6: Exceptions

CSCI 4526 / 6626 Spring 2016

1 Goals

- To define and use exception handlers instead of using fatal().
- To decide what exceptions you need and where they should be caught.
- To define polymorphic exceptions and use their virtual functions skillfully.

2 Error Handling

Errors are detected in your program in a variety of places. Make a list of them. How many of your errors are serious enough to abort execution? Your list should fall into two general categories: stream errors and game-logic errors. Define two polymorphic exception classes for these error-types. Let each exception base class define a virtual print function and also let it contain as much as possible of the code that is common to all variants.

For each kind of error, decide the best way to handle it. Some may abort. Others will prompt the user to re-enter information. Make your English clear and succinct. In several of the error situations, you will be able to continue execution after handling the error.

For each kind of error, decide where that error can best be handled and install a try block in that place, followed by handlers for the relevant exceptions. In your try block, include all of the lines of code that work together with the one that might throw an exception.

If you wish, put a catch-all handler in main. This is not particularly useful, since the only action possible is to abort, but it will give you practice writing try blocks and catch-all blocks.

3 Changes to your Classes

In the State class, get rid of the remove() function. We will not need it.

Find every call on fatal() and replace it by throwing an exception. Then look through your code. Every user input should be validated exactly once. If you are testing the same thing for the same kind of validity twice, remove one of those tests.

Demonstrate that your program still works and that each of the exceptions that is under your control has been tested.