

Ravesli [Ravesli](#)


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


Урок №13. Параметры и аргументы функций

 [Юрий](#) |

- [Уроки C++](#)

|

 Обновл. 2 Сен 2020 |

 92612

[↓](#)  59

На предыдущем уроке мы говорили о том, что функция может возвращать значение обратно в caller, используя **оператор return**. На этом уроке мы узнаем, что такое аргументы в функции и что такое параметры в функции.

Оглавление:

1. [Параметры и аргументы функций](#)
2. [Как работают параметры и аргументы функций?](#)
3. [Как работают параметры и возвращаемые значения функций?](#)
4. [Еще примеры](#)
5. [Тест](#)
6. [Ответы](#)

Параметры и аргументы функций

Во многих случаях нам нужно будет передавать данные в вызываемую функцию, чтобы она могла с ними как-то взаимодействовать. Например, если мы хотим написать функцию умножения двух чисел, то нам нужно каким-то образом сообщить функции, какие это будут числа. В противном случае, как она узнает, что на что перемножать? Здесь нам на помощь приходят параметры и аргументы.

Параметр функции — это переменная, которая используется в функции, и значение которой предоставляет caller (вызывающий объект). Параметры указываются при объявлении функции в круглых скобках. Если их много, то они перечисляются через запятую, например:

```
1 // Эта функция не имеет параметров
2 void doPrint()
3 {
4     std::cout << "In doPrint()" << std::endl;
```

```
5 }
6
7 // Эта функция имеет один параметр типа int: a
8 void printValue(int a)
9 {
10     std::cout << a << std::endl;
11 }
12
13 // Эта функция имеет два параметра типа int: a и b
14 int add(int a, int b)
15 {
16     return a + b;
17 }
```

Параметры каждой функции действительны только внутри этой функции. Поэтому, если `printValue()` и `add()` имеют параметр с именем `a`, то это не означает, что произойдет конфликт имен. Эти параметры считаются независимыми и никак не взаимодействуют друг с другом.

Аргумент функции — это значение, которое передается из `caller`-а в функцию и которое указывается в скобках при вызове функции в `caller`-е:

```
1 printValue(7); // 7 - это аргумент функции printValue()
2 add(4, 5); // 4 и 5 - это аргументы функции add()
```

Обратите внимание, аргументы тоже перечисляются через запятую. Количество аргументов должно совпадать с количеством параметров, иначе компилятор выдаст сообщение об ошибке.

Как работают параметры и аргументы функций?

При вызове функции, все её параметры создаются как локальные переменные, а значение каждого из аргументов копируется в соответствующий параметр (локальную переменную). Этот процесс называется **передачей по значению**. Например:

```
1 #include <iostream>
2
3 // Эта функция имеет два параметра типа int: a и b
4 // Значения переменных a и b определяет caller
5 void printValues(int a, int b)
6 {
7     std::cout << a << std::endl;
8     std::cout << b << std::endl;
9 }
10
11 int main()
12 {
13     printValues(8, 9); // здесь два аргумента: 8 и 9
14
15     return 0;
16 }
```

При вызове функции `printValues()` аргументы 8 и 9 копируются в параметры `a` и `b`. Параметру `a` присваивается значение 8, а параметру `b` — значение 9.

Результат:

8
9

Как работают параметры и возвращаемые значения функций?

Используя параметры и возвращаемые значения, мы можем создавать функции, которые будут принимать и обрабатывать данные, а затем возвращать результат обратно в `caller`.

Например, простая функция, которая принимает два целых числа и возвращает их сумму:

```
1 #include <iostream>
2
3 // Функция add() принимает два целых числа в качестве параметров и возвращает их сум
4 // Значения a и b определяет caller
5 int add(int a, int b)
6 {
7     return a + b;
8 }
9
10 // Функция main() не имеет параметров
11 int main()
12 {
13     std::cout << add(7, 8) << std::endl; // аргументы 7 и 8 передаются в функцию add
14     return 0;
15 }
```

При вызове функции `add()`, параметру `a` присваивается значение 7, а параметру `b` — значение 8. Затем функция `add()` вычисляет их сумму и возвращает результат обратно в `main()`. И тогда уже результат выводится на экран.

