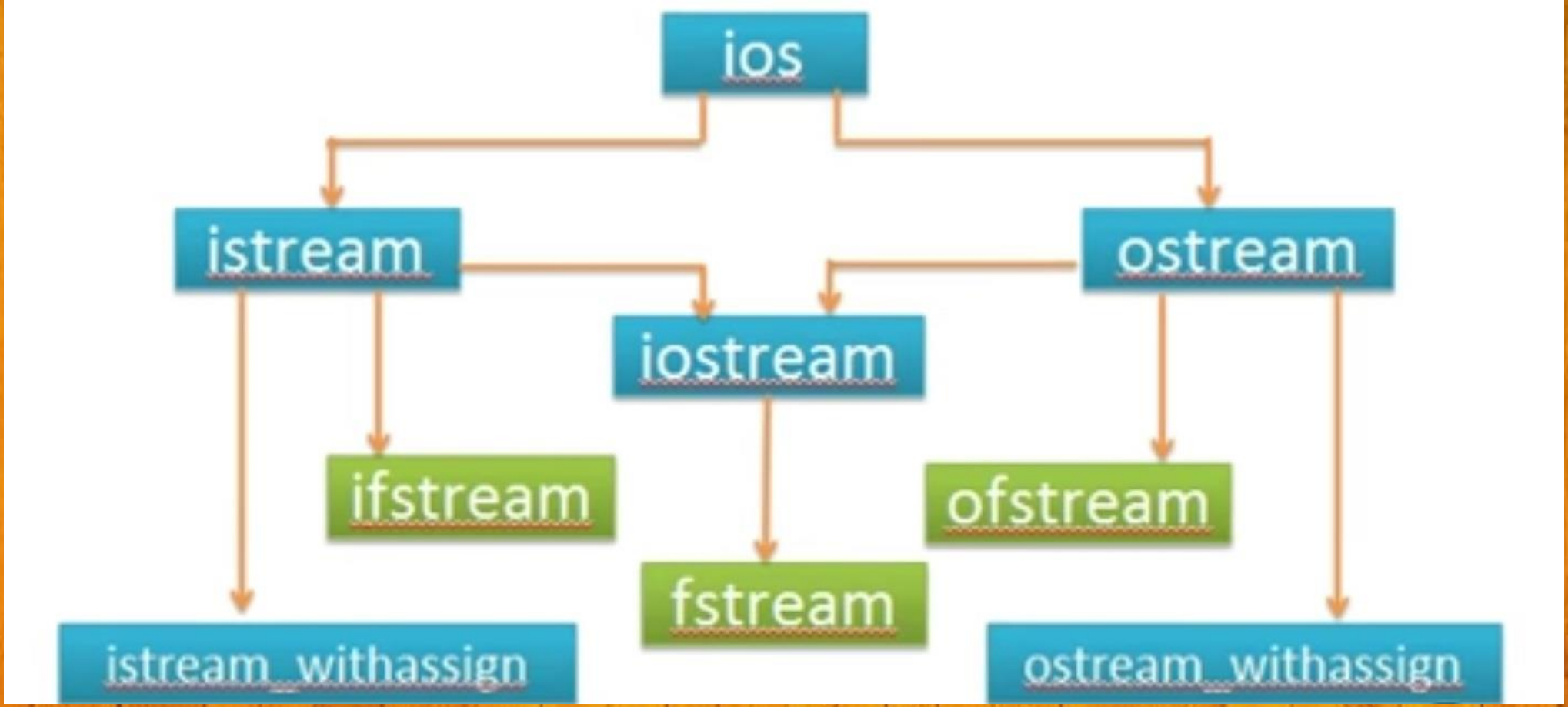
File Handling

File Handling

- Files are used to store data in a storage device permanently.
- File handling provides a mechanism to store the output of a program in a file and to perform various operations on it.
- The fstream library allows us to work with files.
- A stream is an abstraction that represents a device on which operations of input and output are performed. A stream can be represented as a source or destination of characters of indefinite length depending on its usage.

