1. Introduction
   - ➔ File : Sequence of stream.
   - ➔ A file contains the information which is stored under a specific name on a storage device.
   - ➔ Stream represents sequence of bytes. Stream represents the flow of data.
   - ➔ Different stream is used to represent different flow of data
     - i. Input Stream: Stream that supplies data to the program.
     - ii. Output stream : Output Stream receives data from the program
     - iii.
   - ➔ Functions of File Stream Class

Class	Functions
filebuf	It sets the file buffers to read and write. It contains close() and open() member functions.
ifstream	Provides Input operations. Supports get(),getline(),read() , seekg() and tellg().
ofstream	It provides output operations. It inherits put() and write() functions along with functions supporting random access(seekp() and tellp()
fstream	It is an input-output file stream class. It support for simultaneous input and output operations. It inherits all the functions from istream and ostream classes through iostream class

- ➔ A file can store information in two ways:

i. Text File : It stores the information in ASCII characters. Each line of text is terminated with a special character called as delimiter ie End Of Line (EOL).

ii. Binary File : A binary file contains information is in the same format in which the information is held in memory. Delimiter is not used. Binary files are faster and easier to program compared to text files.

➔ A file can be opened in two ways:

i. By using the constructor

ii. By using the open() function

o **the constructor method:**
This will use default streams for file input or output. This method is preferred when file is opened in input or output mode only.
Example :
ofstream file(“student.dat”); or ifstream file(“student.dat”);

o **the open() member function of the stream**. It will preferred when file is opened in various file modes such as ios::in, ios::out, ios::app,
Example:
ios::ate.**file.open(“book.dat”, ios::in | ios::out | ios::binary);**

File modes:
• **ios::out** It open file in output mode (i.e write mode) and place the file pointer in beginning, if file already exist it will overwrite the file.
• **ios::in** It open file in input mode(read mode) and permit reading from the file.
• **ios::app** It open the file in write mode, and place file pointer at the end of file i.e to add new contents and retains previous contents. If file does not
exist it will create a new file.
• **ios::ate** It open the file in write or read mode, and place file pointer at the end of file i.e input/ output operations can performed anywhere in the file.
• **ios::trunc** It truncates the existing file (empties the file).
• **ios::nocreate** If file does not exist this file mode ensures that no file is
created and open() fails.
• **ios::noreplace** If file does not exist, a new file gets created but if the file
already exists, the open() fails.
• **ios::binary** Opens a file in binary mode.

eof(): This function determines the end-of-file by returning true(non-zero) for end of file otherwise returning false(zero).
close(): This function terminates the connection between the file and stream associated with it.
Stream_object.close(); e.g file.close();
Text File functions:
Char I/O :
get() – read a single character from text file and store in a buffer. e.g file.get(ch);
put() - writing a single character in textfile e.g. file.put(ch);
getline() - read a line of text from text file store in a buffer. e.g file.getline(s,80);

We can also use file>>ch for reading and file<<ch writing in text file. But >> operator does not accept white spaces.
Binary file functions:

read()- read a block of binary data or reads a fixed number of bytes from the specified stream and store in a buffer.
Syntax : Stream_object.read((char *)& Object, sizeof(Object));
e.g file.read((char *)&s, sizeof(s));

write() – write a block of binary data or writes fixed number of bytes from a specific memory location to the specified stream.
Syntax : Stream_object.write((char *)& Object, sizeof(Object));
e.g file.write((char *)&s, sizeof(s));
Note:
Both functions take two arguments.
• The first is the address of variable, and the second is the length of that variable in bytes. The
address of variable must be type cast to type char*(pointer to character type)
• The data written to a file using write( ) can only be read accurately using read( ).

File Pointer: The file pointer indicates the position in the file at which the next input/output is to
occur.
Moving the file pointer in a file for various operations viz modification, deletion , searching
etc. Following functions are used:

seekg(): It places the file pointer to the specified position in input mode of file.
e.g file.seekg(p,ios::beg); or file.seekg(-p,ios::end), or file.seekg(p,ios::cur)
i.e to move to p byte position from beginning, end or current position.

seekp(): It places the file pointer to the specified position in output mode of file.
e.g file.seekp(p,ios::beg); or file.seekp(-p,ios::end), or file.seekp(p,ios::cur)
i.e to move to p byte position from beginning, end or current position.

tellg(): This function returns the current working position of the file pointer in the input mode.
e.g int p=file.tellg();
tellp(): This function returns the current working position of the file pointer in the output mode.
e.f int p=file.tellp();

Steps To Create A File

- Declare an object of the desired file stream class(ifstream, ofstream, or fstream)
- Open the required file to be processed using constructor or open function.
- Process the file.
- Close the file stream using the object of file stream.