Результат выполнения программы:

15

Еще примеры

Рассмотрим еще несколько вызовов функций:

```
1 #include <iostream>
2
3 int add(int a, int b)
4 {
5     return a + b;
6 }
7
```

```
8 int multiply(int c, int d)
9 {
10     return c * d;
11 }
12
13 int main()
14 {
15     std::cout << add(7, 8) << std::endl; // внутри функции add(): a = 7, b = 8, значи
16     std::cout << multiply(4, 5) << std::endl; // внутри функции multiply(): c = 4, d
17
18     // Мы можем передавать целые выражения в качестве аргументов
19     std::cout << add(2 + 3, 4 * 5) << std::endl; // внутри функции add(): a = 5, b =
20
21     // Мы можем передавать переменные в качестве аргументов
22     int x = 4;
23     std::cout << add(x, x) << std::endl; // будет 4 + 4
24
25     std::cout << add(1, multiply(2, 3)) << std::endl; // будет 1 + (2 * 3)
26     std::cout << add(1, add(2, 3)) << std::endl; // будет 1 + (2 + 3)
27
28     return 0;
29 }
```

Результат выполнения программы:

```
15
20
25
8
7
6
```

С первыми двумя вызовами всё понятно.

В третьем вызове параметрами являются выражения, которые сначала нужно обработать. $2 + 3 = 5$ и результат 5 присваивается переменной *a*. $4 * 5 = 20$ и результат 20 присваивается переменной *b*. Результатом выполнения функции `add(5, 20)` является значение 25.

Следующая пара относительно лёгкая для понимания:

```
1 int x = 4;
2 std::cout << add(x, x) << std::endl; // будет 4 + 4
```

Здесь уже $a = x$ и $b = x$. Поскольку $x = 4$, то `add(x, x) = add(4, 4)`. Результат — 8.

Теперь рассмотрим вызов посложнее:

```
1 std::cout << add(1, multiply(2, 3)) << std::endl; // будет 1 + (2 * 3)
```

При выполнении этого **стейтмента** процессор должен определить значения параметров *a* и *b* функции `add()`. С параметром *a* всё понятно — мы передаем значение 1 ($a = 1$). А вот чтобы определить значение параметра *b*, нам необходимо выполнить операцию умножения: `multiply(2, 3)`, результат — 6. Затем `add(1, 6)` возвращает число 7, которое и выводится на экран.

Короче говоря:

`add(1, multiply(2, 3)) => add(1, 6) => 7`

Последний вызов может показаться немного сложным из-за того, что параметром функции `add()` является другой вызов `add()`:

```
1 | std::cout << add(1, add(2, 3)) << std::endl; // будет 1 + (2 + 3)
```

Но здесь всё аналогично вышеприведенному примеру. Перед тем, как процессор вычислит внешний вызов функции `add()`, он должен обработать внутренний вызов функции `add(2, 3)`. `add(2, 3) = 5`. Затем процессор обрабатывает функцию `add(1, 5)`, результатом которой является значение 6. Затем 6 передается в `std::cout`.

Короче говоря:

`add(1, add(2, 3)) => add(1, 5) => 6`

Тест

Задание №1: Что не так со следующим фрагментом кода?

```
1 | void multiply(int a, int b)
2 | {
3 |     return a * b;
4 | }
5 |
6 | int main()
7 | {
8 |     std::cout << multiply(7, 8) << std::endl;
9 |     return 0;
10| }
```

Задание №2: Какие здесь есть две проблемы?

```
1 | #include <iostream>
2 |
3 | int multiply(int a, int b)
4 | {
5 |     int product = a * b;
6 | }
7 |
8 | int main()
9 | {
10|     std::cout << multiply(5) << std::endl;
11|     return 0;
12| }
```

