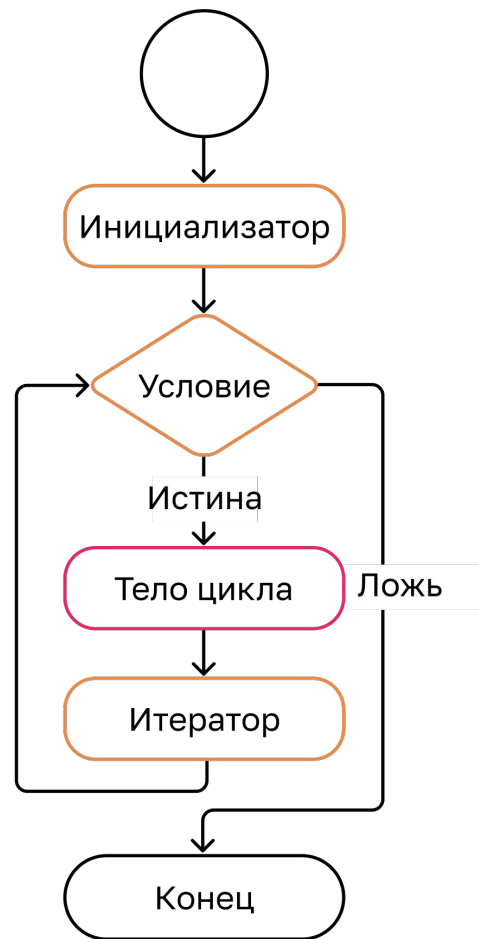
Оператор **for** выполняет оператор или блок операторов, пока определённое логическое выражение равно значению **true**.

```
for (инициализатор; условие; итератор)
{
    тело цикла
}
```

**Инициализатор** выполняется один раз перед входом в цикл.

**Условие** — условное выражение, вычисляется и проверяется перед каждым шагом.

**Итератор** выполняется после каждого шага цикла



# Цикл for

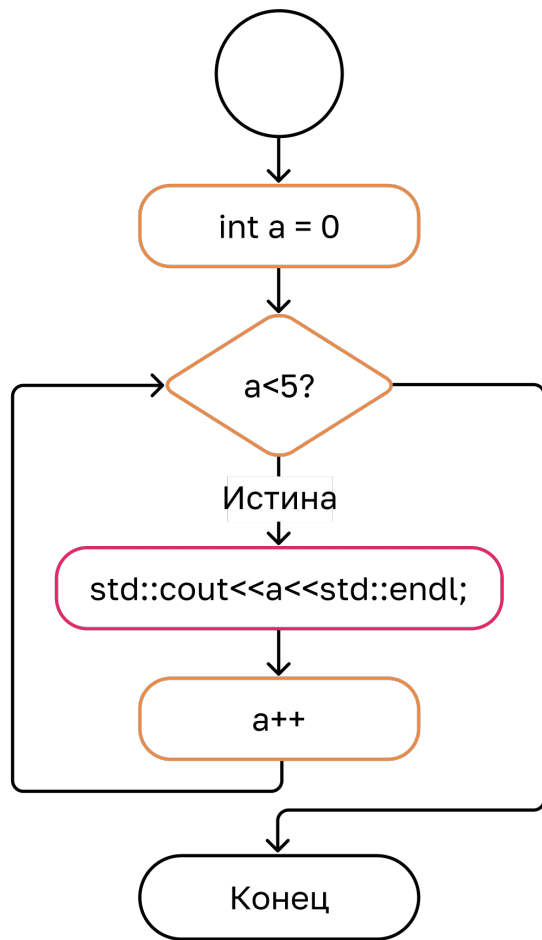
Оператор **for** выполняет оператор или блок операторов, пока определённое логическое выражение равно значению **true**.

```
for (int a = 0; a < 5; a++)  
{  
    std::cout << a << std::endl;  
}
```

