

Ravesli [Ravesli](#)

- [Уроки по C++](#)
- [OpenGL](#)
- [SFML](#)
- [Qt5](#)
- [RegExр](#)
- [Ассемблер](#)
- [Купить .PDF](#)




## Урок №69. Цикл for

 [Юрий](#) |

- [Уроки C++](#)

|

 Обновл. 15 Сен 2020 |

 84117

[↑](#)  39

Безусловно, наиболее используемым циклом в языке C++ является цикл for.

Оглавление:

1. [Цикл for](#)
2. [Выполнение цикла for](#)
3. [Еще примеры циклов for](#)
4. [Ошибка неучтенной единицы](#)
5. [Пропущенные выражения в цикле](#)
6. [Объявление переменных в цикле for](#)
7. [Использование циклов for в старых версиях C++](#)
8. [Вложенные циклы for](#)
9. [Заключение](#)
10. [Тест](#)

## Цикл for

Цикл for в языке C++ идеален, когда известно необходимое количество итераций. Выглядит он следующим образом:

for (объявление переменных; условие; инкремент/декремент счетчика)  
    тело цикла;

Или, преобразуя for в эквивалентный цикл while:

```
{ // обратите внимание, цикл находится в блоке
  объявление переменных;
  while (условие)
  {
    тело цикла;
    инкремент/декремент счетчика;
  }
} // переменные, объявленные внутри цикла, выходят из области видимости здесь
```

Переменные, определенные внутри цикла for, имеют специальный тип области видимости: **область видимости цикла**. Такие переменные существуют только внутри цикла и недоступны за его пределами.

## Выполнение цикла for

Цикл for в C++ выполняется в 3 шага:

- ➔ **Шаг №1: Объявление переменных.** Как правило, здесь выполняется определение и инициализация счетчиков цикла, а точнее — одного счетчика цикла. Эта часть выполняется только один раз, когда цикл выполняется впервые.
- ➔ **Шаг №2: Условие.** Если оно равно false, то цикл немедленно завершает свое выполнение. Если же условие равно true, то выполняется тело цикла.
- ➔ **Шаг №3: Инкремент/декремент счетчика цикла.** Переменная увеличивается или уменьшается на единицу. После этого цикл возвращается к шагу №2.

Рассмотрим пример цикла for и разберемся детально, как он работает:

```
1 #include <iostream>
2
3 int main()
4 {
5     for (int count = 0; count < 10; ++count)
6         std::cout << count << " ";
7
8     return 0;
9 }
```

Сначала мы объявляем переменную count и присваиваем ей значение 0. Далее проверяется условие count < 10, а так как count равен 0, то условие 0 < 10 имеет значение true. Следовательно, выполняется тело цикла, в котором мы выводим в консоль переменную count (т.е. значение 0).

Затем выполняется выражение `++count`, т.е. **инкремент** переменной. Затем цикл снова возвращается к проверке условия. Условие `1 < 10` имеет значение `true`, поэтому тело цикла выполняется опять. Выводится 1, а переменная `count` увеличивается уже до значения 2. Условие `2 < 10` является истинным, поэтому выводится 2, а `count` увеличивается до 3 и так далее.

В конце концов, `count` увеличивается до 10, а условие `10 < 10` является ложным, и цикл завершается. Следовательно, результат выполнения программы:

0 1 2 3 4 5 6 7 8 9

Циклы `for` могут быть несколько сложны для новичков, однако опытные кодеры любят их, так как эти циклы очень компактны и удобны. Для наглядности, давайте преобразуем цикл `for`, приведенный выше, в эквивалентный цикл `while`:

```
1 #include <iostream>
2
3 int main()
4 {
5     { // внешние скобки нужны для обеспечения области видимости цикла
6         int count = 0;
7         while (count < 10)
8         {
9             std::cout << count << " ";
10            ++count;
11        }
12    }
13
14    return 0;
15 }
```

Обратите внимание, внешние фигурные скобки здесь необходимы, так как переменная `count` выходит из области видимости при завершении цикла.

