ADARSH BCA COLLEGE-BOTAD
UNIT : 3
C++ (NEP)

Unit-3 : Stream Handling and File Handling

- **Input output stream class and stream handling (Formatting input and output)**
- **Formatted Console I/O Operations with functions and Manipulators –width( ), precision( ), fill), setf(), unsetf( )**
- **File stream classes (ifstream, ofstream and fstream)**
- **Basic File operations (opening, reading, writing and closing)**
- **Random access file**
- **Command line argument**

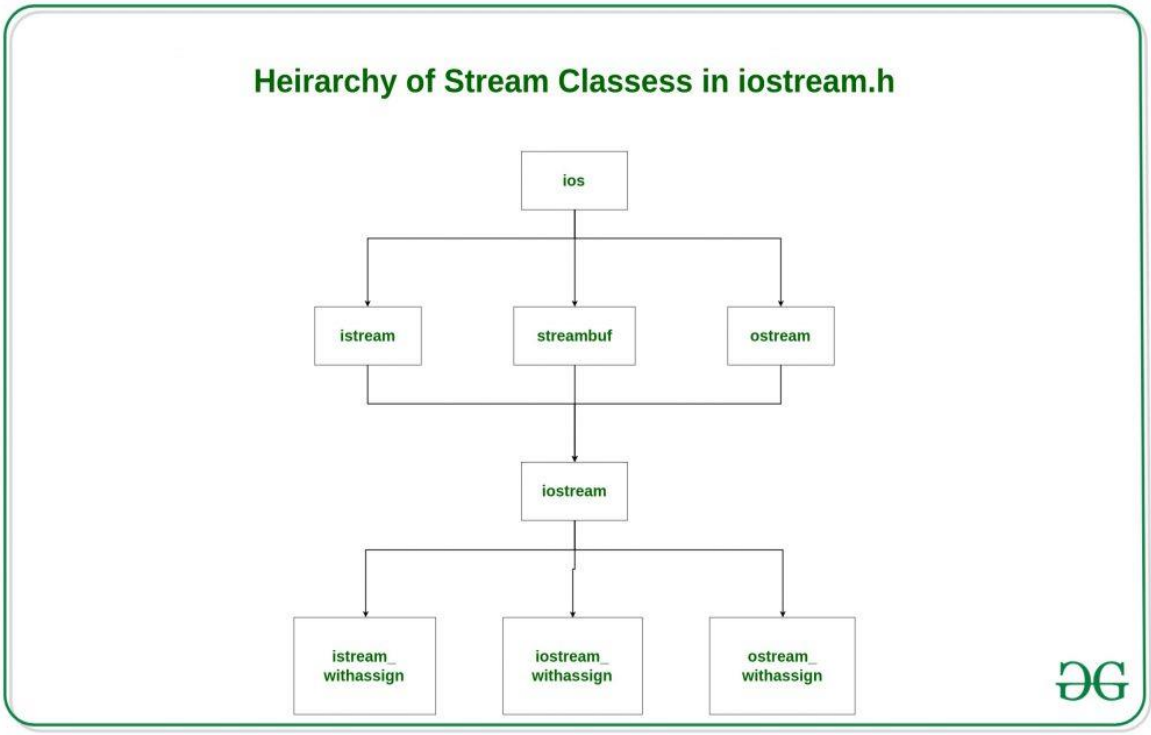
Q- Explain the c++ stream class.

Ans :

In C++ there are number of stream classes for defining various streams related with files and for doing input-output operations.

All these classes are defined in the file **iostream.h**.

There are following figure that shows the hierarchy of these classes.



1. **ios class** is topmost class in the stream classes hierarchy.
   It is the base class for **istream, ostream,** and **streambuf** class.

2. **istream** and **ostream** serves the base classes for **iostream** class
   The class **istream** is used for input and
   The class **ostream** is used for the output.

3. Class **ios** is indirectly inherited to **iostream** class using **istream** and **ostream**. To avoid the duplicity of data and member functions of **ios** class, it is declared as virtual base class when inheriting in **istream** and **ostream** as

```
class istream: virtual public ios
{
};
class ostream: virtual public ios
{
};
```

Facilities provided by these stream classes.

1. **The ios class:** The ios class is responsible for providing all input and output facilities to all other stream classes.

   It provides function of both **istream class** and o**stream class** for handling chars, strings and objects such as **get, getline, read, ignore, putback, put, write** etc..

2. **The istream class:** This class is responsible for handling input stream.
   It provides number of function for handling chars, strings and objects such as **get, getline, read, ignore, putback** etc..

   Example:

```
#include <iostream>

using namespace std;

int main()

{

    char x;

    cin.get(x);

    cout << x;

}
```

3. **The ostream class:** This class is responsible for handling output stream.

   It provides number of function for handling chars, strings and objects

   such as **write, put** etc..