**Инициализатор** выполняется один раз перед входом в цикл.

**Условие** — условное выражение, вычисляется и проверяется перед каждым шагом.

**Итератор** выполняется после каждого шага цикла



Напишем программу  
с помощью for, которая  
выводит степени  
(квадрат, куб, четыре)  
введённого  
пользователем числа

Готовый пример кода

```
const log = require('log');
let embed;

function transform(x, n) {
  // Promise.resolve to promise
  return transform(x, n + 1);
}

function removeLinkHeader(prev) {
  return prev.then(() => {
    $(':header').remove();
    const children = $(':header').children();
    if (children.length) {
      $(header).text(children);
      $(children).remove();
    }
    return header;
  });
}
```

# Оператор continue

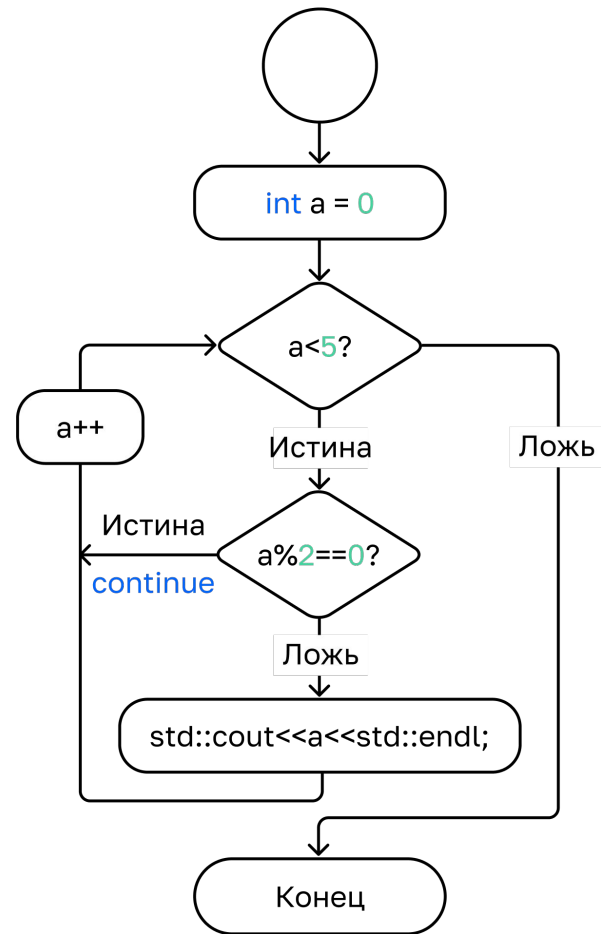


2

# Оператор continue

Оператор `continue` передаёт управление следующей итерации цикла `while`, `do-while` или `for`, в котором она встречается

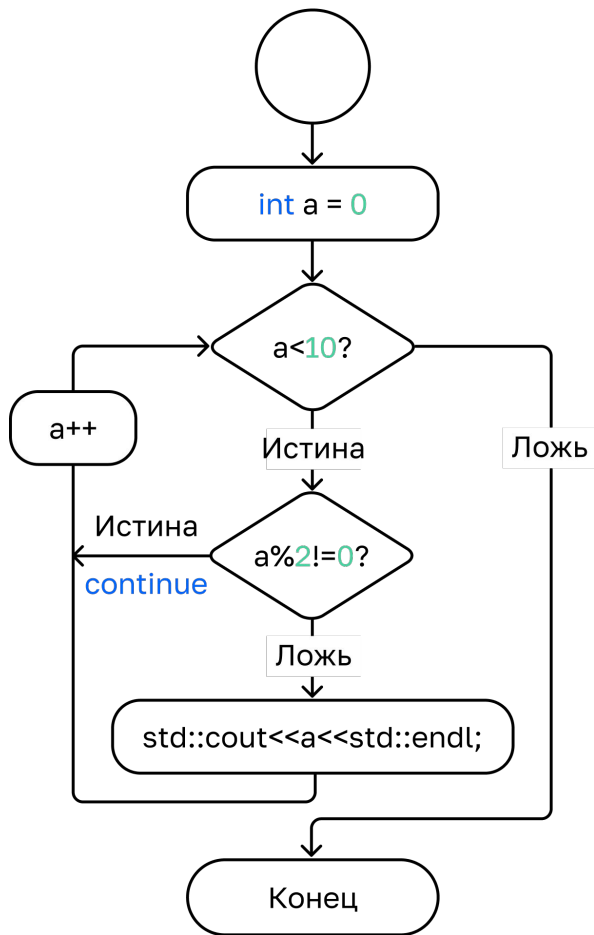
```
for (int a = 0; a < 5; a++)  
{  
    if (a % 2 == 0)  
    {  
        continue;  
    }  
    std::cout << a << std::endl;  
}
```