## Еще примеры циклов for

Давайте, используя цикл `for`, напомним функцию вычисления значений в степени `n`:

```
1 int pow(int base, int exponent)
2 {
3     int total = 1;
4
5     for (int count=0; count < exponent; ++count)
6         total *= base;
7
8     return total;
9 }
```

Функция возвращает значение `baseexponent` (число в степени `n`). `base` — это число, которое нужно возвести в степень, а `exponent` — это степень, в которую нужно возвести `base`.

- Если экспонент равен 0, то цикл for выполняется 0 раз, и функция возвращает 1.
- Если экспонент равен 1, то цикл for выполняется 1 раз, и функция возвращает  $1 * \text{base}$ .
- Если экспонент равен 2, то цикл for выполняется 2 раза, и функция возвращает  $1 * \text{base} * \text{base}$ .

Хотя в большинстве циклов используется инкремент счетчика, мы также можем использовать и декремент счетчика:

```
1 #include <iostream>
2
3 int main()
4 {
5     for (int count = 8; count >= 0; --count)
6         std::cout << count << " ";
7
8     return 0;
9 }
```

Результат:

8 7 6 5 4 3 2 1 0

Также с каждой новой итерацией мы можем увеличить или уменьшить значение счетчика больше, чем на единицу:

```
1 #include <iostream>
2
3 int main()
4 {
5     for (int count = 9; count >= 0; count -= 2)
6         std::cout << count << " ";
7
8     return 0;
9 }
```

Результат:

9 7 5 3 1

## Ошибка неучтенной единицы

Одна из самых больших проблем с которой приходится сталкиваться начинающим программистам в циклах for (а также и в других типах циклов) — это **ошибка на единицу** (или «*ошибка неучтенной единицы*»). Она возникает, когда цикл повторяется на 1 раз больше или на 1 раз меньше нужного количества итераций. Это обычно происходит из-за того, что в условии используется некорректный оператор сравнения (например, > вместо >= или наоборот). Как правило, эти ошибки трудно отследить,

так как компилятор не будет жаловаться на них, программа будет работать, но её результаты будут неправильными.

При написании циклов `for` помните, что цикл будет выполняться до тех пор, пока условие является истинным. Рекомендуется тестировать циклы, используя разные значения для проверки работоспособности цикла. Хорошей практикой является проверять циклы с помощью данных ввода (чисел, символов и прочего), которые заставляют цикл выполниться 0, 1 и 2 раза. Если цикл работает исправно, значит всё ОК.

**Правило: Тестируйте свои циклы, используя входные данные, которые заставляют цикл выполниться 0, 1 и 2 раза.**

## Пропущенные выражения в цикле

Также в циклах можно пропускать одно или сразу все выражения, например:

```
1  #include <iostream>
2
3  int main()
4  {
5      int count = 0;
6      for (; count < 10; )
7      {
8          std::cout << count << " ";
9          ++count;
10     }
11
12     return 0;
13 }
```

Результат:

0 1 2 3 4 5 6 7 8 9

Инициализацию счетчика мы прописали вне тела цикла, а инкремент счетчика — внутри тела цикла. В самом операторе `for` мы указали только условие. Иногда бывают случаи, когда не требуется объявлять счетчик цикла (потому что у нас он уже есть) или увеличивать его (так как мы увеличиваем его каким-то другим способом).

Хоть это и не часто можно наблюдать, но в операторе `for` можно вообще ничего не указывать. Стоит отметить, что подобное приведет к бесконечному циклу:

```
for (;;)
    тело цикла;
```

Вышеприведенный пример эквивалентен:

```
while (true)
    тело цикла;
```

## Объявления переменных в цикле for

Хотя в циклах for обычно используется только один счетчик, иногда могут возникать ситуации, когда нужно работать сразу с несколькими переменными. Для этого используется [оператор Запятая](#). Например:

```
1 #include <iostream>
2
3 int main()
4 {
5     int aaa, bbb;
6     for (aaa = 0, bbb = 9; aaa < 10; ++aaa, --bbb)
7         std::cout << aaa << " " << bbb << std::endl;
8
9     return 0;
10 }
```

Этот цикл присваивает значения двум ранее объявленным переменным: `aaa = 0` и `bbb = 9`. Только с каждой итерацией переменная `aaa` увеличивается на единицу, а `bbb` — уменьшается на единицу.

