UNIT-IV

File Handling in C++ Ⓒ .

* What is a file?

- The computer system stores programs and data permanently in secondary storage in the form of files.
- File is a collection of related data.
  (ie) programs (both source & object forms) and data items.

* File streams

- File operations in C++ are handled using file streams that acts as an interface between programs and files.

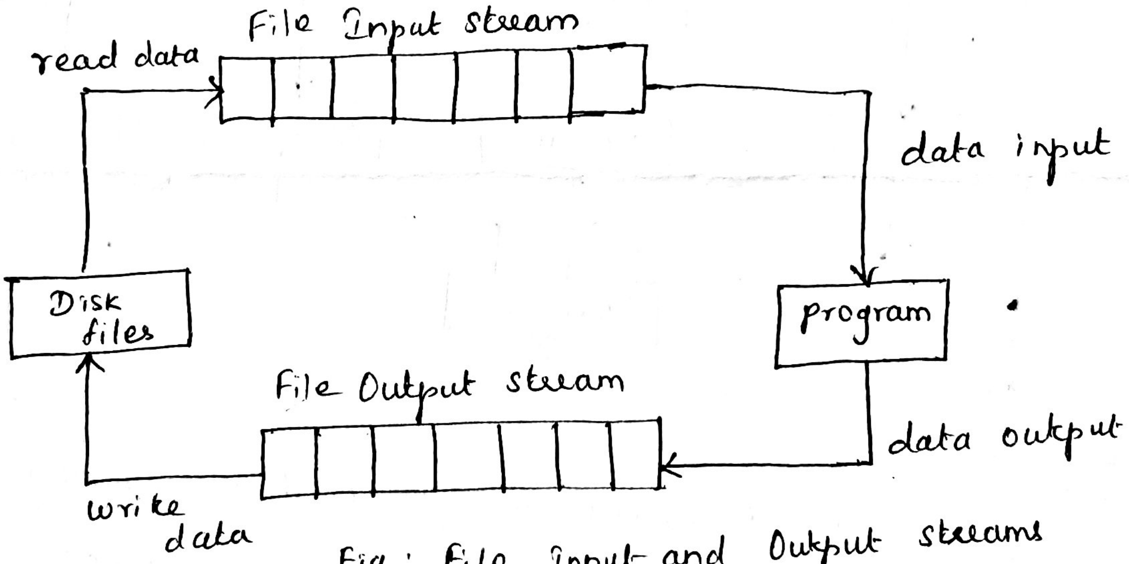


Fig : File Input and Output streams

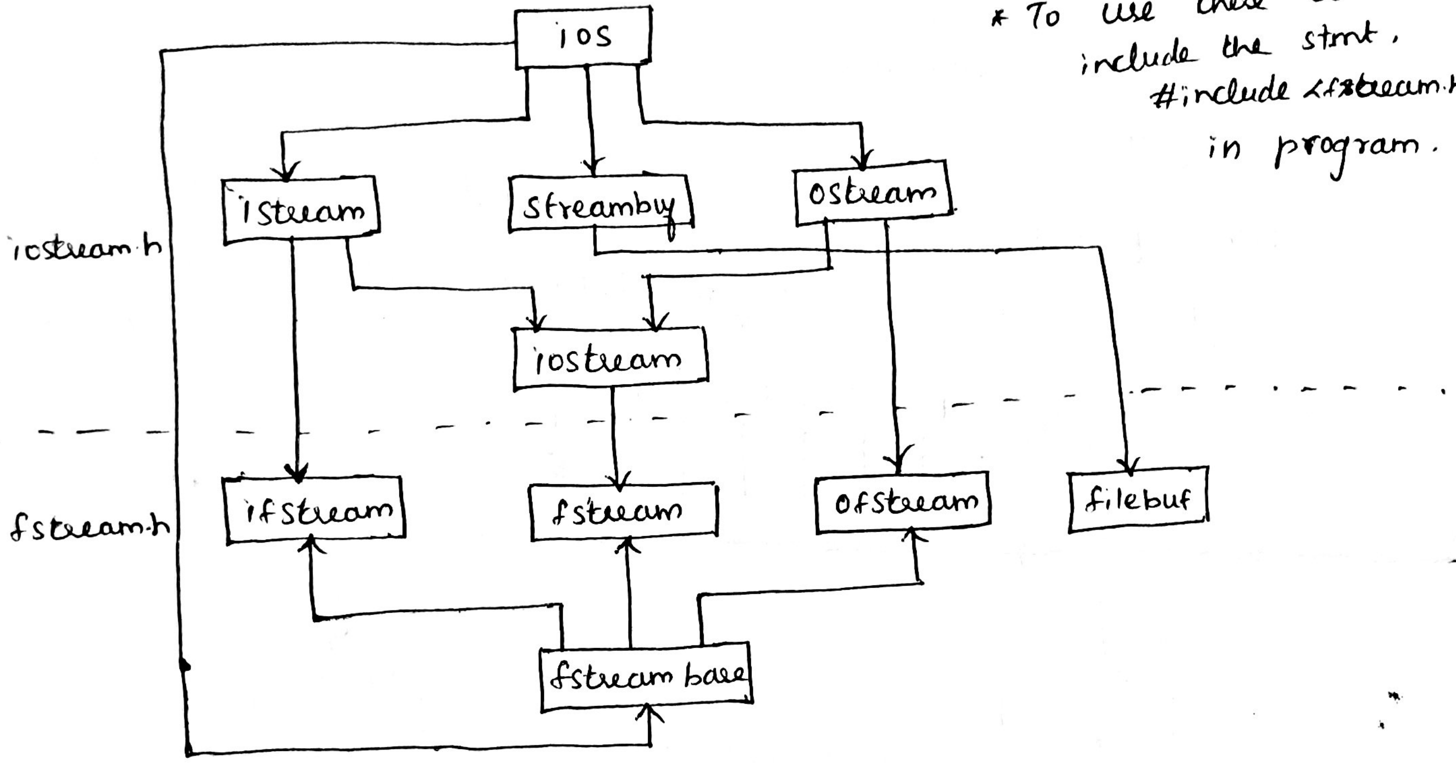
- Input stream supplies data to the program (or) reads data from file
- Output stream receives data from the program (or) writes data to file.

