C++ Slides - 6

File handling: Formatted I/O, Hierarchy of file stream classes, Opening and closing a file, Working with multiple files, file modes, file pointers, Text vs Binary Files.

Formatting I/O

- Set precision
- Text justification
- Display +/- sign
- …

```
//Formatting example
#include<iostream>
using namespace std;
int main(){
cout<<showpos<<10.1234<<endl;       //show +/- sign
cout.precision(4);                  //total display digits
cout<<-12.34567<<endl;
cout.width(5);                  // Right justify with 5 char
cout << 'c' <<endl;
}
```

Output

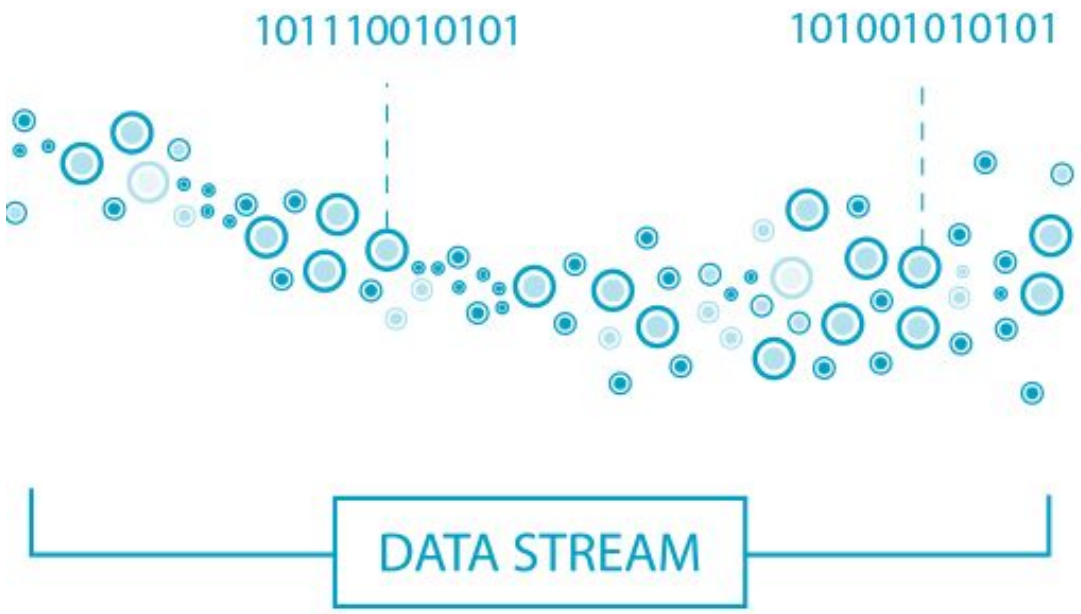
+10.1234

-12.35

c

Stream

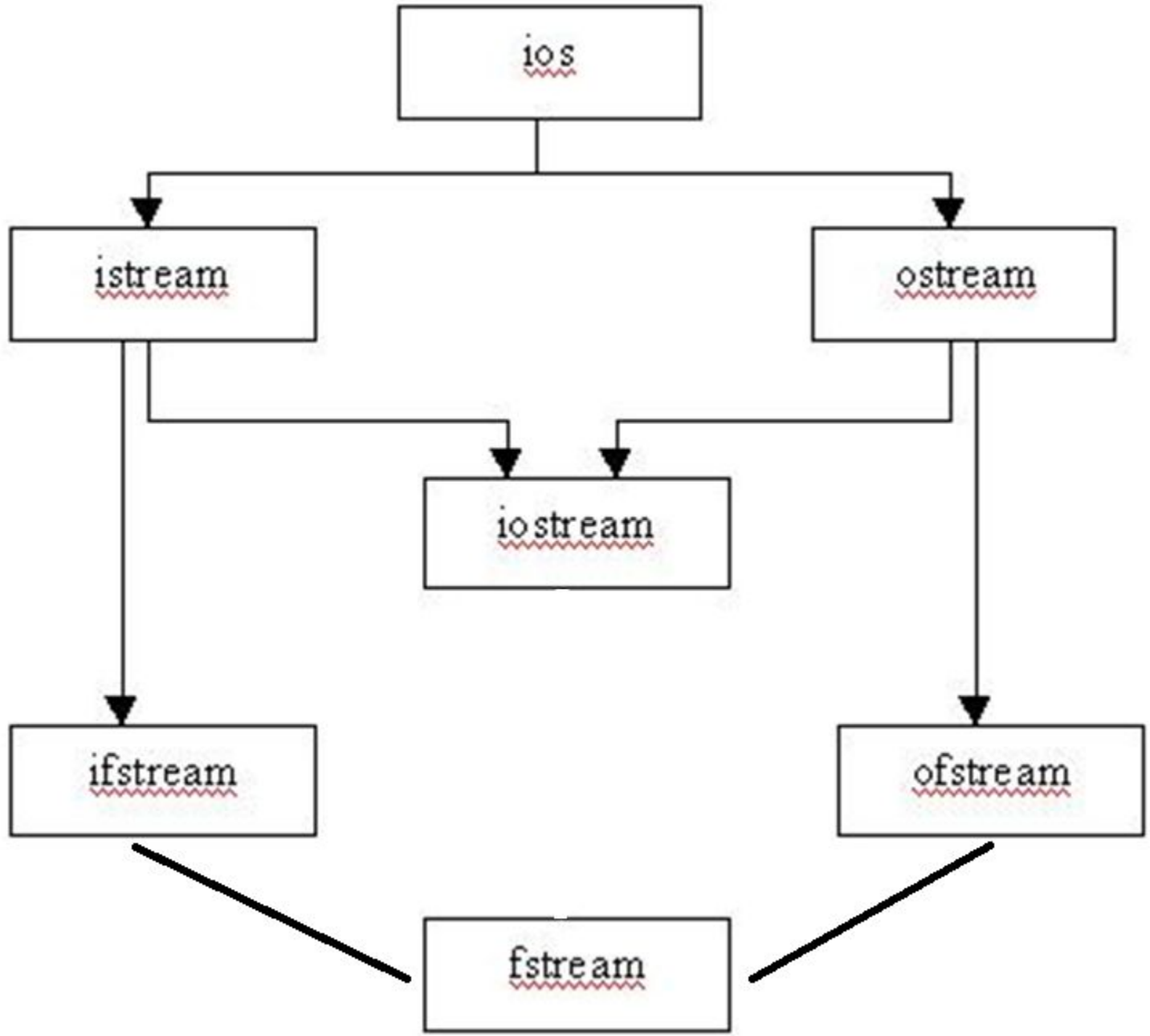
Stream is the flow of digital data (bytes) as input or output through an abstract device (for example screen or files).



File stream **classes**

1. **ios** - all I/O operations
2. **istream** is for input e.g. as getline()
3. **ostream** is for output e.g. write()
4. **ifstream** – Input from file
5. **ofstream** – write to a file
6. **fstream** – I/O with a file


Remember!


ios
istream
ostream
iostream
ifstream
ofstream
fstream

Consider a simple text file MySecrets.txt

This is
a simple text
consisting words
numbers 1234.09872
and symbols !@#$%^&*()

```
// Read characters from file
#include <iostream>
#include<fstream>
using namespace std;
int main() {
char c;
ifstream fin("MySecrets.txt");
