

Ravesli [Ravesli](#)


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

Урок №39. Арифметические операторы

 [Юрий](#) |

- [Уроки C++](#)

|

 Обновл. 7 Сен 2020 |

 69563

[↑](#)  43

На этом уроке мы рассмотрим арифметические операторы и их использование в языке C++.

Оглавление:

1. [Унарные арифметические операторы](#)
2. [Бинарные арифметические операторы](#)
3. [Деление целых чисел и чисел типа с плавающей точкой](#)
4. [Использование оператора `static_cast` в операциях деления](#)
5. [Деление с остатком](#)
6. [Отрицательные числа в операциях деления до C++11](#)
7. [Арифметические операторы присваивания](#)
8. [Где оператор возведения в степень?](#)
9. [Тест](#)

Унарные арифметические операторы

Существуют два унарных арифметических оператора: плюс (+) и минус (-). Унарные операторы — это операторы, которые применяются только к одному операнду.

Оператор	Символ	Пример	Операция
Унарный плюс	+	+x	Значение x
Унарный минус	-	-x	Отрицательное значение x

Унарный оператор + возвращает значение операнда. Другими словами, $+5 = 5$ или $+x = x$. Унарный плюс вам, скорее всего, не придется использовать. Его по большей части добавили в качестве симметрии с унарным оператором минус. Унарный оператор минус возвращает операнд, умноженный на -1 . Например, если $x = 5$, то $-x = -5$.

Оба этих оператора пишутся непосредственно перед самым операндом, без пробела ($-x$, а не $- x$).

Не следует путать унарный оператор минус с бинарным оператором вычитания, хоть они и используют один и тот же символ. Например, в выражении $x = 5 - -3$; , первый минус — это оператор вычитания, а второй — унарный минус.

Бинарные арифметические операторы

Бинарные операторы — это операторы, которые применяются к двум операндам (слева и справа). Существует 5 бинарных операторов.

Оператор	Символ	Пример	Операция
Сложение	+	$x + y$	x плюс y
Вычитание	-	$x - y$	x минус y
Умножение	*	$x * y$	x умножить на y
Деление	/	x / y	x разделить на y
Деление с остатком	%	$x \% y$	Остаток от деления x на y

Операторы сложения, вычитания и умножения работают так же, как и в обычной математике. А вот деление и деление с остатком рассмотрим детально.

Деление целых чисел и чисел типа с плавающей точкой

Оператор деления имеет два режима. Если оба операнда являются целыми числами, то оператор выполняет целочисленное деление. Т.е. любая дробь (больше/меньше) отбрасывается и возвращается целое значение без остатка, например, $7 / 4 = 1$.

Если один или оба операнда типа с плавающей точкой, то тогда будет выполняться деление типа с плавающей точкой. Здесь уже дробь присутствует. Например, выражения $7.0 / 3 = 2.333$, $7 / 3.0 = 2.333$ или $7.0 / 3.0 = 2.333$ имеют один и тот же результат.

Попытки деления на 0 (или на 0.0) станут причиной сбоя в вашей программе, и это правило не следует забывать!

Использование оператора `static_cast` в операциях деления

На [уроке №35](#) мы уже использовали оператор `static_cast` для вывода ASCII-символов в виде целых чисел.

Аналогичным образом мы можем использовать `static_cast` для конвертации целого числа в число типа с плавающей точкой. Таким образом, вместо целочисленного деления выполнится деление типа с плавающей точкой. Например:

```
1 #include <iostream>
2
3 int main()
```

```
4 {  
5     int x = 7;  
6     int y = 4;  
7  
8     std::cout << "int / int = " << x / y << "\n";  
9     std::cout << "double / int = " << static_cast<double>(x) / y << "\n";  
10    std::cout << "int / double = " << x / static_cast<double>(y) << "\n";  
11    std::cout << "double / double = " << static_cast<double>(x) / static_cast<double>(y) << "\n";  
12  
13    return 0;  
14 }
```

