

Ravesli [Ravesli](#)


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


Урок №96. Параметры и аргументы функций

 [Юрий](#) |

- [Уроки C++](#)

|

 Обновл. 26 Сен 2020 |

 28597

[1](#)  [3](#)

В первой главе этого tutorials мы рассматривали функции на следующих уроках:

- [Урок №12. Функции и оператор возврата return](#)
- [Урок №13. Параметры и аргументы функций](#)
- [Урок №19. Прототип функции и Предварительное объявление](#)
- [Урок №20. Многофайловые программы](#)
- [Урок №21. Заголовочные файлы](#)

Перед тем как продолжить, вы должны быть знакомы с концепциями, обсуждаемыми на этих уроках.

Параметры vs. Аргументы

На следующих 3-х уроках мы поговорим о параметрах и аргументах, поэтому давайте вкратце вспомним их определения.

Параметр функции (или «*формальный параметр*») — это переменная, создаваемая в объявлении функции:

```
1 void boo(int x); // объявление (прототип функции). x - это параметр
2
3 void boo(int x) // определение (также объявление). x - это параметр
4
```

```
5 | {  
   | }  
   | }
```

Аргумент (или «*фактический параметр*») — это значение, которое передает в функцию вызывающий объект (caller):

```
1 | boo(7); // 7 - это аргумент, который передается в параметр x  
2 | boo(y+1); // выражение y+1 - это аргумент, который передается в параметр x
```

Когда функция вызывается, все параметры функции создаются как переменные, а значения аргументов копируются в параметры. Например:

```
1 | void boo(int x, int y)  
2 | {  
3 | }  
4 |  
5 | boo(4, 5);
```

При вызове функции `boo()` с аргументами 4 и 5, создаются параметры `x` и `y` функции `boo()` и им присваиваются соответствующие значения: 4 и 5. Результатом будет `x = 4` и `y = 5`.

Примечание: В примере, приведенном выше, порядок обработки параметров в функции `boo()` будет справа налево, т.е. сначала создастся переменная `y` и ей присвоится значение 5, а затем уже создастся переменная `x` и ей присвоится значение 4. Порядок, в котором инициализируются параметры в круглых скобках функции, определяет каждый компилятор отдельно, так как C++ явно не указывает этот порядок обработки. С параметрами-переменными это не столь важно и критично, но если вы будете использовать в качестве параметров функции вызовы других функций (что является плохой практикой и не рекомендуется к использованию), то результат может быть неожиданным.

Рассмотрим следующую программу:

```
1 | #include <iostream>  
2 |  
3 | int prinX()  
4 | {  
5 |     std::cout << "x = 4\n";  
6 |     return 0;  
7 | }  
8 |  
9 | int prinY()  
10 | {  
11 |     std::cout << "y = 5\n";  
12 |     return 0;  
13 | }  
14 |  
15 | void prinAll(int a, int b) {}  
16 |  
17 | int main() {  
18 |  
19 |     prinAll(prinX(), prinY()); // в качестве параметров функции используются вызовы ф
```

```
20 |     return 0;  
21 | }
```

Результат выполнения программы:

```
y = 5  
x = 4
```

Хотя параметры не объявлены внутри блока функции, они имеют локальную область видимости. Это означает, что они создаются при вызове функции и уничтожаются, когда блок функции завершается:

```
1 | void boo(int x, int y) // x и y создаются здесь  
2 | {  
3 | } // x и y уничтожаются здесь
```

Существует 3 основных способа передачи аргументов в функцию:

- ➔ передача по значению;
- ➔ передача по ссылке;
- ➔ передача по адресу.

Мы рассмотрим каждый из этих способов по порядку.

Оценить статью:

★★★★★ (177 оценок, среднее: 4,90 из 5)

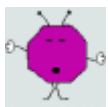


[← Глава №6. Итоговый тест](#)

[Урок №97. Передача по значению](#) ➔



Комментариев: 3



1. [Александр:](#)
[24 февраля 2019 в 14:32](#)

Я бы еще уточнил, что передавать результат выполнения функций в качестве аргумента вполне безопасно в случае, если у этих функций нет побочных эффектов... Или эти побочные эффекты полностью независимы.

[Ответить](#)2. *Алексей:*[30 марта 2018 в 14:52](#)

Просто хочу уточнить, что здесь описано не совсем верно. Функция считывает параметры в обратном порядке (по крайней мере в Visual Studio 2017).

И если для переменных это не критически важно, то при передаче функции в качестве аргумента (такое конечно лучше не творить, но все же) могут возникнуть ошибки, например:

```
1 int prinX() { std::cout << "x = 4\n"; return 1;}
2
3 int prinY() { std::cout << "y = 5\n"; return 1;}
4
5 void prinAll(int a, int b){}
6
7 int main() {
8     prinAll(prinX(), prinY());
9     system("pause");
10    return 0;
11 }
```

выводом будет:

y = 5

x = 4

[Ответить](#)1. *Юрий:*[2 апреля 2018 в 19:38](#)

Вы правы. C++ явно не указывает порядок инициализации параметров в функции и уже каждый компилятор сам решает в каком порядке ему их инициализировать. Статью обновил.

[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