# Object-Oriented Programming

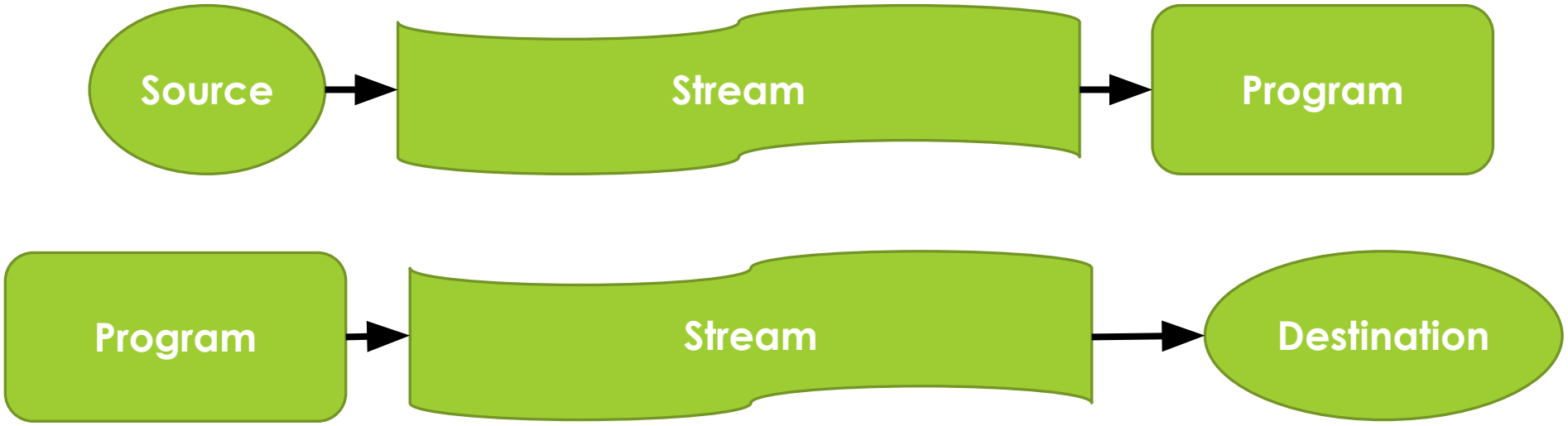
WEEK 12

Abeeha Sattar

# Streams

- What are streams?
- …not these ones!!
- But there is a similar concept here…

# Stream in Computer Programming



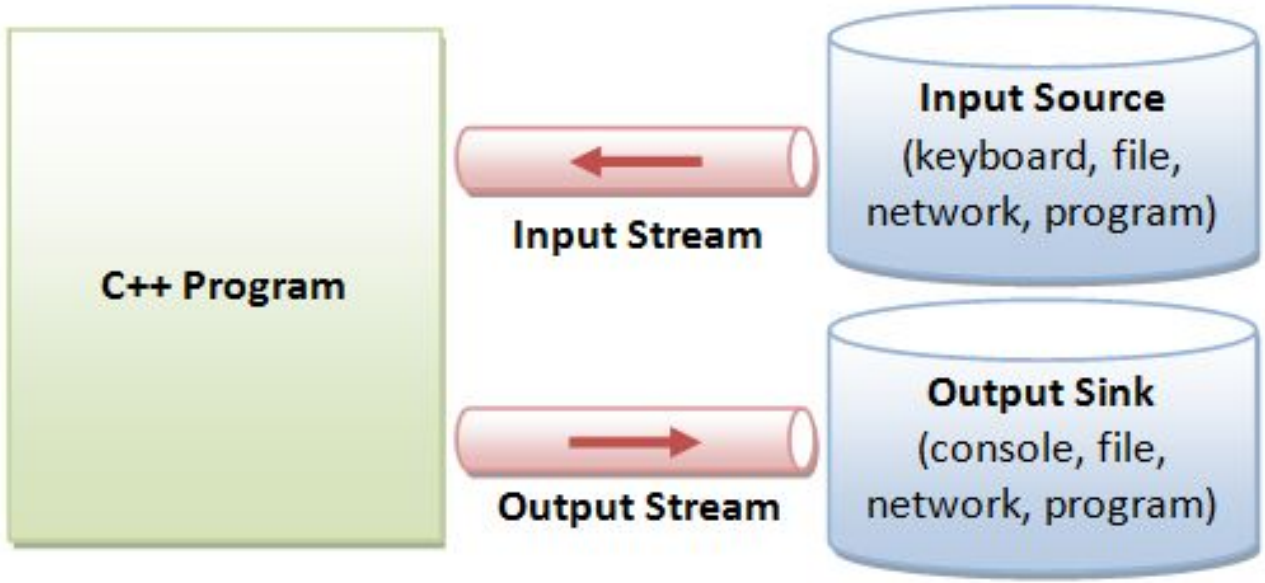
# What is a Stream?