Результат выполнения программы:

```
int / int = 1  
double / int = 1.75  
int / double = 1.75  
double / double = 1.75
```

Деление с остатком

Оператор деления с остатком (%) работает только с целочисленными операндами и возвращает остаток от целочисленного деления. Например:

- ➔ *Пример №1:* $7 / 4 = 1$ с остатком 3, таким образом, $7 \% 4 = 3$.
- ➔ *Пример №2:* $25 / 7 = 3$ с остатком 4, таким образом, $25 \% 7 = 4$. Остаток составляет не дробь, а целое число.
- ➔ *Пример №3:* $36 \% 5 = 1$ с остатком 1. В числе 36 только 35 делится на 5 без остатка, поэтому $36 - 35 = 1$, 1 — это остаток и результат.

Данный оператор чаще всего используют для проверки деления без остатка одних чисел на другие. Если $x \% y == 0$, то x делится на y без остатка.

Например, мы хотим написать программу, которая выводит числа от 1 до 100 по 20 значений в каждой строке. Мы можем использовать оператор деления с остатком для создания разрыва строк. Несмотря на то, что мы еще не рассматривали цикл `while`, в следующей программе всё максимально просто и понятно:

```
1 #include <iostream>  
2  
3 int main()  
4 {  
5     // Переменная count хранит текущее число для вывода  
6     int count = 1; // начинаем с 1  
7  
8     // Повторение операции (цикл) до тех пор, пока count не будет равен 100  
9     while (count <= 100)  
10    {  
11        std::cout << count << " "; // вывод текущего числа  
12    }  
13 }
```

```

14 // Если count делится на 20 без остатка, то вставляем разрыв строки и продо
15 if (count % 20 == 0)
16     std::cout << "\n";
17
18     count = count + 1; // переходим к следующему числу
19 } // конец while
20
21 return 0;
} // конец main()

```

Результат выполнения программы:

```

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40
41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60
61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80
81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100

```

О while мы еще поговорим на соответствующем уроке.

Отрицательные числа в операциях деления до C++11

До C++11, если любой из операндов целочисленного деления является отрицательным, то компилятор округляет результат самостоятельно! Например, результатом $-5 / 2$ может быть либо -3 , либо -2 . Однако большинство современных компиляторов округляют числа в сторону нуля (например, в $-5 / 2$ результатом будет -2). В спецификации C++11 определили, что компилятор должен всегда округлять к нулю (или, проще говоря, просто отбрасывать дробь).

Также до C++11, если один из операндов оператора деления с остатком является отрицательным, то результат может быть как положительным, так и отрицательным! Например, результатом $-5 \% 2$ может быть как 1 , так и -1 . В спецификации C++11 решили сделать так, чтобы результат $a \% b$ был того же знака, что и значение a .

Арифметические операторы присваивания

Оператор	Символ	Пример	Операция
Присваивание	=	$x = y$	Присваиваем значение y переменной x
Сложение с присваиванием	+=	$x += y$	Добавляем y к x
Вычитание с присваиванием	-=	$x -= y$	Вычитаем y из x
Умножение с присваиванием	*=	$x *= y$	Умножаем x на y
Деление с присваиванием	/=	$x /= y$	Делим x на y
Деление с остатком и с присваиванием	%=	$x \% = y$	Присваиваем остаток от деления x на y переменной x

До этого момента, когда нам нужно было добавить число 5 к определенной переменной, мы делали следующее:

```
1 x = x + 5;
```

Это работает, но требуется два оператора для выполнения.

Так как стейтменты типа $x = x + 5$ являются очень распространенными, то C++ предоставляет 5 арифметических операторов присваивания для нашего удобства. Вместо $x = x + 5$, мы можем записать:

```
1 x += 5;
```

Вместо:

```
1 x = x * y;
```

Мы можем записать:

```
1 x *= y;
```

Где оператор возведения в степень?