# Оператор continue

Оператор `continue` передаёт управление следующей итерации цикла `while`, `do-while` или `for`, в котором она встречается

```
int a = 0;
while (a < 10)
{
    if (a % 2 != 0)
    {
        a++;
        continue;
    }
    std::cout << a << std::endl;
    a++;
}
```





# Оператор break

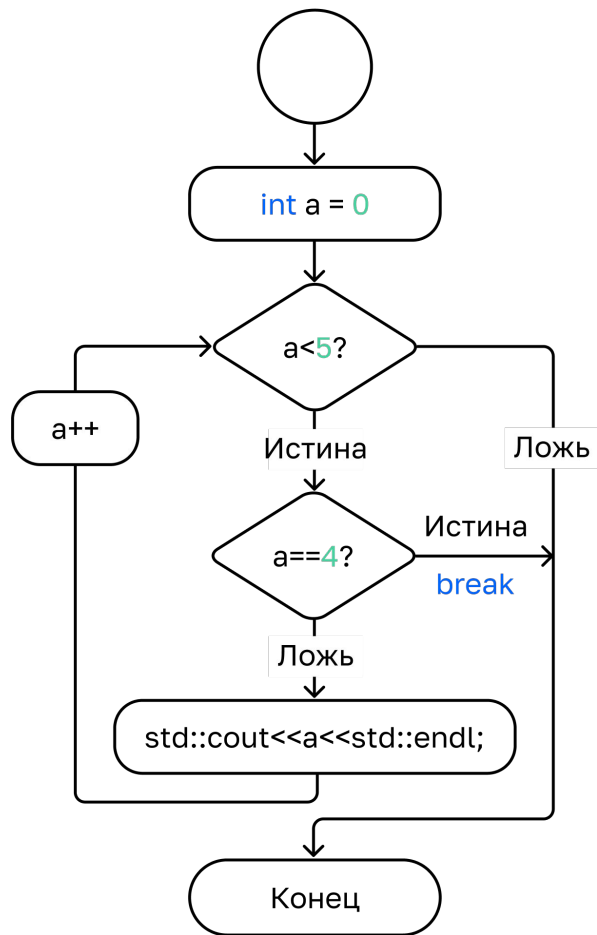


3

# Оператор break

Оператор `break` завершает выполнение цикла `while`, `do-while` или `for`, в котором он встречается. Управление передаётся оператору, который расположен после завершённого цикла

