INTRODUCTION

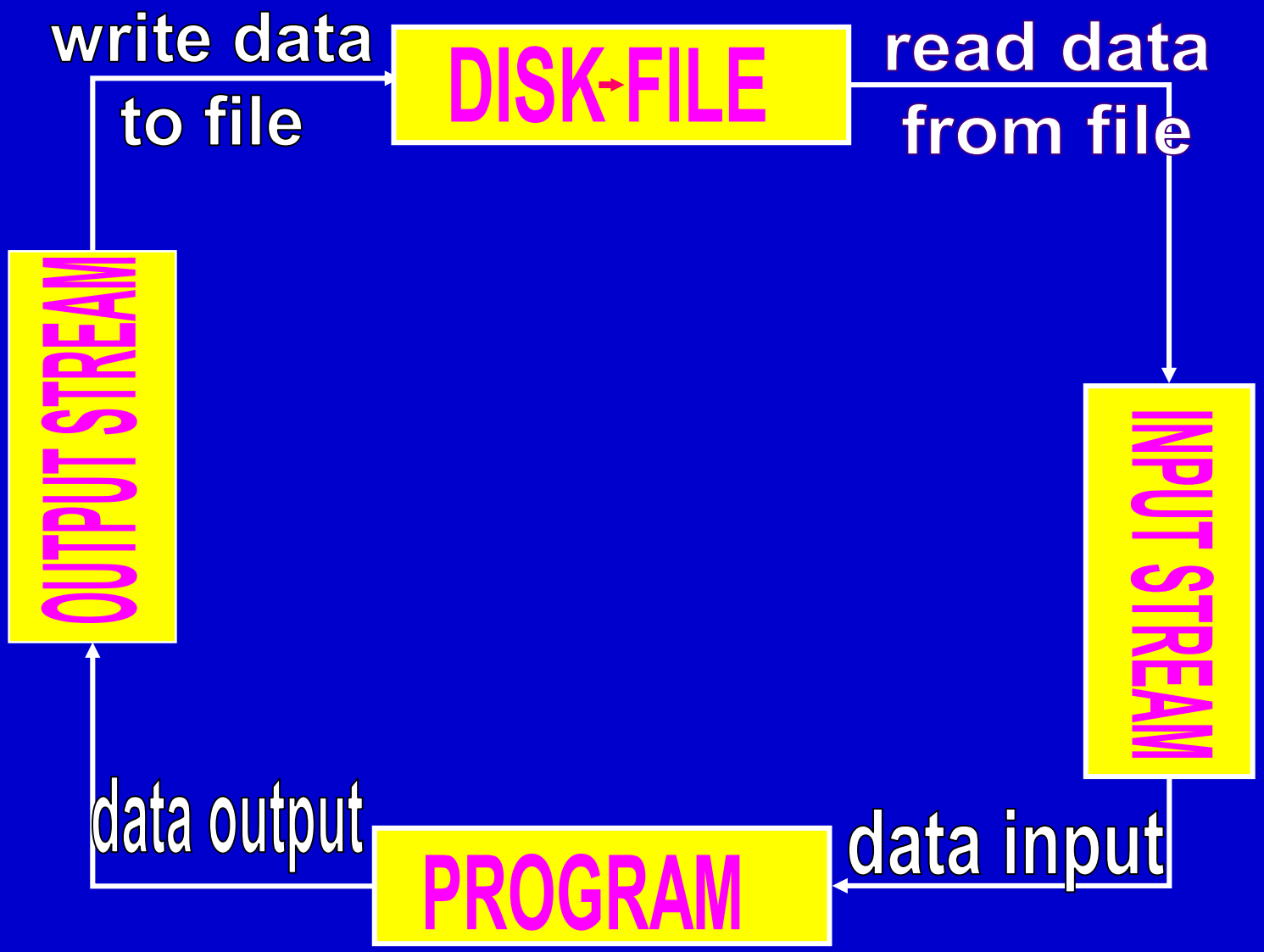
- Computer programs are associated to work with files as it helps in storing data & information permanently.
- File - itself a bunch of bytes stored on some storage devices.
- In C++ this is achieved through a component header file called *fstream.h*
- The I/O library manages two aspects- as interface and for transfer of data.
- The library predefine a set of operations for all file related handling through certain classes.

