EE 200 Lecture 16: Exceptions

Steven Bell

2 November 2023

Tufts UNIVERSITY

UnluckyNumber demo

Throw an unnamed temporary

Throw an unnamed temporary

If you allocate memory on the heap, all possible exception handlers will have to know to free it.

Catch by reference

Catch by reference

If you catch by value, the exception will have to be copied.
This is a) wasteful, and b) might cause another exception.

Declare handlers from most to least specific

Declare handlers from most to least specific

The code will use the first one that matches, so any specific ones have to come first, or else they will be "hidden" by the generic handlers.

Inherit from std::exception

Inherit from std::exception

This makes it possible to reasonably catch everything:

```
try{
    somethingDangerous();
} catch (std::exception e){
    cout << e.what() << endl;
}
```

Destructors should never throw exceptions

Destructors should never throw exceptions

The destructor may get called while handling another exception, and throwing a second exception will cause the program to crash immediately.

Levels of guarantees

None: carnage and chaos may result

Weak: at least you didn't break anything

Strong: nothing is broken and the object wasn't modified

No-throw: Strong, plus we won't ever throw an exception

Classwork 13 is in your Github repo

Upgrade your Array class to:

1) Throw a BoundsException when the user tries to access elements out of bounds
2) Provide the best exception guarantees you can.
   We will make `malloc` and `new` fail, and you should handle this.

   "While such code would get by fine in an introductory course..."