Результат выполнения программы:

```
0 9
1 8
2 7
3 6
4 5
5 4
6 3
7 2
8 1
9 0
```

**Примечание:** Вышеприведенный цикл можно переписать следующим образом:

```
1 #include <iostream>
2
3 int main()
4 {
5     for (int aaa = 0, bbb = 9; aaa < 10; ++aaa, --bbb)
6         std::cout << aaa << " " << bbb << std::endl;
7
8     return 0;
9 }
```

В этом случае запятая в объявлении переменных является частью синтаксиса, а не использованием оператора Запятая. Но эффект идентичен.

## Использование циклов for в старых версиях C++

В старых версиях C++ переменные, объявленные в цикле for, не уничтожались при завершении этого цикла. Т.е. у вас могло получиться что-то вроде следующего:

```
1 for (int count=0; count < 10; ++count)
2     std::cout << count << " ";
3
4 // В старых версиях C++ переменная count здесь не уничтожается
5
6 std::cout << "\n";
7 std::cout << "I counted to: " << count << "\n"; // поэтому мы можем использовать count ,
```

Позднее это было запрещено, но вы все еще можете увидеть подобное в старом коде.

## Вложенные циклы for

Подобно другим типам циклов, одни циклы for могут быть вложены в другие циклы for. В следующем примере мы разместили один for внутри другого for:

```
1 #include <iostream>
2
3 int main()
4 {
5     for (char c = 'a'; c <= 'e'; ++c) // внешний цикл по буквам
6     {
7         std::cout << c; // сначала выводим букву
8
9         for (int i = 0; i < 3; ++i) // внутренний цикл по числам
10             std::cout << i;
11
12         std::cout << '\n';
13     }
14
15     return 0;
16 }
```

С одной итерацией внешнего цикла выполняется три итерации внутреннего цикла. Следовательно, результат выполнения программы:

a012  
b012  
c012  
d012  
e012

## Заключение

Циклы `for` являются наиболее часто используемыми циклами в языке C++. Несмотря на то, что их синтаксис, как правило, немного запутывает начинающих программистов, вы очень скоро к нему привыкните и ощутите всю мощь и удобство этих циклов.

## Тест

### Задание №1

Напишите цикл `for`, который выводит каждое четное число в диапазоне от 0 до 20.

#### Ответ №1

```
1 #include <iostream>
2
3 int main()
4 {
5     for (int count = 0; count <= 20; count += 2)
6         std::cout << count << std::endl;
7
8     return 0;
9 }
```

### Задание №2

Напишите функцию `sumTo()`, которая принимает целочисленный параметр с именем `value` и возвращает сумму всех чисел от 1 до значения `value`.

Например, если значением `value` является 5, то `sumTo(5)` должен вернуть 15, исходя из  $1 + 2 + 3 + 4 + 5$ .

**Подсказка:** Используйте переменную вне тела цикла для хранения суммы чисел, как в примере с функцией `pow()` в подзаголовке «Еще примеры циклов `for`».

#### Ответ №2

```
1 int sumTo(int value)
2 {
3     int total(0);
4     for (int count=1; count <= value; ++count)
5         total += count;
6
7     return total;
8 }
```

### Задание №3

Что не так со следующим циклом?



```
1 // Выводим все числа от 8 до 0
2 for (unsigned int count=8; count >= 0; --count)
3     cout << count << " ";
```

### Ответ №3

Этот цикл `for` выполняется до тех пор, пока `count >= 0`. Другими словами, он работает до тех пор, пока переменная `count` не станет отрицательным числом. Однако, поскольку `count` является типа `unsigned`, то эта переменная никогда не сможет быть отрицательной. Следовательно, этот цикл бесконечный! Как правило, рекомендуется избегать использования типов `unsigned` в цикле, если на это нет веских причин.

