

Ravesli [Ravesli](#)


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


Урок №81. Нулевые указатели

 [Юрий](#) |

- [Уроки C++](#)

|

 Обновл. 12 Авг 2020 |

 53949

[1](#)  [9](#)

Как и в случае с обычными переменными, [указатели](#) не инициализируются при создании. Если значение не было присвоено, то указатель по умолчанию будет указывать на любой адрес, содержимым которого является мусор.

Оглавление:

1. [Нулевое значение и нулевые указатели](#)
2. [Разыменование нулевых указателей](#)
3. [Макрос NULL](#)
4. [Ключевое слово nullptr в C++11](#)
5. [Тип данных std::nullptr_t в C++11](#)

Нулевое значение и нулевые указатели

Помимо адресов памяти, есть еще одно значение, которое указатель может хранить: значение null.

Нулевое значение (или «*значение null*») — это специальное значение, которое означает, что указатель ни на что не указывает. Указатель, содержащий значение null, называется **нулевым указателем**.

В языке C++ мы можем присвоить указателю нулевое значение, инициализируя его/присваивая ему литерал 0:

```
1 int *ptr(0); // ptr теперь нулевой указатель
2
3 int *ptr1; // ptr1 не инициализирован
4 ptr1 = 0; // ptr1 теперь нулевой указатель
```

Поскольку значением нулевого указателя является ноль, то это можно использовать внутри **условного ветвления** для проверки того, является ли указатель нулевым или нет:

```
1 #include <iostream>
2
3 int main()
4 {
5     double *ptr(0);
6
7     if (ptr)
8         std::cout << "ptr is pointing to a double value.";
9     else
10        std::cout << "ptr is a null pointer.";
11
12    return 0;
13 }
```

Совет: Инициализируйте указатели нулевым значением, если не собираетесь присваивать им другие значения.

Разыменование нулевых указателей

Как мы уже знаем из предыдущего урока, разыменование указателей с мусором приведет к неожиданным результатам. С разыменованием нулевого указателя дела обстоят так же. В большинстве случаев вы получите сбой в программе.

В этом есть смысл, ведь разыменование указателя означает, что нужно «перейти к адресу, на который указывает указатель, и достать из этого адреса значение». Нулевой указатель не имеет адреса, поэтому и такой результат.

Макрос NULL

В языке Си (но не в C++) есть специальный макрос препроцессора с именем NULL, который определен как значение 0. Хотя он и не является частью языка C++, его использование достаточно распространено, и должно работать в каждом компиляторе C++:

```
1 int *ptr(NULL); // присваиваем адрес 0 указателю ptr
```

Однако, поскольку NULL является макросом препроцессора и, технически, не является частью C++, то его не рекомендуется использовать.

Ключевое слово nullptr в C++11

Обратите внимание, значение 0 не является типом указателя, и присваивание указателю значения 0 для обозначения того, что он является нулевым — немного противоречиво, вам не кажется? В редких

случаях, использование 0 в качестве аргумента-литерала может привести к проблемам, так как компилятор не сможет определить, используется ли нулевой указатель или целое число 0:

```
1 | doAnything(0); // является ли 0 аргументом-значением или аргументом-указателем? (компил.
```

Для решения этой проблемы в C++11 ввели новое **ключевое слово nullptr**, которое также является константой [r-value](#).

Начиная с C++11, при работе с нулевыми указателями, использование nullptr является более предпочтительным вариантом, нежели использование 0:

```
1 | int *ptr = nullptr; // примечание: ptr по-прежнему остается указателем типа int, просто
```

Язык C++ [неявно преобразует](#) nullptr в соответствующий тип указателя. Таким образом, в вышеприведенном примере, nullptr неявно преобразуется в указатель типа int, а затем значение nullptr присваивается ptr.

nullptr также может использоваться для вызова функции (в качестве аргумента-литерала):

```
1 | #include <iostream>
2 |
3 | void doAnything(int *ptr)
4 | {
5 |     if (ptr)
6 |         std::cout << "You passed in " << *ptr << '\n';
7 |     else
8 |         std::cout << "You passed in a null pointer\n";
9 | }
10 |
11 | int main()
12 | {
13 |     doAnything(nullptr); // теперь аргумент является точно нулевым указателем, а не це.
14 |
15 |     return 0;
16 | }
```

Совет: В C++11 используйте nullptr для инициализации нулевых указателей.

Тип данных std::nullptr_t в C++11

В C++11 добавили новый **тип данных std::nullptr_t**, который находится в [заголовочном файле](#) cstdint. std::nullptr_t может иметь только одно значение — nullptr! Хотя это может показаться немного глупым, но это полезно в одном случае. Если вам нужно написать функцию, которая принимает аргумент nullptr, то какой тип параметра нужно использовать? Правильно! std::nullptr_t. Например:

```
1 | #include <iostream>
2 | #include <cstdint> // для std::nullptr_t
3 |
4 | void doAnything(std::nullptr_t ptr)
```

```
5 | {  
6 |     std::cout << "in doAnything()\n";  
7 | }  
8 |  
9 | int main()  
10 | {  
11 |     doAnything(nullptr); // вызов функции doAnything() с аргументом типа std::nullptr_  
12 |  
13 |     return 0;  
14 | }
```

Вам, вероятно, никогда это не придется использовать, но знать об этом стоит (на всякий пожарный).

