

Re: unit testing C++ code from perl

Source: <http://coding.derkeiler.com/Archive/General/comp.programming/2006-07/msg00146.html>

- *From:* "Nick Keighley" <nick_keighley_nospam@xxxxxxxxxxxxx>
 - *Date:* 7 Jul 2006 05:03:52 -0700
-

I've glued a couple of Philip's post together to make the reply all in one chunk. If it got messed up in the process I apologise.

Philip wrote:

Nick Keighley wrote:

do you have to read a book to understand any of this?

Unfortunately yes.

Beck's "Test Driven Development"? I was meaning to read it.

Are we supposed to just **know** what TEST or TEST_FIXTURE does?

Nope. The authors of a **Unit* should document such things. UnitTest++ has 1,300 downloads and no tutorial on their little home page, so these probably represent people who already use CppUnit or CppUnitLite.

I'll take a look at CppUnit*

To write a general purpose library intended for a wide audience with[out] **any** examples of usage or explanation of what the various parts do, just seems bizzare to me!

Re: unit testing C++ code from perl

Okay, there is the src/test folder. Read its Main.cpp, that's how you launch the test run. Read any TEST() macro instance, that's how you write a test case. So here's "Hello World":

```
TEST(whatever)
{
CHECK(false);
}
```

so macros arn't evil :-)

Noel and Charles might not feel motivated to spend their time writing a manual that would be just the same as any other *Unit manual, but with a few edits to match their own system. They are all generally the same.

a pointer to the other manual? I don't know why I'm arguing with you.
it's
not your package!

Yes, I could puzzle it all out. But the temptation is to use the same time to write my own.

And thus the problem. I wrote my itch-list because they formerly used my rig, which had all these features that they neglected to understand or use. So if you wrote your own, you could muddle through too, but you would miss subtle techniques and conveniences that fully polished test rigs have.

but hacking code is such *fun*!

what! Its *good* to have no comments?! Not even a one liner saying what it's *for*?

Ahem.

```
void ReportAssert(char const* description, char const* filename, int const
lineNumber)
```

Ookay. We have a function called ReportAssert, and it takes a description. That's probably the complaint string for an error. "The frob doesn't fit its Wibble socket", or whatever.

Re: unit testing C++ code from perl

Then we have a filename and lineNumber. They are probably just the same as the filenames and lineNumbers we have seen ever since we were little kiddie-programmers.

Oh, and we didn't get creative with the names, like fileN or lNumber. Those would have required comments. Like fileN // this is the filename, and lNumber // this is the line number.

ok. I hate comments like that. I was arguing each *class* should have 1..few lines saying what it's for. It's not always easy to tell what is for use of a user of a package and what is purely internal. I'm a hater of unnecessary and unhelpful comments, but *no* comments!!

I just looked at a file I created yesterday. It started with a one line title "a foo class for reprobating", Three lines explaining in more detail what it did and then an example of usage. I wasn't trying to flood with comments it just seemed natural.

Let's try another one.

```
template<typename ValueType>
void FormatToStream(MemoryOutputStream& stream, char const* format, ValueType const&
value)
{
char txt[32];
std::sprintf(txt, format, value);
stream << txt;
}
```

Hmm. That seems to format any primitive value as a string, then insert it into a string. It probably wraps the "%2.0i" format flags that sprintf() is famous for. And we are not production code, so we don't need snprintf() or something more over-flow safe. But any comment I wrote describing everything going on would be bigger than the function!

If you ask "what about the hard, complex methods?" that's the point – there are none. All the functions are short, cheap, narrow, and obvious like these. (There are those who claim unit testing helps generate such code!)

Treat all comments as a failed opportunity to improve the code's clarity.

Re: unit testing C++ code from perl

well we might have to agree to differ on this one...

BTW I wrote a tutorial for UnitTest++, here:

<http://wiki.orbzone.org/TestFirstCiao>

excellent, I'll give it a look.

Disregard the CIAO stuff. It would need ... lots of comments!

--

Nick Keighley

.