File Handling(I/O) in C++

C++ File Handling

- As we know, at the time of execution, every program comes in main memory to execute.
- Main memory is volatile and the data would be lost once the program is terminated.
- If we need the same data again, we have to store the data in a file on the disk.
- A file is sequential stream of bytes ending with an *end-of-file* marker.

Types of File(s)

- Types of file supported by C++:
  - Text Files
  - Binary Files

Difference between text file and binary file

- Text file is human readable because everything is stored in terms of text.
- In binary file everything is written in terms of 0 and 1, therefore binary file is not human readable.

- A newline(\n) character is converted into the carriage return-linefeed combination before being written to the disk.
- In binary file, these conversions will not take place.

Difference between text file and binary file

- In text file, a special character, whose ASCII value is 26, is inserted after the last character in the file to mark the end of file.
- There is no such special character present in the binary mode files to mark the end of file

Table 1: ASCII Reference: Nonprintable Characters

Dec	Hex	Oct	Char	Comment
0	0	0	NUL	Null
1	1	1	SOH	Start of Heading
2	2	2	STX	Start of Text
3	3	3	ETX	End of Text
4	4	4	EOT	End of Transmission
5	5	5	ENQ	Enquiry
6	6	6	ACK	Acknowledge
7	7	7	BEL	Bell
8	8	10	BS	Backspace
9	9	11	TAB	Horizontal Tab
10	A	12	LF	Line Feed
11	B	13	VT	Vertical Tab
12	C	14	FF	Form Feed
13	D	15	CR	Carriage Return
14	E	16	SO	Shift Out
15	F	17	SI	Shift In
16	10	20	DLE	Data Link Escape
17	11	21	DC1	Device Control 1
18	12	22	DC2	Device Control 2
19	13	23	DC3	Device Control 3
20	14	24	DC4	Device Control 4
21	15	25	NAK	Negative Acknowledgement
22	16	26	SYN	Synchronous Idle
23	17	27	ETB	End of Transmission Block
24	18	30	CAN	Cancel
25	19	31	EM	End of Medium
26	1A	32	SUB	Substitute
27	1B	33	ESC	Escape
28	1C	34	FS	File Separator
29	1D	35	GS	Group Separator
30	1E	36	RS	Record Separator
31	1F	37	US	Unit Separator

The fstream.h Header File

- C++'s standard library called **fstream**, defines the following classes to support file handling.

- **ofstream class :** Provides methods for writing data into file. Such as, open(), put(), write(), seekp(), tellp(), close(), etc.

- **ifstream class :** Provides methods for reading data from file. Such as, open(),get(), read(), seekg(), tellg(), close(), etc.

- **fstream class :** Provides methods for both writing and reading data from file. The fstream class includes all the methods of ifstream and ofstream class.

Opening a file using open() member function

- The open() function takes file-name argument.
- The purpose of opening the file i.e, whether for reading or writing, depends on the object associated with open() function.

```
ofstream fout;
fout.open("filename");    // Open file for writing

ifstream fin;
fin.open("filename");     // Open file for reading

fstream f;
f.open("filename",mode);

// Using fstream, we can do both read and write, therefore,
// mode specifies the purpose for which the file is opened.
```

File Opening Modes

Mode	Purpose
ios::in	Open a text file for reading.
ios::out	Open a text file for writing. If it doesn't exist, it will be created.
ios::ate	Open a file and move the pointer at the end-of-file.
ios::app	Open a text file for appending. Data will be added at the end of the existing file. If file doesnt exist, it will be created.
ios::binary	Open file in a binary mode.
ios::nocreate	The file must already exist. If file doesn't exist, it will not create new file.
ios::trunc	If the file already exists, all the data will lost.

Reading and Writing into File

- We can read data from file and write data to file in four ways.

1. Reading or writing characters using get() and put() member functions.

2. Reading or writing formatted I/O using insertion operator (<<) and extraction operator (>>).

3. Reading or writing object using read() and write() member functions.

C++ put() function

- The put() function is member of ofstream class. It is used to write singal character into file

Syntax of put() function

```
ofstream obj;
obj.put(char ch);
```

```
#include<fstream.h>
#include<conio.h>

void main()
{
    ofstream fout;
    char ch;

    fout.open("demo.txt");          //Statement 1

    do                              //Statement 2
    {

        cin.get(ch);
        fout.put(ch);

    }while(ch!=EOF);

    fout.close();

    cout<<"\nData written successfully...";

}
```

Output :

```
Hello friends, my name is kumar.^Z
Data written successfully...
```

In this example, statement 1 will create a file named demo.txt in write mode. Statement 2 is do-while loop, which take characters from user and write the character to the file, until user press the ctrl+z to exit from the loop.

C++ get() function

- The get() function is member of ifstream class. It is used to read character form the file.
- The get() function reads a single character from the file and puts that value in ch.

Syntax of get() function

```
ifstream obj;
obj.get(char &ch);
```

```
#include<fstream.h>
#include<conio.h>

void main()
{
    ifstream fin;
    char ch;

    fin.open("demo.txt");              //Statement 1

    cout<<"\nData in file...";

    while(!fin.eof())                  //Statement 2
    {
        fin.get(ch);
        cout<<ch;
    }

    fin.close();

}
```

Output :

```
Data in file...
Hello friends, my name is kumar.
```

In this example, Statement 1 will open an existing file demo.txt in read mode and statement 2 will read all the characters one by one upto EOF(end-of-file) reached. The eof() function is member of ifsream class. It returns zero, if end-of-file reached and returns non-zero value, if any character found.