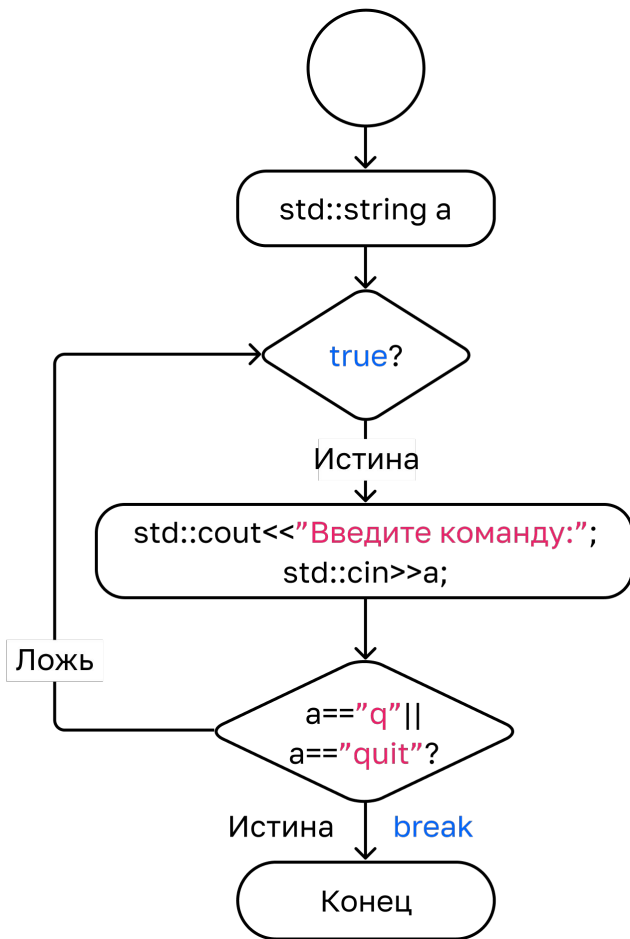
```
for (int a = 0; a < 5; a++)  
{  
    if (a == 4)  
    {  
        break;  
    }  
    std::cout << a << std::endl;  
}
```



# Оператор break и бесконечный цикл while

Бесконечным является цикл, у которого нет явного условия выхода

```
std::string a;  
while (true)  
{  
    std::cout << "Введите команду: ";  
    std::cin >> a;  
    if (a == "q" || a == "quit")  
    {  
        break;  
    }  
}
```



# Вложенные циклы



4

# Вложенный цикл

**Применение:** когда на каждом шаге одного цикла нужно выполнить другой цикл.

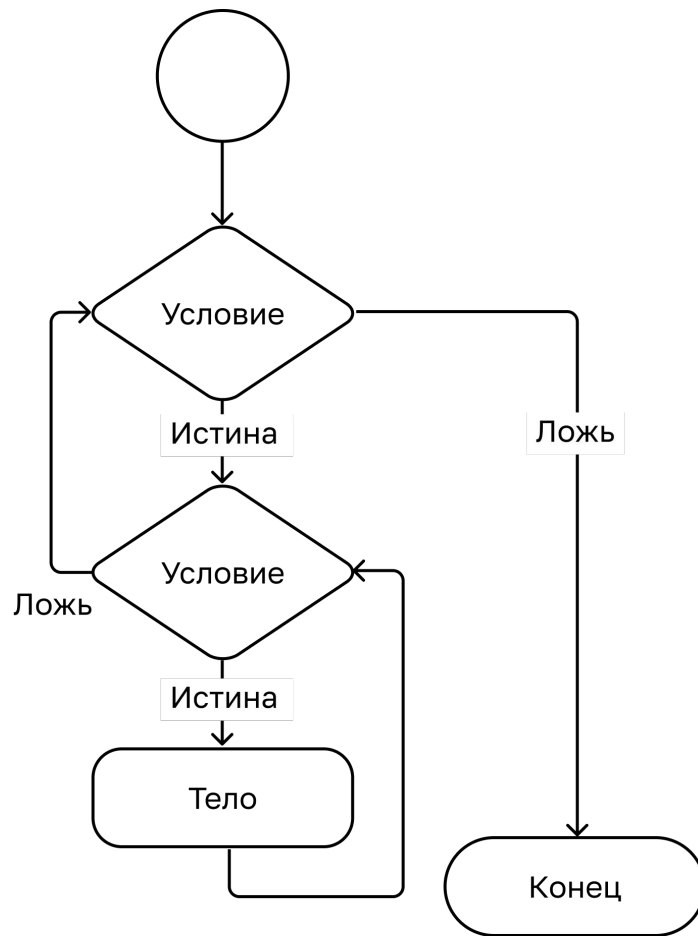
**Задача:** когда на каждом шаге одного цикла нужно выполнить другой цикл.

