Programming Fundamentals

Aamina Batool

Class

- Header file
- Source file
- Stream class

Stream

- a **stream** is a sequence of characters from the source to the destination. There are two types of streams:
- **Input stream:** A sequence of characters from an input device to the computer.
- **Output stream:** A sequence of characters from the computer to an output device.

Stream types

- ***stream*** - a sequence of characters
  - interactive (iostream)
    - **cin** - input stream associated with **keyboard.**
    - **cout** - output stream associated with **display.**
  - file (fstream)
    - **ifstream** - defines new input stream (normally associated with a file).
    - **ofstream** - defines new output stream (normally associated with a file).

String and char

- Cin reads and stores value in variables based on space

```
#include <iostream>
#include <string>
using namespace std;
int main()
{
char a;
string b;
cin>>a>>b;
cout<<a<<endl;
cout<<b;
return 0;
}
```

iostream

- To receive data from the keyboard and send output to the screen, every C++ program must use the header file **`iostream`**. This header file contains, among other things, the definitions of two data types, **`istream`** (input stream) and **`ostream`** (output stream). The header file also contains two variable declarations:
- one for **`cin`** (pronounced ‘‘see-in’’), which stands for **common input**, and
- one for **`cout`** (pronounced ‘‘see-out’’), which stands for **common output**
- These variable declarations are similar to the following C++ statements:
- **`istream cin;`**
- **`ostream cout;`**

istream and ostream datatypes

- Variables of type **`istream`** are called **input stream variables**;
- variables of type **`ostream`** are called **output stream variables**.
- A **stream variable** is either an input stream variable or an output stream variable.

cin.get

- `char ch1, ch2;`
- `int num;`
- and the input:
- `A 25`
- Now consider the following statement:
- `cin >> ch1 >> ch2 >> num;`

- `cin.get(varChar);`

cin.get

- `cin.get(ch1);`
- `cin.get(ch2);`
- `cin >> num;`

- `OR`

- `cin >> ch1;`
- `cin.get(ch2);`
- `cin >> num;`

cin.getline

- `string name; //variable declaration`
- `cin >> name; //input statement`
- `Alice Wonderland`
- then after the statement:
- `cin >> name;`
- executes, the value of the variable **`name`** is **`"Alice"`**.

cin.getline

- To read a string containing blanks, you can use the function **`getline`**.
- The syntax to use the function **`getline`** is:
- **`getline(istreamVar, strVar);`**
- **`getline(cin, myString);`**

Using input/output files

- File - sequence of bytes stored on some storage devices.
- The data of a file is stored in either readable form or in binary code called as text file or binary file.
- Computer programs are associated to work with files as it helps in storing data & information permanently.
- In C++ this is achieved through a library called **fstream**
- The library predefines a set of operations for all file related handling through certain classes.

C++ support for file handling

- C++ provides the following classes to perform output and input of characters to/from files:
- **ofstream**: Stream class to write on files
- **ifstream**: Stream class to read from files
- **fstream**: Stream class to both read and write from/to files.

Data Type	Description
ofstream	This data type represents the output file stream and is used to create files and to write information to files.
ifstream	This data type represents the input file stream and is used to read information from files.
fstream	This data type represents the file stream generally, and has the capabilities of both ofstream and ifstream which means it can create files, write information to files, and read information from files.

File I/O

- File I/O is a five-step process:
- 1. Include the header file **`fstream`** in the program.
- 2. Declare file stream variables.
- 3. Associate the file stream variables with the I/O sources.
- 4. Use the file stream variables with **>>**, **<<**, or other I/O functions.
- 5. Close the files.

Step 1

- Step 1 requires that the header file **`fstream`** be included in the program. The following
- statement accomplishes this task:
- **`#include <fstream>`**

Step 2

- Step 2 requires you to declare file stream variables. Consider the following
- statements:
- **`ifstream inData;`**
- **`ofstream outData;`**
- The first statement declares **`inData`** to be an input file stream variable. The second
- statement declares **`outData`** to be an output file stream variable.

Step 3

- Step 3 requires you to associate file stream variables with the I/O sources. This step
- is called **opening the files**. The stream member function **`open`** is used to open files.
- The syntax for opening a file is:
- `fileStreamVariable.open(sourceName);`
- `inData.open("progIn.txt"); //open the input file; Line 1`
- `outData.open("progOut.txt"); //open the output file; Line 2`

Step 4

- Step 4 typically works as follows. You use the file stream variables with **>>**, **<<**, or other I/O functions
- **`inData >> payRate;`** reads the data from the file **`progIn.txt`** and stores it in the variable **`payRate`**.
- The statement: **`outData << "The paycheck is: $" << pay << endl;`** stores the output—**`The paycheck is: $565.78`**—in the file **`progOut.txt`**

Step 5

- Step 5 requires closing the files. Closing a file means that the
- file stream variables are disassociated from the storage area and are freed.

- `inData.close();`
- `outData.close();`

open function

- 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.

Stream objects as arguments of functions

- **int ReadQuizData (ifstream& inData, int x);**

Basic Syntax

- 1. To access file handling routines:

```
#include <fstream>
using namespace std;
```

- 2. To declare variables that can be used to access file:

```
ifstream ifile;
ofstream ofile;
```

- 3. To connect your program's variable to a file :

```
ifile.open(“my_file.txt");
ofile.open("out_file.txt");
```

Reading till end of file.

- Predefined function .eof() is used to check if end of file has occurred.
- If reading integers from file, consider the following code:

```
ifstream fin;
fin.open(“numbers.txt”);
int num;
int index=0;
while(!fin.eof())
{
fin>>num;
cout<<num;
}
```

```
// basic file operations
 #include <iostream>
 #include <fstream>
using namespace std;
 int main ()
 {
ofstream fout;
fout.open (“C:\\example.txt");
 fout << "Writing this to a file.\n";
 fout.close();
 return 0;
}
```

Opening a file in different modes

ios::in

Open a file for reading.

ios::out

Open a file for writing.

ios::app

Append mode. All output to that file to be appended to the end.

How to use ios::

- ofstream fout;
- fout.open ("example.bin", ios::out | ios::app);

```
int main ()
 {
ofstream myfile ("example.txt");
 if (myfile.is_open())
{
myfile << "This is a line.\n";
 myfile << "This is another line.\n";
 myfile.close();
}
 else
cout << "Unable to open file";
return 0;
}
```

```
int main ()
{ string line;
ifstream myfile ("example.txt");
if (myfile.is_open())
 {
while ( getline (myfile,line) )
 {
  cout << line << '\n';
 }
myfile.close();
}
else
cout << "Unable to open file";
return 0;
}
```