Chapter 12. Streams & File I/O

Stream: flow of data processed sequentially at a time.

- In <io*stream*>, we used:
  - cout: output stream connected to the screen
  - cin: input stream connected to the keyboard

<fstream> Basics

- To send output to a file, use the **ofstream** object.
- To read input from a file, use the **ifstream** object.
- Both objects are defined in the <fstream> library.
- To use the i/ofstream, declare them as variables.

```
Example:    ifstream fin;        // fin is a variable name
            ofstream fout;       // fout is a variable name
```

- Open function:
  - To connect a stream to a file, use the **open** function. The function takes in a single parameter, a C-string of the filename. You need to specify which stream is accessing the file by having the object call the open function.
    - For example, an ifstream object with a variable name **fin** was declared above. To specify that **fin** is accessing a file named "sample.txt", the code is written as: **fin.open("sample.txt");**
  - The parameter takes in a C-String and cannot accept regular Strings. To convert a regular string into a C-String, use the **c_str()** function.
    - Example:

```
string filename;
cout << "Enter a filename: ";
cin >> filename;
ifstream fin;
fin.open(filename.c_str());
```

- Reading from/writing to a file:
  - is used the same way as using cin/cout. In place of cin/cout, use the i/fstream variable name.
  - Example: Reading from a file (input)

```
string word;
fin >> word;
```

  - Example: Writing to a file (output)

```
fout << "Have a wonderful day!";
```

- **Close** the stream once you're done using it.

```
fin.close();
fout.close();
```

Ofstream & Open

- When an ofstream object calls the open function on a non-existing file, C++ will create a file of that name in the current directory.
- When an ofstream object calls the open function on an existing file, the previous contents in the file will be erased.
- To append (add onto) text to the existing file, use the 2-parameter open function:
  OUTPUT_STREAM.open(*FILENAME*, ios::app);
  Example: fout.open("sample.txt", ios::app);

More Ifstream

- The ifstream object reads content from an existing file using the **open** function.
- The **fail()** function checks whether the file was opened successfully. It returns a Boolean value: TRUE if the file *failed* to open, FALSE otherwise. This check should be used after calling the open function from both ifstream and ofstream objects.

```
Example:    fin.open("sample.txt");
            if (fin.fail()) {
                cout << "File did not open successfully." << endl;
                exit(1);
            }
```

- Once our file has been opened successfully, we wish to read the inner contents of the file. The ifstream object can call the .eof() "end of file" function to check whether the stream has reached the end of the file. It returns a Boolean value: TRUE if it reached the end of file; FALSE otherwise.
  - Example: Reading numbers from a file and adding to a sum

```
int read, sum = 0;
fin >> read;
while ( ! fin.eof() ) {
    sum += read;
    fin >> read;
}
```

  - Another way to write the above:

```
int read, sum = 0;
while (fin >> read) {
    sum += read;
}
```

Flags to format precision output

- cout.setf(ios::fixed) → specifying regular decimal points (no scientific notation)
- cout.setf(ios::showpoint) → always include decimal point for floating point numbers
- cout.precision(2) → 2 significant figures OR two digits after decimal point, depending on compiler (most recent compilers will use the latter)
- cout.unsetf → unset a flag

More ios flags:

- ios::fixed
- ios::scientific
- ios::showpoint
- ios::showpos
- ios::right
- ios::left
- ios::dec
- ios::oct
- ios::hex
- ios::uppercase
- ios::showbase