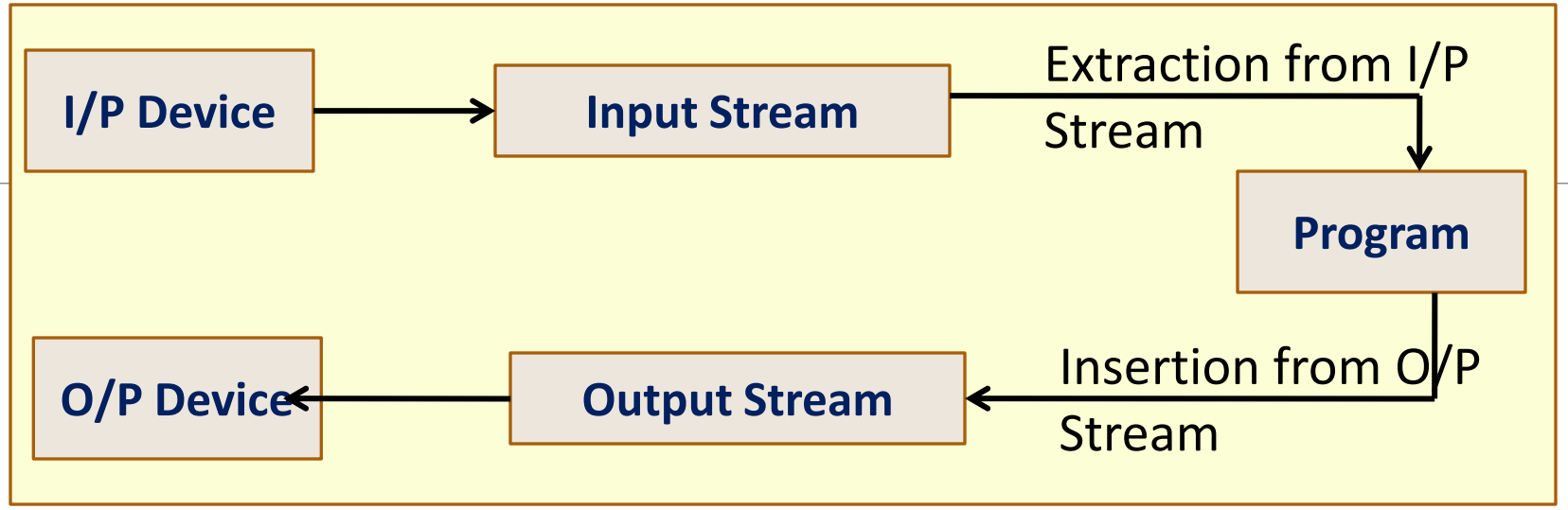
I/O and File Handling



Data Stream- Console IO

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

Unformatted IO Operations

>> and << are overloaded for all built-in data types in corresponding istream and ostream classes

Use get()/put() to fetch/display a character including blank space, tab and newline character

Use cin.get();

Use cout.put(ch);

Use getline() / write() to read / write a whole line of text

Use cin.getline(line, size);

Use cout.write(line, size);

Formatted IO Operations

Using ios class functions and flags

- width() specify field width
- precision() specify number of digits after decimal point
- fill() specify character to be used to fill unused field
- setf() used to set flags for say left justification / right justification
- unsetf() clear the set flags

Formatted IO Operations

Using Manipulators-

- setw() specify field width
- setprecision() specify number of digits after decimal point
- setfill() specify character to be used to fill unused field
- setiosflags() used to set flags for say left justification / right justification
- resetiosflags() clear the set flags

include <iomanip.h>

Using setf()

Syntax : cout.setf (arg1, arg2);

- arg1 is known as formatting flag of ios class
- arg2 is known as bitfield, specifies the group to which formatting flag belongs

Formatting Flags	Bit field
ios::left	adjustfield
ios::right	adjustfield
ios::scientific	floatfield
ios:fixed	floatfield
ios::dec	basefield
ios::oct	basefield
ios::hex	basefield

few more formatted flags for which no bitfield is required

Formatted Flags	Description
ios::showpos	print + before positive numbers
ios::showpoint	show trailing decimal point and zeros
ios::uppercase	use uppercase letters for hex output

File

A file is a collection on information, usually stored on a computer's disk. Information can be saved to files and then later reused.

File Name-

- All files are assigned a name that is used for identification purposes by the operating system and the user.

File Name and Extension	File Contents
MYPROG.BAS	BASIC program
MENU.BAT	DOS Batch File
INSTALL.DOC	Documentation File
CRUNCH.EXE	Executable File
BOB.HTML	HTML (Hypertext Markup Language) File
3DMODEL.JAVA	Java program or applet
INVENT.OBJ	Object File
PROG1.PRJ	Borland C++ Project File
ANSI.SYS	System Device Driver
README.TXT	Text File

File Handling Steps

Using a file in a program is a simple three-step process

- The file must be opened. If the file does not yet exits, opening it means creating it.
- Information is then saved to the file, read from the file, or both.
- When the program is finished using the file, the file must be closed.