Задание №3: Какой результат выполнения следующей программы?

```
1 | #include <iostream>
2 |
3 | int add(int a, int b, int c)
```

```
4  {
5      return a + b + c;
6  }
7
8  int multiply(int a, int b)
9  {
10     return a * b;
11 }
12
13 int main()
14 {
15     std::cout << multiply(add(3, 4, 5), 5) << std::endl;
16     return 0;
17 }
```

Задание №4: Напишите функцию `doubleNumber()`, которая принимает целое число в качестве параметра, удваивает его, а затем возвращает результат обратно в caller.

Задание №5: Напишите полноценную программу, которая принимает целое число от пользователя (используйте [std::cin](#)), удваивает его с помощью функции `doubleNumber()` из предыдущего задания, а затем выводит результат на экран.

Ответы

Чтобы просмотреть ответ, кликните на него мышкой.

Ответ №1

Функция `multiply()` имеет тип возврата `void`, что означает, что эта функция не возвращает значения. Но, так как она все равно пытается вернуть значение с помощью оператора `return`, мы получим ошибку от компилятора. Функция должна иметь тип возврата `int`.

Ответ №2

Проблема №1: `main()` передает один аргумент в `multiply()`, но `multiply()` имеет два параметра.

Проблема №2: `multiply()` вычисляет результат и присваивает его локальной переменной, которую не возвращает обратно в `main()`. А поскольку тип возврата функции `multiply()` — `int`, то мы получим ошибку (в некоторых компиляторах) или неожиданные результаты (в остальных компиляторах).

Ответ №3

Функция `multiply()` принимает следующие параметры: `a = add(3, 4, 5)` и `b = 5`. Сначала процессор обрабатывает `a = add(3, 4, 5)`, т.е. $3 + 4 + 5 = 12$. Затем уже выполняет операцию умножения, результатом которой является `60`. Ответ: 60.

Ответ №4

```
1  int doubleNumber(int a)
2  {
3      return 2 * a;
4  }
```

Ответ №5

```
1  #include <iostream>
2
3  int doubleNumber(int a)
4  {
5      return 2 * a;
6  }
7
8  int main()
9  {
10     int a;
11     std::cout<<"Enter a number: ";
12     std::cin >> a;
13     std::cout << doubleNumber(a) << std::endl;
14     return 0;
15 }
16
17 /*
18 // Следующее решение является альтернативным:
19 int main()
20 {
21     int a;
22     std::cout<<"Enter a number: ";
23     std::cin >> a;
24     a = doubleNumber(a);
25     std::cout << a << std::endl;
26     return 0;
27 }
28 */
```

Примечание: У вас могут быть и другие решения заданий №4 и №5 — это ок. В программировании есть много случаев, когда одну задачу можно решить несколькими способами.

Оценить статью:

★★★★★ (604 оценок, среднее: 4,95 из 5)



[← Урок №12. Функции и оператор возврата return](#)

[Урок №14. Почему функции — полезны, и как их эффективно использовать?](#)



Комментариев: 59



1. *Rock:*

[16 июля 2020 в 00:33](#)

Да простят меня те кто буду это читать)))

```
1 #include<iostream>
2 int doubleNumber(int a){return a*2;}
3 void main(){int a;std::cin>>a;std::cout<<doubleNumber(a);}
```

[Ответить](#)



2. *Fray:*

[5 июля 2020 в 15:28](#)

Вопрос по второму заданию.

Решил дописать в конце тела объявления функции multiply следующее: «return a * b;» и мне выдало следующую ошибку: «unused variable 'product'», что можно перевести как «неиспользованная переменная 'product'». Но какая разница какая переменная, ведь возвращаемое значение есть и это "return a * b".

