Error Handling
Methods of Error Handling

- Some different ways that will be covered further:
  - Flags and return codes
  - Assertions
  - Exceptions
- Some other ways that won't be covered more:
  - setjmp and longjmp - A predecessor to exceptions (Don't use this!)
  - Signals
Flags and return codes

• A function might return a special value or set a global variable (a flag) to indicate an error

• The math functions which set `errno` are an example of using a flag:

```cpp
double x = -5;
errno = 0; // Clear any existing error flag

double d = sqrt(x);

if( errno == EDOM )
    cout << "Imaginary numbers! Ahhh!" << endl;
```
Assertions

- An **assertion** checks a boolean expression
  - If the result is true, nothing happens
  - If the result is false, the program is terminated immediately and a message is printed to stdout

```c
#include <cassert>  // or <assert.h>
...
assert( studentCount > 0 );
average = sumOfScored / studentCount;
```

- If studentCount < 0, the program stops with this message:
  ```
```
Assertions cont.

- Assertions can be turned off with the preprocessor directive `#define NDEBUG` before `assert.h` is included:

  ```
  #define NDEBUG
  #include <assert.h>
  ...
  ```

- If `NDEBUG` is defined before `assert.h` is included, the asserts will have no effect on the program's execution.
Exceptions

• Exception handling uses **try** and **catch** blocks.
  • An exception that is thrown in the **try** block is caught in a **catch** block.
  • A **catch** block can execute any statements.
• C++ exceptions can be any data type.
  • The exception is caught by a **catch** block for the thrown data type.
  • If no **catch** block is found for the thrown data type, the program will abort.
Example

```c
float average( int count, float total ) {
    if( count < 1 )
        throw "Count is too small";
    return total / count;
}
...
try {
    float a = average( 5, 234.8 );
}
catch( char * s ) {
    cout << s << endl;
    exit( 1 );
}
```
Exceptions cont.

- ... can be used to catch anything:

```java
try {
    // Could throw ANYTHING
} 
catch( ... ) {
    // Can catch anything
}
```

- The order of `catch` blocks matters!
  - Each `catch` is checked in order.
  - The ... must be the last `catch` for a `try` block.
Exceptions cont.

- C++ defines a few exceptions
- These are defined in the `<exception>` header

```cpp
#include <exception>
...

float * data;
try {
    data = new float[32768];
}
catch( std::bad_alloc ) {
    cerr << "Couldn't allocate array" << endl;
    exit( 1 );
}
```