601.220 Intermediate Programming

Introduction to OO

C++ class - a brief review of OO programming

- We use classes to define new data types
- These are like C 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 functions called *desctructors* 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
```
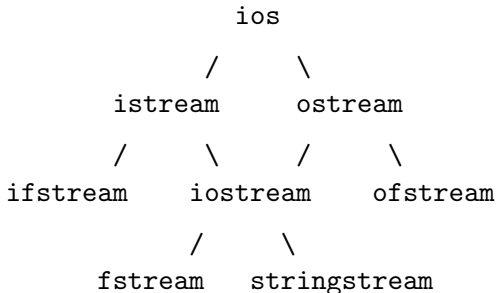
<< is the stream insertion operator; used for output

>> is the stream extraction operator; used for input

C++ file I/O

- In C, `printf` writes to `stdout` and `scanf` read from `stdin`
    - `fprintf` and `fscanf` are 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/form a file (inherits from `iostream`, i.e. from both `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
1  #include <iostream>
2  #include <fstream>
3  int main(){
4      std::ofstream ofile( "hello.txt" );
5      if (!ofile.is_open()) {
6          return 1;
7      }
8      ofile << "Hello, World!" << std::endl;
9      return 0;
10 }
```

```
$ g++ -o io1 io1.cpp -std=c++11 -pedantic -Wall -Wextra

$ ./io1


$ cat hello.txt

Hello, World!
```

C++ file output (std::ofsteam)

- 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 ofsteam 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++ ifstream usage

```cpp
// 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++ -o io2 io2.cpp -std=c++11 -pedantic -Wall -Wextra

$ ./io2

Hello,
World!
```

C++ file input (std::ifsteam)

- ifstream 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 **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

```cpp
// io3.cpp

1   #include <iostream>
2   #include <fstream>
3   #include <string>
4
5   const std::ios::openmode mode =
6       std::ios_base::in | std::ios_base::out | std::fstream::app;
7
8   int main() {
9       std::fstream fs;
10      fs.open("data.txt", mode);
11      fs << "Hello CS 220" << std::endl;
12      fs.clear();
13      fs.seekg(0);
14      std::string a, b;
15      int n;
16      fs >> a >> b >> n;
17      std::cout << "Read: " << a << " " << b << " " << n << std::endl;
18      return 0;
19  }
```

```
$ g++ -o io3 io3.cpp -std=c++11 -pedantic -Wall -Wextra

$ ./io3

Read: Hello CS 220

$ cat data.txt

Hello CS 220
```

C++ stringstream usage

```
// io4.cpp
1  #include <string>
2  #include <iostream>
3  #include <sstream>
4  int main(){
5      std::stringstream ss;
6      ss << "Hello" << ' ' << 35 << " world";
7      std::string word1, word2;
8      int num;
9      ss >> word1 >> num >> word2;
10     std::cout << word1 << ", " << word2 << '(' << num << ")!" << std::endl;
11     return 0;
12 }
```

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

$ ./io4

Hello, world(35)!
```

C++ file input (std::stringstream)

- since stringstream inherits from iostream, which inherits from istream and ostream, both << and >> are defined for reading/writhing 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
```