    if(!fin) {cout<<"File Does not Exist";  return 0; }
    while(!fin.eof()) { // eof – end of the file
        fin.get(c); cout<<c;   // print text on screen
    }
fin.close();}
```

```
// Writing text into file

#include <iostream>
#include <fstream>
using namespace std;
int main () {
ofstream file("Simplefile.txt");
file << "Writing to a file in C++....";
file.close();
}
```

Working with multiple files

```
// MyFile.cpp
int MyValue() {
return -9999;
}
```

```
#include “MyFile.cpp”
#include <iostream>
using namespace std;

int main() {
cout<<MyValue();
}
```

File modes

1. **ios::app –** Always write in the end of a file
2. **ios::ate –** Take the control at the end just once
3. **ios::in -** Open a file for reading
4. **ios::out -** Open a file for writing
5. **ios::trunc –** remove the old contents

```
// Writing text into file using ios::out flag


#include <iostream>
#include <fstream>
using namespace std;
int main () {
fstream file("Simplefile.txt“,ios::out);
file << "Writing to a file in C++....";
file.close();
}
```

```
// Reading + writing in a file
#include<iostream>
#include<fstream>
using namespace std;
int main(){
string sen;
ofstream fout("MySecrets.txt");
fout<<"hello 123";
fout.close();
ifstream fin("MySecrets.txt");
getline(fin, sen); cout<<sen;
fin.close();
}
```

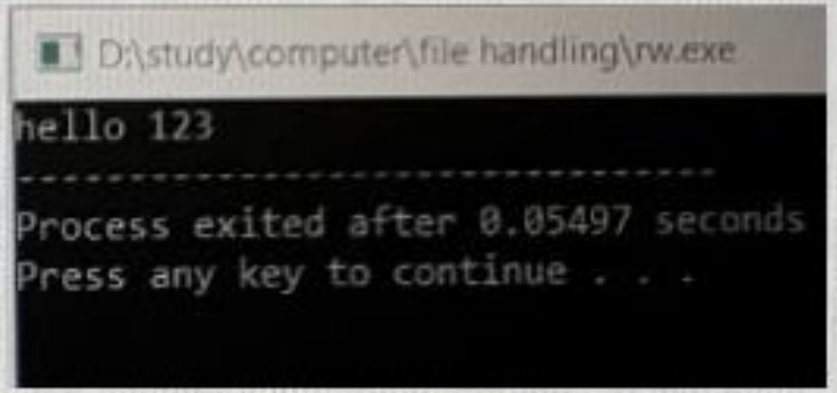
Q:Try to read and write a file using ios::in and ios::out

