

🏠 (<http://cppstudio.com>)

/ Стандартные заголовочные файлы из Си в C++ (<http://cppstudio.com/cat/309/>)

/ Заголовочный файл cassert (assert.h) (<http://cppstudio.com/cat/309/312/>)

/ assert: функция сообщения об ошибке в C++

assert: функция сообщения об ошибке в C++



Оценка: 5,00 (голосов: 3)

Чтобы проголосовать, вы должны зарегистрироваться.

Прототип функции assert:

```
1 void assert(int expression);
```

Заголовочный файл:

Название	Язык
assert.h	C
cassert	C++

Описание

Функция `assert` оценивает выражение, которое передается ей в качестве аргумента, через параметр `expression`. Если аргумент-выражение этого макроса в функциональной форме равно нулю (т.е. выражение ложно), сообщение записывается на стандартное устройство вывода ошибок и вызывается функция `abort` (`/spravochnik/standartnye-zagolovochnye-fajly-iz-si-v-s/zagolovochnyj-fajl-cstdlib-stdlib-h/funkciya-abort/`), работа программы прекращается.

Содержание сообщения об ошибке зависит от конкретной реализации компилятора, но любое сообщение должно состоять из: выражения, которое `assert` оценивает, имя файла с ошибкой и номер строки, где произошла ошибка. Обычный формат сообщения об ошибке :

filename: line number: expression: assertion failed:

Функция `assert` не будет доступна, если в момент включения `<cassert>` (`/spravochnik/standartnye-zagolovochnye-fajly-iz-si-v-s/zagolovochnyj-fajl-cassert-assert-h/`) макрос с именем `NDEBUG` уже будет определен. То есть, при отладке программы, мы можем неограниченное количество раз вызывать `assert`. После отладки программы, не будет надобности в вызванных нами функциях `assert`. Вручную выискивать каждую функцию – неблагодарный труд. В C++ определен макрос `NDEBUG`, который отключит все `assert` функции за нас. Наша задача – просто включить эту строку в код программы:

```
1 #define NDEBUG // в начале файла исходного кода, перед включением заголовочного
2 <cassert>
```

Параметры:

- **expression**

Выражение для оценки. Если логическое выражение в параметре `expression` равно 0, функция `assert` немедленно завершает программу.

Возвращаемое значение

нет

Пример: исходный код программы

```
1 // пример использования функции assert
2 #include <iostream> // для оператора cout
3 #include <cassert> // для функции assert
4
5 void print_adds(int* value)
6 {
7     assert(value != NULL);
8     std::cout << "Адрес значения в памяти = "
9               << value << std::endl;
10 }
11
12 int main()
13 {
14     int a = 10;
15     int *ptr1 = &a; // указатель на переменную a
16     int *ptr2 = NULL; // нулевой указатель
17
18     print_adds(ptr1); // вызов функции с ненулевым указателем
19     print_adds(ptr2); // вызов функции с нулевым указателем
20
21     return 0;
22 }
```


Пример работы программы


В этом примере, функция `assert` используется, чтобы прервать выполнение программы, если функции `print_adds` в качестве аргумента передаётся нулевой указатель. В программе, первый вызов функции `print_adds` завершается успешно. Второй вызов, в качестве аргумента, принимает нулевой указатель `ptr2`. И как раз в этот момент срабатывает функция `assert`, она оценивает выражение, в данном случае нулевой указатель. Нулевой указатель не указывает ни на какой блок памяти, поэтому `assert` сигнализирует об ошибке и немедленно завершает работу программы.


CppStudio.com


Адрес значения в памяти = 0x7fff5900f8ac

er: ../er/main.cpp:7: void print_adds(int*): Assertion `value != __null' failed.

 Обсудить на форуме (/topics/)

 Автор: admin (/forums/users/admin/)

 Дата: 05.09.2012

 Поделиться:

Похожие статьи:

1. Операции инкремента и декремента в C++ (<http://cppstudio.com/post/282/>)
2. Ссылки в C++ (<http://cppstudio.com/post/429/>)
3. Форматированный ввод/вывод в C++ (<http://cppstudio.com/post/319/>)
4. Операции присваивания в C++ (<http://cppstudio.com/post/279/>)
5. Наследование классов (<http://cppstudio.com/post/10103/>)







Комментарии



Оставить комментарий

Вы должны войти (<http://cppstudio.com/wp-login.php?>

[redirect_to=http%3A%2F%2Fcppstudio.com%2Fpost%2F946%2F](http://cppstudio.com/wp-login.php?redirect_to=http%3A%2F%2Fcppstudio.com%2Fpost%2F946%2F)), чтобы оставить комментарий.

Translation

-  (/post/946/)Русский (/post/946/)
-  (/uk/post/946/)Українська (/uk/post/946/)
-  (/en/post/946/)English (/en/post/946/)
-  (/de/post/946/)Deutsch (/de/post/946/)
-  (/be/post/946/)Беларуская (/be/post/946/)
-  (/kk/post/946/)Қазақ тілі (/kk/post/946/)

 (/uz/post/946/) O'zbek tili (/uz/post/946/)
 (/tr/post/946/) Türkçe (/tr/post/946/)

Новое

➤ Особенности Qt: слоты и сигналы, описание QObject и QApplication, виды окон и т.д.
(<http://cppstudio.com/post/11167/>)

➤ Первая программа на Qt:
(<http://cppstudio.com/post/11127/>)

➤ Введение – графическая библиотека Qt
(<http://cppstudio.com/post/11097/>)

➤ Наследование классов
(<http://cppstudio.com/post/10103/>)

➤ Перегрузка операторов в C++ (часть 2)
(<http://cppstudio.com/post/10058/>)

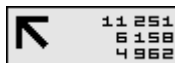
Популярное

Sorry. No data so far.

© 2022 CppStudio – Программирование для начинающих на C++



(<https://plus.google.com/u/0/106109650739084338784>)



(<http://www.liveinternet.ru/click>)



(<http://orphus.ru>)