

Ravesli [Ravesli](#)


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


## Урок №87. Указатели и const

 [Юрий](#) |

- [Уроки C++](#)

|

 Обновл. 16 Авг 2020 |

 38579

[1](#)  [5](#)

На этом уроке мы рассмотрим указатели на константные переменные, указатели на константные значения, константные указатели и константные указатели на константные значения в языке C++.

Оглавление:

1. [Указатели на константные переменные](#)
2. [Указатели на константные значения](#)
3. [Константные указатели](#)
4. [Константные указатели на константные значения](#)
5. [Заключение](#)

## Указатели на константные переменные

До этого момента все [указатели](#), которые мы рассматривали, были неконстантными указателями на неконстантные значения:

```
1 int value = 7;
2 int *ptr = &value;
3 *ptr = 8; // изменяем значение value на 8
```

Однако, что произойдет, если указатель будет указывать на [константную](#) переменную?

```
1 const int value = 7; // value - это константа
2 int *ptr = &value; // ошибка компиляции: невозможно конвертировать const int* в int*
3 *ptr = 8; // изменяем значение value на 8
```

Фрагмент кода, приведенный выше, не скомпилируется: мы не можем присвоить неконстантному указателю константную переменную. Здесь есть смысл, ведь на то она и константа, что её значение нельзя изменить. Гипотетически, если бы мы могли присвоить константное значение неконстантному указателю, то тогда мы могли бы разыменовать неконстантный указатель и изменить значение этой же константы. А это уже является нарушением самого понятия «константа».

## Указатели на константные значения

**Указатель на константное значение** — это неконстантный указатель, который указывает на неизменное значение. Для объявления указателя на константное значение, используется **ключевое слово const перед типом данных**:

```
1 const int value = 7;
2 const int *ptr = &value; // здесь всё ок: ptr - это неконстантный указатель, который ук
3 *ptr = 8; // нельзя, мы не можем изменить константное значение
```

В примере, приведенном выше, ptr указывает на константный целочисленный тип данных.

Пока что всё хорошо. Рассмотрим следующий пример:

```
1 int value = 7; // value - это не константа
2 const int *ptr = &value; // всё хорошо
```

Указатель на константную переменную может указывать и на неконстантную переменную (как в случае с переменной value в примере, приведенном выше). Подумайте об этом так: указатель на константную переменную обрабатывает переменную как константу при получении доступа к ней независимо от того, была ли эта переменная изначально определена как const или нет. Таким образом, следующее в порядке вещей:

```
1 int value = 7;
2 const int *ptr = &value; // ptr указывает на "const int"
3 value = 8; // переменная value уже не константа, если к ней получают доступ через некон
```

Но не следующее:

```
1 int value = 7;
2 const int *ptr = &value; // ptr указывает на "const int"
3 *ptr = 8; // ptr обрабатывает value как константу, поэтому изменение значения переменной
```

Указателю на константное значение, который сам при этом не является константным (он просто указывает на константное значение), можно присвоить и другое значение:

```
1 int value1 = 7;
2 const int *ptr = &value1; // ptr указывает на const int
3
4 int value2 = 8;
5 ptr = &value2; // хорошо, ptr теперь указывает на другой const int
```

## Константные указатели

Мы также можем сделать указатель константным. **Константный указатель** — это указатель, значение которого не может быть изменено после инициализации. Для объявления константного указателя используется **ключевое слово const** между звёздочкой и именем указателя:

```
1 int value = 7;
2 int *const ptr = &value;
```

Подобно обычным константным переменным, константный указатель должен быть инициализирован значением при объявлении. Это означает, что он всегда будет указывать на один и тот же адрес. В вышеприведенном примере `ptr` всегда будет указывать на адрес `value` (до тех пор, пока указатель не выйдет из **области видимости** и не уничтожится):

```
1 int value1 = 7;
2 int value2 = 8;
3
4 int * const ptr = &value1; // ок: константный указатель инициализирован адресом value1
5 ptr = &value2; // не ок: после инициализации константный указатель не может быть изменен
```

Однако, поскольку переменная `value`, на которую указывает указатель, не является константой, то её значение можно изменить путем разыменования константного указателя:

```
1 int value = 7;
2 int *const ptr = &value; // ptr всегда будет указывать на value
3 *ptr = 8; // ок, так как ptr указывает на тип данных (неконстантный int)
```

## Константные указатели на константные значения

Наконец, можно объявить константный указатель на константное значение, используя **ключевое слово const** как перед типом данных, так и перед именем указателя:

```
1 int value = 7;
2 const int *const ptr = &value;
```

Константный указатель на константное значение нельзя перенаправить указывать на другое значение также, как и значение, на которое он указывает, — нельзя изменить.

## Заключение

Подводя итоги, вам нужно запомнить всего лишь 4 правила:

- ➔ Неконстантный указатель можно перенаправить указывать на любой другой адрес.
- ➔ С помощью указателя на неконстантное значение можно изменить это же значение (на которое он указывает).

- Константный указатель всегда указывает на один и тот же адрес, и этот адрес не может быть изменен.
- Указатель на константное значение обрабатывает значение как константное (даже если оно таковым не является) и, следовательно, это значение через указатель изменить нельзя.

А вот с синтаксисом может быть немного труднее. Просто помните, что тип значения, на который указывает указатель, всегда находится слева (в самом начале):

```
1 int value = 7;  
2 const int *ptr1 = &value; // ptr1 указывает на "const int", поэтому это указатель на константу  
3 int *const ptr2 = &value; // ptr2 указывает на "int", поэтому это константный указатель  
4 const int *const ptr3 = &value; // ptr3 указывает на "const int", поэтому это константный указатель на константу
```

Указатели на константные значения в основном используются в параметрах функций (например, при передаче массива) для гарантии того, что функция случайно не изменит значение(я) переданного ей аргумента.

Оценить статью:

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



← [Урок №86. Динамические массивы](#)

[Урок №88. Ссылки](#) →



## Комментариев: 5

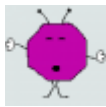


1. Алексей:

[13 декабря 2019 в 13:35](#)

Для лучшего понимания очередности написания типа данных, звездочки и слова const в объявлении указателя, можно прибегнуть к простому приему — прочитать полученную конструкцию справа налево: const int \*ptr1 — значит ptr1 — указатель на int константный int \*const ptr2 — значит ptr2 — константный указатель на int

[Ответить](#)



2. Александр:

[22 февраля 2019 в 15:10](#)

Есть ли разница между

```
1 | const int
```

и

```
1 | int const
```

?

Это просто вопрос договоренностей и стиля или есть различия между этими формами?

[Ответить](#)



1. *fan78:*

[11 марта 2019 в 01:25](#)

Нет никакой разницы. Сама const может быть в любой части. Даже можно в середине:  
unsigned const int

[Ответить](#)



3. *Евгений:*

[16 октября 2017 в 08:45](#)

Спасибо Вам за этот цикл по C++ !!! Читаю его и понимаю что мыслить начинаю так, как должен мыслить программист пишущий на C++. )) Шаг за шагом становится всё интересней. Если раньше открывая программу на C++ с ужасом закрывал её, то теперь понимаю ход мыслей человека который её написал.

[Ответить](#)



1. *Юрий:*

[16 октября 2017 в 10:47](#)

Спасибо 😊 У меня с самого начала такое же ощущение. Всё валится не сразу и наповал, а действительно, шаг за шагом.

[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