THE *FSTREAM.H* HEADER FILE

- A stream is a general term used to name flow of data.
- Streams act as an interface between files and programs.
- A Stream is sequence of bytes.
- They represent as a sequence of bytes and deals with the flow of data.
- Every stream is associated with a class having member functions and operations for a particular kind of data flow.

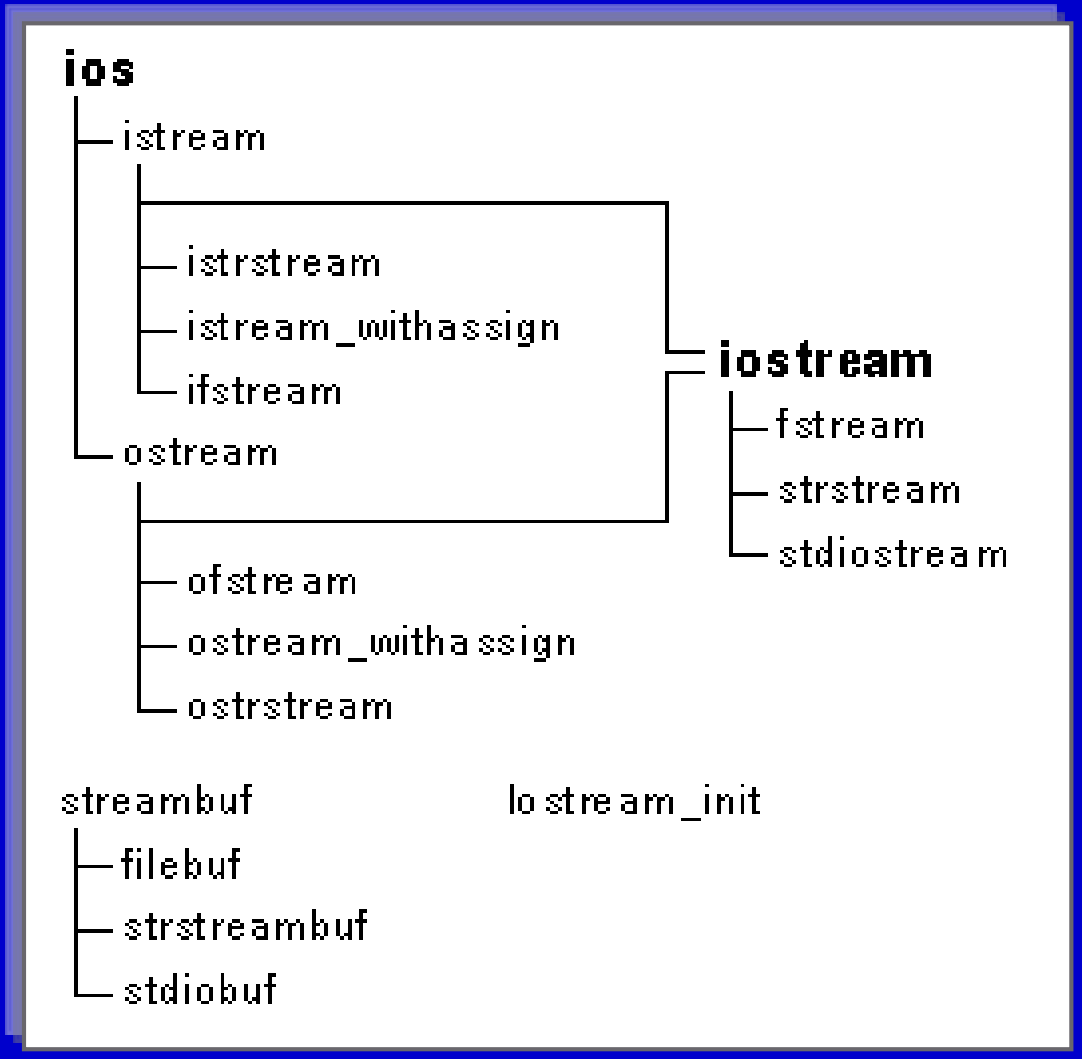
- File → Program (Input stream) - reads
- Program → File (Output stream) – write

- All designed into fstream.h and hence needs to be included in all file handling programs.
- Diagrammatically as shown in next slide


write data
to file
DISK FILE
read data
from file
OUTPUT STREAM
INPUT STREAM
data output
PROGRAM
data input