вот отредактированный код, чтобы лучше было видно:

```
1 #include <iostream>
2
3 int multiply(int a, int b)
4 {
5     int product = a * b;
6
7     return a * b;
8 }
9
10 int main()
11 {
12     std::cout << multiply(5, 6) << std::endl;
13     return 0;
14 }
```

Однако если написать вместо "return a * b;" — "return product", то все работает как надо.

То есть, объявляя переменную в объявлении функции и присваивая ей определенное значение, нужно вернуть именно ее в caller? Потому что чтобы я не писал в return, мне выдавало ошибку «unused variable 'product'».

Прошу прощения за свою невнимательность, если где то что то пропустил или мои вопросы звучат глупо, я еще в этом пытаюсь разобраться.

Еще раз спасибо за предоставленные материалы.

[Ответить](#)



1. *Александр:*

[5 августа 2020 в 22:40](#)

Проверил твой код, у меня все норм работает, ide — Visual Studio 2019 Community

[Ответить](#)2. *Никита:*[18 августа 2020 в 13:51](#)

Скорее всего всё из-за настроек из предыдущих уроков. Если делали всё по урокам, то все Предупреждения(Warnings) распознаются как ошибки. В качестве Warning он ругается, дескать зачем объявили переменную, если не использовали вообще в функции — странно это, батенька. А так как Warning = ошибка при заданных по урокам настройках IDE, то он и не идёт дальше, сообщая об ошибке.

Собственно ничего страшного нет и всё будет работать, но для этого автор и говорил о важности обращать внимания на предупреждения и подчищать хвосты.

[Ответить](#)3. *Spectre:*[18 августа 2020 в 16:13](#)

По сути ты занял память переменной "product", которую не используешь, поэтому компилятор ругается, критической ошибки нет, просто ты нерационально используешь ресурсы.

[Ответить](#)3. *Asklepiy:*[17 апреля 2020 в 20:25](#)

Я параллельно вашим урокам начал читать другую литературу по C++. Где-то вычитал, что в идеале функция main() должна лишь вызывать другие функции, не принимая участия в вычислениях. Это верно?

[Ответить](#)1. *Руслан:*[12 июня 2020 в 21:20](#)

Ну как, и да и нет. Если тебе удобно всё делать через функции — делай) Данный метод хорош при крупных проектах, где каждый программист пишет свою часть кода, а уже после подключают в главной функции.

[Ответить](#)4. *Хей.Лонг:*[23 марта 2020 в 01:45](#)

Читаю и вспоминаю любимую перегрузку функций. Вроде лёгкое явление, но препод постоянно акцентировал внимание на этом, видать большая тема для профессиональных разработчиков.

[Ответить](#)



5. *Олександра:*
[20 марта 2020 в 16:36](#)

И почему вы раньше мне не попадались??! От многих заумных фраз других сайтов в состоянии "Взрыв мозга"

[Ответить](#)



6. *Вадим:*
[12 февраля 2020 в 21:57](#)

Сколько лучше читать уроков в день? Что бы не загружаться лишний раз и запомнить то что я прочитал)

[Ответить](#)



7. *Daler:*
[28 января 2020 в 16:38](#)

это 4

```
1 #include <iostream>
2 int doubleNumber(int x) {
3     return 2 * x;
4 }
```

вот 5

```
1 #include <iostream>
2 int doubleNumber(int x)
3 {
4     return 2 * x;
5 }
6 int main()
7 {
8     setlocale(LC_ALL, "Russian");
9     int a;
10    std::cout << "введите число" << std::endl;
11    std::cin >> a;
12    std::cout << doubleNumber(a) << std::endl;
13    return 0;
14 }
```

[Ответить](#)



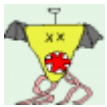
8. *Oleg:*
[22 января 2020 в 21:22](#)

Спасибо вам огромное за бесплатные уроки! Мне очень нравится, всё понятно и в конце каждого урока тест, а это важно. Я до этого немного изучал Java, поэтому мне пока что всё

