601.220 Intermediate Programming

C++ Intro to OO

C++ class - an overview

- We use classes to define new data types
- These are like structs, but include functions that operate directly on the fields
- Classes add protection levels for the fields and functions (e.g., private) to provide data hiding and encapsulation - good object-oriented principles
- Special functions called *constructors* are used to initialize class objects (also called *instances* - variables declared to be of a class type)
- Special fuctions called *destructors* are used to perform clean-up operations just before the lifetime of a class instance ends
- We can use inheritance to define a class by extending an existing class

C++ I/O refresher

`iostream` is the main C++ library for input and output

```
#include <iostream>

using std::cin;   // default input stream
using std::cout;  // default output stream
using std::endl;  // end of line, flushes buffer
```

also

```
using std::cerr;  // default error output stream
```

<< is the stream insertion operator; used for output

>> is the stream extraction operator; used for input

C++ File I/O

- In C, printf wrote to stdout and scanf read from stdin
    - fprintf and fscanf were their counterparts for files
- In C++, we have std::cout and std::cin
    - std::ofstream and std::ifstream are their counterparts for files
    - These are defined in the file-stream header
        - `#include <fstream>`
    - and define classes:
        - ofstream: for writing to a file (inherits from ostream)
        - ifstream: for reading from a file (inherits from istream)
        - fstream: for reading and writing to/from a file (inherits from ostream and istream)

C++ stream class hierarchy

Inheritance: class A inherits from class B if every class A object "is-a" class B object also.

```
                 ios
               /     \
       istream        ostream
       /     \        /     \
ifstream     iostream     ofstream
              /     \
         fstream   stringstream
```

C++ I/O class relationships

- istream and ostream are both derived from ios
- iostream inherits from both istream and ostream
    - multiple inheritance is allowed in C++
- stream extraction operator (>>) defined for all istreams
- stream insertion operator (<<) defined for all ostreams
- fstream and stringstream are both derived from iostream
    - can use both >> and << on them for input or output

C++ ofstream usage

```
// io1.cpp:
#include <iostream>
#include <fstream>
int main(){
        std::ofstream ofile( "hello.txt" );
        if (!ofile.is_open()) {
             return 1;
        }
        ofile << "Hello, World!" << std::endl;
        return 0;
}
```

```
$ g++ -c io1.cpp -std=c++11 -pedantic -Wall -Wextra
$ g++ -o io1 io1.o
$ ./io1
$ cat hello.txt
Hello, World!
```

C++ File Output (std::ofstream)

- ofstream has a constructor taking a string specifying the filename
    - Calling the constructor with a filename string is the same as calling fopen with the filename using a “w” flag
    - Will create a new file or overwrite an existing one
- Since ofstream inherits from ostream, anything we can “<<” to an ostream, we can “<<” to the ofstream
- ofstream has a destructor that closes the file
    - When an ofstream object’s lifetime ends, it automatically closes itself

C++ istream usage

```
// io2.cpp:
#include <iostream>
#include <fstream>
#include <string>
int main(){
        std::ifstream ifile( "hello.txt" );
        if (!ifile.is_open()) {
              return 1;
        }
        std::string word;
        while( ifile >> word )
               std::cout << word << std::endl;
        return 0;
}
```

```
$ g++ -c io2.cpp -std=c++11 -pedantic -Wall -Wextra
$ g++ -o io2 io2.o
$ ./io2
Hello,
World!
```

C++ File Input (std::ifstream)

- ifstream has a constructor taking a string specifying the filename
    - Calling the constructor with a filenamestring is the same as calling fopen with the filename using a “r” flag
    - The file must already exist
- Since ifstream inherits from istream, anything we can “>>” to an istream, we can “>>” to the ifstream
- ifstream has a destructor that closes the file
    - When an ifstream object’s lifetime ends, it automatically closes itself

C++ fstream usage

```
// io3.cpp:
#include <iostream>
#include <fstream>
#include <string>

const std::ios::openmode mode =
  std::ios_base::in | std::ios_base::out | std::fstream::app;

int main() {
  std::fstream fs;
  fs.open("data.txt", mode);
  fs << "Hello CS 220" << std::endl;
  fs.clear();
  fs.seekg(0);
  std::string a, b;
  int n;
  fs >> a >> b >> n;
  std::cout << "Read: " << a << " " << b << " " << n << std::endl;
  return 0;
}
```

```
$ g++ -std=c++11 -pedantic -Wall -Wextra io3.cpp -o io3
$ rm -f data.txt && ./io3
Read: Hello CS 220
$ cat data.txt
Hello CS 220
```

C++ stringstream strings

`std::stringstream`

Instead of reading or writing to console or file, it reads and writes to a temporary string (“buffer”) stored inside

```
// io4.cpp:
#include <string>
#include <iostream>
#include <sstream>
int main(){
        std::stringstream ss;
        ss << "Hello" << ' ' << 35 << " world";
        std::string word1, word2;
        int num;
        ss >> word1 >> num >> word2;
        std::cout << word1 << ", " << word2 << '!' << std::endl;
        return 0;
}
```

```
$ g++ -c io4.cpp -std=c++11 -pedantic -Wall -Wextra
$ g++ -o io4 io4.o
$ ./io4
```

C++ stringstream details

- stringstream inherits from istream and ostream, so operators << and >> are defined for reading/writing from/to a stringstream
- use member function .str() to get the string out of the object

C++ stream class hierarchy

```
                           ios
                         /     \
         __________istream     ostream_________
        /          /     \     /      \        \
istringstream  ifstream  iostream  ofstream  ostringstream
                         /     \
                    fstream  stringstream
```