Section 22 : I/O Streams

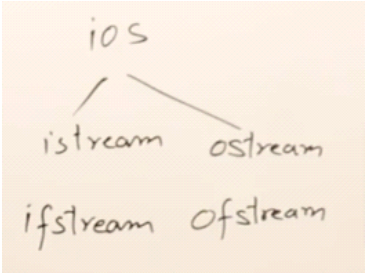
15 April 2022 16:49

© Rajat Kumar
https://www.linkedin.com/in/imRajat/
https://github.com/im-Rajat

Section 22 : I/O Streams

Steams :

- Must include : **#include <fstream>**
- Stream is a flow of data or flow of characters
- Streams are used for accessing the data from outside the program. That is from external sources or destination.
- So for accessing the data from outside world of a program, we use streams.
- I/O Streams = Input/Output Streams.
- There are building classes available in C++ for accessing input output stream classes.
- IOS :
  - Istream - ifstream
  - Ostream - ofstream



- We can use the same insertion (cin>>) and extraction (cout<<) operators for reading and writing into the file.
- Ofstream outfile("my.txt");
  - It's same as Ofstream outfile("my.txt", ios::trunc); // trunc = truncate
- If we are opening a file that's already existed, it will just open it. If it don't exist it will create a new file with same name.
  - If already file is there and it's having some content, then it will truncate/remove the content.
- If we want the content also, then we can change the mode to append.
  - Ofstream outfile("my.txt", ios::app);
- For writing anything to the file :
  - Output<<"Hello"<<endl;
  - Output<<"How are you?"<<endl;
- We must close the file after using it.
  - Output.close();

Writing into a File :

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

int main()
{
    ofstream ofs("my.txt", ios::trunc);
    ofs<<"RJ"<<endl;
    ofs<<22<<endl;
    ofs<<"Cricket"<<endl;

    ofs.close();

    return 0;
}
```

Reading Data from a File :

- Ifstream infile;
- Infile.open("my.txt");
- Modes/Flags available:
  - Ios::in
  - Ios::out
- If we are reading from a file, the file must exist. So we need to check it
- If (!infile) { cout<<"File cannot open" };
- Or if (infile.is_open()) {…}
- Reading data :
  - Infile>>str; // string str;

- Infile>>x; // int x;
- Check end of file reached
  - If (infile.eof()) { cout<<"End of file reached"; }
  - Infile.close();
- Point to remember is when we read the data from a file, we must know the format, if we are reading int, string or anything else.

```
// Reading the data;
ifstream ifs("my.txt");
//ifstream ifs;        // same as above
//ifs.open("my.txt");
if (ifs.is_open())       // same as if(ifs)
{
    cout<<"File is Opened"<<endl;
}
string name;
int roll;
string sports;
ifs>>name>>roll>>sports;
ifs.close();

cout<<"Name: "<<name<<endl;
cout<<"Roll No.: "<<roll<<endl;
cout<<"Sports: "<<sports<<endl;
```

Serialization :

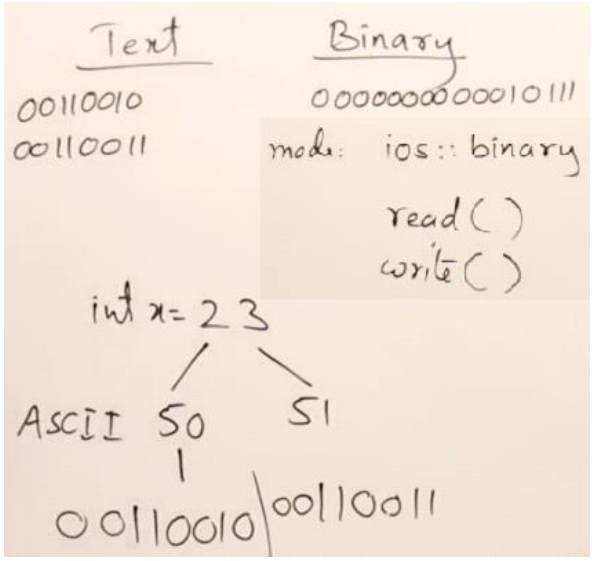
- It is a process of storing and retrieving the state of an object.

