Programming Fundamentals

LECTURE 02: FILE HANDLING

BY: ZUPASH AWAIS

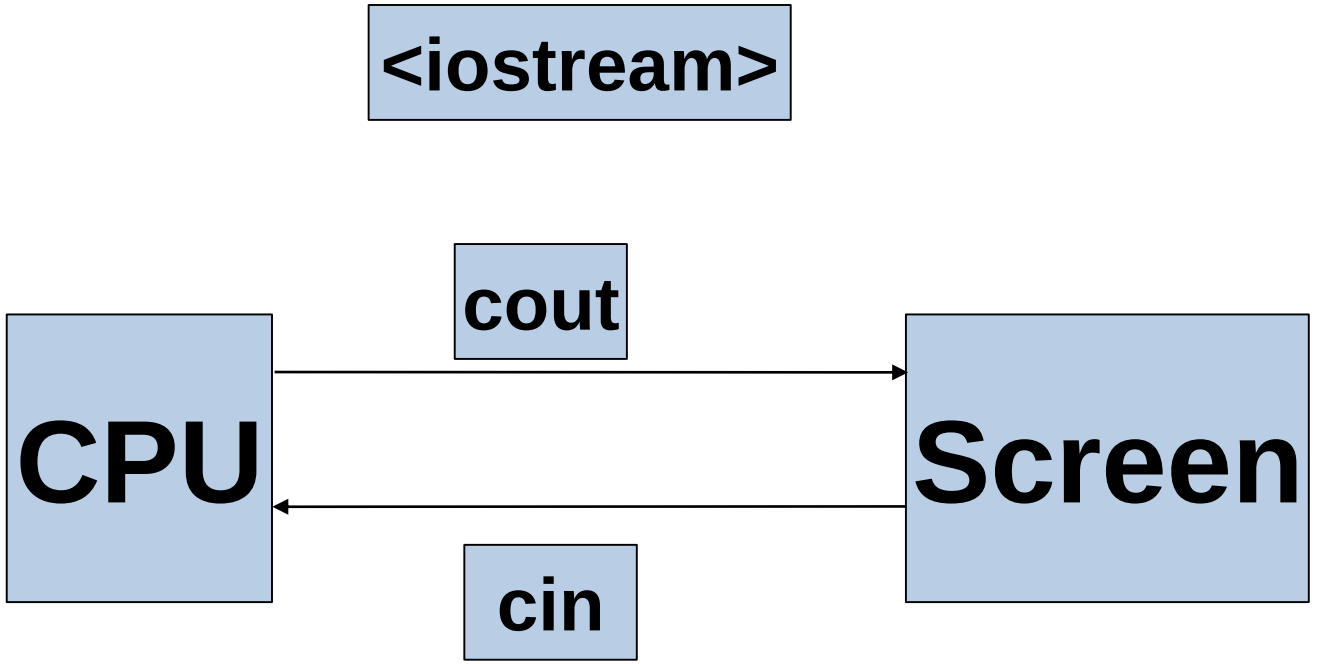
WEEK 02

File

- A set of data stored on a computer, often on a disk drive
- Programs can read from or write to files
- Used in many applications:
  - Word processing
  - Databases
  - Spreadsheets