Example:

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

int main()
{
        char x;

        cin.get(x);

        cout.put(x);
}
```

Q- Explain the c++ file function with example.

Ans : There are following c++ file function

1. **width():** The width method or function is used to set the required field width. The output will be displayed in the given width

   EXAMPLE :

```
#include<bits/stdc++.h>

using namespace std;

void IOS_width()
{
   cout << "-------------------------\n";
   cout << "Implementing ios::width\n\n";

   char c = 'A';
   cout.width(5);

   cout << c <<"\n";
```

```
int temp = 10;

cout<<temp;
cout << "\n-------------------------\n"; }
```

Output:

```
-------------------------
Implementing ios::width
   A
10
```

2. **precision():** The precision method is used to set the number of the decimal point to a float value

EXAMPLE :

```
 void IOS_precision()

{
  cout << "\n-------------------------\n";
  cout << "Implementing ios::precision\n\n";

  cout.precision(2);
  cout<<3.1422;
  cout << "\n-------------------------\n";
}
```

3. **fill():** The fill method is used to set or fill a character in the blank space.

Example :

```
 void IOS_fill()

{

  char ch = 'a';


  cout.fill('*');

  cout.width(10);
  cout<<ch <<"\n";


  int i = 1;
```

```
    cout.width(5);
  cout<<i;
  cout << "\n------------------------\n";
}
```

OUTPUT :

```
*********a
****1
```

4. **setf():** The setf method is used to set **various flags** for formatting output.

EXAMPLE :

```
        void IOS_setf()

{
  cout << "\n------------------------\n";
  //cout << "Implementing ios::setf\n\n";
  int val1=100,val2=200;
  cout.setf(ios::showpos);
  cout<<val1<<" "<<val2;
  cout << "\n------------------------\n";
}
```

OUTPUT ;

```
 Implementing ios::setf


+100 +200
```

5. **unsetf():** The unsetf method is used To remove the flag setting.

```
void IOS_unsetf()
{
  cout << "\n------------------------\n";
  cout << "Implementing ios::unsetf\n\n";
  cout.setf(ios::showpos|ios::showpoint);
```

```
    cout.unsetf(ios::showpos);
    cout<<200.0;
    cout << "\n------------------------\n";
}
```

OUTPUT :

```
Implementing ios::unsetf


200.000
```

Q- Exlain the file stream class.

Ans ;

There are following file stream class:

1) ofstream.
2) ifstream.
3) fstream.

1) Ofstream.

Ans :

"This class is generally used to write in a file".

- This class provides output operations.
- It is the file stream class.
- It is very important class in a file operation.
- It contains open() function with default output mode.
- Inherits the functions put(), write(), seekp() and tellp() functions from the ostream.

2) **Ifstream** :

" This class is generally used to read from the file"

- This class provides input operations.
- It is the file stream class.
- It is very important class in a file operation.

- It contains open() function with default input mode.
- Inherits the functions get(), getline(), read(), seekg() and tellg() functions from the istream.

3) Fstream :

" This class is generally used in both read and write operation.

- This class provides support for simultaneous input and output operations.
- It is the file stream class.
- It is very important class in a file operation.

- Inherits all the functions from istream and ostream classes through iostream.

Q- Explain the Basic file operation .

Ans : There are following basic file operation :

STEP 1-Naming a file
STEP 2-Opening a file
STEP 3-Writing data into the file
STEP 4-Reading data from the file
STEP 5-Closing a file.

1) Naming a file :

We can use the ofstream class to create a file and given the name it.

We can also use the open function to create a file.

Syntax :

Ofstream my_file.open("sum.txt");

In the above syntax,

- ofstream is the class name.
- my_file .open is the command to create new file.
- Sum.txt is the file name that we want to create.

2) Opening a file :

We can use the ofstream class to open a file.

We can also use the open function to open a file.

Syntax :

Ofstream my_file.open(“sum.txt”);

In the above syntax,

- ofstream is the class name.
- my_file .open is the command to open file.
- Sum.txt is the file name that we want to open.

Example :

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

using namespace std;

int main() {

   // opening a text file for writing
   ofstream my_file("example.txt");

   // close the file
   my_file.close();


   return 0;
}
```

3) Writing Data into a file :

- We can use the ofstream or fstream class to write in a file.
- First we can create or open the file.
- To write in a file, we can use the insertion operator "<<".

