Introduction to Software Development – CS 6010
Lecture 9 –File I/O

Master of Software Development (MSD) Program

Varun Shankar

Fall 2022

Miscellaneous

- How is ‘a’ stored in the computer?
  - 97
- What about ‘z’?
  - 122 // or 97 + 25 (ie: 25 more letters)
  - // or ‘a’ + 25
- How to loop over all the (lower case) letters?
- for(char c = 97; c <= 122; c++) // BAD – Never do this!
- for(char c = ‘a’; c <= ‘z’; c++) // GOOD. Google “ascii table”

Lecture 9 – Sorting / File I/O

- File I/O
- Catch up on missing labs/assignments

File Input/Output (I/O)

- Reading from and writing to the console is great for interacting with the user, but…
  - For large amounts of data, reading from (and writing to) files is more convenient.
    - The user does not have to type things over and over
    - We can save/load large chunks of data easily

C++ File Streams – Ifstream / Ofstream

- if == input file, of == output file
- C++ (and most languages) uses *file streams* to read/write data from/to disk
- Note: Once we have opened a file (for reading/writing) we will treat it just like *cin* / *cout*!
- The library that handles this for us is *fstream* – How do we use a library?

#include <fstream> // File Stream

std::ifstream myStream(“file_name.txt”); // We have created a variable named myStream

// myStream is an “if” – **input** file stream. We use it just like *cin*…

// Assuming the file has an integer in it (as the first value)

int data;

myStream >> data; // Instead of *cin* we have just used our file stream (variable) myStream

Input File Stream - Opening

```
// When opening an input stream (to read data from), verify that it opened successfully!
#include <cstdlib> // For exit() function

ifstream myStream( "fileThatDoesntExist.txt" );

if( myStream.fail() ) {
	cout << "Failed to open file.\n";
	exit( 1 );  // Kill the program and exit with a return value of 1
}
```

Input File Stream – Opening BETTER

```
string filename = “fileThatDoesntExist.txt”;
ifstream myStream( filename );

if( myStream.fail() ) {
        cout << “Failed to open file: “ << filename << “.\n”;
        exit( 1 );  // Kill the program and exit with a return value of 1
}
```

- Why is this better?
  - Used a variable to store the name of the file!

Input File Stream(ing)

```
// Read in the 1st two words from myStream.
// Remember, myStream is attached to file_name.txt
// Individual strings are separated by spaces, so…
string word1, word2;
myStream >> word1 >> word2;
// Or better yet…
while( myStream >> word1 ) { // true if data was read in
        cout << “Read in: “ << word1 << “\n”;
} // Loop reads all data (assuming it is all strings) in file.
```

file_name.txt

Four score and
seven years ago…

Input File – Read Was Successful?

// What happens if we try to read in an integer?

```
int i;
myStream >> i;
cout << “Found?:  “ << myStream.good() << “\n”;
string word1, word2;
myStream >> word1 >> word2;
cout << “Read ‘” << word1 << “’ and ‘”
     << word2 << “’\n”
// Outputs:
Found?:  0      // What is the 0?
Read ‘’ and ‘’ // File pointer moved to end of
               // file when failed to find the int (as 1st thing in file)
```

file_name.txt

Four score and
seven years ago…

File Pointer

- The stream keeps track of where in the file it is currently looking
- To reset the file pointer:

```
myStream.clear(); // Clear any fail / EoF bits
myStream.seekg( 0, std::ios::beg ); // Move file pointer to the beginning of the file.
```

Output File Stream(ing)

- Output – similar to creating the input stream…
  - As usual, these library functions/objects are in the *std::* namespace. I am leaving it off here (ie, using namespace std) to help with clarity.

```
// Create a variable of type ofstream (output file stream) - As always: <type> <name>
ofstream myOutputStream( “output_file_name.txt” );
// Note, if output_file_name.txt exists, it will be erased and overwritten.

// How to just append to the file?
ofstream myAppendingOutputStream( “file_name.txt”, std::ios_base::app );

// Now to save some data to disk (just like writing to the console)…
myOutputStream  <<  “Hello World will now be written in the file.\n”;
```

Close the Stream

- It is good practice to explicitly close any file stream that you have opened.

```
myStream.close();
myOutputStream.close(); // likewise, close the output file
```

- If you don't close the (usually) output stream, it is possible that (some of the) data you wrote to the file will not actually get to the file. For example, if the program crashes before it ends.
- This is because data is *buffered* before it is actually written, so the data in the buffer may be lost.

- You can force the buffer to be written with:

```
myOutputStream.flush(); // Also, data is written on std::endl (but not "\n")
```

- Why would data be buffered before being written to disk?
  - Efficiency (disk writes are slow, do fewer of them)

Quick Quiz Questions

- What is the type of myStream? Of myOutputStream?
  - streams, or better yet file streams, or better yet input/output file streams repectively
- What is the “.close()”?
  - A method. This also reminds (tells?) us that these variables are *Objects*
- What is an object?
  - A variable that both holds data and provides functions on that data.

Location of Files

- Streams opens files (by default) in the current working directory
    - pwd

```
ifstream myStream( “filename.txt” );
```

- “filename.txt” would be in the directory you ran the program from.

```
ifstream myStream( “dirname/filename.txt” );
```

- This is a *relative* path (start in the current dir), and so is this:

```
ifstream myStream( “../filename.txt” );
```

- But go up a directory before looking for filename.txt

```
ifstream myStream( “/filename.txt” );
```

- Absolute path (in this case “filename.txt” would be in the root directory)
- Always starts with a “/”

Location of Files (XCode)

- By default, XCode places files in:

/Users/<username>/Library/Developer/Xcode/DerivedData/dummy-<*>/Build/Products/Debug/<file_name.txt>

- Sigh…
- So, let's set the working directory in XCode
  - Product −> Scheme −> Edit Scheme…
  - Run −> Options (Tab) −> Working Directory (Check Box)
    - Set to the project directory.

Today’s Assignment(s)

- Code Review: Vectors
- Catch up on previous work.
- This could be labs, assignments, whatever.
- Take a stab at tomorrow’s assignment if you’d like (involves File IO).