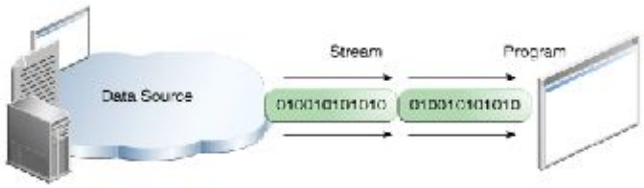
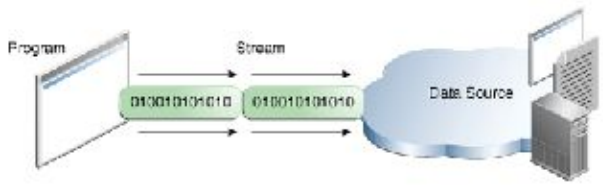
Disk input/output in C++

- While writing application there is need to store data/information permanently.
- Permanent / Auxiliary /secondary storage device allow us to save data forever until you delete it.
- If we want to save data permanently we have understand ***file system*** that is supported by operating system
- Permanent storage media stores its information in the data structure called **file**.
- Disk Management is the responsibility of the *file system which is a module of operating system.*
- The technology C++ allows us to store data/information to disk and for does this there are *Disk management* ***classes.***
- ***We can perform these common operations***
    - List of file and folders
    - Create file folder
    - File Create
    - File Open
        - File Contents Read
        - File Contents Write
        - File Contents Append
        - File Contents Modify
        - File Contents Delete
    - File Close
    - FILE Delete
    - Rename The File etc.

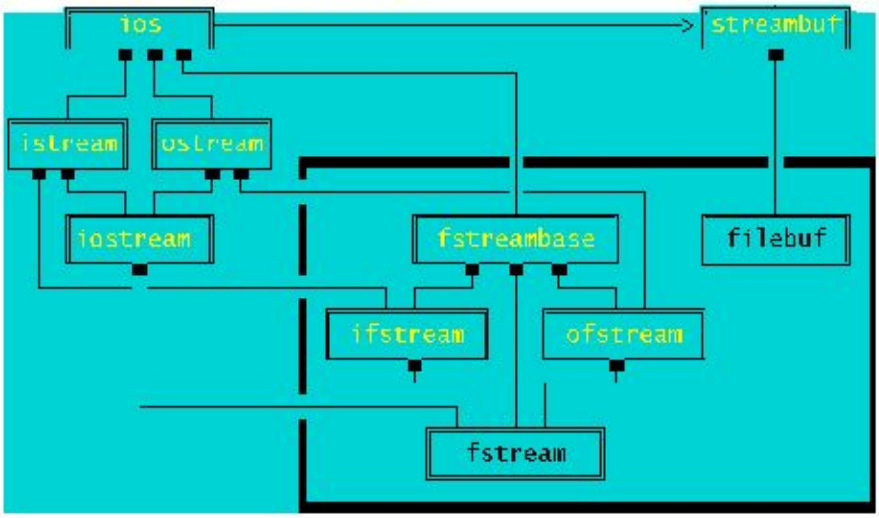
- Note
    - C++ technology uses ***stream object*** to perform input/output through the file system by using STREAM.
    - Stream is ***flowable objects.***
    - I/O Streams
        - A stream is a ***sequence of data***. A program uses an input stream to read data from a source, one item at a time. Input steam used to read information into a program.



        - A program uses an output stream to write data to a destination, one item at time. Output stream is used to Write information from a program.



- As we know a stream represents sequence of data for data. Stream based I/O have the following advantages over conventional I/O.
    - Abstraction
    - Flexibility
    - Performance
- ***Fstream class summary***



- C++ provides the following classes to perform output and input to/from files
    - ***ifstream***: Stream class to read from files
    - ***ofstream***: Stream class to write on files
    - ***fstream***: Stream class to both read and write from/to files.
- These classes are derived directly or indirectly from the classes istream and ostream.

- ***Ifstream class***
    - ***Its object allows to read data from file***
        - Constructors:
            - ifstream::ifstream()
            - ifstream::ifstream(const char*, int, int = filebuf::openprot);
        - Member Functions
            - ifstream::open()
            - ifstream::rdbuf()
            - get()

e.g.

```
#include<iostream>
#include<fstream>
using namespace std;
main ()
{
        char ch ;
        ifstream infile("c:/myfile/data.txt");
        while (infile)
        {
                infile.get(ch);
                cout << ch ;
        }
```

```
        infile.close();

}
e.g.
#include<iostream>
#include<fstream>
using namespace std;
main ()
{
        char ch ;
        ifstream infile;
        infile.open("c:/myfile/data.txt");
        while (infile)
        {
                infile.get(ch);
                cout << ch ;
        }
        infile.close();
}
e.g.
#include<iostream>
#include<fstream>
using namespace std;
main ()
{
        char ch ;
        ifstream infile;
        infile.open("c:/myfile/data.txt");
        while (infile)
        {
                infile >>ch ;
                cout << ch ;
        }
        infile.close();
}
```

- Note
  - Method operator >> ignores white spaces so there is an another method get () get used to fetch data from file with white space.