EXAMPLE :

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

int main() {
  // Create and open a text file
  ofstream MyFile("filename.txt");

  // Write to the file
  MyFile << "Files can be tricky, but it is fun enough!";

  // Close the file
  MyFile.close();
}
```

4) READING DATA FROM A FILE :

We can use the fstream or ifstream class for reading data from a file.

First, we can open a file for reading.

We can use the "GETLINE()" to read the file line by line and to print the content of the line.

Example :

```
string myText;

// Read from the text file
ifstream MyReadFile("filename.txt");

// Use a while loop together with the getline() function to read the file line by line
while (getline (MyReadFile, myText)) {
  // Output the text from the file
  cout << myText;
}

// Close the file
MyReadFile.close();
```

5) Closing a file :

We can use the fstream class to close a file.

We can use the close() function to close the file.

Example :

```
My_file.close();
```

Write a program to create file and write in it and read from it.

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

int main () {
  // Create a text file
  ofstream myfile("sum.txt");

  // Write to the file
  myfile << "HELLO"<<endl;
  myfile<<”HELLO HOW ARE YOU”<<endl;
  // Close the file
  myfile.close();

  // Create a text string, which is used to output the text file
  string line;

  // Read from the text file
  ifstream myfile1("sum.txt");
```

```
// Use a while loop together with the getline() function to read the file line by line

while (getline (myfile1, line)) {

  // Output the text from the file

  cout << line;

 }


 // Close the file

 Myfile1.close();

}
```

Q- Explain the Random Access File.

Ans :

In C++, a **random access file** refers to a file that allows you to read and write data at any where (position) within the file, without having to process the file sequentially.

This is achieved using file pointers and special file stream functions.

To work with random access files in C++, you generally use fstream (from the <fstream> library).

The function for random access is the seekg() and seekp() functions which is used to move the file pointer to specific locations in the file.

This is useful for efficiently handling large files, databases, or situations where you need to jump to a specific record or piece of data.

The operation of Random access file in c++ :

1) Open a file

   You can open a random access mode by using fstream class and specifying the ios::in, ios::out flag along with the ios::binary flag.

2) Move the file pointer :

We can move the file pointer using the seekp() and seekg() function.

Seekp() function is used for writing data.

Seekg() function is used for reading data.

3) Close a file :

We can close a file by using the file.close() function.

Example :

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


struct Student {
    int id;
    char name[50];
};
int main() {
    // Open a file in binary mode
    std::fstream file("students.dat", std::ios::in | std::ios::out | std::ios::binary);


    if (!file) {
        std::cerr << "Error opening file!" << std::endl;
        return 1;
    }
```

```
    // Writing to the file (adding a few student records)
    Student student1 = {1, "Alice"};
    Student student2 = {2, "Bob"};
    Student student3 = {3, "Charlie"};
    file.seekp(0, std::ios::beg); // Move to the beginning of the file
    file.write(reinterpret_cast<char*>(&student1), sizeof(Student));

    file.seekp(sizeof(Student), std::ios::beg); // Move to the second record
    file.write(reinterpret_cast<char*>(&student2), sizeof(Student));

    file.seekp(2 * sizeof(Student), std::ios::beg); // Move to the third record
    file.write(reinterpret_cast<char*>(&student3), sizeof(Student));

    // Reading from the file (reading a specific record)
    int recordToRead = 2; // We want to read the second student (Bob)
    file.seekg((recordToRead - 1) * sizeof(Student), std::ios::beg); // Move to the
second record
    Student readStudent;
    file.read(reinterpret_cast<char*>(&readStudent), sizeof(Student));
    std::cout << "Student ID: " << readStudent.id << "\n";
    std::cout << "Student Name: " << readStudent.name << "\n";
    file.close(); // Close the file
    return 0;
}
```

Q- Explain the Command Line Argument with example.

Ans : In a C++, command-line arguments are passed to the main function as parameters.

The syntax of the main function when accepting command-line arguments looks like this:

Syntax :

```
int main(int argc, char* argv[])
```

In above syntax ,

1) argc: ArgC stands for *argument count* and It is an integer that represents the number of command-line arguments (including the name of the program itself).
2) argv: Argv stands for *argument vector* and It is an array of strings (character pointers), where each element is a command-line argument.
   The first element (argv[0]) is the name of the program.

Example: Reading a file from command-line arguments

Here's a simple C++ program that accepts a file name as a command-line argument and opens the file:

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

int main(int argc, char* argv[])
{
   // Check if the user provided a file name
   if (argc != 2) {
      std::cerr << "Usage: " << argv[0] << " <filename>\n";
      return 1;
   }

   std::string filename = argv[1];

   // Open the file
   std::ifstream file(filename);
```

```
    if (!file) {
        std::cerr << "Error opening file: " << filename << '\n';
        return 1;
    }

    // File is successfully opened, now read its contents
    std::string line;
    while (std::getline(file, line)) {
        std::cout << line << '\n';  // Print the file content to the console
    }

    file.close();  // Close the file
    return 0;
}
```

How it works:

1. The program expects exactly one command-line argument: the file name.
2. It checks if the correct number of arguments (argc == 2), where the first is the program name and the second is the file name.
3. It opens the file specified by argv[1] (the second command-line argument).
4. If the file is opened successfully, it reads and prints the file's contents.

Example Usage:

Shell

```
$ ./myprogram myfile.txt
```

This will read the contents of myfile.txt and print them to the console.

If you don’t provide the correct number of arguments, it will display an error message like:

Bash

```
Usage: ./myprogram <filename>
```