просто. Вот моё решение №5:

```
1  #include <iostream>
2
3  using namespace std;
4
5  int doubleNumber(int a)
6  {
7      return a * 2;
8  }
9
10 int main()
11 {
12     cout << "Введите число: ";
13     int x;
14     cin >> x;
15     cout << x;
16     cout << " * 2 = ";
17     cout << doubleNumber(x) << endl;
18 }
```

[Ответить](#)

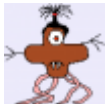


9. Олег:

[15 января 2020 в 13:09](#)

```
1  #include <iostream>
2
3  int enterNumber() {
4      int x;
5      std::cout << "Please enter a number..." << std::endl;
6      std::cin >> x;
7      return x;
8  }
9
10 int doubleNumber(int a) {
11     return a * 2;
12 }
13
14 int main() {
15     std::cout << "Your number after doubling is " << doubleNumber(enterNumber());
16     return 0;
17 }
```

[Ответить](#)



10. Владимир:

[7 января 2020 в 00:24](#)

Я в шоке, это лучшие уроки что я видел. Очень просто о том, что ранее не вмещалось в башку. Пока все понятно, и тесты реально закрепляют знания. Спасибо!

[Ответить](#)

1. Юрий:

[7 января 2020 в 01:18](#)

Пожалуйста 😊

[Ответить](#)

11. Владимир:

[8 декабря 2019 в 13:17](#)

Спасибо за это замечательное объяснение, подобного нигде не встретил, замечательно!

[Ответить](#)

1. Юрий:

[8 декабря 2019 в 14:08](#)

Пожалуйста)

[Ответить](#)

12. diflikator:

[25 ноября 2019 в 08:47](#)

Вариант выполнения задания №2.

```
1 #include <iostream>
2
3 int doubleNumber(int x)
4 {
5     return x * 2;
6 }
7
8 int cinx()
9 {
10     std::cout << "Enter a number: ";
11     int x;
12     std::cin >> x;
13     return x;
14 }
15
16 int main()
17 {
18     std::cout << "doubleNumber = " << doubleNumber(cinx());
19 }
20 }
```

[Ответить](#)

13. *Владимир:*[4 ноября 2019 в 17:02](#)

Здравствуйте, я может что то не так делаю, но VS 19 не работает если имя тела программы какое то кроме как `int main`. `int doubleNumber()` вот так уже не работает только `main`? в чем может быть проблема?

[Ответить](#)14. *Aleksandr:*[18 октября 2019 в 13:21](#)

Я все усложнил в 1000 раз, оказывается можно было намного проще код написать. Все верно получилось, но написал вот так:

```
1  int add() {
2      cout << "Enter number: " << endl;
3      int x;
4      cin >> x;
5      return x;
6  }
7
8  int doubleNumber(int a) {
9      return 2 * a;
10 }
11
12 int main() {
13     int a = add();
14     cout << doubleNumber(a) << endl;
15     return 0;
16 }
```

[Ответить](#)1. *Sergey:*[19 ноября 2019 в 13:19](#)

```
1  #include <iostream>
2
3  int doubleNumber(int a)
4  {
5      return 2 * a;
6  }
7
8
9  int main()
10 {
11     std::cout << "Enter a number: ";
12     int a = 0;
13     std::cin >> a;
```

```

14 |     std::cout << "Otvet: " << a << " * 2 = " << doubleNumber(a) << std::endl;
15 |     return 0;
16 | }

```

[Ответить](#)



15. *Дмитрий:*

[1 октября 2019 в 21:55](#)

Чувак спасибо :))

```

{
    благодарочка
}

```

[Ответить](#)



1. *Юрий:*

[2 октября 2019 в 08:33](#)

```

{
    Пожалуйста 😊
}

```

[Ответить](#)



16. *steyrin:*

[20 сентября 2019 в 08:44](#)

```

1 | #include "pch.h"
2 | #include <iostream>
3 |
4 | int doubleNumber(int b)
5 | {
6 |
7 |     return b * 2;
8 | }
9 |
10 | int main()
11 | {
12 |     int a;
13 |     std::cout << "Enter number ";
14 |     std::cin >> a;
15 |     std::cout << "Otvet " << doubleNumber(a) << std::endl;
16 |     return 0;
17 |
18 | }

