What are assertions? and why would you use them?

Asked 16 years, 1 month ago Modified 16 years, 1 month ago

Viewed 12k times

15

How are assertions done in c++? Example code is appreciated.

c++ assert

Share

Improve this question

Follow

edited Oct 31, 2008 at 11:46

Omar Kooheji

55.6k ● 71 ● 186 ● 243

asked Oct 31, 2008 at 11:28

yesraaj

47.8k ● 70 ● 198 ● 252

8 Answers

Sorted by: Highest score (default)

36

Asserts are a way of explicitly checking the assumptions that your code makes, which helps you track down lots of bugs by narrowing down what the possible problems could be. They are typically only evaluated in a special "debug" build of your application, so they won't slow down the final release version.

Let's say you wrote a function that took a pointer as an argument. There's a good chance that your code will assume that the pointer is non-NULL, so why not explicitly check that with an assertion? Here's how:

```
#include <assert.h>

void function(int* pointer_arg)
{
    assert(pointer_arg != NULL);

    ...
}
```

An important thing to note is that the expressions you assert must never have side effects, since they won't be present in the release build. So never do something like this:

```
assert(a++ == 5);
```

Some people also like to add little messages into their assertions to help give them meaning. Since a string always evaulates to true, you could write this:

```
assert((a == 5) && "a has the wrong value!!");
```

Share Improve this answer Follow

answered Oct 31, 2008 at 11:34

andygeers
6,966 ● 9 ● 52 ● 63

I haven't seen the anding with a string before. Real useful!
– David Holm Oct 31, 2008 at 11:36

That sort of trick (or a similar variation) is used all the time in Perl-land, namely, using the example given: a == 5 || die("a has the wrong value"); Hurrah for loose boolean typing.
– Matthew Scharley Oct 31, 2008 at 11:54

Another trick along the lines of this string is when validating values in release code, but asserting in debug code. For example, use an if(x>10) check first, and then if the erroneous condition passes, just assert(!"x out of bounds") and then print to a logfile, etc. – Marcin Oct 31, 2008 at 13:33

1 @David: Usually an assert statement will print the failed condition when it fails, along with file/line info. So what it gives you is a human-readable message along with the actual. Equivalently, you could write your own assert with two parameters: a condition and a message. – Steve Jessop Oct 31, 2008 at 14:54

1 David, your IDE will typically display the expression that failed as part of the error message it displays to you, so if your expression contains a string then that'll pop up
– andygeers Nov 3, 2008 at 12:42

6

Assertion are boolean expressions which should typically always be true.

They are used to ensure what you expected is also what happens.

```
void some_function(int age)
{
    assert(age > 0);
}
```

You wrote the function to deal with ages, you also 'know' for sure you're always passing sensible arguments, then you use an assert. It's like saying "I know this can never go wrong, but if it does, I want to know", because, well, everyone makes mistakes.

So it's not to check for sensible user input, if there are scenario's where something can go wrong, don't use an assert. Do real checks and deal with the errors.

Asserts are typically only for debug builds, so don't put code with side effects in asserts.

Share Improve this answer Follow

edited Oct 31, 2008 at 11:38

answered Oct 31, 2008 at 11:33

Pieter

17.7k ● 8 ● 53 ● 90

5

Assertions are used to verify design assumptions, usually in terms of input parameters and return results. For example

```
// Given customer and product details for a sale, gene

Invoice ProcessOrder(Customer Cust,Product Prod)
{
  assert(IsValid(Cust));
  assert(IsValid(Prod);
'
'
'
  assert(IsValid(RetInvoice))
   return(RetInvoice);

}
```

The assert statements aren't required for the code to run, but they check the validity of the input and output. If the input is invalid, there is a bug in the calling function. If the input is valid and output is invalid, there is a bug in this code. See design by contract for more details of this use of asserts.

Edit: As pointed out in other posts, the default implementation of assert is not included in the release run-time. A common practice that many would use, including myself, is to replace it with a version that is included in the release build, but is only called in a diagnostics mode. This enables proper regression testing on release builds with full assertion checking. My version is as follows;

```
extern  void _my_assert(void *, void *, unsigned);

#define myassert(exp)                                  \
{                                                      \
    if (InDiagnostics)                                 \
        if ( !(exp) )                                  \
            _my_assert(#exp, __FILE__, __LINE__);      \
}                                                      \
```

There is a small runtime overhead in this technique, but it makes tracking any bugs that have made it into the field much easier.

Share Improve this answer Follow

edited Oct 31, 2008 at 11:55

answered Oct 31, 2008 at 11:38

SmacL
22.9k ● 15 ● 99 ● 151

3

Use assertions to check for "can't happen" situations.

Typical usage: check against invalid/impossible arguments at the top of a function.

Seldom seen, but still useful: loop invariants and postconditions.

Share Improve this answer Follow

answered Oct 31, 2008 at 12:07

mfx
7,358 ● 1 ● 29 ● 29

2

Assertions are statements allowing you to test any assumptions you might have in your program. This is especially useful to document your program logic (preconditions and postconditions). Assertions that fail usually throw runtime errors, and are signs that something is VERY wrong with your program - your assertion failed because something you assumed to be true was not. The usual reasons are: there is a flaw in your function's logic, or the caller of your function passed you bad data.

Share Improve this answer

Follow

answered Oct 31, 2008 at 11:34

Maxam

4,053 ● 2 ● 26 ● 25

1

An assertion is something you add to your program that causes the program to stop immediately if a condition is met, and display an error message. You generally use them for things which you believe can never happen in your code.

Share Improve this answer

Follow

answered Oct 31, 2008 at 11:34

Dave Markle

97.6k ● 20 ● 151 ● 171

1

This doesn't address the *assert* facility which has come down to us from early C days, but you should also be aware of [Boost StaticAssert]() functionality, in the event that your projects can use Boost.

The standard C/C++ assert works during runtime. The Boost StaticAssert facility enables you to make some classes of assertions at compile time, catching logic errors and the like even earlier.

Share Improve this answer

Follow

answered Oct 31, 2008 at 16:24

Don Wakefield
8,812 ● 3 ● 38 ● 54

0

Here is a definition of what an assertion is and here is some sample code. In a nutshell an assertion is a way for a developer to test his (or her) assumptions about the state of the code at any given point. For example, if you were doing the following code:

```
mypointer->myfunct();
```

You probably want to assert that mypointer is not NULL because that's your assumption--that mypointer will never be NULL before the call.

Share Improve this answer Follow

answered Oct 31, 2008 at 11:34
Onorio Catenacci
15.3k ● 16 ● 84 ● 134