File Handling

Instructor: Isra Naz

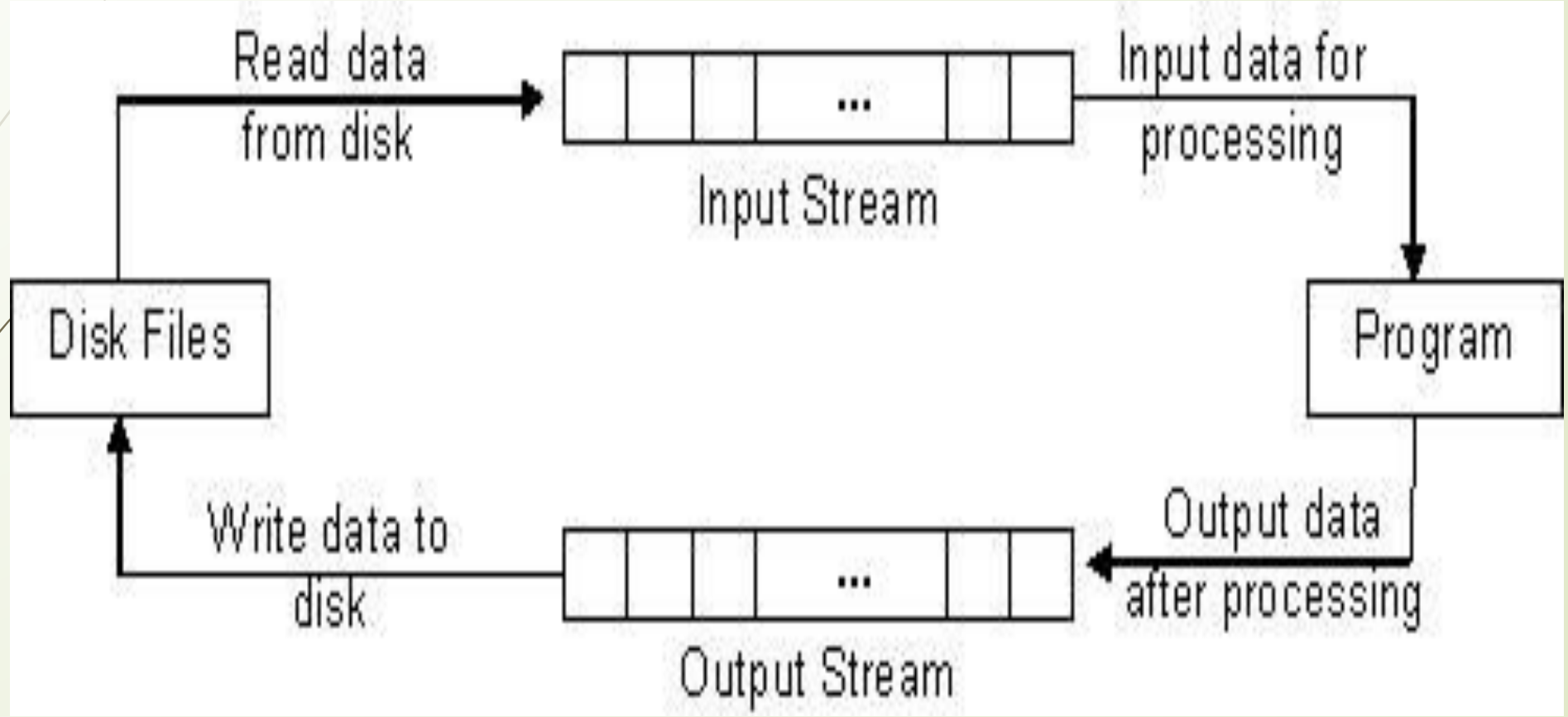
Introduction

- 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
    - It is an object of istream class
    - It is connected to the standard input devices such as keyboard
  - cout
    - It is an object of ostream class
    - It is connected to the standard output device such as monitor
- Can define other streams
  - To or from files
  - Used similarly as cin, cout

