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.