General program structure used for creating a Text File

To create a text file using strings I/O

```
#include<fstream.h>                **//header file for file operations**
void main()
{
char s[80], ch;
ofstream file("myfile.txt"); **//open myfile.txt in default output mode**
do
{ cout<<" enter line of text";
gets(s);                     **//standard input**
file<<s;                     **// write in a file myfile.txt**
cout<<" more input y/n";
cin>>ch;
}while(ch!='n'||ch!='N');
file.close();
}                            **//end of main**
```

To create a text file using characters I/O

```
#include<fstream.h>                //header file for file operations
void main()
{
char ch;
ofstream file("myfile.txt");       //open myfile.txt in default output mode
do{

        ch=getche();
        if (ch==13)                //check if character is enter key
        cout<<' ';
        else
        file<<ch;                  // write a character in text file 'myfile.txt '
        }
 while(ch!=27);              // check for escape key
file.close();
}                            //end of main
```

Text files in input mode:

To read content of 'myfile.txt' and display it on monitor.

```
#include<fstream.h>                          //header file for file operations
void main()
{
char ch;
ifstream file("myfile.txt");                    //open myfile.txt in default input mode
while(file)
{
 file.get(ch)                            // read a character from text file 'myfile.txt'
cout<<ch;                               // write a character in text file 'myfile.txt '
}
file.close();
}                                       //end of main
```

Binary file using Objects and other file operations:

Consider the following class declaration then write c++ function for following file operations
viz create_file, read_file, add new records, modify record, delete a record, search for a record.

```
        #include<iostream.h>
        class student
        {
                 int rno;
                char name[30];
                int age;
               public:
              void input()
                {
                   cout<<" enter roll no";
                   cin>>rno;
                   cout<<" enter name ";
                   gets(name);
                   cout<<" enter age";
                   cin>>age;
               }

           void output()
           {
               cout<< " roll no:"<<rno;
               cout<< " name :"<<name;
               cout<< " age:"<<age;
          }
          int getrno()
          {
            return rno;
           }
        };
```

```
void create_file()
{
ofstream fout;
char ch;
fout.open(“student”, ios::out | ios:: binary);

clrscr();
student s;
if(!fout)
{cout<<“File can’t be opened”;
break;
}
do
{
s.input();
cout<<” more record y/n”;
cin>>ch;
}while(ch!=’n’ || ch!=’N’);
fout.close();
}
void read_file()
{
ifstream fin;
student s;
fin.open(“student.dat”,ios::in | ios:: binary);
fin.read((char *) &s,sizeof(student));
while(file)
{
s.output();
cout<< “ ”;
fin.read((char *) & s,sizeof(student));
}
fin.close();
}
void modify_record()
{ student s;
fstream file;
file.open(“student.dat”,ios::in|ios::out|ios::ate|ios::binary);
int r,pos=-1;
cout<<” enter the rollo no of student whom data to be modified”;
cin>>r;
file.read((char *)&s,sizeof(s));
while(file)
{
if (r==s.getrno())
{c
out<<” record is “;
s.output();
pos =file.tellg()-size(s);
break;
}
file.read((char *)&s,sizeof(s));
}
if(pos>-1)
{c
out<< “ enter new record”;
s.input();
file.seekp(pos,ios::beg);
file.write((char *)&s,sizeof(s));
cout<< “ record modified successfully”;
}
else
cout<< “ record not exist”;
}
```

```
void delete_record()
{f
stream file(“student.dat”, ios::in|ios::binary);
fstream newfile(“newstu.dat”,ios::out|ios::binary);
student s;
cout<<” enter the rollno no of student whom record to be deleted”;
cin>>r;
file.read((char *)&s,sizeof(s));
while(file)
{
if (r!=s.getrno())
{
newfile.write((char *)&s,sizeof(s));
}
file.read((char *)&s,sizeof(s));
}
file.close();
newfile.close();
}
void search_record()
{
student s;
fstream file;
file.open(“student.dat”,ios::in|os::binary);
int r,flag=-1;
cout<<” enter the rollo no of student whom record to be searched”;
cin>>r;
file.read((char *)&s,sizeof(s));
while(file)
{
if (r==s.getrno())
{c
out<<” record is “;
s.output();
flag=1;
break;
}
file.read((char *)&s,sizeof(s));
}
if(flag==1)
cout<< “ search successfull”;
else
cout<< “ search unsuccessfull”;
file.close();
}
```