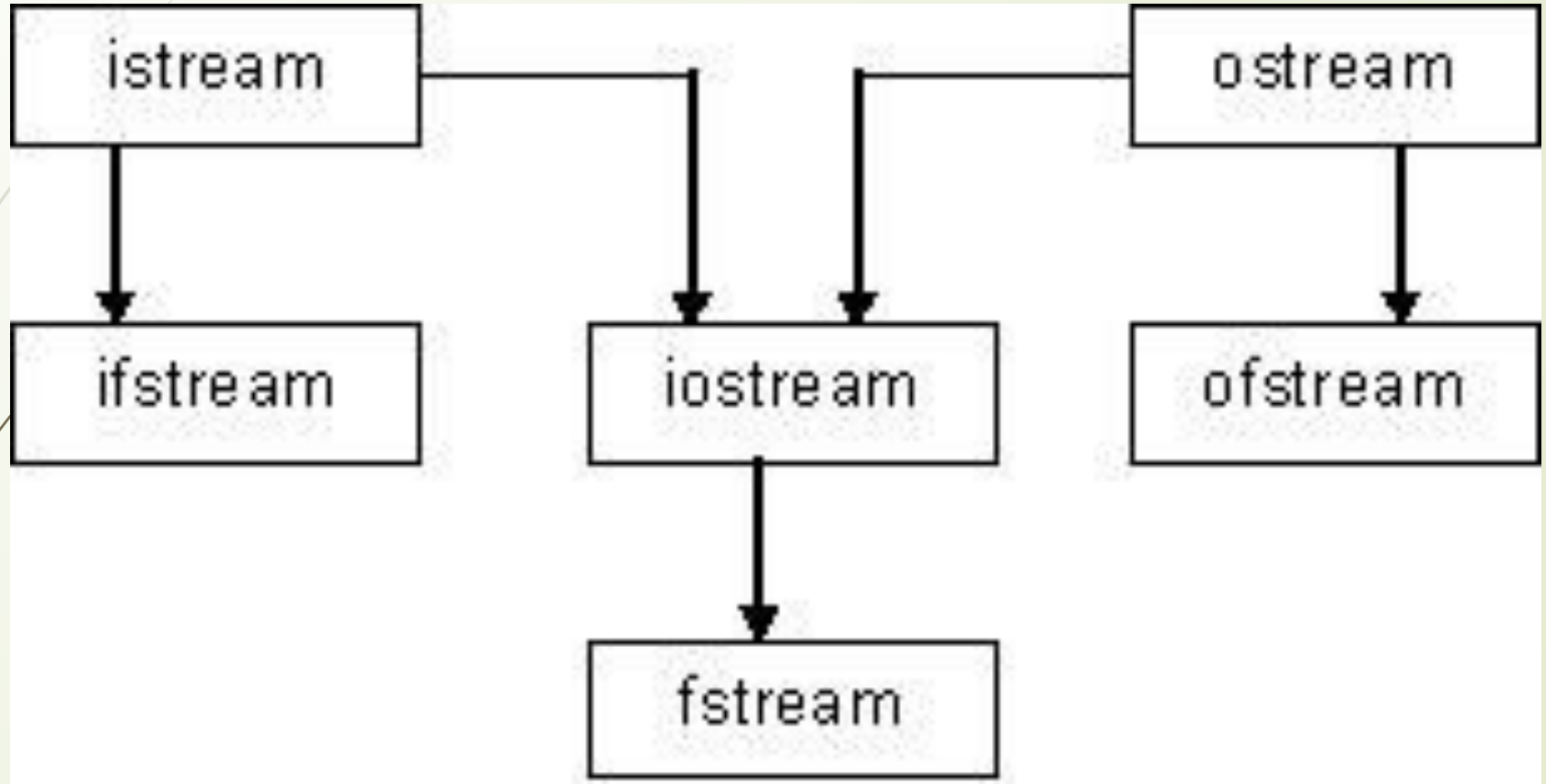
File input and output streams



Streams

- File Input Stream – reads data from disk file to the program.
- File output Stream – writes data to the disk file from the program.
- C++ provides the following classes to perform input and output of characters with files
  - ifstream – provides input/read operations on files
  - ofstream – provides output/write operations on files
  - fstream – supports for simultaneous input and output or read and write operations on files
- These classes are derived directly or indirectly from the classes istream and ostream.

Stream Classes



General File I/O Steps

- Declare a file name variable
- Associate the file name variable with the disk file name
- Open the file
- Use the file
- Close the file

Open File

- A file should be opened before it can be processed.
- A file pointer is declared and associated with the file to be opened.
- A file can be opened by first creating an object of ifstream, ofstream or fstream.
- The object is then associated with a real file.
- Any input or output operation performed on this stream object is applied to the physical file associated to it.

Open File

- **Syntax**
  - The member function open() of stream object is used to open a file as follows

    open(filename, mode);
  - *Filename:* It is name of file to be opened
  - *Mode:* It is the mode in which the file is to be opened. It is optional parameter.
- Default Open Modes :
  - ifstream ios::in
  - ofstream ios::out
  - fstream ios::in | ios::out
- We can combine the different modes using or symbol | .

Open File

- **Example**
  - ofstream new_file;
  - new_file.open(“new_file.txt”, ios::in | ios::app);

- Here, input mode and append mode are combined which represents the file is opened for writing and appending the outputs at the end.

File Opening Mode

File Mode	Meaning
ios::app	When file is opened in append mode, the data in the file remains and the new data is appended to the end of file.
ios::ate	A file is opened with the file pointer set to the end of file.
ios::binary	A file opened in binary mode.
ios::nocreate	When a file is opened in out mode and the file is not there, no new file created.
ios::noreplace	If the file is already available, it cannot be opened. If the file is not there, a new file is created.
ios::trunc	When a file is opened, the data in the file gets deleted.
ios::in	Opens the file for input or to read from file. This is default for ifstream.
ios::out	Opens the file for output or to write to the file. This is default for ofstream.

