

Ravesli [Ravesli](#)


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


Урок №103. Параметры по умолчанию

 [Юрий](#) |

- [Уроки C++](#)

|

 Обновл. 27 Сен 2020 |

 37031

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

На этом уроке мы рассмотрим, что такое параметры по умолчанию в языке C++ и то, как их использовать.

Оглавление:

1. [Параметры по умолчанию](#)
2. [Несколько параметров по умолчанию](#)
3. [Объявление параметров по умолчанию](#)
4. [Параметры по умолчанию и перегрузка функций](#)
5. [Заключение](#)

Параметры по умолчанию

Параметр по умолчанию (или «*необязательный параметр*») — это параметр функции, который имеет определенное (по умолчанию) значение. Если пользователь не передает в функцию значение для параметра, то используется значение по умолчанию. Если же пользователь передает значение, то это значение используется вместо значения по умолчанию. Например:

```
1 #include <iostream>
2
3 void printValues(int a, int b=5)
4 {
5     std::cout << "a: " << a << '\n';
6     std::cout << "b: " << b << '\n';
7 }
```

```
8 |  
9 | int main()  
10 | {  
11 |     printValues(1); // в качестве b будет использоваться значение по умолчанию - 5  
12 |     printValues(6, 7); // в качестве b будет использоваться значение, предоставляемое  
13 | }
```

Результат выполнения программы:

```
a: 1  
b: 5  
a: 6  
b: 7
```

В первом вызове функции мы не передаем аргумент для `b`, поэтому функция использует значение по умолчанию — 5. Во втором вызове мы передаем значение для `b`, поэтому оно используется вместо параметра по умолчанию.

Параметр по умолчанию — это отличный вариант, когда функция нуждается в значении, которое пользователь может переопределить, а может и не переопределить. Например, вот несколько [прототипов функций](#), для которых могут использоваться параметры по умолчанию:

```
1 | void openLogFile(std::string filename="default.log");  
2 | int rollDie(int sides=6);  
3 | void printStringInColor(std::string str, Color color=COLOR_RED); // Color - это перечислитель
```

Несколько параметров по умолчанию

Функция может иметь несколько параметров по умолчанию:

```
1 | void printValues(int a=10, int b=11, int c=12)  
2 | {  
3 |     std::cout << "Values: " << a << " " << b << " " << c << '\n';  
4 | }
```

При следующих вызовах функции:

```
1 | printValues(3, 4, 5);  
2 | printValues(3, 4);  
3 | printValues(3);  
4 | printValues();
```

Результат следующий:

```
Values: 3 4 5  
Values: 3 4 12
```

```
Values: 3 11 12
Values: 10 11 12
```

Обратите внимание, предоставить аргумент для параметра `c`, не предоставляя при этом аргументы для параметров `a` и `b` — нельзя (перепрыгивать через параметры не разрешается). Это связано с тем, что язык C++ не поддерживает следующий синтаксис вызова функции: `printValues(, , 5)`. Из этого вытекают следующие два правила:

Правило №1: Все параметры по умолчанию в прототипе или в определении функции должны находиться справа. Следующее вызовет ошибку:

```
1 void printValue(int a=5, int b); // не разрешается
```

Правильно:

```
1 void printValue(int a, int b=5);
```

Правило №2: Если имеется более одного параметра по умолчанию, то самым левым параметром по умолчанию должен быть тот, который с наибольшей вероятностью (среди всех остальных параметров) будет явно переопределен пользователем.

Объявление параметров по умолчанию

Как только параметр по умолчанию объявлен, повторно объявить его уже нельзя. Это значит, что для функции с предварительным объявлением и определением, параметр по умолчанию объявить можно либо в предварительном объявлении, либо в определении функции, но не в обоих местах сразу.

Например:

```
1 void printValues(int a, int b=15);
2
3 void printValues(int a, int b=15) // ошибка: переопределение параметра по умолчанию
4 {
5     std::cout << "a: " << a << '\n';
6     std::cout << "b: " << b << '\n';
7 }
```

Рекомендуется объявлять параметры по умолчанию в предварительном объявлении, а не в определении функции, так как предварительные объявления можно использовать в нескольких файлах — при таком раскладе параметры по умолчанию будет легче увидеть (особенно, если предварительное объявление находится в [заголовочном файле](#)). Например:

boo.h:

```
1 #ifndef BOO_H
2 #define BOO_H
3 void printValues(int a, int b=15);
4 #endif
```

main.cpp:

```
1 #include "boo.h"
2 #include <iostream>
3
4 void printValues(int a, int b)
5 {
6     std::cout << "a: " << a << '\n';
7     std::cout << "b: " << b << '\n';
8 }
9
10 int main()
11 {
12     printValues(7);
13
14     return 0;
15 }
```

Обратите внимание, в примере, приведенном выше, используется параметр по умолчанию `b` для функции `printValues()`, так как `main.cpp` подключает `boo.h`, который имеет предварительное объявление функции `printValues()` с объявленным параметром по умолчанию.

Правило: Объявляйте параметры по умолчанию в предварительном объявлении функции, в противном случае (если функция не имеет предварительного объявления) — объявляйте в определении функции.

Параметры по умолчанию и перегрузка функций

Функции с параметрами по умолчанию могут быть перегружены. Например:

```
1 void print(std::string string);
2 void print(char ch=' ');
```

Если пользователь вызовет просто `print()` (без параметров), то выведется пробел, что будет результатом выполнения `print(' ')`.

Однако, стоит отметить, что параметры по умолчанию НЕ относятся к параметрам, которые учитываются при определении уникальности функции. Следовательно, следующее не допускается:

```
1 void printValues(int a);
2 void printValues(int a, int b=15);
```

При вызове `printValues(10)` компилятор не сможет определить, хотите ли вы вызвать `printValues(int)` или `printValues(int, 15)` (со значением по умолчанию).

Заключение

Параметры по умолчанию — это полезный механизм указания параметров, при котором пользователь может переопределять значения по умолчанию, либо не переопределять их вообще. Они часто

используются в языке C++, и их применение вы увидите уже на следующих уроках.

Оценить статью:

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



← [Урок №102. Перегрузка функций](#)

[Урок №104. Указатели на функции](#) →



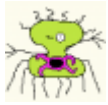
Комментариев: 2



1. *Снова я:*
[28 января 2020 в 11:44](#)

То что при перегрузке невозможно использовать параметры по умолчанию это конечно ужасно.
Есть выход из этой ситуации?

[Ответить](#)



2. *Илья:*
[6 января 2020 в 09:23](#)

Ребят поздравляю всех кто дошёл до этого момента!!
Вы прошли уже половину пути в изучении C++!!!
(Впрочем как и я 😊)

[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](#)
[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