Hierarchy of file stream classes

* There are three classes for handling files
  - ifstream ↝ for handling input files
  - ofstream ↝ for handling output files.
  - fstream ↝ for handling files on which both i/p & o/p can be performed

* Their declarations exist in the header file fstream.h.

* To use these classes include the stmt,
#include <fstream.h>
in program.



* ifstream
  - Supports input operations. It contains open() with default input mode
  - inherits get(), getline(), read(), seekg() and tellg() functions from istream.

* Ofstream
  - Supports output operations. Contains open() with default output mode
  - inherits put(), write(), seekp(), tellp() functions from ostream.

fstream
- supports simultaneous input and output operations.
- inherits all functions from istream and ostream classes through iostream.

* filebuf
  - sets the file buffer to read and write.

* fstreambase
  - Provides operations common to the file streams.
  - serves as a base for fstream, ifstream and ofstream.

Steps for file manipulation

* There are 3 steps for file manipulation:
  (i) Opening a file
  (ii) Reading (or) writing a file
  (iii) Closing a file.

Opening a file :

* A file has to be opened before the read and write operations.
* To open a file,
  - create a file stream using ifstream (to read) or ofstream (to write)
  - link the stream to a file name.
* A file can be opened in two ways:
  (i) Using constructor function of the class (implicit)
  ↳ use only one file in the stream.

(Eg) ofstream fout ("student.txt");

↳ creates fout as an ofstream object that manages o/p stream.

↳ opens the file "Student.txt' and attaches it to o/p stream fout.

ifstream fin ("Student.txt");

↳ declares fin as an ifstream object and attaches it to file 'Student.txt' for reading.

(Eg) #include <iostream.h>

```
# include <fstream.h>
void main()
{
 char name [20];
 int rollno;
 Ofstream  fout ("Student.txt");
 cout<< "Enter Name and rollno\n");
 cin>> name >> rollno;
 fout << name << " " << rollno; // writes to file using ofstream obj.
 fout. close ();     // disconnects the file "Student.txt" from
                       o/p stream fout. But the
 ifstream fin ("Student.txt");
                       Object fout still exist.
 cout<< "Name and rollno:\n";
 fin >> name >> rollno;     // reads from file using ifstream obj.
 cout << name << " " << rollno;
 fin.close();
}
```

(ii) Opening a file explicitly using member function open()

- the function open() can be used to open multiple files using one stream object

Syntax:

file-Stream-class StreamObj ;
streamObj . open ("filename");

```
#include <iostream.h>
#include <fstream.h>
void main( )
{
 char name[20] ;
 int rollno;
 int mark1 , mark2;
 ofstream fout;
 fout.open("Student.txt");
 for(int i=0; i<2; i++ )
 {
 cout << "Enter name and roll no\n";
 cin >> name >> rollno;
 fout <<name <<" "<<rollno<<endl;
 }
 fout.close();   // first file should be closed before opening second file.
                 cox , a stream can be connected to
                 only one file at a time.
 fout.open("Marks.txt");
 for(int i = 0; i<2; i++ )
 {
 cout <<"Enter Mark1 and Mark2\n";
  cin>> mark1 >>mark2;
  fout<< mark1<<" "<<mark2 << endl;
 }
 fout.close ( );
 ifstream fin ;
 fin.open ("Student.txt");
 cout<<"Name and rollno:\n";
 while (1)
 {
  fin >> name >> rollno ;
  if (fin.eof())   // eof() is member fn of ios class. returns non-zero
                      if EOF is encountered.
     break;
  cout<< name << " " <<rollno <<endl;
 }
 fin.close ( );
 char line[50] ;
 int n = 50;
 fin.open ("Marks.txt");
 cout<< "Mark1 and Mark2 :\n";
 while(1)
 {
  fin.getline (line, n) ; //reads a line
  if (fin.eof ()) // checks EOF.
    break ;
  cout << line << endl;
 }
 fin.close ();
}
```

