Streams

CS A150 – C++ Programming I

STREAMS

- A **stream** is a flow of characters (or other kind of data)
- **Input stream**:
  - Flow going *into* your program
- **Output stream**:
  - Flow going *out* of your program

- We have seen streams:
  - **`cin`** → input stream connected to the keyboard
  - **`cout`** → output stream connected to the screen

File I/O

- The files we will use for **I/O** (input/output) are **text** files
- Reading from file:
  - Your program takes input from a file
- Writing to a file:
  - Your program sends output to a file

How It Reads and Writes

- There are ***several*** ways to read from a file
- We will use a method that
  - Reads the file from beginning to end (***cannot*** back up)
  - Writes output into the file starting at the beginning of file and processing forward

Steps to Read from a File

- To **read** data from a file you connect the file to an object (**stream**) of the class **ifstream**
  - **Include**
    - the **fstream** library
    - **std namespace**
  - **Declare streams**
    - An object of the class **ifstream**

Steps to Read from a File (cont.)

- **Open the file to read it**
  - You might need to specify a path name
  - ***Always*** write the **file extension**
  - Note that the file has **<u>two</u> names**
    - **fileName.txt** → name of the file
    - (**stream name**) → object name

Steps to Read from a File (cont.)

- **Read from the file**
  - You use the **extraction operator** **>>**
  - Same as **`cin`**
- **Close the file**
  - Every file should be closed to disconnect the stream from the file.
  - If you do not close it, the system will automatically close it
    - **BUT**, if your program ends with an error, your file will be corrupted.
    - **BEST PRACTICE:** Always close your file.

Example – Reading from a File

- File: IO_Example_1

Steps to Write to a File

- To **write** data to a file you connect the file to an object (stream) of the class **ofstream**
    - **Include** (same as reading)
        - the **fstream** library
        - **std namespace**
    - **Declare streams**
        - An object of the class **ofstream**

Steps to Write to a File

- **Open the file to write on it**
  - You might need to specify a **path** name
  - ***Always*** write the **file extension**
  - **Note** that
    - If file does **<u>not</u>** exist, it **will be created**
    - If file **<u>exists</u>**, it will be **overwritten**

Steps to Write to a File (cont.)

- **Write to the file**
  - You use the **insertion operator** **<<**
  - Same as **cout**
- **Close the file** (same as reading)
  - Every file should be closed to disconnect the stream from the file.

Example – Writing to a File

- File: IO_Example_2

Appending to a File

- When you use

```
outStream.open(outfile.txt);
```

  - you **overwrite** the contents of the text file
- If you need to append contents to existing contents
  - Need to use the **class ios**

    → defined in the **iostream** library
  - The ***open*** statement will have parameters

```
outStream.open("fileName.txt", ios::app);
```

Example – Appending to a File

- File: IO_Example_3

Checking If the File Is Opened

- You need to check whether the file was opened successfully, because:
  - What if you typed the wrong name for the input file that needs to be opened?
  - What if you are trying to open an output file for which you have no writing permissions?
- Use the following:

```
if (inStream.fail()) //or outStream
{
        cerr << … //output error message
        exit(1);  //terminate program gracefully
}
```

Testing for the End of File

- When reading from a file, you need to use a loop that will stop when it reaches the end of the file
- A **while loop** can keep on reading up until the ***end*** of file

```
char next;
inStream.get(next);
while (!inStream.eof())
{
      //read…
}
```

Testing for the End of File

- When reading from a file, you need to use a loop that will stop when it reaches the end of the file
- A **while loop** can keep on reading up until the ***end*** of file

```
char next;
inStream.get(next);
while (!inStream.eof())
{
      //read…
}
```

Example

- File: IO_Example_4
  - Check if file is open
  - Check end of file

File Names as Inputs

- Instead of hard-coding the file name

```
inStream.open("file.txt");
```

  - You can read the name from the keyboard
- Reads it as a **C-string** (cannot use a string)
  - An array of chars where element at *n-1* is NULL
  - There is ***no*** predefined type-cast operator to convert from a string object to a C-string
    - BUT, you can read it as a string and use the `c_str()` to produce the corresponding C-string value

File Names as Inputs (cont.)

- If the user enters the name of the file, you need to ask the extension as well

```
string fileName;

cout << "Enter file name: \n";
cin >> fileName;
inStream.open(fileName.c_str());
```

Example – File Name as Input

- File: IO_Example_5

Streams (end)