Subject: how to input an int into std::string? and an exception question
Posted by bonami on Thu, 03 Jan 2008 03:10:20 GMT
View Forum Message <> Reply to Message

iota is good, but that is for char *.
i used std::stringstream, which accepts int well, but when i convered it into std::string and threw it, my program crashed.
std::stringstream msg;
msg << "failed to bind to " << localSocket->ip() << ":" << localSocket->port()/* this is an int */;
throw std::string(msg.str()); //or simply throw msg.str();

...
catch (std::string err) ...
& i want standard C++ method, not String or alike.
another question is why my functions cannot throw an exception by default

```
child() {... throw std::string("error"); }
  dad() {... child(); ...}/*crashes*/
  mum() {... try { child(); }/*this works*/
    catch (std::string err) { throw err; }}
grand() {... try { dad()/* or mum()*/; }
  catch (std::string err) {....}}
```

today i just ran mum() again, and it crashed, too... (since i had been Debuggin, i could locate the last sentence is throw)

Subject: Re: how to input an int into std::string?
Posted by tvanriper on Thu, 03 Jan 2008 12:18:04 GMT
View Forum Message <> Reply to Message

When you throw an exception, if you have no way to catch that exception, it will crash your program.

That's actually kind of the point behind exceptions.

If you never want your program to crash due to an exception, but want it to die gracefully, you have to provide some way to handle all exceptions. Sometimes, this is accomplished in main with something like:

```
int main( int argc, char** argv )
{
  try
  {
    millions_of_functions();
    even_more_functions();
    maybe_yet_more_functions();
  }
  catch( ... )
  {
    // Some error handling routine here... like maybe:
    std::cerr << "Something catastrophic happened." << std::endl;
  }
}
```

So, based on what I'm reading in your message, I do not see the nature of your problem. It looks to me that your program should be crashing, since you're asking it to do so.

Generally, the folks experienced with C++ that I've talked to seem to agree that exceptions should be used in very rare cases, and ought to be avoided if possible. You might only use exceptions for truly exceptional situations, where you can see that the system will have a serious problem that destabilizes memory or registers, and want to clearly indicate where the problem occurred so you can fix it more easily.

If you can, you should try to indicate most error conditions through other means, perhaps through an error state, return code, or something you change in a parameter.

Subject: Re: how to input an int into std::string?
Posted by mr_ped on Thu, 03 Jan 2008 13:39:45 GMT
View Forum Message <> Reply to Message

bonami: the mum should crash too, as you do another throw in the catch clause, which is not catch anywhere.

tvanriper wrote on Thu, 03 January 2008 13:18Generally, the folks experienced with C++ that I've talked to seem to agree that exceptions should be used in very rare cases, and ought to be avoided if possible. You might only use exceptions for truly exceptional situations, where you can see that the system will have a serious problem that destabilizes memory or registers, and want to clearly indicate where the problem occurred so you can fix it more easily.

If you can, you should try to indicate most error conditions through other means, perhaps through an error state, return code, or something you change in a parameter.

I think it's about how the source code looks.
If the exception version looks shorter and cleaner and is easier to read and understand, you should use it, even if it is not serious problem state what is handled there.
If the "if" version with error return codes looks good enough, the exception version would likely look more complex.

In case of performance critical code you should benchmark final version with profiler, and rewrite only the most critical code parts, with proper comments why you are using more ugly (but faster) code in that space and how it improves performance.

And you should try to create some policy for your project where/why to use exceptions, so they are used in similar context trough whole source (my biggest problem with them, often two different components from me are written in different way, one does use exceptions a lot for error states, other does use if/return extensively, especially if rewrite with different error handling would lead to similar source code and there's no clear winner).

But I don't think there's any major reason to avoid exceptions for all costs. Especially if they will make the source code easier to understand and easier to maintain/change later.

Subject: Re: how to input an int into std::string?
Posted by tvanriper on Thu, 03 Jan 2008 17:58:06 GMT

View Forum Message <> Reply to Message

Yeah, ultimately, it's about whatever reads the best.

I probably spoke mostly to folks who find it less legible to have the execution of the code shift dramatically due to an exception.

But, ultimately, it's about whatever makes sense for your particular context.

Subject: Re: how to input an int into std::string?
Posted by bonami on Fri, 04 Jan 2008 02:46:27 GMT

View Forum Message <> Reply to Message

Mr.Ped,
   mum() is caught in grand() as below,
     grand() { ... try { mum(); } catch (std::string err) {...}}
  that is why it should not crash.
  and i suppose dad() should not crash either, because, though dad() does not catch the exception of child(), it should throw the exception by default, which will be caught in grand().