<u>o/p:</u>

```
Enter name and roll no
 aa  100
Enter name and roll no
 bb  200
Enter mark1 and mark2
   98  90
Enter mark1 and mark2
   70  80
Name and roll no
 aa  100
 bb  200
Mark1 and Mark2 :
 98 90
 70 80
```

Handling two files simultaneously

- Two or more files can be used simultaneously.
- To read from two files simultaneously, create two input streams for handling 2 i/p files.

(Eg)
```
#include <iostream.h>
#include <fstream.h>
void main()
{
char line[50];
int n=50;
ifstream fin1, fin2;
fin1.open ("student.txt");
fin2.open ("Marks.txt");
for (int i=0; i<2; i++)
 {
 cout << "Name and rollno:\n";
 fin1. getline (line, n);
 if (fin1.eof())
   break;
 cout << line << endl;
 cout << "Mark1 and Mark2 :\n";
 fin2.getline (line,n);
 if (fin2.eof())
    break;
 cout << line << endl;
 }
fin1.close();
fin2.close();
}
```

O/P:
```
Name and roll no:
aa  100
Mark1 and Mark2
98  90
Name and roll no:
bb  200
Mark1 and Mark2
70  80
```

open() - File modes

Syntax :
Stream_object . open ("filename", mode);

mode → specifies the purpose for which the file is opened.

Parameter	Meaning
ios::in	Opens file for reading.
ios :: out	Opens file for writing .
ios :: app	Append to EOF .
ios :: trunc	Delete contents of file if it exists .
ios :: ate	go to end-of file on opening .

Note: * Mode parameters are not required for ifstream and ofstream objects .

* Mode can combine two or more parameters using bitwise OR operator (|)

(Eg) fstream fout ;
fout . open ("data.txt", ios :: in | ios :: out);

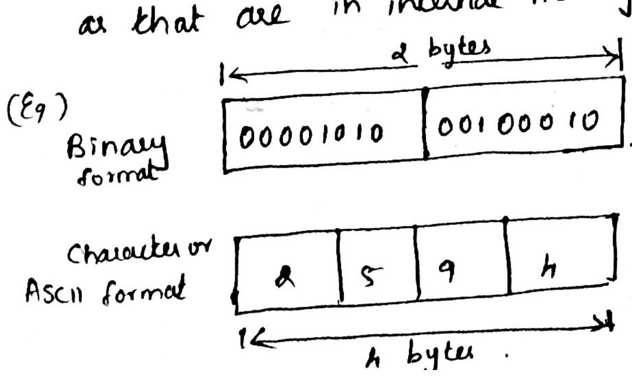
Function for Manipulation of File pointers.

seek g( )	- Moves get pointer (input) to specified location.
seekp()	- Moves put pointer (output) to specified location .
tell g()	- gives current position of get pointer .
tellp()	- gives current position of put pointer.

(Eg) fout . seekg(10); → moves file pointer to byte number 10 .

read and write functions .
- handles data in binary form as that are in internal memory .

Syntax:
fin. read ((char *) &v , sizeof(v));
fout . write ((char *) &v, sizeof (v));
↓
address of variable is cast to type char*



for storing no. 2594

→ * Accurate
* Must faster to store & retrieve
* No conversions

Reading and writing a class object

* C ++ supports features for writing and reading objects from files.
    - This is done using read () and write ( ) binary input and output functions.
* Only data members can be written to disk file and not member function.

(Eg)
```
// include <iostream.h>
# include < fstream.h >

class item
{
  char name [10] ;
  int code ;
 public :
   void getdata ( );
   void writedata( ) ;
 } ;
void item :: getdata ( )
 {
 cout << "Enter name :";
 cin >> name ;
 cout << " Enter code :";
 cout >> code ;
 }
void item :: writedata ( )
 {
 cout << name << " " << code ;
 }
```

```
void main()
{
 item it[2] ;
 fstream file ;
 file.open ("Stock.txt" , ios::in
                         |ios :: out);

 cout << "Enter details \n");
 for (int i =0; i<2; i++)
 {
   it[i]. getdata ( ) ;
 file.write ((char *)  &it[i],
              sizeof (it[i]));
 }
 file.seekg(0) ; //reset to start.
 cout << "\n Output \n";
 for (int i =0 ; i<2 ; i++ )
 {
 file.read ((char *) & it [i],
             sizeof (it[i]));
  it [i] . writedata ( ) ;
 }
 file.close ( ) ;
}
```

Address of variable is cast to type char*