В языке C++ вместо оператора возведения в степень есть **функция pow()**, которая находится в [заголовочном файле](#) `cmath`. `pow(base, exponent)` эквивалентно $base^{exponent}$. Стоит отметить, что параметры `pow()` имеют тип `double`, поэтому вы можете использовать не только целые числа, но и дробные. Например:

```
1 #include <iostream>
2 #include <cmath> // подключаем pow()
3
4 int main()
5 {
6     std::cout << "Enter the base: ";
7     double base;
8     std::cin >> base;
9
10    std::cout << "Enter the exponent: ";
11    double exp;
12    std::cin >> exp;
13
14    std::cout << base << "^" << exp << " = " << pow(base, exp) << "\n";
15
16    return 0;
17 }
```

Тест

Задание №1

Вычислите результат следующего выражения: $6 + 5 * 4 \% 3$.

Ответ №1

Поскольку операторы `*` и `%` имеют более высокий **приоритет**, чем оператор `+`, то оператор `+` будет выполняться последним. Мы можем переписать наше выражение следующим образом: $6 + (5 * 4 \% 3)$. Операторы `*` и `%` имеют одинаковый приоритет, но их ассоциативность слева направо, так что левый оператор будет выполняться первым. Получается: $6 + ((5 * 4) \% 3)$.

$$6 + ((5 * 4) \% 3) = 6 + (20 \% 3) = 6 + 2 = 8$$

Ответ: 8.

Задание №2

Напишите программу, которая просит пользователя ввести целое число, а затем сообщает, является ли его число чётным или нечётным. Напишите функцию `isEven()`, которая возвращает `true`, если целое число является чётным. Используйте оператор деления с остатком, чтобы определить чётность числа.

Подсказка: Используйте ветвление `if` и оператор сравнения (`==`).

Ответ №2

```
1  #include <iostream>
2
3  bool isEven(int x)
4  {
5      // Если x % 2 == 0, то x - это чётное число
6      return (x % 2) == 0;
7  }
8
9  int main()
10 {
11     std::cout << "Enter an integer: ";
12     int x;
13     std::cin >> x;
14
15     if (isEven(x))
16         std::cout << x << " is even\n";
17     else
18         std::cout << x << " is odd\n";
19
20     return 0;
21 }
```

Возможно, вы хотели написать или написали функцию `isEven()` следующим образом:

```
1  bool isEven(int x)
2  {
3      if ((x % 2) == 0)
4          return true;
5      else
6          return false;
7  }
```

Хотя этот способ тоже рабочий, но он сложнее. Посмотрим, как его можно упростить. Во-первых, давайте вытащим условие `if` и присвоим его отдельной переменной **типа `bool`**:

```
1 bool isEven(int x)
2 {
3     bool isEven = (x % 2) == 0;
4     if (isEven) // isEven - true
5         return true;
6     else // isEven - false
7         return false;
8 }
```

В коде, приведенном выше, если переменная `isEven` имеет значение `true`, то возвращаем `true`, в противном случае (если `isEven` имеет значение `false`) — возвращаем `false`. Мы же можем сразу возвращать `isEven`:

```
1 bool isEven(int x)
2 {
3     bool isEven = (x % 2) == 0;
4     return isEven;
5 }
```

Так как переменная `isEven` используется только один раз, то мы можем её вообще исключить:

```
1 bool isEven(int x)
2 {
3     return (x % 2) == 0;
4 }
```

Оценить статью:

★★★★★ (286 оценок, среднее: 4,89 из 5)



[← Урок №38. Приоритет операций и правила ассоциативности](#)

[Урок №40. Инкремент, декремент и побочные эффекты →](#)



Комментариев: 43



1. *Slava:*
[30 августа 2020 в 21:19](#)

Решение задачи

"ifn.h"

```
1 #include <iostream>
2
3 bool isEven(int a)
```

```
4 {
5     if (a % 2 == 0)
6         return true;
7     else
8         return false;
9 }
10
11 void startIntNumber()
12 {
13     std::cout << " Enter integer number : " << std::endl; int rec; std::cin >>
14     bool ba = isEven(rec);
15     if (ba == true)
16         std::cout << " Even number " << std::endl;
17     else
18         std::cout << " The number is odd " << std::endl;
19 }
```

main.cpp

```
1 #include <iostream>
2 #include "ifn.h"
3
4 void startIntNumber();
5
6 int main()
7 {
8     startIntNumber();
9     return 0;
10 }
```