```
class Student {
    public:
        string name;
        int roll;
        string branch;

        friend ofstream & operator<<(ofstream &ofs, Student &s);
        friend ifstream & operator>>(ifstream &ifs, Student &s);
};

ofstream & operator<<(ofstream &ofs, Student &s) {
    ofs<<s.name<<endl;
    ofs<<s.roll<<endl;
    ofs<<s.branch<<endl;

    return ofs;
}

ifstream & operator>>(ifstream &ifs, Student &s) {
    ifs>>s.name>>s.roll>>s.branch;

    return ifs;
}

int main()
{
    Student s1;
    s1.name = "John";
    s1.roll = 10;
    s1.branch = "CS";
    ofstream ofs("Student.txt", ios::trunc);
    ofs<<s1;
    ofs.close();

    ifstream ifs("Student.txt");
    ifs>>s1;
    cout<<"Name: "<<s1.name<<endl;
    cout<<"Roll No.: "<<s1.roll<<endl;
    cout<<"Branch: "<<s1.branch<<endl;
    ifs.close();
```

- And any object we want to store and retrieve in a file, we can use serialization. For that, we must overload these operators (>> and <<).

2 Type of Files :

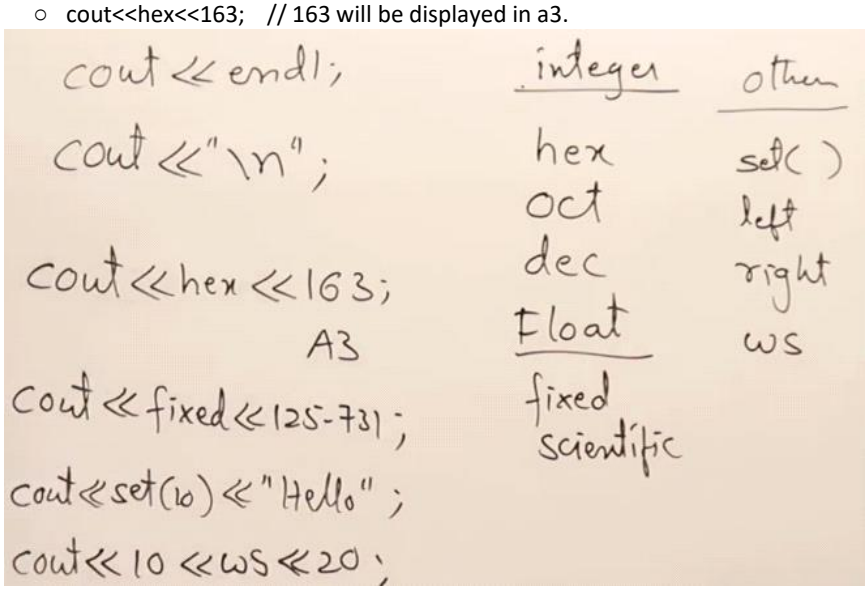
- Text Files
- Binary Files



- For text files : If we open this text file in notepad then, what Notepad will do, for every eight bit, it will convert into ASCII code and display a symbol, for next 8 bit it will do the same.
- For binary file : If we open this file in notepad then, it might print junk data. Because for every 8 bit it doesn't make any meaningful ascii code.
- For reading binary file we need to use **ios::binary** mode. And functions are also available like read() and write().
- Binary files are faster because for reading text file it needed conversion.
- Text files take more space.

Manipulators :

- Manipulators are used for enhancing streams or formatting streams
- When we want to write the data, for writing the data we can adopt some format.
- Example :
  - Cout<<endl;
  - Cout<<"\n";
  - cout<<hex<<163; // 163 will be displayed in a3.



- << operator is used to insert the data into file.
- Seekg function is used to position the pointer backward from the end of file for reading.
- 3 C++ objects are used for taking string as input from keyboard and displaying it on screen.
- Is_open member function is used to determine whether the stream object is currently associated with a file.
- #include <fstream> header file is used for reading and writing to a file.