

Ravesli [Ravesli](#)


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

Урок №109. assert и static_assert

 [Юрий](#) |

- [Уроки C++](#)

|

 Обновл. 8 Сен 2020 |

 34859

[↑](#)  4

Использование [операторов условного ветвления](#) для обнаружения ложного предположения, а также вывода сообщения об ошибке и завершения выполнения программы является настолько распространенным решением возникающих проблем, что C++ решил это дело упростить. И упростил он его с помощью **assert**.

Оглавление:

1. [Стейтмент assert](#)
2. [NDEBUG](#)
3. [static_assert](#)

Стейтмент assert

Стейтмент assert (или «*оператор проверочного утверждения*») в языке C++ — это макрос препроцессора, который обрабатывает условное выражение во время выполнения. Если условное выражение истинно, то стейтмент assert ничего не делает. Если же оно ложное, то выводится сообщение об ошибке, и программа завершается. Это сообщение об ошибке содержит ложное условное выражение, а также имя файла с кодом и номером строки с assert. Таким образом, можно легко найти и идентифицировать проблему, что очень помогает при [отладке программ](#).

Сам assert реализован в [заголовочном файле](#) cassert и часто используется как для проверки корректности переданных параметров функции, так и для проверки возвращаемого значения функции:

```
1 #include <cassert> // для assert()
2
```

```
3 int getArrayValue(const std::array<int, 10> &array, int index)
4 {
5     // Предполагается, что значение index-а находится между 0 и 8
6     assert(index >= 0 && index <= 8); // это строка 6 в Program.cpp
7
8     return array[index];
9 }
```

Если в вышеприведенной программе вызвать `getArrayValue(array, -3);`, то программа выведет следующее сообщение:

Assertion failed: index >= 0 && index <=8, file C:\\VCProjects\\Program.cpp, line 6

Рекомендуется использовать стейтменты `assert`. Иногда утверждения `assert` бывают не очень описательными, например:

```
1 assert(found);
```

Если этот `assert` сработает, то получим:

Assertion failed: found, file C:\\VCProjects\\Program.cpp, line 42

Но что это нам сообщает? Очевидно, что что-то не было найдено, но что именно? Вам нужно будет самому пройти по коду, чтобы это определить.

К счастью, есть небольшой трюк, который можно использовать для исправления этой ситуации. Просто добавьте сообщение в качестве [строки C-style](#) вместе с логическим [оператором И](#):

```
1 assert(found && "Animal could not be found in database");
```

Как это работает? Строка C-style всегда принимает значение `true`. Поэтому, если `found` примет значение `false`, то `false && true = false`. Если же `found` примет значение `true`, то `true && true = true`. Таким образом, строка C-style вообще не влияет на обработку утверждения.

Однако, если `assert` сработает, то строка C-style будет включена в сообщение `assert`:

Assertion failed: found && "Animal could not be found in database", file C:\\VCProjects\\Program.cpp, line 42

Это даст дополнительное объяснение того, что пошло не так.

NDEBUG

Функция `assert()` тратит мало ресурсов на проверку условия. Кроме того, стейтменты `assert` (в идеале) никогда не должны встречаться в релизном коде (потому что ваш код к этому моменту уже должен быть [тщательно протестирован](#)). Следовательно, многие разработчики предпочитают использовать `assert` только в [конфигурации Debug](#). В языке C++ есть возможность отключить все `assert`-ы в релизном коде — использовать директиву `#define NDEBUG`:

```
1 #define NDEBUG
2
3 // Все стейтменты assert будут проигнорированы аж до самого конца этого файла
```

Некоторые IDE устанавливают NDEBUG по умолчанию, как часть параметров проекта в конфигурации Release.

Обратите внимание, **функция exit()** и assert (если он срабатывает) немедленно прекращают выполнение программы, без возможности выполнить дальнейшую любую очистку (например, закрыть файл или базу данных). Из-за этого их следует использовать аккуратно.

static_assert

В C++11 добавили еще один тип assert-а — **static_assert**. В отличие от assert, который срабатывает во время выполнения программы, static_assert срабатывает во время компиляции, вызывая ошибку компилятора, если условие не является истинным. Если условие ложное, то выводится диагностическое сообщение.

Вот пример использования static_assert для проверки размеров определенных типов данных:

```
1 static_assert(sizeof(long) == 8, "long must be 8 bytes");
2 static_assert(sizeof(int) == 4, "int must be 4 bytes");
3
4 int main()
5 {
6     return 0;
7 }
```

Результат на моем компьютере:

```
1>C:\ConsoleApplication1\main.cpp(19): error C2338: long must be 8 bytes
```

Поскольку static_assert обрабатывается компилятором, то условная часть static_assert также должна обрабатываться во время компиляции. Поскольку static_assert не обрабатывается во время выполнения программы, то стейтменты static_assert могут быть размещены в любом месте кода (даже в глобальном пространстве).

В C++11 диагностическое сообщение должно быть обязательно предоставлено в качестве второго параметра. В C++17 предоставление диагностического сообщения является необязательным.

Оценить статью:

★★★★★ (207 оценок, среднее: 4,93 из 5)



[← Урок №108. Обработка ошибок, cerr и exit\(\)](#)

[Урок №110. Аргументы командной строки](#)



Комментариев: 4



1. *Илья:*

[17 августа 2020 в 14:32](#)

```
1 #define NDEBUG // Все стейтменты assert будут проигнорированы аж до самого конца
2 #include <cassert> // для assert()
3 #include <array>
4
5 int getArrayValue(const std::array<int, 10>& array, int index)
6 {
7     // Предполагается, что значение index-а находится между 0 и 8
8     assert(index >= 0 && index <= 8 && "Animal could not be found in database");
9
10    return array[index];
11 }
12
13
14
15 int main()
16 {
17     std::array<int, 10>myarr{ 1,2,4,5,6,7 };
18     myarr = {0,1,2,3,4,5,6,7,8,9};
19     getArrayValue(myarr, 11); //индекс за пределами диапазона от 0 до 8 (по усл
20
21 }
```

[Ответить](#)



2. *Анастасия:*

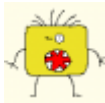
[19 июля 2019 в 13:32](#)

В примере

```
1 assert(found);
```

found — это какая-то булева переменная? Или спец слово, используемое вместе с assert в каких-то не описанных здесь случаях?

[Ответить](#)



1. *Евгений:*
[13 августа 2019 в 14:53](#)

булева переменная

[Ответить](#)



2. *Steindvart:*
[11 мая 2020 в 18:51](#)

Может быть булевой переменной. Или же каким-то объектом, значение которого можно неявно или явно преобразовать в true или false.

[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