How to use external text files as input and output from a C++ script

CS 10A – EXTERNAL FILE IO

Allen Zhao – SRJC CS 10A

The File Stream

- The file stream <fstream> is the library to include when we want to read and write directly to text files with our programs. For convenience, keep your text file where your program is.
- The file stream comes in two parts, which can be used independently if so desired.
  - <ifstream> is the input file stream, for reading files only
  - <ofstream> is the output file stream, for writing to files only
- A file becomes read/write accessible by declaring a stream object, and then linking the file to that stream object. You can consider this a variable type.
- Overall, using <fstream> is similar to using <iostream>
- Side note: add debug text to your program to know where it's at.

Starting Up a File Stream

```
#include <fstream>
int main()
{
        ifstream inputFile0;                    // Declare variable inputFile0 as input file stream object
        inputFile0.open("someFile.txt");        // Links the given file name to the given stream object

        ifstream inputFile1;
        inputFile1.open("C:\\local\\file.txt");  // File path C:\local\file.txt
        // Remember that \ is used for escape characters so \\ allows us to use \ in the file path string

        ofstream outputFile;                    // Output file stream object
        outputFile.open("fileOut.txt");         // File will be created if it does not already exist
                                                // File location is relative to executable's location

        return 0;
}
```

Writing to a File

Program

```
#include <fstream>
int main()
{
        // Output file stream object subs cout
        ofstream outputFile;
        outputFile.open("fileOut.txt");

        cout << "Writing to file…" << endl;
        outputFile << "Example string" << endl;
        for(int i = 0; i < 5; i++)
                outputFile << i << ' ';
        cout << "Done" << endl;          // Debug text
        return 0;
}
```

Console Output and Output File

- ./a.out

```
Writing to file…
Done
```

- vim fileOut.txt

```
Example string
0 1 2 3 4
```

Whitespace Management

- Make use of your escape characters!
  - \t for Tab
  - \n for New Line (Enter)
- There is no standard definition for a tab across multiple computer systems. You can even manually change the definition of a tab in certain applications. Keep this in mind if you want a specific format for your text file output.
- Use the IO manipulation library and loops to control your spacing and formatting.
- File output is not displayed to IO Stream output, so it may be a good idea to also have your program write debug text to the Terminal window during the write process.

Closing a File

```
// Leaving a file stream open, both input and output types, consumes processing power.
#include <fstream>
int main()
{
        ofstream outputFile;
        cout << “Now opening file…” << endl;          // Debug text
        outputFile.open(“someFile.txt”);
        cout << “Done” << endl;                       // Debug text
        // … Code for file manipulation here

        cout << “Now closing file…” << endl;          // Debug text
        outputFile.close();                           // Close the file stream when you’re done with it
        cout << “Done” << endl;                       // Debug text
        return 0;
}
```

Reading from a File – Open and Validate File

```
#include <fstream>
int main()
{		// You can verify whether if you opened the file successfully.
		ifstream inputFile;
		inputFile.open(“someFile.txt”);
		if(inputFile)					// File stream object can act as a Boolean
				cout << “File opened successfully, now reading…” << endl;		// Debug text
		else
		{
				cout << “Error opening file” << endl;	// Debug text
				inputFile.close();
		}
		if(inputFile.fail())				// Alternative method using a member function
		{
				cout << “Error opening file” << endl;	// Debug text
				inputFile.close();			// Close the stream if the file open fails
		}
		return 0;
}
```

Reading from a File – User Defined File

```
#include <fstream>
int main()
{
        cout << “Enter file name: ”;      // Debug text
        string fileName;                  // You can also set a default output file name
        getline(cin, fileName);           // Remember to be inclusive of spaces

        ifstream fileInput;
        fileInput.open(fileName);         // File names and paths are just single strings
        // …
        fileInput.close();
        return 0;                         // All string rules apply, so use concatenating
}                                         // and appending where applicable
```

Reading from a File – Input String Data

Program

```
// Use the file input stream the same way you would use cin
#include <fstream>
int main()
{
        ifstream inputFile;
        inputFile.open("text.txt");
        // Input streams do not alter the original file's contents
        string temp;
        getline(inputFile, temp);
        cout << temp << endl;
        // Contents of a file is read sequentially and only once
        inputFile >> temp;
        cout << temp << endl;
        return 0;
}
```

Input File and Console Output

- vim text.txt

```
Line 1
Line 2
Line 3        // "2" and "Line 3" remain unread
```

- ./a.exe

```
Line 1        // Items that are read once are not read again
Line
```

Reading from a File – Input Numeric Data

Program

```
// If you're expecting a number, use the appropriate numeric data type
#include <fstream>
int main()
{
        int x;
        ifstream inputFile;
        inputFile.open("text.txt");

        cout << "Now reading data…" << endl;
        for(int i = 0; i < 3; i++)
        {       // Works with arrays too
                inputFile >> x;
                cout << x << endl;
        }       // More on sizing later…
        return 0;
}
```

Input File and Console Output

- vim text.txt

```
23 26
14 9
```

- ./a.exe

```
Now reading data…
23
26
14
```

Reading from a File – End of a File

Program

```
#include <fstream>
int main()
{
        ifstream inputFile;
        inputFile.open(“text.txt”);
        int number, size = 0;
        // Brings in a value from file, returns 1 if successful
        while(inputFile >> number)    // >> works the same way as before
        {
                size++;          // How many values in the file?
                cout << number << endl;
        }                // Loop continues until no more values can be found
        // Note: This technique does not work with cin
        inputFile.close();               // All values have been used, so close
        return 0;
}               // You can close and reopen a file multiple times if necessary
```

Input File and Console Output

- vim text.txt

12
24
36

- ./a.exe

12
24
36

File From Where?

- When a path is not mentioned in the string specifying the file name, the program looks for the file where the executable (*.exe or *.out) is located.
- However, some IDEs instead use the location of the program (*.cpp) as the default.
- Execute a file output to verify where the default location is for your setup. Files should go here for text file inputs.
- For execution consistency, your assignment submissions must use the default file location.