Оценить статью:

★★★★★ (224 оценок, среднее: 4,97 из 5)

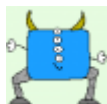


← [Урок №68. Цикл do while](#)

[Урок №70. Операторы break и continue](#) →



### Комментариев: 39



1. Андрей:

[11 ноября 2020 в 14:14](#)

Благодарю, вы объясняете лучше чем учителя в моей школе

[Ответить](#)



1. Юрий:

[11 ноября 2020 в 16:29](#)

Для этого и делалось 😊

[Ответить](#)



2. Ruslan:

[22 сентября 2020 в 16:27](#)

Задание № 2

```
1 #include <iostream>
2
```

```
3 int sum_to (int value)
4 {
5     int sum {};
6     for (int count {}; count <= value; ++count)
7         sum += count;
8     return sum;
9 }
10
11 int main ()
12 {
13     std::cout << "Enter number" << '\n';
14     int value {};
15     std::cin >> value;
16
17     std::cout << sum_to (value) << '\n';
18
19     return 0;
20 }
```

[Ответить](#)3. *Ruslan:*[22 сентября 2020 в 15:55](#)

```
1 Задание № 1
2
3 #include <iostream>
4
5 int main ()
6 {
7     for (int count {-2}; (count < 20); std::cout << (count += 2) << ' ')
8         ; // null-statement
9     std::cout << '\n';
10
11     return 0;
12 }
```

[Ответить](#)4. *Вася:*[11 августа 2020 в 14:22](#)

Кто решит проще?)

```
1 #include <iostream>
2
3 void sumTo(int n) {
4     int gauss = n * (n + 1) / 2;
```

```

5     std::cout << gauss;
6 }
7 int main()
8 {
9     sumTo(5);
10 }

```

[Ответить](#)1. *Владимир:*[28 августа 2020 в 12:02](#)

Проще уже невозможно будет написать

```

1 int sumTo(int value)
2 {
3     return value * (value + 1) / 2;
4 }

```

[Ответить](#)5. *Яна:*[16 июня 2020 в 12:10](#)

```

1 #include <iostream>
2
3 using namespace std;
4
5 int input() // функция для ввода переменной
6 {
7     cout << "Введите значение от 1 до ";
8     int value;
9     cin >> value;
10    return value;
11 }
12 int sumTo(int value) // функция для расчета суммы чисел от 2 до выбранной переменной
13 {
14     int sum = 0;
15
16     for (int start = 1; start <= value; ++start) //счет от числа start до value
17
18         // тоже самое, что sum=sum+start (т.е. сумма(0)= 0 + 1 + 2 + 3 +....+ value)
19         sum+= start;
20     return sum;
21 }
22 int main()
23 {
24     setlocale(0, ""); // кириллица

```

```

25     int value = input(); // возврат функции input()
26
27     // вывод суммы всех чисел от 1 до значения value из sumTo()
28     cout << "Сумма значений от 1 до " << value << " равна " << sumTo(value) << "\n";
29     return 0;
30 }

```

[Ответить](#)

1. Айнур:

[25 сентября 2020 в 09:40](#)

Ваш код не будет работать, так как функция `input()` у вас не сделана, а Вы ее вызываете

[Ответить](#)

6. Павел:

[25 мая 2020 в 18:10](#)

Попробовал написать программу проверки на палиндромность числа. Прога работает как надо, однако ошибка в конце кода вылезит. Не могу понять с чем связано. Подскажите плиз (использую Visual Studio 2019).

Ошибка:

Run-Time Check Failure #2 — Stack around the variable 'q' was corrupted.

