Catching exceptions from a constructor's initializer list

Asked 16 years, 2 months ago Modified 7 years, 2 months ago Viewed 32k times

71

Here's a curious one. I have a class A. It has an item of class B, which I want to initialize in the constructor of A using an initializer list, like so:

```
class A {
    public:
    A(const B& b): mB(b) { };

    private:
    B mB;
};
```

Is there a way to catch exceptions that might be thrown by mB's copy-constructor while still using the initializer list method? Or would I have to initialize mB within the constructor's braces in order to have a try/catch?

c++ exception

Share Improve this question Follow

asked Oct 1, 2008 at 22:57

Head Geek

39.8k ● 22 ● 79 ● 89

5 Answers

Sorted by: Highest score (default)

114

Have a read of http://weseetips.wordpress.com/tag/exception-from-constructor-initializer-list/)

Edit: After more digging, these are called "Function try blocks".

I confess I didn't know this either until I went looking. You learn something every day! I don't know if this is an indictment of how little I get to use C++ these days, my lack of C++ knowledge, or the often Byzantine features that litter the language. Ah well - I still like it :)

To ensure people don't have to jump to another site, the syntax of a function try block for constructors turns out to be:

```
C::C()
try : init1(), ..., initn()
{
  // Constructor
}
catch(...)
{
  // Handle exception
}
```

Share

Improve this answer

Follow

edited Oct 16, 2017 at 9:26

answered Oct 1, 2008 at 23:03

Adam Wright

49.3k ● 12 ● 133 ● 152

2 Ugh. I'm not surprised that there's some way to do that, but it sure is a great example of why I hate C++ initializer syntax... – Mark Bessey Oct 1, 2008 at 23:07

26 NOTE: you can't handle the exception when using function try blocks on constructors. even if your catch(...) block does not re-throw, the exception still escapes to the caller. – Aaron Oct 1, 2008 at 23:37

11 Herb Sutter's "Guru of the Week" articles also has a nice discussion on function-try-blocks: gotw.ca/gotw/066.htm – Void - Othman Sep 23, 2009 at 20:12

2 @Aaron That is true, but still worth mentioning, that you can re-throw a different exception, which is usefull anyway. Say that some generic exceptio nis thrown, but you woudl rather throw your own type, which the caller is already able to handle. – Petr Apr 25, 2019 at 10:57

2 For anyone curious what @Aaron is referring to, essentially if an exception is thrown when initializing the class members from the initialization list of the constructor, then this means the construction failed and therefore cannot continue. There is no valid class object as a result. The C++ specification requires that the same exception be thrown again or a new exception is thrown regardless if you catch the exception with the constructor's function try block. You can learn more here: stackoverflow.com/questions/63158522/… . – Dannon Sep 28, 2022 at 4:24

20

It's not particularly pretty:

```
A::A(const B& b) try : mB(b)
{
    // constructor stuff
}
catch (/* exception type */)
{
    // handle the exception
}
```

Share Improve this answer Follow

answered Oct 1, 2008 at 23:05

Michael Burr
340k ● 52 ● 548 ● 769

4

I know it has been awhile since this discussion started. But that try-and-catch construct mentioned by Adam is part of the C++ standard and is supported by Microsoft VC++ and GNU C++. Here is the program that works. By the way the the catch generates automatically another exception to signal about the constructor failure.

```
#include <iostream>
#include <exception>
#include <string>

using namespace std;

class my_exception: public exception
{
  string message;
public:
  my_exception(const char* message1)
  {
    message = message1;
  }

  virtual const char* what() const throw()
  {
     cout << message << endl;
     return message.c_str();
  }

  virtual ~my_exception() throw() {};
};

class E
{
public:
    E(const char* message) { throw my_exception(message);}
};

class A
{
    E p;
public:
    A()
  try :p("E failure")
    {
            cout << "A constructor" << endl;
    }
  catch (const exception& ex)
    {
        cout << "Inside A. Constructor failure: " << ex.what() << endl;
    }
};
```

```
int main()
{
    try
    {
        A z;
    }
    catch (const exception& ex)
    {
        cout << "In main. Constructor failure: " << ex.what() << endl;
    }
    return 0;
}
```

Share Improve this answer Follow

answered Jul 13, 2011 at 8:46

Mikhail Semenov
963 ● 8 ● 8

1

You could work with lazy initialization, though, that is hold a unique_ptr to Reader in MyClass and create it with new. That way, you do not even need the flag has_reader but you can just see if your unique_ptr is initial or not.

```
#include <iostream>
#include <memory>
using namespace std;

class MyOtherClass
{
public:
    MyOtherClass()
    {
        throw std::runtime_error("not working");
    }
};

class MyClass
{
public:
    typedef std::unique_ptr<MyOtherClass> MyOtherClassPtr;

    MyClass()
    {
        try
        {
            other = std::make_unique<MyOtherClass>();
        }
        catch(...)
        {
            cout << "initialization failed." << endl;
        }

        cout << "other is initialized: " << (other ? "yes" : "no");
    }

private:
    std::unique_ptr<MyOtherClass> other;
};
```

```
int main()
{
    MyClass c;

    return 0;
}
```

Of course, there are also solutions without using exceptions at all but I assumed that this is a prerequisite in your setting.

Share Improve this answer Follow

answered Feb 8, 2016 at 8:51

IceFire

4,119 ● 2 ● 33 ● 56

-1

I don't see how you'd do that with initializer-list syntax, but I'm also a bit sceptical that you'll be able to do anything useful by catching the exception in your constructor. It depends on the design of the classes, obviously, but in what case are you going to fail to create "mB", and still have a useful "A" object?

You might as well let the exception percolate up, and handle it wherever the constructor for A is being called.

Share

Improve this answer

Follow

edited Oct 1, 2008 at 23:07

answered Oct 1, 2008 at 23:02

Mark Bessey

19.8k ● 4 ● 49 ● 69

1 Perhaps this should have been a comment. Anyway, there are still some useful stuff you can do there like diagnose the problem, log it, and throw a more informative exception than the one you caught. You are correct in that you can't do anything with A or its members.
– patatahooligan Aug 21, 2019 at 8:53