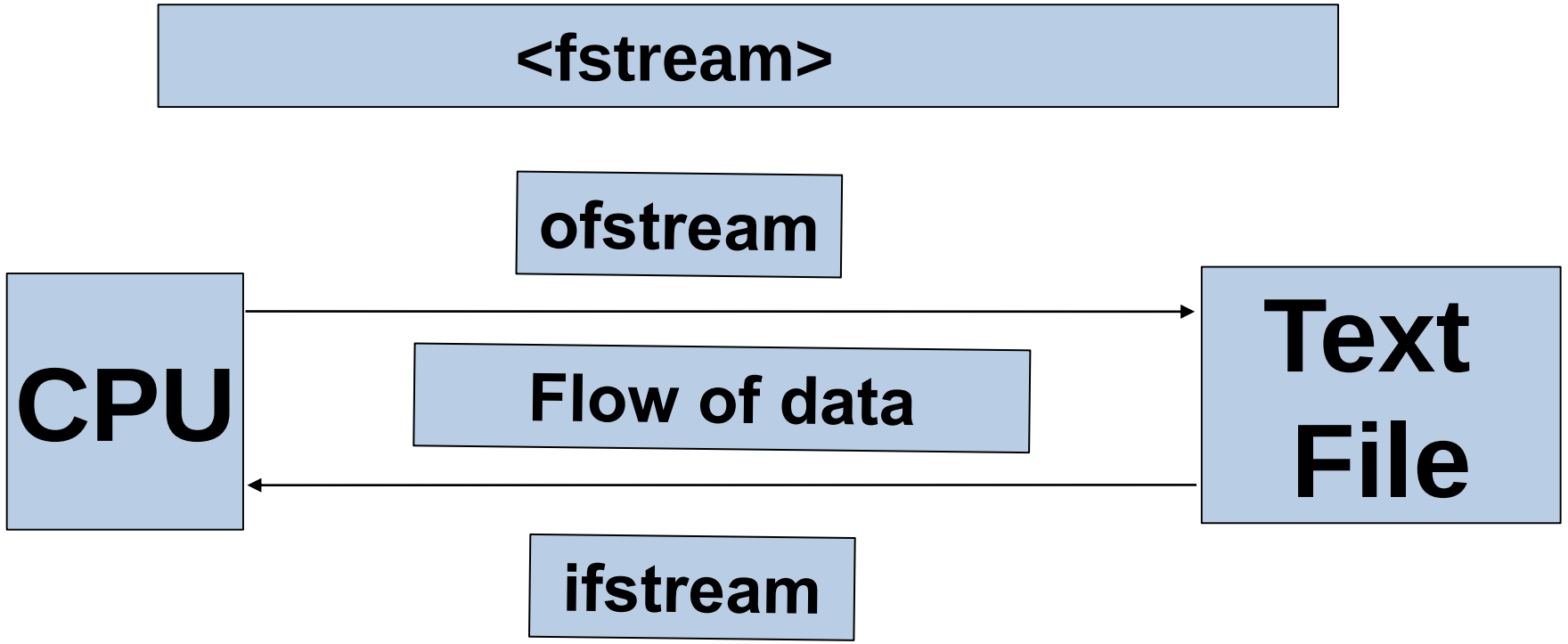
Streams <iostream>

- Stream is the flow of water from one direction to another
- In computer science streams are the flow of data, sometimes from CPU to Screen or the other way around.
- It’s <iostream>



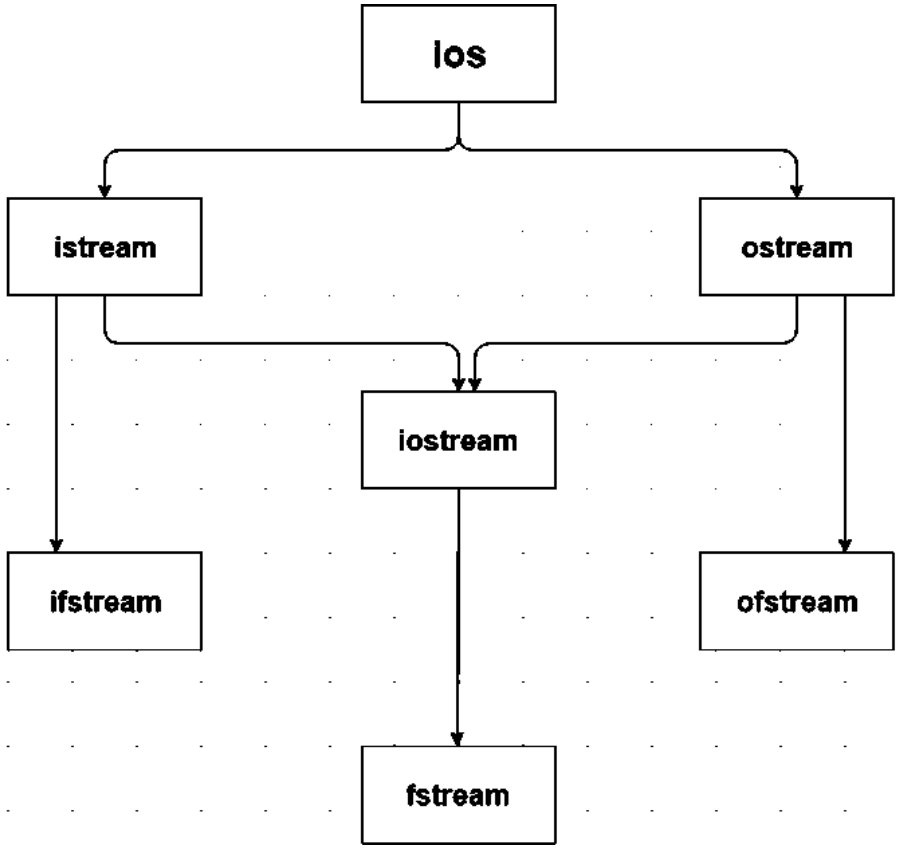
Files Stream <fstream>

- If the data flows to and from other than the screen/console, the stream changes.
- In the case of a file, the stream is fstream.