[Ответить](#)



2. *Виктор:*

[5 июня 2020 в 14:29](#)

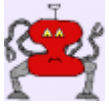
Отошёл от пункта спецификации, где "Напишите функцию isEven(), которая возвращает true" и упростил программу 😊

```
1 #include <iostream>
2
3 using namespace std;
4
5 void isEven(int a)
6 {
7     if (!(a%2))
8         cout<<"The digital is even"<<endl;
9     else
10        cout<<"The digital is odd"<<endl;
11 }
12
13 int main()
14 {
```



```
15     cout<<"Insert your digital"<<endl;
16     int x;
17     cin>>x;
18     isEven(x);
19
20     return 0;
21 }
```

[Ответить](#)



3. Эдуард:

[13 мая 2020 в 21:32](#)

Моё решение на тест 2:

```
1  #include <iostream>
2  #include <cmath>
3  #include <windows.h> //Английский не понимаю, тому я писал русским
4  using namespace std;
5  bool isEven(int x); //Объявление isEven (и функции bool isEven)
6
7  int main()
8  {
9      SetConsoleCP(1251);
10     SetConsoleOutputCP(1251);
11     cout << "Введите число:" << endl;
12     int x;
13     cin >> x;
14     //Не очень знаю что, но по моему- перевод данных
15     bool even = isEven(x);
16     if (even)
17         cout << "Это целое число" << endl;
18     else
19         cout << "Это не целое число" << endl;
20
21     return 0;
22 }
23 //Компилятор жаловался, что не может он сверху положить эту функцию
24 //Пришлось мне её здесь поставить
25 bool isEven(int x)
26 {
27     if (x % 2 == 0)
28         return true;
29     else
30         return false;
31 }
```

[Ответить](#)



4. Sagynysh:

[12 мая 2020 в 14:59](#)

вот такой вот вариантчик

```
1 #include<iostream>
2 #include<cmath>
3 using namespace std;
4 int isEven()
5 {
6     int n;
7     cout << "N = ";
8     cin >> n;
9     return n;
10 }
11 int main()
12 {
13     bool checkbox = false;
14     int n = isEven();
15     if(n%2 == 0)
16     {
17         checkbox = true;
18     }
19     cout << boolalpha << checkbox << endl;
20     system("pause");
21     return 0;
22 }
```

[Ответить](#)



5. *Владимир:*

[21 ноября 2019 в 23:28](#)

Все понял, каюсь и благодарю. Это ж целочисленный формат, епт... Незнаю чего так тупанул в таком простом моменте. Что поделать. Бывает.

[Ответить](#)



1. *Юрий:*

[22 ноября 2019 в 00:41](#)

Ничего страшного, это естественно в процессе изучения чего-либо. Главное — двигаться дальше)

[Ответить](#)



6. *Владимир:*

[19 ноября 2019 в 01:16](#)

Есть вопрос. В разделе "Отрицательные числа в операциях деления до C++11" есть такой фрагмент :

"... до C++11...результатом $-5 \% 2$ может быть как 1, так и -1 ..."

Но как??? Может как 5 так и -5? Откуда там единица?

[Ответить](#)1. *Дмитрий:*[20 ноября 2019 в 08:16](#)

Прежде чем писать комментарий вы бы перечитали весь материал лучше... Деление с остатком не есть деление: $7\%4=3$; $6\%4=2$; $5\%4=1$; $4\%4=0$.

[Ответить](#)1. *владимир:*[20 ноября 2019 в 16:06](#)

Ну правильно все пишете. А $5/2$ сколько будет? Правильно, 2.5. Какой остаток дробный видим? Правильно, 0.5. Так что должно быть результатом выражения $5\%2$ по вышеприведенной вами же логике? Не 5, нет? Откуда 1?

