

Ravesli [Ravesli](#)


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


Урок №89. Ссылки и const

 [Юрий](#) |

- [Уроки C++](#)

|

 Обновл. 16 Авг 2020 |

 33626

[↑](#)  10

Так же, как можно объявить [указатель на константное значение](#), так же можно объявить и [ссылку](#) на константное значение в языке C++.

Оглавление:

1. [Ссылки на константные значения](#)
2. [Инициализация ссылок на константы](#)
3. [Ссылки r-values](#)
4. [Константные ссылки в качестве параметров функции](#)

Ссылки на константные значения

Объявить ссылку на константное значение можно путем добавления **ключевого слова const** перед **типом данных**:

```
1 const int value = 7;  
2 const int &ref = value; // ref - это ссылка на константную переменную value
```

Ссылки на константные значения часто называют просто «**ссылки на константы**» или «**константные ссылки**».

Инициализация ссылок на константы

В отличие от ссылок на неконстантные значения, которые могут быть инициализированы только неконстантными **l-values**, ссылки на константные значения могут быть инициализированы неконстантными l-values, константными l-values и r-values:

```
1 int a = 7;
2 const int &ref1 = a; // ок: a - это неконстантное l-value
3
4 const int b = 9;
5 const int &ref2 = b; // ок: b - это константное l-value
6
7 const int &ref3 = 5; // ок: 5 - это r-value
```

Как и в случае с указателями, константные ссылки также могут ссылаться и на неконстантные переменные. При доступе к значению через константную ссылку, это значение автоматически считается const, даже если исходная переменная таковой не является:

```
1 int value = 7;
2 const int &ref = value; // создаем константную ссылку на переменную value
3
4 value = 8; // ок: value - это не константа
5 ref = 9; // нельзя: ref - это константа
```

Ссылки r-values

Обычно r-values имеют область видимости выражения, что означает, что они уничтожаются в конце выражения, в котором созданы:

```
1 std::cout << 3 + 4; // 3 + 4 вычисляется в r-value 7, которое уничтожается в конце этого
```

Однако, когда константная ссылка инициализируется значением r-value, время жизни r-value продлевается в соответствии со временем жизни ссылки:

```
1 int somefcn()
2 {
3     const int &ref = 3 + 4; // обычно результат 3 + 4 имеет область видимости выражения
4     std::cout << ref; // мы можем использовать его здесь
5 } // и время жизни r-value продлевается до этой точки, когда константная ссылка уничтожится
```

Константные ссылки в качестве параметров функции

Ссылки, используемые в качестве параметров функции, также могут быть константными. Это позволяет получить доступ к аргументу без его копирования, гарантируя, что функция не изменит значение, на которое ссылается ссылка:

```
1 // ref - это константная ссылка на переданный аргумент, а не копия аргумента
2 void changeN(const int &ref)
```

```
3 | {  
4 |     ref = 8; // нельзя: ref - это константа  
5 | }
```

Ссылки на константные значения особенно полезны в качестве параметров функции из-за их универсальности. Константная ссылка в качестве параметра позволяет передавать неконстантный аргумент l-value, константный аргумент l-value, литерал или результат выражения:

```
1 | #include <iostream>  
2 |  
3 | void printIt(const int &a)  
4 | {  
5 |     std::cout << a;  
6 | }  
7 |  
8 | int main()  
9 | {  
10 |     int x = 3;  
11 |     printIt(x); // неконстантное l-value  
12 |  
13 |     const int y = 4;  
14 |     printIt(y); // константное l-value  
15 |  
16 |     printIt(5); // литерал в качестве r-value  
17 |  
18 |     printIt(3+y); // выражение в качестве r-value  
19 |  
20 |     return 0;  
21 | }
```

Результат выполнения программы:

3457

Во избежание ненужного, слишком затратного копирования аргументов, переменные, которые не являются фундаментальных типов данных (типов `int`, `double` и т.д.) или указателями, — должны передаваться по (константной) ссылке в функцию. Фундаментальные типы данных должны передаваться по значению в случае, если функция не будет изменять их значений.

Правило: Переменные не фундаментальных типов данных и которые не являются указателями, передавайте в функцию по (константной) ссылке.

Оценить статью:

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



[← Урок №88. Ссылки](#)[Урок №90. Оператор доступа к членам через указатель →](#)

Комментариев: 10



1. *Дмитрий:*

[11 ноября 2020 в 09:18](#)

Очень хорошие уроки у авторов. Спасибо вам большое за хороший перевод. Очень хорошо дополняет книгу C++ базовый курс (Липпман).

[Ответить](#)



1. *Юрий:*

[11 ноября 2020 в 16:29](#)

Пожалуйста 😊

[Ответить](#)



2. *Константин:*

[14 июня 2020 в 10:29](#)

А ссылку на массив сделать можно?
(Соре я немного тупой)

[Ответить](#)



1. *Andrey:*

[5 октября 2020 в 15:24](#)

```
1  #include <iostream>
2
3  using namespace std;
4
5  int main()
6  {
7      int val1[100]{1000};
8      auto &val = val1;
9      auto *ptr = val;
10     cout << *ptr;
11     cout << "PALANTIR" << endl;
12     return 0;
```

13 | }

Можно через авто, но будет указывать только на первое значение))

[Ответить](#)



3. *Дмитрий:*
[15 июня 2019 в 14:24](#)

А как правильно передать массив в функцию? Писать амперсанта или не обязательно?

[Ответить](#)



1. *Анастасия:*
[28 июня 2019 в 16:48](#)

В главе про связь указателей и массивов было сказано, что лучше всего передавать массив в функцию как указатель. Например, так: `void doSomething(int *array)`

[Ответить](#)



2. *Анастасия:*
[28 июня 2019 в 16:51](#)

В любом случае при передаче массива в функцию он передаётся как указатель и поэтому при изменении его значений, они реально будут меняться. Насколько я понимаю, как константу массив в функцию передать нельзя.

[Ответить](#)



1. *Константин:*
[21 ноября 2019 в 22:45](#)

Если я правильно понял Ваши сомнения:

```

1  #include <iostream>
2
3  void changeArray(const int change[])// функция принимает массив как кон
4  {
5      int x = change[0];// элементы массива можно использовать для чтение
6      x += 3;// попытка написать здесь change[0]+=3; вызовет ошибку при к
7      std::cout << '\n' << x << '\n';
8  }
9
10 int main()
11 {
12     int subj[] = { 1, 2, 3 };// массив subj - неконстантный

```

```
13     for (int i = 0; i < 3; ++i)
14         std::cout << subj[i] << ' ';
15
16     std::cout << std::endl;
17
18     subj[0] += 3; // здесь всё ок
19     for (int i = 0; i < 3; ++i)
20         std::cout << subj[i] << ' ';
21
22
23     changeArray(subj); // передаём массив в функцию, которая будет его р
24
25     return 0;
26 }
```

[Ответить](#)

4. *Vlados_Bro:*
[1 апреля 2019 в 12:25](#)

Дякую, інформативно

[Ответить](#)

1. *Юрий:*
[1 апреля 2019 в 18:47](#)

Будь ласка, заходите — читайте 😊

[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