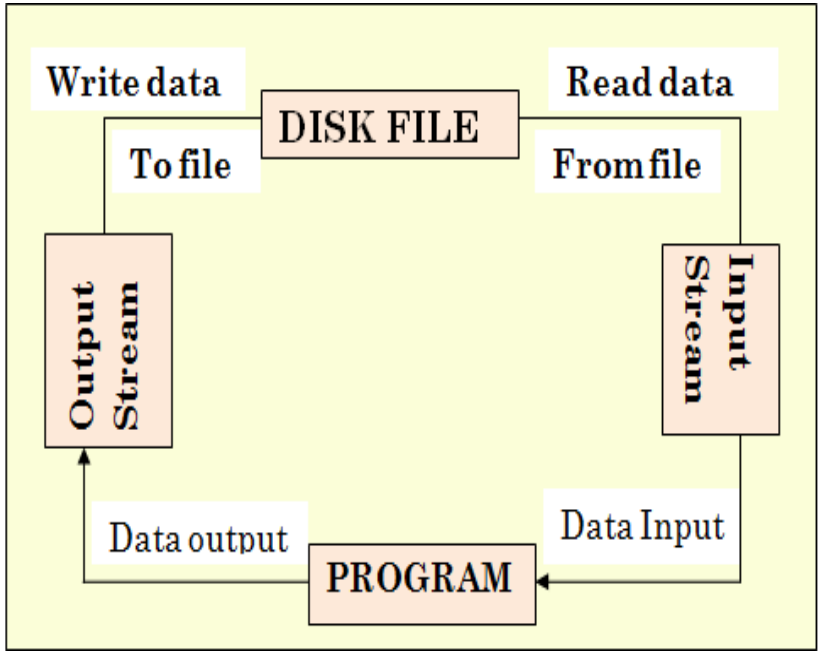
Using Input / Output Stream

Streams act as an interface between files and programs. In C++ . A stream is used to refer to the flow of data from a particular device to the program's variables.

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.


Write data
DISK FILE
Read data
To file
From file
Output Stream
Input Stream
Data output
Data Input
PROGRAM

Using Input / Output File

stream - a sequence of characters

- interactive (iostream)
  - **istream** - input stream associated with **keyboard.**
  - **ostream** - output stream associated with **display.**
- file (fstream)
  - **ifstream** - defines new input stream (normally associated with a file).
  - **ofstream** - defines new output stream (normally associated with a file).

- Stream of bytes to do input and output to different devices.
- Stream is the basic concepts which can be attached to files, strings, console and other devices.
- User can also create their own stream to cater specific device or user defined class.

Opening and Closing File

Before data can be written to or read from a file, the file must be opened.

- ifstream inputFile;

inputFile.open("customer.dat");//opens file in read mode

- ofstream outputFile;

outputFile.open("customer.dat");//opens file in write mode

- fstream iofile;

iofile.open("customer.dat", filemode);// opens file in specified mode

Closing File- A file should be closed when a program is finished using it

- inputFile.close();
- outputFile.close();
- Iofile.close();

File Modes

ios::app	Append mode. If the file already exists, its contents are preserved and all output is written to the end of the file. By default, this flag causes the file to be created if it does not exist.
ios::ate	If the file already exists, the program goes directly to the end of it. Output may be written anywhere in the file.
ios::binary	Binary mode. When a file is opened in binary mode, information is written to or read from it in pure binary format. (The default mode is text.)
ios::in	Input mode. Information will be read from the file. If the file does not exist, it will not be created and the open function will fail.
ios::nocreate	If the file does not already exist, this flag will cause the open function to fail. (The file will not be created.)
ios::noreplace	If the file already exists, this flag will cause the open function to fail. (The existing file will not be opened.)
ios::out	Output mode. Information will be written to the file. By default, the file's contents will be deleted if it already exists.
ios::trunc	If the file already exists, its contents will be deleted (truncated). This is the default mode used by ios::out.

Writing/Reading Data From/To File

Writing data to file-

- The stream insertion operator (<<) may be used to write information to a file.

  outputFile << “I love C++ programming !”

- File output may be formatted the same way as screen output.

Reading Data From File-

- The stream extraction operator (>>) may be used to read information from a file.

End of File Detetcion-

- The eof() member function reports when the end of a file has been encountered.