- A transfer of information in the form of a sequence of byte

# I/O Streams

- Input Stream:
  - A stream that flows from an input device ( i.e.: keyboard, disk drive, network connection) to main memo
- Output Stream:
  - A stream that flows from main memory to an output device ( i.e.: screen, printer, disk drive, network connection)

# C++ I/O Stream




Internal Data Formats:

- Text: `char`, `wchar_t`
- `int`, `float`, `double`, etc.

External Data Formats:

- Text in various encodings (US-ASCII, ISO-8859-1, UCS-2, UTF-8, UTF-16, UTF-16BE, UTF16-LE, etc.)
- Binary (raw bytes)

https://www3.ntu.edu.sg/home/ehchua/programming/cpp/cp10_io.html

# I/O Stream Library Header Files

- `<iostream.h>`
  - Contains **cin** & **cout** objects
- `<fstream.h>`
  - Contains information important to **user-controlled file processing** operations

# File Streams (fstream)

- `ifstream`
  - defines new input stream (normally associated with a file)
- `ofstream`
  - defines new output stream (normally associated with a file).

# General File I/O Steps

- Include the header file **`fstream`** in the program.
- Declare file stream variables.
- Open the file
- Use the file stream variables with **`>>`**, **`<<`**, or other input/output functions.
- Close the file.

# Basic Syntax (We'll add to this afterwards)

```
#include <iostream>
#include <fstream>

using namespace std;

int main()
{
    ifstream fin; // input only
    ofstream fout; // output only
    fstream finout; // input and output                    both

    fin.open("TEXTFILE.txt");
    fout.open("MyOutputFile.txt");

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

# Opening a File…

- Let's understand this:

```
fin.open("TEXTFILE.txt");
```

- fin is our input file stream object
- open is function available for any ifstream/ofstream/fstream object
- The 'open' function accepts a string with the file name or file path in it
- You can directly use the filename if the file you want to open exists in the same directory as your cpp file. Otherwise you will have to provide a complete file path
- You always have to specify the file extension with the file name.

# What Happens When you Open a File Stream?

- Opening a file associates a file stream variable declared in the program with a physical file at the source, such as a disk.
- In the case of an input file:
  - the file must exist before the open statement executes.
  - If the file does not exist, the open statement fails and the input stream enters the fail state
- An output file does not have to exist before it is opened;
  - if the output file does not exist, the computer prepares an empty file for output.
  - If the designated output file already exists, by default, the old contents are erased when the file is opened

# Things you Need to do Before Accessing the Opened File.

- VALIDATE!!!

Method 1:

Check the stream variable

```
if (!fin)
    {
        cout << "Cannot open file.\n";
    }
```

Method 2:

By using is_open() function, which returns a bool value

```
if (!fout.is_open()) {
        cout << "File is not open.\n";
```

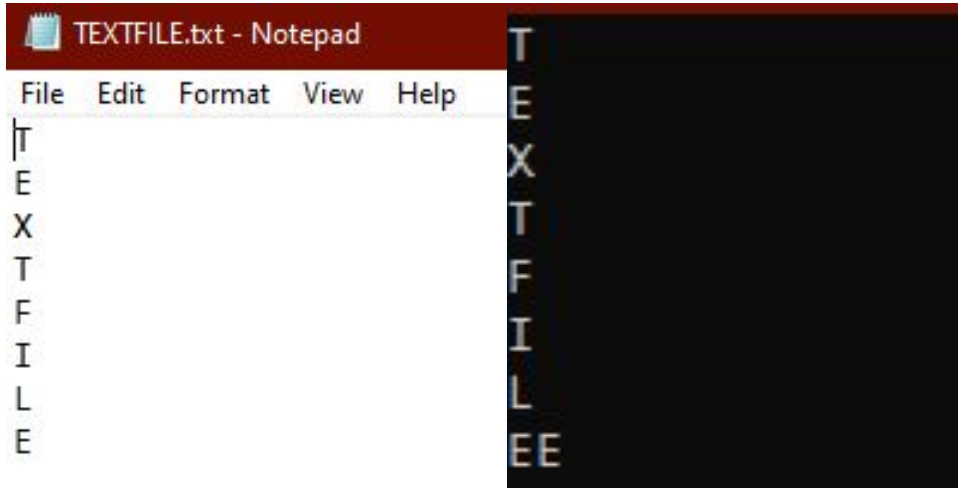
# File Input/Read Related Methods

- **`fin.get(char character)`**
  - extracts next character from the input stream **fin** and places it in the character variable character.

- **`fin.eof()`**
  - tests for the end-of-file condition

# Reading Character by Character

```
int main() {
    //Declare and open a text file
    ifstream fin("TEXTFILE.txt");
    char ch;
    //do until the end of file
    while (!fin.eof()) {
        fin.get(ch); // get one character
        cout << ch; // display the character
    }
    fin.close(); // close the file
    return 0;
}
```



# Reading a Line

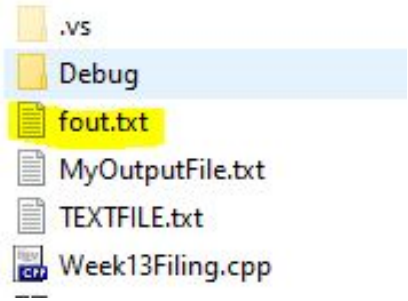
```
int main() {
    //Declare and open a text file
    ifstream fin("TEXTFILE.txt");
    string line;
    while (!fin.eof()) {
        //fetch line from data.txt and put it in a string
        getline(fin, line);
        cout << line << endl;
    }
    fin.close(); // close the file
    return 0;
}
```

# File Output/Write Related Methods

- **`fout.open(const char[] fname)`**
  - connects stream fout to the external file name
- **`fout.put(char character)`**
  - inserts character to the output stream fout
- **`fout.eof()`**
  - tests for the end-of-file condition

# Writing to a File (opening file via constructor)