File Handling

- File handling is used to store a data permanently in computer. The data can be stored in secondary memory (hard disk) using file handling.
- The I/O data can easily transferred from one computer to another by using files.
- The C++ standard library provides fstream class for performing Read and Write operations.



Datatypes and Functions of Fstreams

Datatype	Description
ofstream	Used for creating a file and write data on files.
ifstream	Used for reading the data from files.
fstream	Used for both read and write data from files.

Function	Description
open()	It is used to create a file.
close()	It is used to close an existing file.
get()	It is used to read a single unit character from a file.
put()	It is used to write a single unit character in file.
read()	It is used to read data from file.
write()	It is used to write data into file.

Operations of File Handling

1. Naming a file
2. Opening a file
3. Reading data from file
4. Writing data into a file
5. Closing a file

Writing to a File

The insertion operator (<<) is used to write information in a file.

The ofstream or fstream object is used instead of cout object.

Reading from a File

The extraction operator (>>) is used to read information from a file into your program.

The ifstream or fstream object is used instead of cin object.

Note: ofstream and ifstream are datatype you can make as many objects of both types as you like. It depends on how many files you are working on in your program

Example Writing in File

```
#include <fstream>
#include <iostream>
using namespace std;
int main()
{
   ofstream fout;
   fout.open("text.txt");

   cout << "Writing to the file" << endl;
   fout << "My First File" << endl;
   fout.close();
   // close the opened file.

   return 0;
}
```

Comparison of cout and fout

To print number on CONSOLE screen, cout is used.	**To print number to a file, fout is used.**
cout is pre-defined keyword. So no need to declare it. Like int, float, endl, return etc.	**fout is user-defined keyword. So we need to declare it like we declare a variable.** **ofstream fout;**
cout, by default, is linked with console screen. So no need to open it with cout.	**fout is not linked with any file. So there is a need to open the file with fin.** **fout.open();**
Insertion operator is used << **cout << variable_name;**	**Insertion operator is used <<** **fout << variable_name;**

Example Reading From File

```
#include <fstream>
#include <iostream>
#include<string>
using namespace std;
int main()
{
    ifstream fin;
    fin.open("file.txt");
    for (int i = 0; i < 4; i++)
    {
        fin >> read;
        cout << read<<" ";
    }
    fin.close();
    return 0;
}
```

Example Reading From File

```
#include <fstream>
#include <iostream>
#include<string>
using namespace std;
int main()
{
   ifstream fin;
   fin.("text.txt");
   string read;
   while (getline(fin,read))
   {
      cout << read;
   }
   fin.close();
   return 0;
}
```

Example Problem while Writing in File

```
#include <fstream>
#include <iostream>
using namespace std;
int main()
{
    ofstream fout;
    fout.open("text.txt");

    fout << "Writing to the file" << endl;
    fout << "My First File" << endl;
    ofile.close();
    // close the opened file.

    return 0;
}
```

Modes of File Handling

Name	Description
ios::in	Open file to read
ios::out	Open file to write
ios::app	All the data you write, is put at the end of the file. It calls ios::out
ios::ate	All the data you write, is put at the end of the file. It does not call ios::out
ios::trunc	Deletes all previous content in the file. (empties the file)
ios::nocreate	If the file does not exist, opening it with the open() function gets impossible.
ios::noreplace	If the file exists, trying to open it with the open() function, returns an error.
ios::binary	Opens the file in binary mode.

Example Problem while Writing in File

```
#include <fstream>
#include <iostream>
using namespace std;
int main()
{
   ofstream fout;
   fout.open("text.txt",ios::app);

   fout << "Writing to the file" << endl;
   fout << "My First File" << endl;
   ofile.close();
   // close the opened file.

   return 0;
}
```

Example (End of File)

```
#include <fstream>
#include <iostream>
#include<string>
using namespace std;
int main()
{
   ifstream ifile("text.txt");
   string read;
   while (!ifile.eof())
   {
      getline(ifile,read);
      cout << read;
   }
   ifile.close();
   return 0;
}
```