[Ответить](#)1. *Юрий:*[20 ноября 2019 в 18:36](#)

Владимир, перечитайте урок ещё раз и посмотрите информацию в Интернете по поводу оператора $\%$ (остаток от деления). $5\%2=1$ от того, что $2*2=4$ и $5-4=1$.

2. *Дмитрий:*[20 ноября 2019 в 19:23](#)

Спешу вас удивить владимир, но $5 / 2 = 2!$ а уже $5.0 / 2$ или $5 / 2.0$ или $5.0 / 2.0$ вот тогда ровняется 2.5. Ибо $\text{int} / \text{int} = \text{int}!$ И советую я вам перечитать внимательно не только это урок но и главу 2 (в частности уроки 31, 32, и 33). Да и привыкайте разбираться сами, ибо путь этот тернист подводными камнями 😊

3. *Борис:*[26 апреля 2020 в 22:16](#)

Владимир, кто вам сказал, что остаток от деления — это то что после десятичной точки? Тут и есть ваша ошибка.

7. *Юрий:*[10 ноября 2019 в 16:38](#)

Я написал без применения функции. Но с проблемой — компьютер отвечает с запозданием. Писалось в Eclipse IDE, Manjaro Linux.

```

1 #include <iostream>
2 #include <cmath>
3 #include <iomanip>
4 using namespace std;;
5 //-----
6 int main(/*int argc, char *argv[]*/)
7 {
8     int number;
9     cout << "Введи число: ";
10    cin >> number;
11    if(number % 2 == 0)
12        cout << endl << "true" << endl;
13    if(number % 2 != 0)
14        cout << endl << "false" << endl;
15 }

```

Ответить



8. Алексей:

[4 июля 2019 в 17:13](#)

Даже не знаю как я мог не понять эти деления, но было ясно насчет if.
 Буль можно использовать, но как поставил Ваш вариант — подумал, зачем.

```

1 #include <iostream>
2
3 int val() { std::cout << "Enter value:"; int x; std::cin >> x; return x; }
4
5 int main()
6 {
7     int x = val();
8     if (x % 2 == 0) std::cout << x << " is even!" "\n";
9     else std::cout << x << " is odd!" "\n";
10 }

```

Не знаю кому проще или сложнее, но вроде бы проще уже некуда.

Ответить



1. Danger:

[2 августа 2020 в 15:06](#)

Неудобно только то, что у вас многие стейтменты в одну строчку.

Ответить



9. Кирилл:

[27 апреля 2019 в 20:50](#)

```

1 #include <iostream>
2

```

```
3 using namespace std;
4
5 int main() {
6     int n;
7     cout << "Enter a number: ";
8     cin >> n;
9     cout << boolalpha << !(n & 1);
10 }
```

[Ответить](#)

10. *Дмитрий:*
[19 марта 2019 в 10:12](#)

```
1 #include <iostream>
2
3 using namespace std;
4
5 bool isEven(int a)
6 {
7     return (a % 2) == 0;
8 }
9
10 int getValue()
11 {
12     int a;
13     cin >> a;
14     return a;
15 }
16
17 int main()
18 {
19     int a{ getValue() };
20     bool b{ isEven(a) };
21
22     if (b)
23         cout << true;
24     else
25         cout << false;
26
27     system("pause");
28     return 0;
29 }
```

[Ответить](#)

11. *Вячеслав:*
[28 января 2019 в 22:26](#)

Вот мой вариант. Юрий посмотрите и скажите свое мнение.

```

1  #include "pch.h"
2  #include <iostream>
3  using namespace std;
4
5  bool isEven(int x)
6  {
7      if (x % 2 == 0)
8          return true;
9      return false;
10 }
11
12
13 int main()
14 {
15     int x;
16     cout << "Enter a number:";
17     cin >> x;
18
19     if (isEven(x))
20         cout << x << "The number is even" << "\n";
21     else
22         cout << x << "The number is not even" << "\n";
23     return 0;
24 }

```

[Ответить](#)



12. *Andrey:*

[8 декабря 2018 в 09:03](#)

По примеру задачи из итогового теста (если точнее, то по примеру вашего решения) — сделал вот так.