Verifying File Open

- The function is_open() is used to check if a stream object has opened a file successfully.
- The function has no parameters and returns a value of true if the file is open.
- It returns false if the file is not opened.
- For Example

```
if(! New_file)
cout<< "Error in opening the file.";
```

Close File

- The opened file should be closed when the input or output operations on the file are finished.
- The member function close() of stream object is used to close a file.
- It takes no parameters.
- Syntax
  - New_file.close();

Writing Data to File

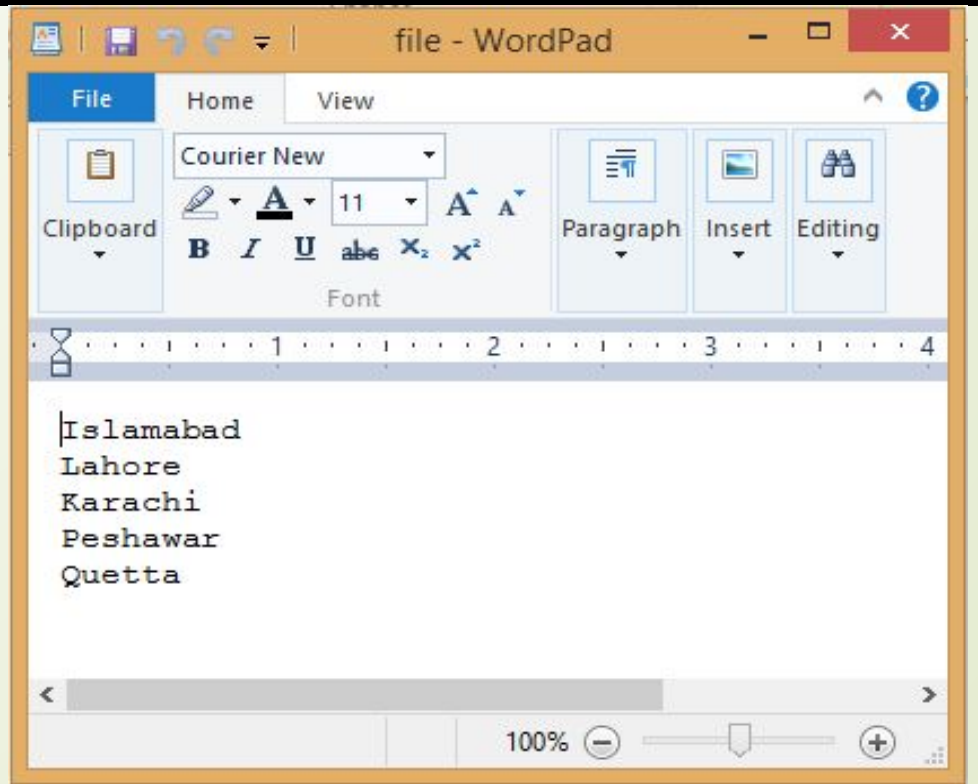
- The **insertion operator <<** is used with **cout object** to write the data in files.
- The operator is used with stream object of ofstream of fstream to write data to files.

Example

```
#include<fstream>
using namespace std;
int main()
{
    char city[50];
    ofstream file("F:\\file.txt");
    if(!file)
    {
    cout<<"File creation failed";
    exit(1);
    }
    else
    {
        for(int i=0; i<5;i++)
        {
            cout<<"Enter the name of any city: ";
            cin>>city;
            file<<city<<endl;
        }
    cout<<"New file created";
    }
    file.close();
}
```

Example

```
Enter the name of any city: Islamabad
Enter the name of any city: Lahore
Enter the name of any city: Karachi
Enter the name of any city: Peshawar
Enter the name of any city: Quetta
New file created
--------------------------------
```



Reading Data from file

- The **extractor operator >>** is used with **cin object** to read data from files.
- It is used with a stream object of ifstream of fstream to read data from files.

Detecting End-of-file

- The word **eof** stands for **end of file.**
- The **eof()** function is used to find if the control has reached the end of file or not.
- It returns true(1) if control has reached the end of file and returns false(0) otherwise.
- This member function is very useful in displaying all records in the file where the number of records are unknown.

Example

```
#include<iostream>
#include<fstream>
using namespace std;
int main()
{
    char city[50];
    ifstream file("F:\\file.txt");
    if(!file)
    {
    cout<<"Error in reading file";
    exit(1);
    }
    cout<<"The name of cities are: \n";
    while(!file.eof())
    {
        file>>city;
        cout<<city<<"\n";
    }
    file.close();
}
```

Example

```
The name of cities are:
Islamabad
Lahore
Karachi
Peshawar
Rawalpindi
--------------------------------
```

Lecture End