File Handling Classes

Hierarchy Diagram



FUNCTIONS OF FILE STREAM CLASSES

- **filebuf** – It sets the buffer to read and write, it contains close() and open() member functions on it.
- **fstreambase** – this is the base class for fstream and, ifstream and ofstream classes. therefore it provides the common function to these classes. It also contains open() and close() functions.
- **ifstream** – Being input class it provides input operations it inherits the functions get( ), getline( ), read( ), and random access functions seekg( ) and tellg( ) functions.
- **ofstream –** Being output class it provides output operations it inherits put( ), write( ) and random access functions seekp( ) and tellp( ) functions.
- **fstream** – it is an i/o class stream, it provides simultaneous input and output operations.

FILE TYPES

- ✓A File can be stored in two ways
- ✓**Text File**
- ✓**Binary File**

Text Files : Stores information in ASCII characters. In text file each line of text is terminated by with special character known as EOL (End of Line) In text file some translations takes place when this EOL character is read or written.

Binary File: it contains the information in the same format as it is held in the memory. In binary file there is no delimiter for a line. Also no translation occur in binary file. As a result binary files are faster and easier for program to read and write.

FILE MODES

- **WHAT IS FILE MODE?**
- **The File Mode describes how a file is to be used ; to read from it, write to it, to append and so on**

Syntax

Stream_object.open("filename",mode);

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.

FILE MODES

- **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.

Closing a File

- A File is closed by disconnecting it with the stream it is associated with. The close( ) function is used to accomplish this task.

Syntax:

```
Stream_object.close( );
```

Example :

```
fout.close();
```

Steps To Create A File

1. Declare an object of the desired file stream class(ifstream, ofstream, or fstream)

2. Open the required file to be processed using constructor or open function.

3. Process the file.

4. Close the file stream using the object of file stream.

eof ( ) Function

This function determines the end-of-file by returning true(non-zero) for end of file otherwise returning false(zero).

Syntax

Stream_object.eof( );

Example :

fout.eof( );

Text File Functions

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.

Program 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<<"\n enter line of text";
gets(s); //standard input
file<<s; // write in a file myfile.txt
cout<<"\n more input y/n";
cin>>ch;
}while(ch!='n'||ch!='N');
file.close();
} //end of main
```

Program 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
```

CBSE QUESTION PATTERN

RELATED

TO

TEXT FILES

2/3 Marks QNO 4 (b)

1. Write a function in C++ to count the number of uppercase alphabets present in a text file “BOOK.txt”

2. Write a function in C++ to count the number of alphabets present in a text file “BOOK.txt”

3. Write a function in C++ to count the number of digits present in a text file “BOOK.txt”

4. Write a function in C++ to count the number of white spaces present in a text file “BOOK.txt”

5. Write a function in C++ to count the number of vowels present in a text file “BOOK.txt”

6. Assume a text file “Test.txt” is already created. Using this file, write a function to create three files “LOWER.TXT” which contains all the lowercase vowels and “UPPER.TXT” which contains all the uppercase vowels and “DIGIT.TXT” which contains all digits.

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));

Binary File Functions

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( ).

Program to create a binary file ‘student.dat’ using structure.

```
#include<fstream.h>
struct student
{
char name[15];
float percent;
};
void main()
{
ofstream fout;
char ch;
fout.open(“student.dat”, ios::out | ios:: binary);
clrscr();
student s;
if(!fout)
{
cout<<“File can’t be opened”;
exit(0);
}
```

```
do
{ cout<<”\n
enter name of student”;
gets(s);
cout<<”\n enter percentage”;
cin>>percent;
fout.write((char *)&s,sizeof(s)); // writing a record in a student.dat file
cout<<”\n more record y/n”;
cin>>ch;
}while(ch!=’n’ || ch!=’N’);
fout.close();
}
```

Program to read a binary file ‘student.dat’ display records on monitor.

```
#include<fstream.h>
struct student
{
char name[15];
float percent;
};
void main()
{
ifstream fin;
student s;
fin.open(“student.dat”,ios::in | ios:: binary);
fin.read((char *) &s, sizeof(student)); //read a record from file ‘student.dat’
```