C++ Insertion operator (<<)

- Insertion operator (<<) is similar to fprintf() function available in C language.
- It is used in situation where we need to write characters, strings and integers in a single file or we can say, it is used to write mixed type in the file.

Syntax of Insertion operator (<<)

```
ofstream obj;
obj << variable-list;
```

operator (<<)

```
#include<fstream.h>
#include<conio.h>

void main()
{
    ofstream fout;

    int roll;
    char name[25];
    float marks;

    char ch;

    fout.open("demo.txt");                    //Statement  1

    do
    {
        cout<<"\n\nEnter Roll : ";
        cin>>roll;

        cout<<"\nEnter Name : ";
        cin>>name;

        cout<<"\nEnter Marks : ";
        cin>>marks;

        fout << roll << "  " << name << "  " << marks << "  ";//Statement  2

        cout<<"\n\nDo you want to add another data (y/n) : ";
        ch = getche();

    }while(ch=='y' || ch=='Y');

    cout<<"\nData written successfully...";

    fout.close();
}
```

Output :

```
Enter Roll : 1
Enter Name : Kumar
Enter Marks : 78.53

Do you want to add another data (y/n) : y

Enter Roll : 2
Enter Name : Sumit
Enter Marks : 89.62

Do you want to add another data (y/n) : n

Data written successfully...
```

In the above example, Statement 1 will open an existing file **demo.txt** in write mode and statement 2 will write all the data in file.

Note : We must insert blank space in file to separate each values, as shown in **statement 2**, because extraction operator terminates at blank space and consider next value as new value.

C++ Extraction operator (>>)

- The extraction operator (>>) is similar to fscanf() function available in C language. It is used to read mixed type form the file.

Syntax of Extraction operator (>>)

```
ifstream obj;
obj >> variable-list;
```

```
#include<fstream.h>
#include<conio.h>

void main()
{
      ifstream fin;

      int roll;
      char name[25];
      float marks;

      char ch;

      fin.open("demo.txt");          //Statement  1

      cout<<"\n\tData in file...\n";

      while(fin>>roll>>name>>marks)            //Statement  2
      {

         cout << "\n\t" << roll << "\t" << name << "\t" << marks;
      }

      fin.close();
}
```

Output :

```
      Data in file...

      1    Kumar       78.53
      2    Sumit       89.62
```

raction operator (>>)

In the above example, Statement 1 will open an existing file demo.txt in read mode and statement 2 will read all the data upto EOF(end-of-file) reached.

C++ write() function

- The write() function is used to write object or record (sequence of bytes) to the file. A record may be an array, structure or class.

Syntax of write() function

```
fstream fout;
fout.write( (char *) &obj, sizeof(obj) );
```

- The write() function takes two arguments.
  &obj : Initial byte of an object stored in memory.
  sizeof(obj) : size of object represents the total number of bytes to be written from initial byte.

C++ write() function

```
#include<fstream.h>
#include<conio.h>

class Student
{

    int roll;
    char name[25];
    float marks;

    void getdata()
    {
        cout<<"\n\nEnter Roll : ";
        cin>>roll;

        cout<<"\nEnter Name : ";
        cin>>name;

        cout<<"\nEnter Marks : ";
        cin>>marks;
    }

    public:
    void AddRecord()
    {

        fstream f;
        Student Stu;

        f.open("Student.dat",ios::app|ios::binary);

        Stu.getdata();

        f.write( (char *) &Stu, sizeof(Stu) );

        f.close();

    }
};
```

```
void main()
{

    Student S;
    char ch='n';

    do
    {
        S.AddRecord();

        cout<<"\n\nDo you want to add another data (y/n) : ";
        ch = getche();

    } while(ch=='y' || ch=='Y');

    cout<<"\nData written successfully...";
}
```

C++ write() function

Output :

```
Enter Roll : 1
Enter Name : Ashish
Enter Marks : 78.53

Do you want to add another data (y/n) : y

Enter Roll : 2
Enter Name : Kaushal
Enter Marks : 72.65

Do you want to add another data (y/n) : y

Enter Roll : 3
Enter Name : Vishwas
Enter Marks : 82.65

Do you want to add another data (y/n) : n

Data written successfully...
```

C++ read() function

- The read() function is used to read object (sequence of bytes) to the file.

Syntax of read() function

```
fstream fin;
fin.read( (char *) &obj, sizeof(obj) );
```

- The read() function takes two arguments.
  &obj : Initial byte of an object stored in file.
  sizeof(obj) : size of object represents the total number of bytes to be read from initial byte.
- The read() function returns NULL if no data read.

C++ read() function

```
#include<fstream.h>
#include<conio.h>

class Student
{

      int roll;
      char name[25];
      float marks;

      void putdata()
      {

         cout<<"\n\t"<<roll<<"\t"<<name<<"\t"<<marks;

      }

      public:

      void Display()
      {

            fstream f;
            Student Stu;

            f.open("Student.dat",ios::in|ios::binary);

            cout<<"\n\tRoll\tName\tMarks\n";

            while( (f.read((char*)&Stu,sizeof(Stu))) != NULL )
                  Stu.putdata();

            f.close();
      }
};
```

```
void main()
{

      Student S;

      S.Display();
}
```

Output :

Roll	Name	Marks
1	Ashish	78.53
2	Kaushal	72.65
3	Vishwas	82.65