```

Выдает ошибка "doubleNumber: функция не принимает 0 аргументов"

[Ответить](#)



1. *Sergey:*

[2 декабря 2019 в 19:18](#)

Как в поговорке а и б сидели на трубе)))

[Ответить](#)



17. *Михалыч:*

[26 июня 2019 в 22:26](#)

альтернатива для пятого задания

```
1 int main()
2 {
3     std::cout << "Enter a number: "; // просим пользователя ввести любое число
4     int a = 0;
5     std::cin >> a; // получаем пользовательское число и сохраняем его в переменную
6     std::cout << "answer: " << a*2;
7     return 0;
8 }
```

[Ответить](#)



1. *Innor:*

[30 июля 2019 в 01:56](#)

В задании было сказано: "Напишите функцию doubleNumber()". А вы её написали? Это задание ведь на проверку усвоения урока "Параметры и аргументы функции". Но согласен: путей решения задачи несколько.

[Ответить](#)

2. *Vladimir Dal':*

[18 сентября 2019 в 16:52](#)

Точно также решил

[Ответить](#)



18. *Алексей:*

[20 июня 2019 в 16:46](#)

Немного другим путем пошел сразу после прочтения "В программировании есть много случаев, когда одну задачу можно решить несколькими способами."

Подумал, так же и есть, а если...

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

```

3 | int doubleNumber() { int a; std::cout << "Enter value:"; std::cin >> a; return a; }
4 |
5 | int main () { std::cout << doubleNumber() << std::endl; }

```

[Ответить](#)



19. *Steve Dekart:*

[12 июня 2019 в 18:49](#)

Я уже дошёл до этой темы и боюсь забыть прошлые темы. Боюсь что на 200 теме буду не помнить какую-нибудь там 30 тему

[Ответить](#)



20. *Данис:*

[8 мая 2019 в 08:43](#)

А можете ли объяснить , я прост не понимаю почему в 3 задание в функции multiply b=5?

[Ответить](#)

21. *олег:*

[28 февраля 2019 в 03:13](#)

```

1 | #include <iostream>
2 |
3 | int doubleNumber()
4 | {
5 |     std::cout << "Введите число:" << std::endl;
6 |     int a = 0;
7 |     std::cin >> a;
8 |     a = a*2
9 |     std::cout << "Ваше число " << a << std::endl;
10 |
11 |     return 0;
12 | }
13 |
14 |
15 | int main()
16 | {
17 |     doubleNumber();
18 |     return 0;
19 | }

```

А такой вариант плох тем что одна функция выполняет 3 действия?

И еще вопрос, почему code::blocks спокойно выводит в консоль русские буквы без добавления set_locale?

[Ответить](#)



1. *Ростик:*
[8 мая 2019 в 14:15](#)

я хз

[Ответить](#)

22. *урнееро:*
[28 января 2019 в 21:10](#)

Один из лучших сайтов где можно выучить C++. Понятно все до мелочей, автору респект!!! Было бы неплохо добавить к примеру видео-уроки)). Еще раз спасибо за ваши старания

[Ответить](#)



1. *Юрий:*
[28 января 2019 в 22:17](#)

Пожалуйста 😊

[Ответить](#)

23. *Eugen:*
[8 января 2019 в 16:35](#)

Добрый день.

Хотелось бы добавить в данный урок более четкое разграничение функций с параметрами и без. Так как я считаю это важным. Делая задания по данному уроку решил немножко усложнить и столкнулся с проблемой понимания работы функции. То есть функция с параметрами требует аргумента, а функция без параметров нет. Но моя программа не работала... Регулярными ударами головой о проблему решил ее. Было интересно. Посему предлагаю рассмотреть данный вопрос.

