File Handling

Introduction

- All programs we looked earlier:
  - input data from the keyboard.
  - output data to the screen.
- Output would be lost as soon as we exit from the program.
- How do we store data permanently?
  - We can use secondary storage device.
  - Data is packaged up on the storage device as data structures called files.

Streams Usage

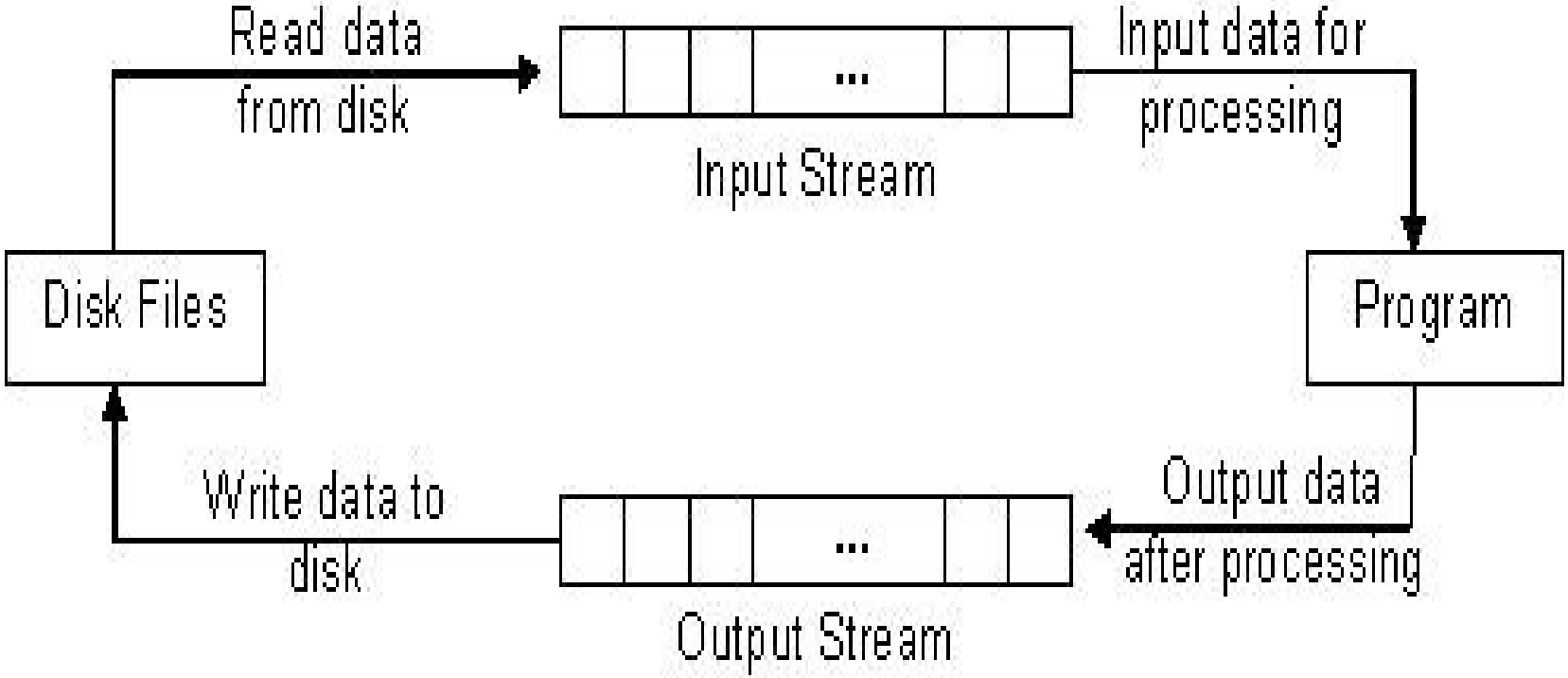
- We've used streams already
  - cin
    - Input from stream object connected to keyboard
  - cout
    - Output to stream object connected to screen
- Can define other streams
  - To or from files
  - Used similarly as cin, cout

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.

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.

In C++ we have a set of file handling methods. These include ifstream, ofstream, and fstream. These classes are derived from fstrembase and from the corresponding iostream class. These classes, designed to manage the disk files, are declared in fstream and therefore we must include fstream and therefore we must include this file in any program that uses files.