Попробовал — работает.

Посмотрел ваше решение тут — узнал опять много нового 😊

Например то, что при инициализации переменной можно делать вот так

```
1 | bool isEven = (x % 2) == 0;
```

(почему-то не подумал бы, что проверка == будет работать при инициализации тоже).

А вот то, что return может и сам посчитать — уже знал, но забыл 😞 Так бы сразу через него и сделал, но получилось вообще через if_o_O

Кстати по прошлым примерам, когда мы все действия записывали каждое в свою ф-цию, сделал всю программу на сколько смог. Пытался и последний if вписать, но с ним что-то не захотело...

😊 Только вот может в больших программах это и имеет смысл, но вот в таких — кажется излишне громоздким и не нужным. Ваш пример здесь намного удобней и красивей смотрится.

Кстати вопрос.

Задачи — тоже с источника, или сами делаете?)

```

1  #include <iostream>
2
3  using namespace std;

```

```
4
5 // Получаем число от пользователя
6 int get_number ()
7 {
8     cout << "Enter number: ";
9     int number = 0;
10    cin >> number;
11    return number;
12 }
13
14 // тут должно быть название isEven
15 // проверяем число пользователя на четность
16 int check_number(int number)
17 {
18     if (number % 2 == 0)
19         return 1;
20     else
21         return 0;
22 }
23
24
25 int main()
26 {
27     const int number = get_number();
28
29     // выводим результат в зависимости от четности числа
30     const bool return_number = check_number(number);
31     if (return_number == 1)
32         cout << "Your number is even.";
33     else
34         cout << "Your number is not even.";
35
36     return 0;
37 }
```

[Ответить](#)



13. Андрей:

[2 ноября 2018 в 09:56](#)

```
1 #include "pch.h"
2 #include <cstdlib>
3 #include <iostream>
4 using namespace std;
5
6 int inputNumb() {
7     cout << "Input your number: ";
8     int numb{ 0 };
9     cin >> numb;
10    return numb;
11 }
```

```
12
13 bool isEven() {
14     if (inputNumb() % 2 == 0) return 1;
15     else return 0;
16 }
17
18 int main()
19 {
20     cout << boolalpha;
21     cout << isEven() << endl;
22     system("Pause");
23     return 0;
24 }
```

Решил сделать вот так, надеюсь, что правильно.

[Ответить](#)



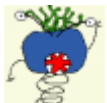
14. *Andrey:*

[29 сентября 2018 в 09:22](#)

немного упростил IsEven

```
1 #include <iostream>
2 using namespace std;
3
4 bool isEven(int a)
5 {
6     return !(a % 2);
7 }
8
9 int main()
10 {
11     cout << "please put number: ";
12     int a;
13     cin >> a;
14     if (isEven(a))
15         cout << a << " is even\n";
16     else
17         cout << a << " is odd\n";
18 }
```

[Ответить](#)



15. *Алексей:*

[25 сентября 2018 в 12:23](#)

```
1 #include <iostream>
2
3 using namespace std;
4
```



```

5 int getVol (); // Получаем целое число от пользователя
6 bool isEven (int vol); // Выясняем четность/нечетность числа
7 void prntRes (bool check); // Выводим результат на экран
8
9 int main()
10 {
11     cout << "Hello!" << endl;
12     int vol = getVol ();
13     bool check = isEven (vol);
14     prntRes (check);
15     return 0;
16 }
17 int getVol ()
18 {
19     cout << "Enter volume please: ";
20     int vol;
21     cin >> vol;
22     return vol;
23 }
24 bool isEven (int vol)
25 {
26     return (vol%2) == 0;
27 }
28 void prntRes (bool check)
29 {
30     if (check == true)
31         cout << "The number is even" << endl;
32     else
33         cout << "The number is odd" << endl;
34 }

```

[Ответить](#)



16. *Dennis:*

[4 сентября 2018 в 22:40](#)

У меня вопрос, а зачем вот так расписывать, когда , если нам нужно вывести просто результат ?

