Exception Handling

Chapter 16

What is an Exception

- An exception is some type of problem that occurs during the execution of a program
  - An example is a divide by zero error
- With exception handling, a program can continue executing as if no error occurred

Types of Errors

- Events during program execution can be:
  - Synchronous - follows normal flow of control executing one statement and then another
  - Asynchronous - events that can operate in parallel or independent of the normal flow of a program
    - Examples are disk I/O, mouse click, keyboard press
- Exception handling is for synchronous errors

Error Checking

- Our focus in this class has not been writing code with good error checking due to the fact that this would probably add hours of time to each coding project
- When coding a real world project, it is important to incorporate all error checking into your solution immediately

Errors

- With exception handling, there is very little performance penalty if there is no error
- If you have program logic where a statement problem occurs at least 30% of the time (Deitel), then use inline error checks as opposed to exception handling

Exception Handling

- Exception Handler: code that processes a certain kind of exception
- It is the case that if a program throws a certain kind of exception,
  - if there is an exception handler for that particular exception, the exception is handled;
  - otherwise, the exception is uncaught and can cause any number of problems.

Try, Catch, and Throw

- C++ provides the following:
  - **try** block - defines a block of code where an exception might occur
  - **catch** block - defines the type of exception(s) that might occur
  - **throw** point - is the point where an exception might occur

Try Block

- If an exception occurs in a try block then
  - the try block of code terminates immediately and
  - the first catch handler is checked to see if it can handle the exception
  - If not, proceed to the next catch handler, and so on