File input and output streams



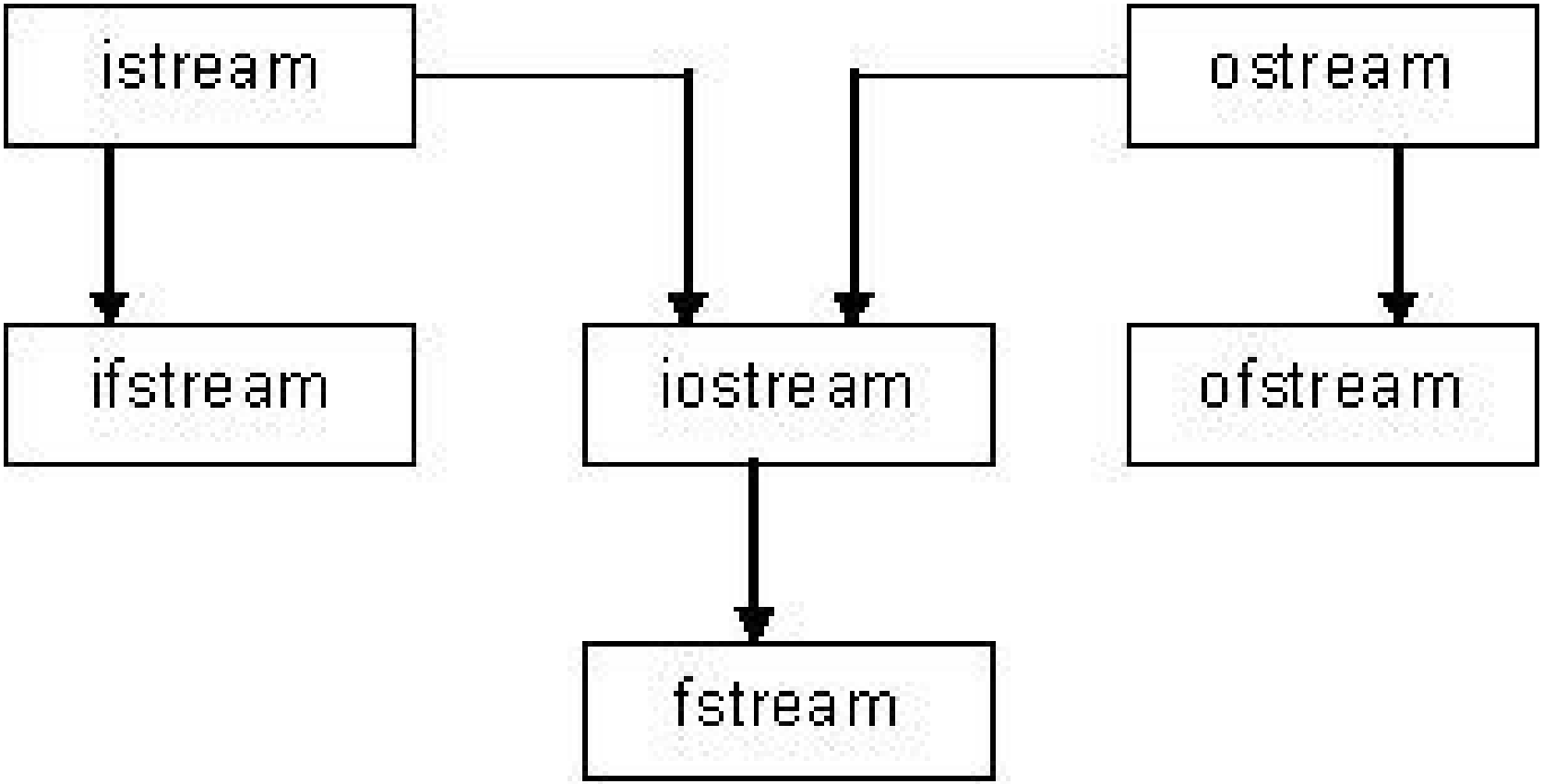
Streams

- File Input Stream – reads data from disk file to the program.

- File output Stream – writes data to the disk file from the program.

- The I/O system of C++ contains:
  - ifstream – provides input operations on files
  - ofstream – provides output operations on files
  - fstream – supports for simultaneous input and output operations on files

Stream Classes



Sr.No	Data Type & Description
1	**ofstream** This data type represents the output file stream and is used to create files and to write information to files.
2	**ifstream** This data type represents the input file stream and is used to read information from files.
3	**fstream** This data type represents the file stream generally, and has the capabilities of both ofstream and ifstream which means it can create files, write information to files, and read information from files.

C++ provides us with the following operations in File Handling:

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

Moving on with article on File Handling in C++

Opening a File

To open a file use

```
open() function
```

Syntax

```
void open(const char* file_name,ios::openmode mode);
```

Here, the first argument of the open function defines the name and format of the file with the address of the file.

The second argument represents the mode in which the file has to be opened. The following modes are used as per the requirements.

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

Example of opening/creating a file using the open() function

```
#include<iostream>
#include <fstream>
using namespace std;
int main()
{
fstream new_file;
new_file.open("new_file",ios::out);
if(!new_file)
{
cout<<"File creation failed";
}
else
{
cout<<"New file created";
new_file.close(); // Step 4: Closing file
}
return 0;
}
```

Explanation

In the above example we first create an object to class fstream and name it ‘new_file’. Then we apply the open() function on our ‘new_file’ object. We give the name ‘new_file’ to the new file we wish to create and we set the mode to ‘out’ which allows us to write in our file. We use a ‘if’ statement to find if the file already exists or not if it does exist then it will going to print “File creation failed” or it will gonna create a new file and print “New file created”.

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;
}
```

Explanation

Here we first create a new file “new_file_write” using open() function since we wanted to send output to the file so, we use ios::out. As given in the program, information typed inside the quotes after Insertion Pointer “<<” got passed to the output file.

Reading from file

```
#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[70];//showing the content
        while(!new_file.eof())
        {
            new_file.getline(ch,70);
            new_file>>ch;
            cout<<ch ;

        }
        new_file.close();
        return 0;
    }
}
```

Close a File

It is simply done with the help of close() function.

Syntax: File Pointer.close()

Example

```
#include <iostream>
#include <fstream>
using namespace std;
int main()
{
fstream new_file;
new_file.open("new_file.txt",ios::out);
new_file.close();
return 0;
}
```