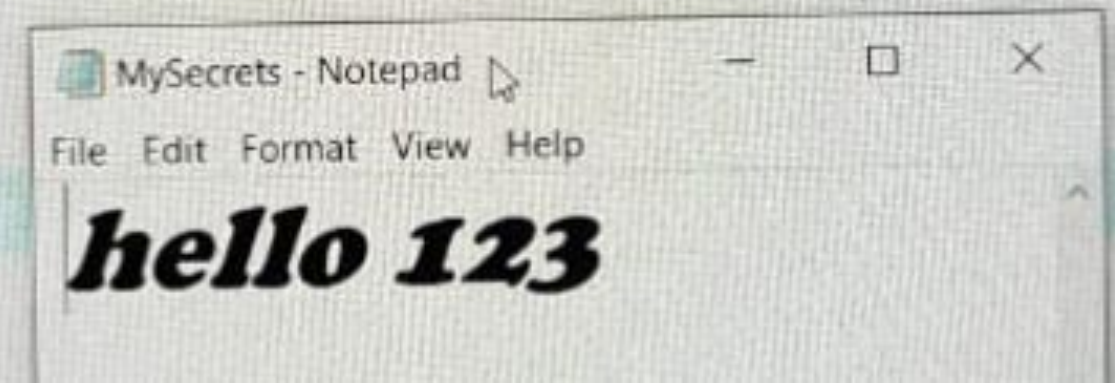
Hint:

fstream fileIO(“MySecrets.txt”,ios::in,ios::out);

You may need file pointers such as *seek()* and *tell()* functions which are covered next.

```cpp
// Reading + writing in a file
#include<iostream>
#include<fstream>
using namespace std;
int main(){
string sen;
fstream iofile("MySecrets.txt", ios::out | ios::in);
iofile<<"hello 123";
iofile.seekg(0,ios::beg);
getline(iofile,sen);
cout<<sen;
iofile.close();
}
```





File pointers – bookmarks in the file

- Get Pointer tells the current *location* during file *reading*
- Put Pointer tells the current *position* during file *writing*
- A file pointer is not like a C++ pointer but works like a book-mark in a book
- These pointers help attain random access in file for faster access in comparison to a sequential access

File pointers – seek and tell

- Tell function - just *examine* the file location
- Seek function – actually *set* the bookmark in a file

G and P in – seekg(), tellg(), seekp(), tellp()

- G – (as Get) is for file read operation
- P – (as Put) is for file write operation

File pointers – seekg(), tellg(), seekp(), tellp()

- seekg() and tellg() functions allow you to set and examine the get_pointer in the given file

- seekp() and tellp() functions perform these operations on the put_pointer.

Current, beginning and & end

References for seek and tell functions

- ios::beg
- ios:cur
- ios:end

Few examples

- fin.seekg(30); or fin.seekg(30, ios::beg);

// go to byte no. 30 from beginning of file

- fin.seekg(-2, ios::cur); // back up 2 bytes from the current position of get pointer

- fin.seekg(-4, ios::end); // backup 4 bytes from the end of the file

Text vs Binary Files

- In *text* files various character translations are performed such as “\r+\f” is converted into “\n”, whereas in binary files no such translations are performed.

- By default, C++ opens the files in text mode.

Text files versus Binary files

- ofstream out (“myfile.txt”);

- Storage is more e.g. floating point numbers (IEEE 754) 12345.1234 will take 10 char bytes.
- Access is slower using sequential search

- ofstream out (“myfile.txt”,ios::binary);

- Memory efficient for storage.
- Each binary file maintains a header of content index for faster search as compared to text files. More info is required but access is faster.

Useful codes for file programs

```
// how to count characters in a file
int count = 0; char ch; while(!fin.eof()) { fin.get(ch); count++; }

// how to count words in a file
int count = 0; char word[30]; while(!fin.eof()) { fin >> word; count++; }

// how to count lines in a file
int count = 0;  char str[80];
while(!fin.eof()) {  fin.getline(str,80); count++; }
```

```
// Copy contents of one file into another
#include<iostream>
#include<fstream>
using namespace std;
int main() {
char ch;
ifstream fin("MySecrets.txt");
ofstream fout("CopyMySecrets.txt");
while(!fin.eof()) {
fin.get(ch);
fout << ch;
}
fin.close(); fout.close();
}
```