Код программы:

```

1  Int main()
2  {
3  setlocale(LC_ALL, "ru");
4
5  char q[] = {0};
6  int n;
7
8  cout << "\t\t\tПрограмма Палиндром в C++ ";
9  cout << "\n\nВведите целое число: ";
10 cin >> q;
11
12 n = strlen(q) - 1;
13
14 for (int i = 0; i <= n; i++)
15 {
16 if (q[i] == q[n])
17 {
18 cout << q[i] << " " << q[n] << "\n";
19 n--;
20 }
21

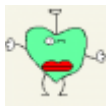
```

```

22 else
23 {
24 cout << "\n\nЭто число НЕ полиндром";
25 return 0;
26 }
27
28 //cout << "\n\n" << i << " " << n;
29 }
30
31 cout << "Это число полиндром";
32
33 return 0;
34 }

```

[Ответить](#)



7. *Павел Карнов:*  
[28 февраля 2020 в 21:32](#)

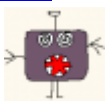
Вот моя программа, оцените)

```

1  #include <iostream>
2
3  int sumTo(int value)
4  {
5      int x(0);
6      for (; 1 <= value; --value)
7          x += value;
8
9      return x;
10 }
11 int main()
12 {
13     int value;
14     std::cout << "Вы хотите узнать сумму от 1 до ";
15     std::cin >> value;
16     std::cout << "Сумма от 1 до " << value << " = " << sumTo(value) << "\n";
17     return 0;
18 }

```

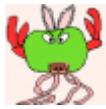
[Ответить](#)



1. *Валера:*  
[4 апреля 2020 в 20:39](#)

Не делай запутанные программы, на продакшене это никому не надо

[Ответить](#)



8. *armus1:*

[3 декабря 2019 в 23:20](#)

Задача №2

```
1 #include <iostream>
2
3 int valueOut() //пишем функцию valueOut() для ввода пользователем переменной value
4 {
5     using namespace std;
6     int value;
7     cout << "Enter an integer:";
8     cin >> value;
9     return value;
10 }
11
12 int sumTo(int value) // пишем функцию для подсчета суммы чисел от 1 до числа введенного
13 {
14     int total = 0;
15     for (int number = 1; number <= value; ++number)
16         total += number;
17     return total;
18 }
19
20 int main()
21 {
22     using namespace std;
23     int value = valueOut();
24     cout << "Sum of all numbers from 1 to the entered number:" << sumTo(value) <<
25     return 0;
26 }
```

[Ответить](#)



9. *Игорь:*

[20 сентября 2019 в 15:51](#)

2-я задача:

```
1 #include "pch.h"
2 #include <iostream>
3 using namespace std;
4
5 int sumTo(int value)
6 {
7
```

```
8     int start = 0;
9     for (int ii = 1; ii <= value; ii++)
10         start += ii;
11     return start;
12 }
13
14 int main()
15 {
16     cout << "Enter value: ";
17     int a;
18     cin >> a;
19     if (a < 0)
20         cout << "\nError: You entered negative number!";
21     else
22         cout << "\nSumTo its: " << sumTo(a)<<"\n";
23 }
```

### Ответить



10. *Игорь:*

20 сентября 2019 в 15:08

пришло на ум 2 решения первой задачи

1-е решение:

```
1 #include "pch.h"
2 #include <iostream>
3 using namespace std;
4
5 int main()
6 {
7     for (int ch = 0; ch <= 20; ch += 2)
8         cout << ch << " ";
9
10 }
```

2-е решение(с использованием ветвления if):

```
1 #include "pch.h"
2 #include <iostream>
3 using namespace std;
4
5 int main()
6 {
7     for (int ch = 0; ch <= 20; ch++)
8     {
9         if (ch % 2 == 0)
10             cout << ch << " ";
11     }
```

```
12 | }
```

[Ответить](#)11. *Алексей:*[16 августа 2019 в 01:35](#)

Решение задания № 2:

```
1  #include <iostream>
2
3  int toSum(int value)
4  {
5      for (int i = value - 1; i > 0; i--)
6          value += i;
7      return value;
8  }
9
10 int main()
11 {
12     int a = 4;
13     std::cout << a << "\n";
14     std::cout << toSum(a) << "\n";
15     std::cout << a << "\n";
16
17     return 0;
18 }
```

Результат:

```
4
10
4
```

Program ended with exit code: 0

Вопрос:

Когда я прогоняю переменную "a" через функцию toSum(), я меняю ее значение.