```
int main()
{
/* declare and automatically open thefile*/
    ofstream fout("fout.txt");
//behaves just like cout, puts the word into the file
    fout << "Hello World!";
    fout.close();
    return 0;
}
```




# Writing to a File (using the open function)

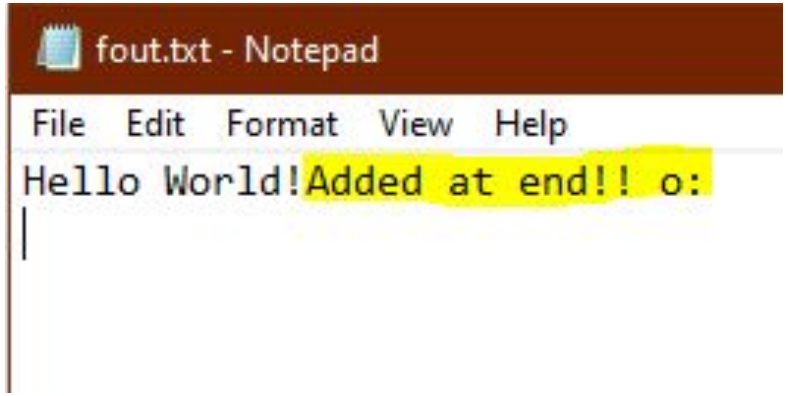
```
int main(){
// declare output file variable
    ofstream fout;
// open an existing file fout.txt
     fout.open("fout.txt");
//behaves just like cout, puts the word into the file
     fout << "Hello World!";
     fout.close();
    return 0;
}
```

# File Open Modes

| Name | Description |
|---|---|
| ios::in | Open file to read |
| ios::out | Open file to write |
| ios::app | All the data you write, is put at the end of the file. It calls ios::out |
| ios::ate | All the data you write, is put at the end of the file. It does not call ios::out |
| ios::trunc | Deletes all previous content in the file. (empties the file) |
| ios::nocreate | If the file does not exists, opening it with the open() function gets impossible. |
| ios::noreplace | If the file exists, trying to open it with the open() function, returns an error. |
| ios::binary | Opens the file in binary mode. |

# Example of Open Modes

```
int main() {
    ofstream outFile("fout.txt", ios::out | ios::app);
    outFile << "Added at end!! o:" << endl;
    outFile.close();
    return 0;
}
```



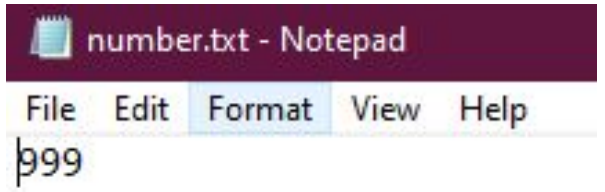

You can add more open modes by adding the OR/Pipeline symbol “ | ”

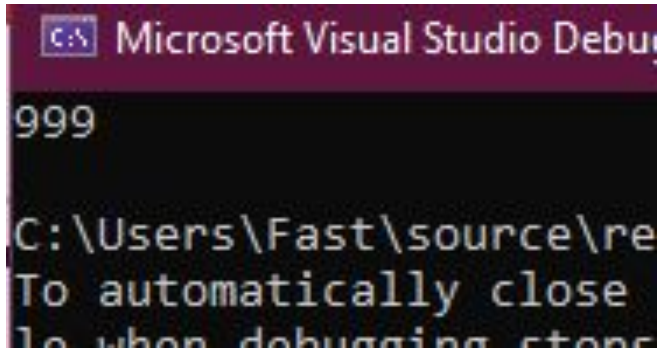
# Reading Numbers from a File

```
int main() {
    ifstream fin("number.txt");
    string line;
    int total = 0;

    while (getline(fin, line)) {
    stringstream(line) >> total;
    cout << total << endl;

    }
}
```



# Writing/Reading Objects to/from File

```
class Person {
    public:
        char name[10];
        int age;
};
int main() {
    Person person;
    cin >> person.name;
    person.age = 25;

//writing object to a file
    ofstream file("person.bin", ios::binary);
    file.write((char*)&person, sizeof(Person));
    file.close();
    cout << "File written successfully." << endl;
//reading object from a file
    Person otherPerson;
    ifstream file2;
    file2.open("person.bin", ios::in);
    file2.seekg(0);
    file2.read((char*)&otherPerson, sizeof(Person));
    cout << "\nName : " << otherPerson.name << endl;
```

# Fin.