CONTD....

```
while(file)
{
cout<<s.name;
cout<<“\n has the percent: ”<<s.percent;
fin.read((char *) &s, sizeof(student));
}
fin.close();
}
```

CBSE QUESTION PATTERN

RELATED

TO

BINARY FILES
3 MARKS
QNO 4 (c)

QNO 4 (C) Write a function in c++ to search for details (Phoneno and Calls) of those Phones which have more than 800 calls from binary file "phones.dat". Assuming that this binary file contains records/ objects of class Phone, which is defined below. CBSE 2012

```
class Phone
{
    Char Phoneno[10]; int Calls;
    public:
    void Get() {gets(Phoneno); cin>>Calls;}
    void Billing() { cout<<Phoneno<< "#"<<Calls<<endl;}
    int GetCalls() {return Calls;}
};
```

Ans :

```
void Search()
{
Phone P;
fstream fin;
fin.open( “Phone.dat”, ios::binary| ios::in);
while(fin.read((char *)&P, sizeof(P)))
{
if(p.GetCalls() >800)
p.Billing();
}
Fin.close(); //ignore
}};
```

Write a function in C++ to add new objects at the bottom of a binary file “STUDENT.DAT”, assuming the binary file is containing the objects of the following class.

```
class STUD
{
    int Rno;
    char Name[20];
    public:
    void Enter()
    {cin>>Rno;gets(Name);}
    void Display(){cout<<Rno<<Name<<endl;}
};
```

Ans.

```
 void searchbook(int bookno)
     {ifstream ifile(“BOOK.DAT”,ios::in|ios::binary);
     if(!ifile)
     {cout<<”could not open BOOK.DAT file”; exit(-1);}
     else
     {BOOK b; int found=0;
     while(ifile.read((char *)&b, sizeof(b)))
     {if(b.RBno()==bookno)
          {b.Display(); found=1; break;}
     }
     if(! found)
     cout<<”record is not found “;
     ifile.close();
     }
}
```

Given a binary file PHONE.DAT, containing records of the following class type

```
class Phonlist
{
    char name[20];
    char address[30];
    char areacode[5];
    char Phoneno[15];
    public:
    void Register()
    void Show();
    void CheckCode(char AC[])
    {return(strcmp(areacode,AC);
};
```

Write a function TRANSFER( ) in C++, that would copy all those records which are having areacode as “DEL” from PHONE.DAT to PHONBACK.DAT.

Ans

```
 void TRANSFER()
{
    fstream File1,File2;
    Phonelist P;
    File1.open(“PHONE.DAT”, ios::binary|ios::in);
    File2.open(“PHONEBACK.DAT”, ios::binary|ios::OUT)
    while(File1.read((char *)&P, sizeof(P)))
    { if( p.CheckCode( “DEL”))
    File2.write((char *)&P,sizeof(P)); }
    File1.close();
    File2.close();
}
```

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.

File Pointer

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();

CBSE QUESTION PATTERN

RELATED

TO

FILE POINTER

1 MARK

QNO 4 (a)

4(a) Observe the program segment carefully and answer the question that follows:

```
class stock
{
int Ino, Qty; Char Item[20];
public:
void Enter() { cin>>Ino; gets(Item); cin>>Qty;}
void issue(int Q) { Qty+=0;}
void Purchase(int Q) {Qty-=Q;}
int GetIno() { return Ino;}
};
```

```
void PurchaseItem(int Pino, int PQty)
{ fstream File;
File.open("stock.dat", ios::binary|ios::in|ios::out);
Stock s;
int success=0;
while(success= = 0 && File.read((char *)&s,sizeof(s)))
{
If(Pino= = ss.GetIno())
{
s.Purchase(PQty);
_____ // statement 1
_____ // statement 2
Success++;
}
}
```

```
if (success = =1)
cout<< “Purchase Updated”<<endl;
else
cout<< “Wrong Item No”<<endl;
File.close() ;
}
```

Ans

i) Statement 1 to position the file pointer to the appropriate place so that the data updation is done for the required item.

```
File.seekp(File.tellg()-sizeof(stock);
OR
File.seekp(-sizeof(stock),ios::cur);
```

ii) Staement 2 to perform write operation so that the updation is done in the binary file.

```
File.write((char *)&s, sizeof(s)); OR
File.write((char *)&s, sizeof(stock));
```