## Алгоритм:

- ① Подойти к следующей пустой бочке
  - Проверить, заполнена ли бочка
  - Проверить, заполнена ли бочка
  - Наполнить ведро водой из реки
  - Вернуться с полным ведром к бочке
  - Вылить воду из ведра в бочку
  - Перейти к пункту а
- ② Перейти к пункту 1

# Вложенный цикл

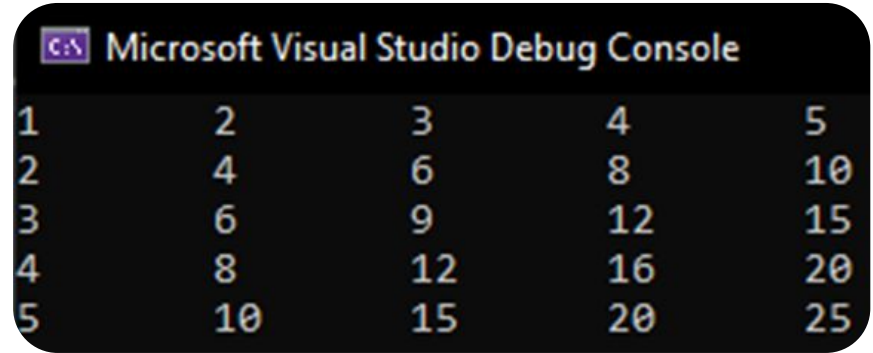
Цикл, размещённый в теле другого цикла, называется вложенным



# Вложенный цикл

Программа, которая выводит на консоль таблицу умножения заданного размера

```
int size = 5;
for (int i = 1; i <= size; i++)
{
    for (int j = 1; j <= size; j++)
    {
        std::cout << i * j << "\t";
    }
    std::cout << std::endl;
}
```



The screenshot shows the Microsoft Visual Studio Debug Console with a multiplication table for size 5. The table is displayed with rows and columns numbered 1 to 5. The values are the products of the row and column indices.

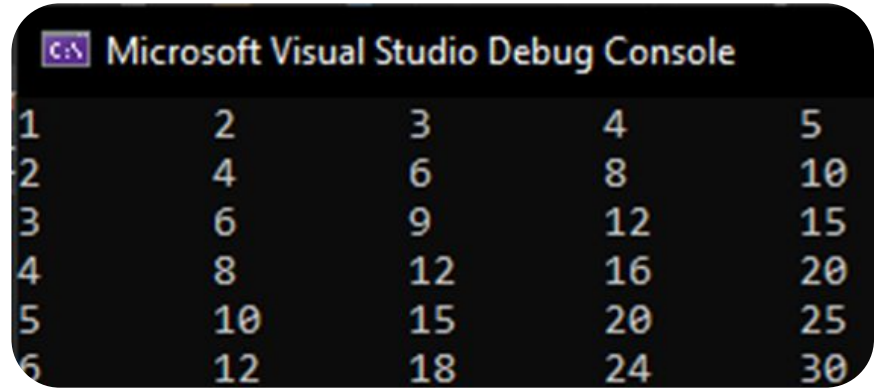
1	2	3	4	5
2	4	6	8	10
3	6	9	12	15
4	8	12	16	20
5	10	15	20	25

\* В качестве самостоятельной работы добавьте в консольный вывод заголовки колонок и строк, чтобы было понятно, что на что умножается

# Вложенный цикл

Программа, которая выводит на консоль таблицу умножения заданного размера

```
int n = 6;
for (int i = 1; i <= n; i++)
{
    while (j < n)
    {
        std::cout << i * j << "\t";
        j++;
    }
    std::cout << std::endl;
}
```



The screenshot shows the Microsoft Visual Studio Debug Console with a multiplication table for n=6. The table is displayed with rows and columns separated by tabs, matching the code output. The values range from 1 to 30.