Однако, после выполнения функции, переменная имеет свое прежнее значение.

Как так? Переменная "a" создана во внешнем блоке, во вложенном блоке (в функции) преобразуется.

Почему она не изменилась?

Или функцию нельзя рассматривать как вложенный блок? (Не баг, а фича.)

[Ответить](#)1. *Анастасия:*[19 августа 2019 в 20:14](#)



"a" передаётся в функцию по значению, то есть в параметр value из Вашей функции копируется значение "a", далее вы меняете само value в функции и получившееся значение возвращаете в main тоже в качестве простого числа. "a" при этом не страдает.

[Ответить](#)



12. Максим:

[4 августа 2019 в 21:51](#)

Зацените задумку

Задание 2

```
1  #include <iostream>
2
3  int getCount()
4  {
5      std::cout << "Enter an integer: ";
6      int count;
7      std::cin >> count;
8      return count;
9  }
10
11 int sumTo(int count)
12 {
13     int value(0);
14
15     for (; count >= 1; --count)
16     {
17         value += count;
18     }
19     return value;
20 }
21
22 int main()
23 {
24     int count = getCount();
25
26     std::cout << "The sum of numbers from 1 to " << count << " is " << sumTo(count);
27
28     return 0;
29 }
```

[Ответить](#)



13. Фёдор:

[14 июня 2019 в 12:38](#)

Помогите разобраться, почему в обоих случаях total имеет одно и то же значение, хотя счётчик count различается на единицу.

```
1  #include <iostream>
2  int main()
3  {
4      int value;
5      std::cin >> value;
6
7      int totalOne = 0;
8      for (int count = 0; count <= value; count++)
9          totalOne += count;
10
11     int totalTwo = 0;
12     for (int count = 1; count <= value; count++)
13         totalTwo += count;
14
15     if (totalOne == totalTwo)
16         std::cout << "It's equals";
17     else
18         std::cout << "It's not equals";
19     return 0;
20 }
```

Заранее спасибо)

[Ответить](#)



1. Анастасия:  
[15 июня 2019 в 16:37](#)

totalOne при первой итерации прибавляет 0 к 0, то есть total останется прежним (0), а на второй итерации он уже догонит totalTwo, и дальше они работают одинаково, поэтому и результат одинаковый.

[Ответить](#)



14. Владимир:  
[15 мая 2019 в 22:22](#)

задача №2

```
1  int sss(int a)
2  {
3      int q = 0;
4      for (; a > 0; a--)
5          q += a;
6      return q;
```

```

7   }
8
9   void ttt()
10  {
11      using namespace std;
12      cout << "Введите число: ";
13      int a;
14      cin >> a;
15      cout << "Сумма чисел от 1 до " << a << " = " << sss(a) << endl;
16  }
17
18  int main()
19  {
20      setlocale(0, "");
21      ttt();
22      return 0;
23  }

```

### [Ответить](#)



15. **Вячеслав:**

[13 марта 2019 в 21:49](#)

разобрался с примером задания 2 как у Александра и немного добавил и вот что получилось:

```

1  #include "pch.h"
2  #include <iostream>
3  //функция суммы всех чисел арифметической прогрессии
4  //с параметром целочисленного значения
5  int sumTo(int value)
6  {
7      using namespace std;
8      //инициализируем переменную total значением 0
9      int total(0);
10     //цикл for: 1) инициализируем счетчик цикла = 1
11     //2) условие: тело цикла выполнится, если счетчик <= параметра функции
12     //3) с каждой итерацией увеличиваем счетчик на 1
13     for (int count = 1; count <= value; ++count)
14     { //тело цикла
15         //переменная total с каждой итерацией увеличивается на
16         // значение счетчика
17         total += count;
18         //выводим в консоль значение счетчика
19         cout << count;
20         // доп. условие если счетчик меньше значения параметра функции
21         if (count < value)
22             // то выводится оператор +
23             cout << " + ";