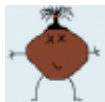
```

1 int a{ 0 };
2 cout << "Please, entre number : " << endl;
3 cin >> a;
4 cout << " result : " << !(a & 1);

```

Если только для подачи материала ? если так — то гуд.

[Ответить](#)



1. *Saturn21:*

[31 мая 2019 в 21:05](#)

Согласен. Но:
please ENTRE number
Эммм... вы француз?))
(Я знаю, что E и R находятся рядом, но это смешно)

[Ответить](#)



17. *SuRprizZe:*

[23 июня 2018 в 16:22](#)

Думаю это ,не плохое решение.

```
1 #include "stdafx.h"
2 #include <iostream>
3 using namespace std;
4
5 bool isEven()
6 {
7     int a;
8     cin >> a;
9     if (a % 2 == 0)
10         cout << "4islo 4etnoe!!!";
11     else
12         cout << "4islo ne 4etnoe";
13     return a;
14 }
15 int main()
16 {
17     cout << isEven() << "\n";
18     return 0;
19 }
```

[Ответить](#)



1. *Юрий:*

[24 июня 2018 в 14:52](#)

Плохое. Одна функция должна выполнять одну задачу. У вас все задачи выполняет одна функция.

[Ответить](#)



1. *Алексей:*

[18 июля 2018 в 20:21](#)

И красивее будет, если `setlocale(LC_ALL, "ru");` Без нее сейчас как без рук. Для понимания лучше.

[Ответить](#)

2. *Алексей:*[18 июля 2018 в 20:24](#)

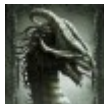
И вот вопрос: при вводе 5, сообщается о нечетности. А при вводе 5.5... тоже сообщается о нечетности)))) почему? Переменная `int`.

[Ответить](#)1. *Юрий:*[18 июля 2018 в 22:21](#)

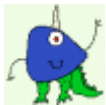
Потому что любая дробь отбрасывается, а не округляется. Будь у вас 5.99 — эта дробь отбросится и останется 5.

2. *Alexey:*[2 февраля 2019 в 16:34](#)

А еще функция должна возвращать значение `bool`, а возвращает `int`.

[Ответить](#)1. *ХейЛонг:*[19 апреля 2020 в 18:29](#)

Это не проблема, на самом деле, тут вообще не было разницы, какой тип функции. Её можно было сделать `void`, и просто выполнить. Но меня заинтересовало, что она выводит в `cout`, если на возврат ставится введённое пользователем число. Вполне ожидаемо, что ненулевое значение трансформировалось в логическую единицу. Так что сюрпризу спасибо за интересный эксперимент.

[Ответить](#)18. *Максим:*[27 февраля 2018 в 04:26](#)

Я так сделал

```

1  #include "stdafx.h"
2  #include <iostream>
3
4  void isEven(int x) {
5      if (x % 2 == 0)
6          std::cout << "Entered number is even";
7      else
8          std::cout << "Entered number is not even";
9  }
10
11 int getX() {

```

```
12     std::cout << "Enter the X-number: ";
13     int x;
14     std::cin >> x;
15     return x;
16 }
17
18 int main() {
19     isEven(getX());
20     return 0;
21 }
```

[Ответить](#)

1. *Юрий:*
[27 февраля 2018 в 13:33](#)

Хорошее решение, действительно.

[Ответить](#)

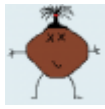
2. *somebox:*
[16 ноября 2018 в 02:18](#)

У меня похожий вариант:

```
1  #include <stdio.h>
2  #include <iostream>
3
4  using namespace std;
5
6  void isEven(int x) {
7      if (x % 2 == 0)
8          cout << x << " число четное\n";
9      else
10         cout << x << " число нечетное\n";
11 }
12
13 int main() {
14     cout << "Введите целое число: ";
15     int x(0);
16     cin >> x;
17
18     isEven(x);
19
20     return 0;
21 }
```

Не понимаю, зачем использовать с bool. По мне, это усложняет код.