И конечно мой код:

```
1  #include <iostream>
2
3  using namespace std;
4
5  int Get_Number()
6  {
7      int a;
8      cout << "Введите число " << endl;
9      cin >> a;
10     return a;
11 }
12
13 int doubleNumber()
14 {
```

```
15     return 2*Get_Number();
16
17 }
18
19 void Print_Func()
20 {
21     int b = doubleNumber() ;
22     cout << "Удвоенное число равно " << b << endl;
23 }
24
25 int main()
26 {
27
28     Print_Func();
29
30     return 0;
31 }
```

[Ответить](#)

24. *Илья:*

[6 июня 2018 в 17:07](#)

Вот тебе, админ, для примера альтернативная программа, ты в ответе предлагал вводить число в консоль, используя функцию `main()`, я написал альтернативную, где ввод происходит через функцию `doubleNumber()`, есть ли у такого способа минусы перед твоим, и если да, то какие??? Код скомпилировался, работает как надо.

```
1 #include <stdafx.h>
2 #include <iostream>
3
4 int doubleNumber(int a)
5 {
6     std::cout << "Enter a number: " << std::endl;
7     std::cin >> a;
8     return a * 2;
9 }
10 int main()
11 {
12     std::cout << doubleNumber(7) << std::endl;
13     return 0;
14 }
```

[Ответить](#)

1. *Илья:*

[6 июня 2018 в 17:13](#)

P.S:

В таком решении, когда мы в функции `main` вызываем функцию `doubleNumber()`, её аргументом можно прописать произвольное число при этом считать умножаться будет то

число, которое мы пропишем в консоли, а не аргумент. Кстати с аргументом "a" в функции `doubleNumber` код не компилируется.

[Ответить](#)



2. *Юрий:*

[8 июня 2018 в 15:19](#)

Вот беру и смотрю.

Правило хорошего программирования: одна функция должна выполнять одну задачу. (об этом в следующих уроках рассказывается детальнее).

У тебя `doubleNumber` выполняет две задачи: принимает число и умножает его. Для такой простенькой программы, как эта (это даже не программа — 13 строк кода всего лишь), можно использовать и такой, как у тебя вариант, и такой, как у меня, и другие, которые предлагали в комментариях. В более сложных программах оптимальным вариантом было бы сделать отдельную функцию для получения значения от пользователя и отдельную для умножения значения.

[Ответить](#)



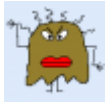
3. *Alexey:*

[24 января 2019 в 13:40](#)

```
1  #include "pch.h"
2  #include <iostream>
3  #include <Windows.h>
4
5  int doubleNumber(int a) {
6      return a * 2;
7  }
8  int userInput() {
9      std::cout << "Введите число " << std::endl;
10     int a = 0;
11     std::cin >> a;
12     return a;
13 }
14
15 int main()
16 {
17     SetConsoleCP(1251);
18     SetConsoleOutputCP(1251);
19     std::cout << doubleNumber(userInput()) << std::endl;
20     return 0;
21 }
```

так будет правильнее

[Ответить](#)



25. Ray:

[1 июня 2018 в 16:13](#)

У Вас, в примерах к уроку 13 указывается: `using namespace std;` и дальше везде все равно прописываются `std`: Корректно ли использование `std`: в добавок к `using namespace std;` ? Думаю, что они должны быть взаимоисключающими, я не прав ?

[Ответить](#)



1. Юрий:

[1 июня 2018 в 18:36](#)

Вы правы. Спасибо за бдительность, исправил.

[Ответить](#)



26. master114:

[24 апреля 2018 в 14:02](#)

Сделал через функции, так сказать с заделом на правильность — <http://www.onlinegdb.com/B1-Zzq3hM>.

[Ответить](#)



1. Юрий:

[28 апреля 2018 в 00:05](#)

Всё правильно и согласно рекомендациям из этого tutorиала. Радует, что уловили суть и пишете программы с самого начала правильным образом.