```

```
24
25     }
26     return total; // возвращаем значение переменной total
27
28 }
29
30 int main()
31 {
32     setlocale(0, "");
33     using namespace std;
34     //запрос ввода числа
35     cout << "Введите число: ";
36     //объявляем переменную x, которая будет присвоена
37     //параметру value функции sumTo()
38     int x;
39     //запрашиваем число из консоли в функцию
40     cin >> x;
41     //инициализируем переменную y функцией sumTo(x)
42     int y = sumTo(x);
43     cout << " = " << y;
44
45     return 0;
46 }
```

### [Ответить](#)



16. **Вячеслав:**

[13 марта 2019 в 00:14](#)

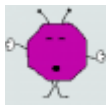
во втором задании программа ничего не выводит что я сделал не так?

```
1  #include "pch.h"
2  #include <iostream>
3
4  int sumTo(int value)
5  {
6      int total(0);
7      for (int count = 1; count <= value; ++count)
8          total += count;
9
10     return total;
11
12 }
13
14 int main()
15 {
16     sumTo(5);
17     return 0;
18 }
```

18 | }

[Ответить](#)1. *Владимир:*[14 марта 2019 в 21:27](#)

Она и не будет ничего выводить. В коде нет ни одного `std::cout`, который бы выводил сообщение в консоль.

[Ответить](#)17. *Александр:*[8 февраля 2019 в 12:11](#)

Цикл `for` может быть сложным для новичков потому, что это не `for` 😊

В том смысле, что он ничего не имеет общего с классическими циклами `for` (базовые алгоритмические структуры, реализация в Паскале, Бейсике и т.п.)

Я обычно говорю ученикам (особенно переходящим на C++ с паскаля), что "цикла `for` в C++ нет. Зато есть отличный фороимитатор. Мы пишем `for`, но на самом деле это няшная и удобная форма `while`"

[Ответить](#)1. *Константин:*[22 апреля 2019 в 03:30](#)

Ты ещё маленького ребёнка (который букву "р" "невыговаливает") подучи это слово произносить!:-)Шутка!

[Ответить](#)18. *Андрей:*[23 января 2019 в 22:33](#)

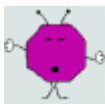
Объясните, пожалуйста, для чего Вы инициализировали переменную `total`? (Для проверки условия `true` или `false`?)

```
1 | int total = 1;
```

и можете подробно объяснить строчку

```
1 | total *= base;
```

[Ответить](#)



1. *Александр:*

[8 февраля 2019 в 12:15](#)

```
1 total = 1; // это начальное значение для возведения в натуральную степень
2 total *= base; // это умножение текущего найденного результата на основание с
```

математика процесса:

$$a^0 = 1$$

$$a^n = a * a^{(n - 1)}$$

[Ответить](#)



19. *corsair123:*

[5 ноября 2018 в 23:40](#)

Я попался с unsigned и отрицательной переменной в цикле. Долго глаза "ломал", а тут статья. Спасибо огромное!!! Здорово всё объяснил. Удачи.

[Ответить](#)



1. *Юрий:*

[6 ноября 2018 в 00:16](#)

Рад, что смог помочь 😊

[Ответить](#)



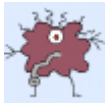
2. *Владимир:*

[1 декабря 2018 в 20:22](#)

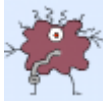
Поломай ещё с вот этим:

```
1 //Нарращиваем переменную k с 0 до 10 (включительно);
2 int main()
3 {
4     int k = 0;
5     for (int i = 0; i < 10; i++)
6         std::cout << '\n';
7         k++;
8         cout << k << '\n';
9     return 0;
10 }
```

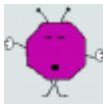
Подсказка: с этим циклом тоже что-то не так. Тем не менее, он так же скомпилируется, как и цикл в тесте.