[Ответить](#)



1. *Saturn21:*
[31 мая 2019 в 21:08](#)

Сетлокейл не забываем))

```
1 | setlocale(LC_ALL, "ru")
```

[Ответить](#)



2. *Дмитрий:*
[9 июня 2019 в 11:13](#)

Суть тестового задания не только в том что бы программа работала, скорее так: смыслом условий написания программ из тестов есть целенаправленное закрепления пройденного материала. То есть код хорош тогда, когда он удовлетворяет заданные условия из теста и в нем не нарушаются рекомендации указанные ранее.

[Ответить](#)



1. *Юрий:*
[9 июня 2019 в 11:29](#)

Именно!



3. *Хей Лонг:*
[19 апреля 2020 в 18:33](#)

Усложнение простого кода поможет в будущем упростить сложный код. Парадоксально немного, но так оно и работает. Тяжело в учении — легко в бою)

[Ответить](#)



19. *Ka4:*
[11 февраля 2018 в 12:46](#)

так можно же?)

```
1 | #include "stdafx.h"
2 | #include <iostream>
3 | #include <locale.h>
4 | #include <cmath>
5 | using namespace std;
6 |
7 | int chislo()
8 | {
9 |     cout << "Введите целое число: ";
10 |     int a;
11 |     cin >> a;
```

```
12     return a;
13 }
14
15
16 bool isEven(int v)
17 {
18     cout << boolalpha;
19     if (v % 2 == 0)
20         cout << true << endl;
21     else
22         cout << false << endl;
23     return 0;
24 }
25
26 int main()
27 {
28     setlocale(LC_ALL, "RUS");
29
30     int x = chislo();
31     isEven(x);
32
33     system("pause");
34     return 0;
35 }
```

[Ответить](#)



1. Юрий:

[13 февраля 2018 в 18:23](#)

Да, можно и так. Как вариант.

[Ответить](#)



2. Александр:

[19 февраля 2018 в 11:55](#)

Возвращение к единице обратно

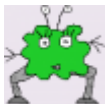
```
1  #include "stdafx.h"
2  #include <iostream>
3  #include <locale.h>
4
5  int main()
6  {
7      // переменная count хранит текущее число для вывода
8      int count = 1; // начинаем с 1
9
10     // Повторение операции (цикл) до тех пор, пока count не будет = 1000
11     while (count <= 1000)
12     {
```

```

13     std::cout << count << " "; // вывод текущего числа
14
15     // если count делится нацело на 10, без остатка - разрыв строки,
16     if (count % 10 == 0)
17         std::cout << "\n";
18
19     count = count + 1; // переходим к следующему числу
20 }
21
22 // переменная count хранит текущее число для вывода
23 int countm = 999; // продолжаем с 999
24
25 // Повторение операции (цикл) до тех пор, пока count не будет = 1
26 while (countm >= 1)
27 {
28     std::cout << countm << " "; // вывод текущего числа
29
30     // если count делится нацело на 10, без остатка - разрыв строки,
31     if (countm % 10 == 0)
32         std::cout << "\n";
33
34     countm = countm - 1; // переходим к следующему числу
35 }
36 std::cin.clear();
37 std::cin.ignore(32767, '\n');
38 std::cin.get();
39 return 0;
40 }

```

[Ответить](#)



20. Андрей:

[5 декабря 2017 в 18:19](#)

А как быть с нечетными 1 и -1?

[Ответить](#)



1. Юрий:

[5 декабря 2017 в 18:45](#)

Так ведь программа правильно работает. 1 и -1 — нечетные числа.

[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 «Same Game»](#)
-  [Урок №4. Установка IDE \(Интегрированной Среды Разработки\)](#)

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