[Ответить](#)

27. Parenochek1488:

[26 марта 2018 в 15:32](#)

```
1 #include "stdafx.h"
2 #include <iostream>
3 using namespace std;
4 int a;
5 int main()
6 {
7     cin >> a;
8     cout << a * 2;
9 }
```

Я какой-то не правильный(

[Ответить](#)



1. Юрий:

[26 марта 2018 в 16:36](#)

У вас тут несколько замечаний (по поводу `using namespace std` и объявления переменных в глобальной области видимости). Продолжайте читать уроки — постепенно всё узнаете и сами поймете, что у вас не так.

[Ответить](#)

1. Parenochek1488:

[27 марта 2018 в 03:49](#)

Никак не могу запомнить, как работают параметры. Это решение единственное, которое пришло на ум. Но, как по мне, главное, что оно работает и введённое "a" умножается на два. Но урок я явно не усвоил, так что буду пробовать ещё. Спасибо за такой удобный сайт.

[Ответить](#)



1. Юрий:

[27 марта 2018 в 11:57](#)

Пожалуйста. По поводу параметров есть не один урок, всё, что нужно — рассматривается.



28. Семён:

[23 февраля 2018 в 20:47](#)

Спасибо ! Но по мне так лучше пока все поэтапно Но их плюс в том , что их можно повторять много раз .

[Ответить](#)

29. Александр:

[15 января 2018 в 06:51](#)

Для чего костыли с `std::` ? Если можно реализовать все быстрее без него с помощью `using namespace std;`,

[Ответить](#)



1. Юрий:

[15 января 2018 в 14:21](#)

Ответ на ваш вопрос в уроках [«Конфликт имен и std namespace»](#) и [«Using statements»](#).

[Ответить](#)

30. Дарья:

[7 января 2018 в 20:20](#)

Все прекрасно! Спасибо Вам! Не думала, что когда-нибудь возьмусь за изучение и что моей головы на это хватит! Написала решение к 5 заданию и какого было мое удивление когда она заработала!))

```
1 #include "stdafx.h"
2 #include <iostream>
3 #include <Windows.h>
4
5 int doubleNumber(int x)
6 {
7     return x * 2;
8 }
9
10 int main()
11 {
12     setlocale(LC_ALL, "Rus");
13     int a;
14     std::cout<<"Введите число: ";
15     std::cin >> a;
16     std::cout << doubleNumber(a) << std::endl;
17     return 0;
18 }
```

[Ответить](#)



1. Юрий:

[9 января 2018 в 15:48](#)

Никогда не знаешь, пока не попробуешь 😊

[Ответить](#)



2. Максим:

[6 декабря 2018 в 15:07](#)

какого пачему твой код сильно похож на мой? рандом наверно

```
1 #include <iostream>
2 int doubleNumber(int x)
3 {
4     return x + x;
5 }
6 int main()
7 {
8     int a;
9     std::cout << "enter a number:" ;
10    std::cin >> a;
```

```
11 | std::cout << doubleNumber(a) ;  
12 | return 0;  
13 | }
```

[Ответить](#)

31. *Юрий:*
[24 октября 2017 в 19:40](#)

По началу так и делал. Но практика такая не прижилась, как видите 😊

[Ответить](#)

32. *Сергей:*
[24 августа 2017 в 14:19](#)

Нужно добавить на сайте систему (лайк, дизлайк) к комментариям.

[Ответить](#)

1. *Юрий:*
[24 августа 2017 в 17:52](#)

Ну это уже в перспективе.

[Ответить](#)

33. *Сергей:*
[16 июля 2017 в 20:36](#)

Один из лучших уроков легко и элементарно объясняющий правило использования параметров в функции. Автору огромное спасибо, даже в книгах для новичков я не встречал раздела про параметры функции.

[Ответить](#)

1. *Юрий:*
[17 июля 2017 в 00:24](#)

Спасибо, приятно 😊

[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.

Отправить комментарий






TELEGRAM  КАНАЛ

Электронная почта



[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