Оценить статью:

★★★★★ (253 оценок, среднее: 4,92 из 5)



[← Урок №80. Указатели](#)



[Урок №82. Указатели и массивы →](#)

Комментариев: 9



1. *Владимир:*
[23 декабря 2019 в 18:59](#)

с этими указателями....какие-то скороговорки для мозга..

[Ответить](#)



2. *somebox:*
[30 мая 2019 в 19:57](#)

Так и не понял, для чего нужен std::nullptr_t. Зачем его ввели.

[Ответить](#)



1. *Алексей:*
[31 мая 2019 в 01:17](#)

Судя по тому, что я узнал за 81 урок и какое мнение сложилось о C++, ответ на Ваш вопрос: "просто так, на всякий пожарный")")))))))

ps. 3 допустимых вида инициализации переменных, значит, вас не смущают?")))))))

Ответить



1. Юрий:

[6 июня 2019 в 14:56](#)

В этих уроках объясняется как база, так и нюансы, которые вы, скорее всего, не очень часто будете использовать на практике, но знать об этом стоит.

Ответить



2. C++11:

[3 сентября 2019 в 16:28](#)

Каждая переменная или константа должна быть определенного типа.

Следовательно, у nullptr (как константы, а не ключевого слова) тоже должен быть какой-то тип.

В MSDN есть пример, показывающий зачем ввели дополнительный тип:

Например, если для функции `func(std::pair<const char *, double>)` произвести вызов `func(std::make_pair(NULL, 3.14))`, возникнет ошибка времени компиляции. Макрос `NULL` разворачивается в `0`, поэтому вызов `std::make_pair(0, 3.14)` возвращает значение `std::pair<int, double>`, которое невозможно преобразовать к типу параметра `std::pair<const char *, double>` функции `func()`. Вызов `func(std::make_pair(nullptr, 3.14))` успешно компилируется, поскольку `std::make_pair(nullptr, 3.14)` возвращает значение `std::pair<std::nullptr_t, double>`, которое допускает преобразование в тип `std::pair<const char *, double>`.

>> <https://docs.microsoft.com/ru-ru/cpp/cpp/nullptr>

>> <https://msdn.microsoft.com/ru-ru/windows/desktop/jj651642>

Зачем еще оно нужно:

Приходите на собеседование, и дадут вам такой код 😊

```

1 //=====
2 #include <iostream>
3
4 void foo(const int)
5 {
6     std::cout << "foo(const int)" << std::endl;
7 }
8
9 void foo(const int*)
10 {
11     std::cout << "foo(const int*)" << std::endl;
12 }
```

```

13
14 int main()
15 {
16     foo(NULL);
17     foo(nullptr);
18     return 0;
19 }
20 //=====

```

Из <http://www.cyberforum.ru/cpp-beginners/thread638684.html#post3365236>

Ответ: <http://ideone.com/Av14Ae>

Ответить



3. *Борис:*

[10 мая 2020 в 20:53](#)

Зачем он нужен — понятно (см. конец). Непонятно — зачем кому-то писать функцию, принимающую этот nullptr как параметр? Он же не несёт никакую информацию в функцию.

Ответить



1. *Павел:*

[23 мая 2020 в 16:25](#)

Вроде я допёхал. Он нужен тупо для вызова функции без каких либо значений. Ну к примеру для проги выше нам надо объявит переменную, затем присвоить ей значение и только после этого засунуть её в функцию. Примерно так:

```

1 #include <iostream>
2 void foo(const int)
3 {
4     std::cout << "foo(const int)" << std::endl;
5 }
6
7 void foo(const int*)
8 {
9     std::cout << "foo(const int*)" << std::endl;
10 }
11
12 int main()
13 {
14     int i = 1;
15     int* ptr = &i;
16     foo(i);
17     foo(ptr);
18     return 0;
19 }

```

То есть + 2 строчки из которых 1 указатель, а второй переменная, а так мы тупо без каких либо объявлений пишем эти значения и функция выполняется. Как я понимаю используется в чём то типо кнопок.

[Ответить](#)



4. *Алексей:*

[27 сентября 2020 в 18:19](#)

Думаю, что это можно использовать с целью производительности для особого поведения при передаче параметра nullptr в метод класса, просто перегружаете метод, который принимает, например, строку и некий необязательный указатель.

Грубый пример:

```
1 void foo(std::string str,int* ptr)
2 {
3   doSomethingWithStr(str);
4   doSomethingWithPtr(ptr);
5 }
6
7 void foo(std::string,std::nullptr_t)
8 {
9   doSomethingWith(str);
10  //и тут второй метод не выполняется
11 }
```

Это позволит не выполнять лишних действий если указатель=nullptr.

p.s. Если это вам не понятно, то дочитайте эти уроки до конца и вернитесь сюда.

[Ответить](#)



3. *Веня:*

[8 октября 2018 в 16:47](#)

Можете добавить примеры с применением нулевых указателей, в том числе с передачей тех в функцию как аргумент?

[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