- ***ofstream class***
  - ***its object allow you to write the data to the file***
    - ***Constructors:***
      - ofstream::ofstream();
      - ofstream::ofstream(const char*, int=ios::out, int = filebuf::openprot);
    - ***Member Functions***
      - ofstream::ofstream
      - ofstream::open
      - ofstream::rdbuf

- ***File copy program***

```
#include<iostream>
#include<fstream>
using namespace std;
main ()
{
    ifstream infile("c:/myfile/data.txt");
    ofstream outfile("c:/myfile/siddhant.txt");
    char ch ;
    while ( infile )
    {
            infile.get(ch);
            outfile << ch ;

    }
    infile.close();
    outfile.close();
    cout << "\n1 file copied(s)";
}
```

- ***File modes***
    - There are several other modes in which stream object can be opened.
    - Each mode is representing by a bit pattern in IOS class.
    - We can combine these bits using ***logical OR*** operator. Each combination of mode bits specifies various aspect of how a stream object will be opened.
        - ios::in
            - Open for reading
            - default for ifstream
        - ios::out
            - Open for writing
            - Default for ofstream
        - ios::ate
            - start reading or writing at end of file
        - ios::app
            - start writing at end of file
        - ios::trunc
            - trancate file to 0 length it exists
            - Default for ofstream
        - ios::nocreate
            - Error when opening if file does not already exists
        - ios::noreplace
            - Error when opening for output if file already exist unless ate or app is set
        - ios::Binary
            - Open file in binary mode

- ***String transfer***

e.g.

```
#include<iostream>
#include<fstream>
#include<string.h>
#include<stdio.h>
using namespace std;
main ()
{
    ofstream outfile("c:/myfile/strdata.txt");
    char str[80];
    cout << "\nEnter poem ";
    while (strlen(gets(str)) > 0 )
    {
            outfile << str << endl ;
    }
    outfile.close();
}
```

- ***Reading string from file***

e.g.

```
#include<iostream>
#include<fstream>
using namespace std;
main ()
{
    ifstream infile("c:/myfile/strdata.txt");
    char str[80];
    cout << "\nyour poem ";
    while ( infile)
    {
            infile.getline(str,80);
            cout << str << endl ;

    }
    infile.close();

}
```

- ***Serialiation***
  - It is require writing an object to a flat file or sending it over a network.
  - To send an object over a wire (for example, to a file, over a network), the system must deconstruct the object into stream(flatten it out), send it over the wire, and then reconstruct it on the other end of the wire.This process is called serializing an object.
  - The act of actually sending the object across a wire is called marshaling an object.A serialized object..

- ***Sending object to a file***

e.g.

```
#include<iostream>
```

```
#include<string.h>
#include<fstream>
#include<stdio.h>
#include<conio.h>
using namespace std;
class book
{
        private :
                char name[20] ;
                int page ;
                float price;
        public:
                book(){}
                book(char *b , int p , float pri)
                {
                        strcpy(name , b ) ;
                        page = p ;
                        price = pri ;
                }
                void getdata()
                {

                        cout << "\nEnter book name" ;
                        cin.ignore();
                        cin.getline(name,20);
                        cout << "\nEnter book page" ;
                        cin >> page;
                        cout << "\nEnter book price" ;
                        cin >> price;

                }
                void set(char *b , int p , float pri)
                {
                        strcpy(name , b ) ;
                        page = p ;
                        price = pri ;
                }

                void display()
                {
                        cout << endl << "book information";
                        cout << endl << "name = " << name ;
                        cout << endl << "page = " << page ;
                        cout << endl << "price = " << price ;
                }

};

main ()
{
        ofstream outfile;
```

```
        outfile.open("c:/myfile/recdata",ios::out|ios::binary);
        book b;
        char another ;
        do
        {
                b.getdata();
                outfile.write ((char*) &b, sizeof (b));
                cout << "\ndo you want to another record";
                another = getche();

        } while (another =='y');

        outfile.close();

}
```

- ***Reading record from file***

e.g.

```
#include<iostream>
#include<string.h>
#include<fstream>
#include<stdio.h>
#include<conio.h>
using namespace std;
class book
{
        private :
                char name[20] ;
                int page ;
                float price;
        public:
                book(){}
                book(char *b , int p , float pri)
                {
                        strcpy(name , b ) ;
                        page = p ;
                        price = pri ;
                }
                void getdata()
                {

                        cout << "\nEnter book name" ;
                        cin.ignore();
                        cin.getline(name,20);
                        cout << "\nEnter book page" ;
                        cin >> page;
                        cout << "\nEnter book price" ;
                        cin >> price;

                }
                void set(char *b , int p , float pri)
```

```
        {
                strcpy(name , b ) ;
                page = p ;
                price = pri ;
        }

        void display()
        {
                cout << endl << "book information";
                cout << endl << "name = " << name ;
                cout << endl << "page = " << page ;
                cout << endl << "price = " << price ;
        }

};

main ()
{
        ifstream infile;
        infile.open("c:/myfile/recdata",ios::in|ios::binary);
        book b;
        while( infile.read( (char*) &b , sizeof (b)) )
        {
                b.display();
        }
        infile.close();
}
```

- ***Repositioning file pointer***
  - ***ofstream outfile : outfile.seekp***
    - it is use to reset the put pointer on a specific position

  e.g.
  outfile.seekp(0L,ios::beg) ; to set the pointer to begin
  outfile.seekp(0L,ios::end); to set the pointer to end
  outfile.seekp(0L,ios::cur)
  outfile.seekp(20L,ios::beg) ; to set the pointer to begin and move 20 byte ahead
  outfile.seekp(-20L,ios::end); to set the pointer to end and move 20 byte back

  - ***ifstream infile : infile file.seekg***
    - it is use to reset the put pointer on a specific position

  e.g.
  infile.seekg(0L,ios::beg) ; to set the pointer to begin
  infile.seekg(0L,ios::end); to set the pointer to end
  infile.seekg(0l,ios::cur)
  infile.seekg(20L,ios::beg) ; to set the pointer to begin and move 20 byte ahead
  infile.seekg(-20L,ios::end); to set the pointer to end and move 20 byte back

- commonly used function
  - ***rewind(fs)***
    - It set the file pointer to begin
  - ***remove("filename")***
    - It removes the file from the disk
  - rename ("oldfile","newfile")
    - It rename the file.