1	2	3	4	5
2	4	6	8	10
3	6	9	12	15
4	8	12	16	20
5	10	15	20	25
6	12	18	24	30

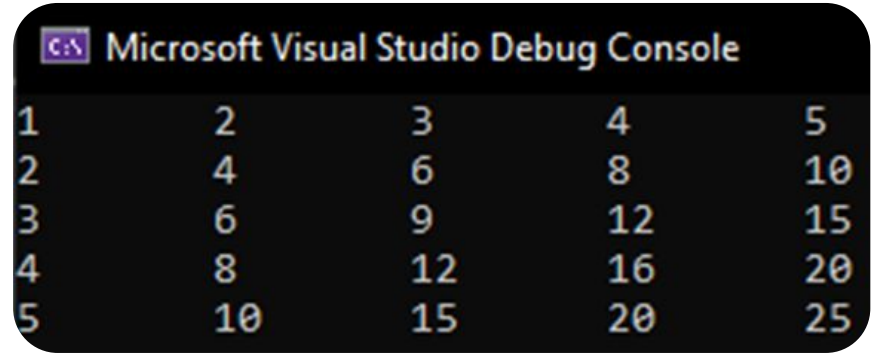
\* В качестве самостоятельной работы добавьте в консольный вывод заголовки колонок и строк, чтобы было понятно, что на что умножается



# Вложенный цикл

Программа, которая выводит на консоль таблицу умножения заданного размера

```
int n = 6;  
int i = 1;  
while(i < n)  
{  
    int j = 1;  
    while(j < n)  
    {  
        std::cout << i * j << "\t";  
        j++;  
    }  
    std::cout << std::endl;  
    i++;  
}
```



The screenshot shows the Microsoft Visual Studio Debug Console with a 5x5 multiplication table. The table is displayed with rows and columns numbered 1 to 5. The values are the products of the row and column indices.

1	2	3	4	5
2	4	6	8	10
3	6	9	12	15
4	8	12	16	20
5	10	15	20	25

\* В качестве самостоятельной работы добавьте в консольный вывод заголовки колонок и строк, чтобы было понятно, что на что умножается

# Итоги

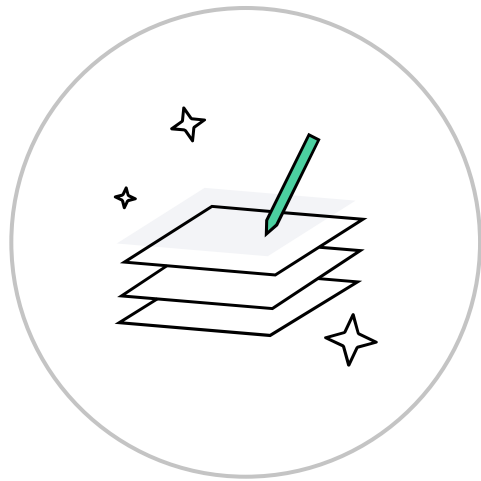
- 1 Научились выполнять произвольное количество действий с помощью небольшого количества кода
- 2 Узнали, что такое циклы, и разобрались с тем, как записывать цикл `while`, `do..while` и `for`
- 3 Научились пользоваться операторами `break` и `continue` для управления циклом



# Домашнее задание

Давайте посмотрим ваше домашнее задание.

- 1 Вопросы по домашней работе задавайте в чате группы
- 2 Задачи можно сдавать по частям
- 3 Зачёт по домашней работе ставят после того, как приняты все задачи



# Дополнительные материалы

- Циклы `while`, `for`
- Вложенные циклы



# Задавайте вопросы и пишите отзыв о лекции

Михаил Марков  
C++ - разработчик