Streams


ios
istream
ostream
iostream
ifstream
ofstream
fstream
istream_withassign
ostream_withassign

There are three classes included in the fstream library,which are used to create, write or read files:

Class	Description
`ofstream`	Creates and writes to files
`ifstream`	Reads from files
`fstream`	A combination of ofstream and ifstream: creates, reads, and writes to files

Operations in File Handling:

- Creating a file: open()
- Reading data: read()
- Writing new data: write()
- Closing a file: close()

Modes of opening file:

Modes	Description
in	Opens the file to read(default for ifstream)
out	Opens the file to write(default for ofstream)
binary	Opens the file in binary mode
app	Opens the file and appends all the outputs at the end
ate	Opens the file and moves the control to the end of the file
trunc	Removes the data in the existing file
nocreate	Opens the file only if it already exists
noreplace	Opens the file only if it does not already exist

Default Open Modes :

→ifstream ios::in →ofstream ios::out → fstream ios::in | ios::out

Create and Write To a File

Generally, the first operation performed on an object of one of these classes is to associate it to a real file. This procedure is known to open a file.

To write to the file, use the insertion operator (<<).

Example:

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

int main() {
  // Create and open a text file
  ofstream MyFile("filename.txt");

  // Write to the file
  MyFile << "Files can be tricky, but it is fun enough!";

  // Close the file
  MyFile.close();
}
```

Another Example of writing to file:

Writing to a File

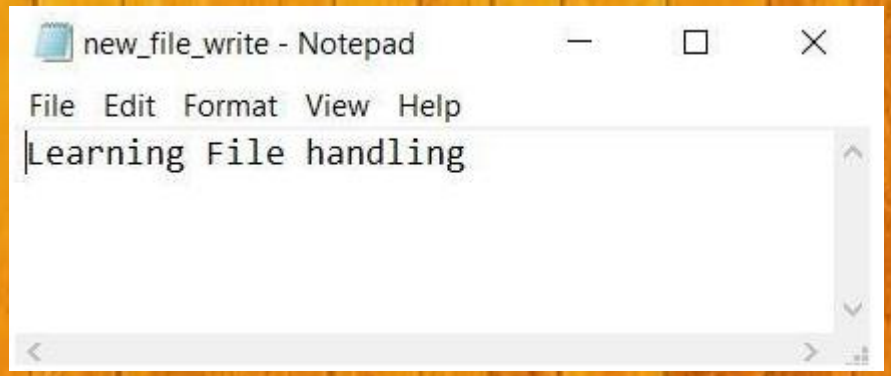
Example:

```
#include <iostream>
#include <fstream>
using namespace std;
int main()
{
fstream new_file;
new_file.open("new_file_write.txt",ios::out);
if(!new_file)
{
cout<<"File creation failed";
}
else
{
cout<<"New file created";
new_file<<"Learning File handling";    //Writing to file
new_file.close();
}
return 0;
}
```

Output:

```
C:\Users\Lenovo\Desktop>g++ new_file.cpp -o new_file.exe

C:\Users\Lenovo\Desktop>new_file.exe
New file created
C:\Users\Lenovo\Desktop>
```



Read a File

To read from a file, use either the ifstream or fstream class, and the name of the file

- For opening file in read mode we need to check whether the file is present in the storage or not, if no such named file is there and if we are not checking then it may give error.

Reading from a File

Example

```
#include <iostream>
#include <fstream>
using namespace std;
int main()
{
fstream new_file;
new_file.open("new_file_write.txt",ios::in);
if(!new_file)
cout<<"No such file"; } else { char ch; while (!new_file.eof()) { new_file >>ch;
cout << ch;
}
new_file.close();
return 0;
}
```

Output:

```
LearningFilehandlingg[Finished in 1.7s]
```

Closing a File

- It is simply done with the help of close() function.
- Why do we close the file?
- Closing a file will save the changes which we do in particular file.

→It is considered good practice, and it can clean up unnecessary memory space.

Example

```
1  #include <iostream>
2  #include <fstream>
3  using namespace std;
4  int main()
5  {
6  fstream new_file;
7  new_file.open("new_file.txt",ios::out);
8  new_file.close();
9  return 0;
10 }
```

Output: The file gets closed.

Thank You!!