[Ответить](#)1. *Игорь:*[23 февраля 2019 в 21:32](#)

Помоему должно быть ++i. При i++ значение i=0 побывает в цикле 2 раза и в результате выведутся числа от 0 до 11... или я что-то путаю в инкрементах?

[Ответить](#)1. *Игорь:*[23 февраля 2019 в 22:02](#)

Ан нет, я переосмыслил )) от 0 до 10 и есть ведь 11 чисел... всё, что после if и до return нужно взять в блок, иначе в цикле выполнится только строка после if (11 раз \n) и 1 раз вывод значения k

2. *Александр:*[2 марта 2019 в 12:52](#)

если конструкции ++i и i++ не входят в состав других выражений, то разницы в результате нет (есть различия в реализации и т.п., но результат будет одинаковым)

2. *Геннадий:*[1 марта 2019 в 20:42](#)

Привет!

Должно быть в цикле `int i = 1;`[Ответить](#)3. *Геннадий:*[1 марта 2019 в 20:49](#)

Прошу прощения))... Должно быть так в Вашем коде:

```

1  int k = 0;
2      for (int i = 0; i < 10; i++) {
3          std::cout << "\n";
4          {
5              cout << k << '\n';
6              k++;
7          }
8      }

```

Тогда от 0 до 9 показывает

[Ответить](#)



20. Алексей:

[5 октября 2018 в 05:57](#)

Задача 2

```
1  #include <iostream>
2
3  using namespace std;
4
5  //сумма членов арифметической прогрессии
6  int sumTo(int value)
7  {
8      int result(0);
9      for(int cnt = 1; cnt <= value; ++cnt)
10     {
11         result += cnt;
12         cout << cnt;
13         //ограничиваем вывод "+"
14         if (cnt < value)
15             cout << " + ";
16     }
17     return result;
18 }
19
20 int main ()
21 {
22     int x;
23     cin >> x;
24     int y = sumTo(x);
25     cout << " = " << y;
26
27
28     return 0;
29 }
```

[Ответить](#)



21. Дмитрий:

[11 сентября 2018 в 12:11](#)

Решение на первый вопрос, не много странный но работает 😊

```
1  int main()
2  {
```



```
3   for (int alf = 0; alf <= 22; alf++)
4       if (int alf1=alf%2)
5           cout << alf-1 << " ";
6   return 0;
7 }
```

[Ответить](#)22. *beksheikh:*[9 марта 2018 в 20:02](#)

Хотя я прошел эту часть.(в учебе)но все грамотно говорите.и нашел кое что новое)спасибо за инфу.С уважением Я.

[Ответить](#)1.  *Юрий:*[9 марта 2018 в 20:28](#)

Пожалуйста 😊

[Ответить](#)

## Добавить комментарий

Ваш E-mail не будет опубликован. Обязательные поля помечены \*






Имя \* Email \* 

Комментарий

☐ Сохранить моё Имя и E-mail. Видеть комментарии, отправленные на модерацию☐ Получать уведомления о новых комментариях по электронной почте. Вы можете [подписаться](#) без комментирования.[TELEGRAM](#)  [КАНАЛ](#)[ПАБЛИК](#) 



## ТОП СТАТЬИ

-  [Словарь программиста. Сленг, который должен знать каждый кодер](#)
-  [Урок №1. Введение в программирование](#)
-  [70+ бесплатных ресурсов для изучения программирования](#)
-  [Урок №1: Введение в создание игры «SameGame» на C++/MFC](#)
-  [Урок №4. Установка IDE \(Интегрированной Среды Разработки\)](#)

- [Ravesli](#)
- - [О проекте/Контакты](#) -
- - [Пользовательское Соглашение](#) -
- - [Все статьи](#) -
- Copyright © 2015 - 2020