

Ravesli [Ravesli](#)


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


## Урок №97. Передача по значению

 [Юрий](#) |

- [Уроки C++](#)

|

 Обновл. 1 Сен 2020 |

 23466

[1](#)  [2](#)

По умолчанию, аргументы в C++ передаются по значению. Когда аргумент **передается по значению**, то его значение копируется в параметр функции. Например:

```
1 #include <iostream>
2
3 void boo(int y)
4 {
5     std::cout << "y = " << y << std::endl;
6 }
7
8 int main()
9 {
10     boo(7); // 1-й вызов
11
12     int x = 8;
13     boo(x); // 2-й вызов
14     boo(x + 2); // 3-й вызов
15
16     return 0;
17 }
```

В первом вызове функции boo() аргументом является **литерал** 7. При вызове boo() создается переменная y, в которую копируется значение 7. Затем, когда boo() завершает свое выполнение, переменная y уничтожается.

Во втором вызове функции `boo()` аргументом является уже переменная `x = 8`. Когда `boo()` вызывается во второй раз, переменная `y` создается снова и значение 8 копируется в `y`. Затем, когда `boo()` завершает свое выполнение, переменная `y` снова уничтожается.

В третьем вызове функции `boo()` аргументом является выражение `x + 2`, которое вычисляется в значение 10. Затем это значение передается в переменную `y`. При завершении выполнения функции `boo()` переменная `y` вновь уничтожается.

Таким образом, результат выполнения программы:

```
y = 7
y = 8
y = 10
```

Поскольку в функцию передается копия аргумента, то исходное значение не может быть изменено функцией. Это хорошо проиллюстрировано в следующем примере:

```
1  #include <iostream>
2
3  void boo(int y)
4  {
5      std::cout << "y = " << y << '\n';
6
7      y = 8;
8
9      std::cout << "y = " << y << '\n';
10 } // y уничтожается здесь
11
12 int main()
13 {
14     int x = 7;
15     std::cout << "x = " << x << '\n';
16
17     boo(x);
18
19     std::cout << "x = " << x << '\n';
20     return 0;
21 }
```

Результат:

```
x = 7
y = 7
y = 8
x = 7
```

В начале функции `main()` `x` равно 7. При вызове `boo()` значение `x` (7) передается в параметр `y` функции `boo()`. Внутри `boo()` переменной `y` сначала присваивается значение 8, а затем `y` уничтожается. Значение `x` не изменяется, даже если изменить `y`.

Параметры функции, переданные по значению, также могут быть **const**. Тогда уже будет 100% гарантия того, что функция не изменит значение параметра.

## Плюсы и минусы передачи по значению

### Плюсы передачи по значению:

- ➕ Аргументы, переданные по значению, могут быть переменными (например, `x`), литералами (например, `8`), выражениями (например, `x + 2`), **структурами**, классами или **перечислителями** (т.е. почти всем, чем угодно).
- ➕ Аргументы никогда не изменяются функцией, в которую передаются, что предотвращает возникновение **побочных эффектов**.

### Минусы передачи по значению:

- Копирование структур и классов может привести к значительному снижению производительности (особенно, когда функция вызывается много раз).

### Когда использовать передачу по значению:

- ➔ При передаче фундаментальных типов данных и перечислителей, когда предполагается, что функция не должна изменять аргумент.

### Когда не использовать передачу по значению:

- ➔ При передаче **массивов**, структур и классов.

В большинстве случаев, передача по значению — это наилучший способ передачи аргументов фундаментальных типов данных, когда функция не должна изменять исходные значения. Передача по значению является гибкой и безопасной, а в случае фундаментальных типов данных еще и эффективной.

Оценить статью:

★★★★★ (195 оценок, среднее: 4,96 из 5)

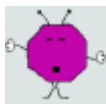


← [Урок №96. Параметры и аргументы функций](#)

[Урок №98. Передача по ссылке](#) ➔



Комментариев: 2



1. *Александр:*

[24 февраля 2019 в 14:38](#)

В минусы передачи по значению обязательно нужно добавить передачу структур, которые содержат динамические параметры (при условии, что мы не уверены, насколько качественно эта структура прописана). Так как это может приводить к абсолютно непредсказуемым результатам.

[Ответить](#)



2. *Роман:*

[5 августа 2018 в 20:00](#)

Спасибо вам за уроки, всё понятно, мне всего 14 а я уже на пол пути в изучении C++, одна мне непонятные перечисления и классы, я не до конца понял как ими пользоваться, а главное зачем, почти все тесты и задания я писал без них.

[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