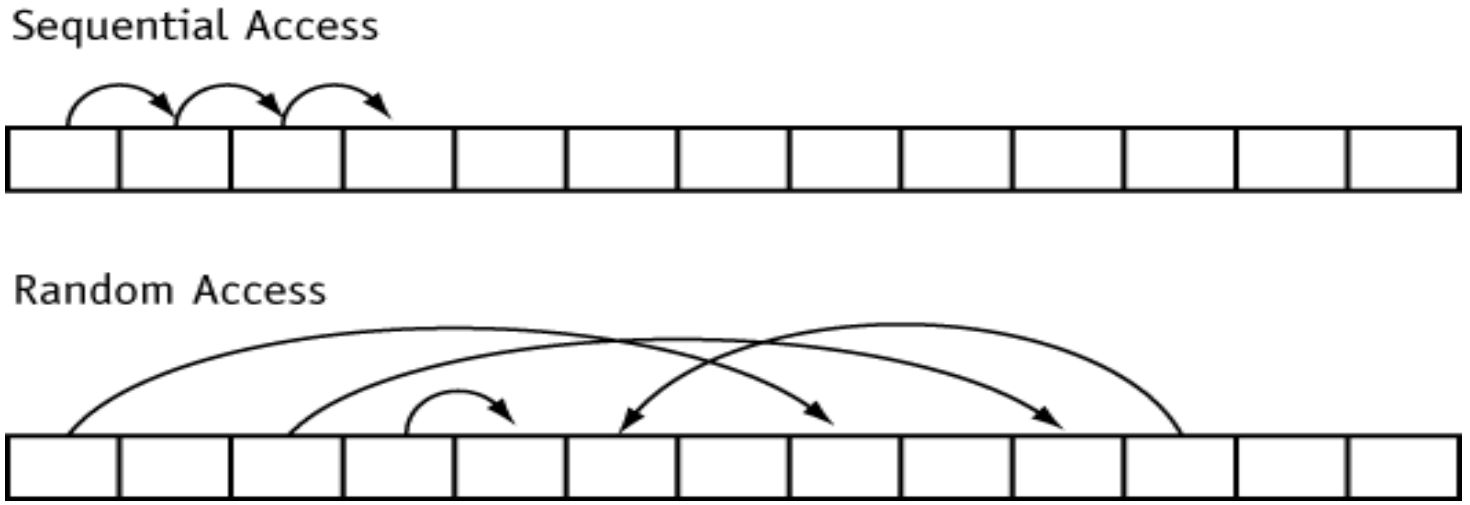
```
if (inFile.eof())
  inFile.close();
```

Random Access File

Random Access means non-sequentially accessing information in a file.

Different modes to operate Random AccessFile-

- ios::beg : The offset is calculated from the beginning of the file.
- ios::end : The offset is calculated from the end of the file.
- ios::cur : The offset is calculated from the current position.



seekp() and seekg()

Statement	How it affects the position
File.seekp(32L, ios::beg);	Sets the write position to the 33rd byte (byte 32) from the beginning of the file.
file.seekp(-10L, ios::end);	Sets the write position to the 11th byte (byte 10) from the end of the file.
file.seekp(120L, ios::cur);	Sets the write position to the 121st byte (byte 120) from the current position.
file.seekg(2L, ios::beg);	Sets the read position to the 3rd byte (byte 2) from the beginning of the file.
file.seekg(-100L, ios::end);	Sets the read position to the 101st byte (byte 100) from the end of the file.
file.seekg(40L, ios::cur);	Sets the read position to the 41st byte (byte 40) from the current position.
file.seekg(0L, ios::end);	Sets the read position to the end of the file.

tellp() and tellg()

tellp returns a long integer that is the current byte number of the file's write position.

tellg returns a long integer that is the current byte number of the file's read position.

```
#include<fstream>
#include<iostream>
using namespace std;
int main()
{
        char ch;
        ofstream f1("first.txt");

        f1<<"Hi";
        f1<<" Hello";
        f1<<" Good Afternoon";
        f1.put('C');

        f1.close();
```

```
        ifstream f2("first.txt");
        while((!f2.eof()))
        {
                ch=f2.get();
                cout<<ch;
        }
        f2.close();
}
```

Program to write to file

```
#include<fstream>
#include<iostream>
using namespace std;
int main()
{
        char arr[]="Infoway Technologies Pvt. Ltd., Pune";
        ofstream f1("text1.txt");
  f1<<arr;
/*      or
  for(int i=0;arr[i]!='\0';i++)
        {
                f1.put(arr[i]);
        }
*/      f1.seekp(5,ios::beg);
        f1.put('P');
        f1.put('p');
}
```

Program to read from file

```
#include<fstream>
#include<iostream>
using namespace std;
int main()
{
        char ch;
        ifstream f1("text1.txt");
    f1.seekg(2,ios::beg);
        while(f1)
        {
                f1.get(ch);
                cout<<ch;
        /*      if(ch=='E')
                  f1.seekg(1,ios::cur);
     f1.get(ch);
                cout<<ch;
        */}
}
```

Binary / Record File

```
#include<fstream>
#include<iostream>
using namespace std;
class student{
  int rno;
  char name[15];
  public:
   void set_data()
   {
            cout<<"Enter Rno";
            cin>>rno;
            cout<<"\nEnter Name";
            cin>>name;
   }
   void display()
  {
     cout<<"Rno: "<<rno;
     cout<<"Name: "<<name<<endl;
   }
};
```

```
int main()
{
  ofstream f1;
  ifstream f2;
  f1.open("mydata.dat");
  student obj1,obj2;
  obj1.set_data();
  f1.write((char *)&obj1,sizeof(obj1));
  obj1.display();
  f1.close();

  f2.open("mydata.dat");
  f2.read((char *)&obj2,sizeof(obj2));
  obj2.display();
  f2.close();
}
```