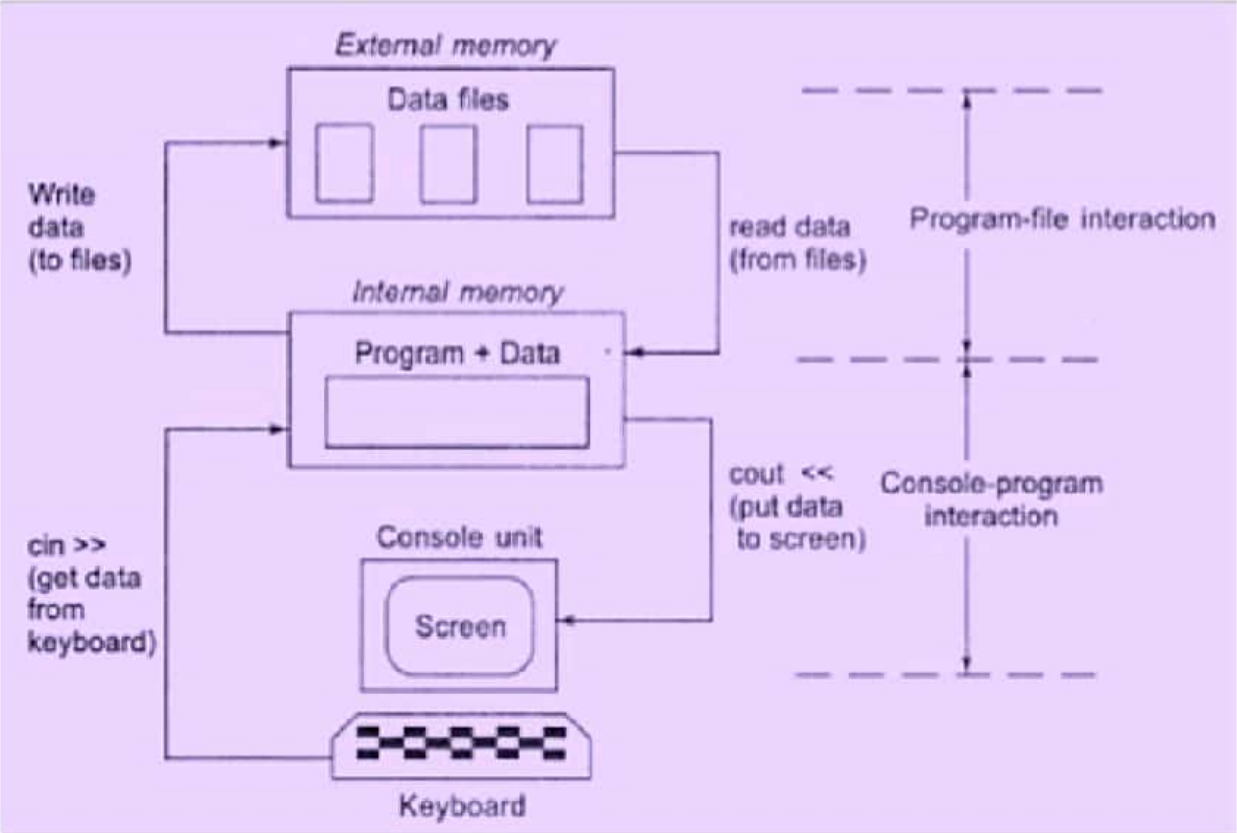
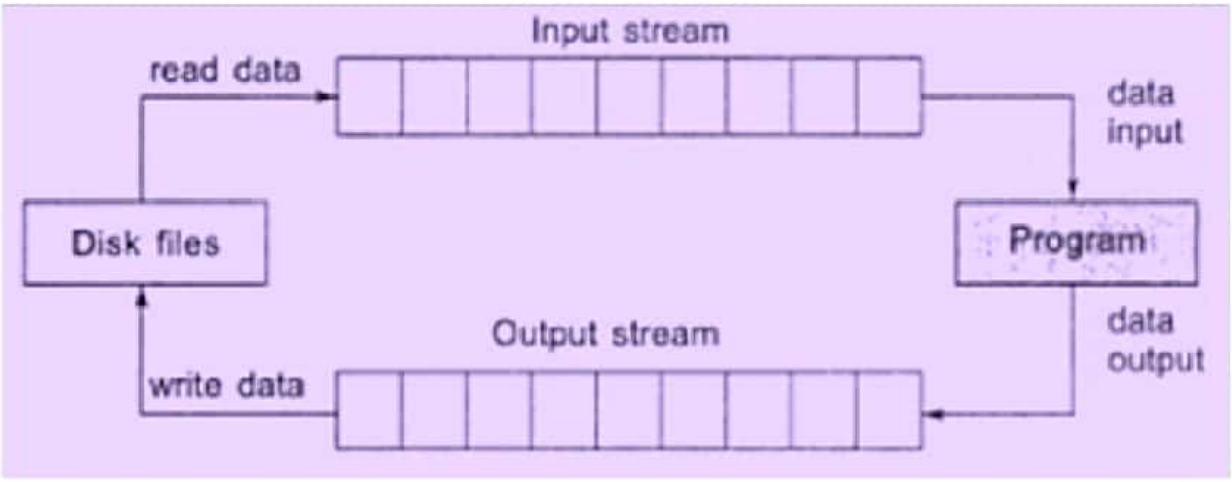
Introduction

- A file is a collection of related data stored on a particular area on the disk.
- Programs can be designed to perform the read and write operations on these files.
- Programs can involves either or both following communication:
  1. Data transfer between console unit and program
  2. Data transfer between program and disk file

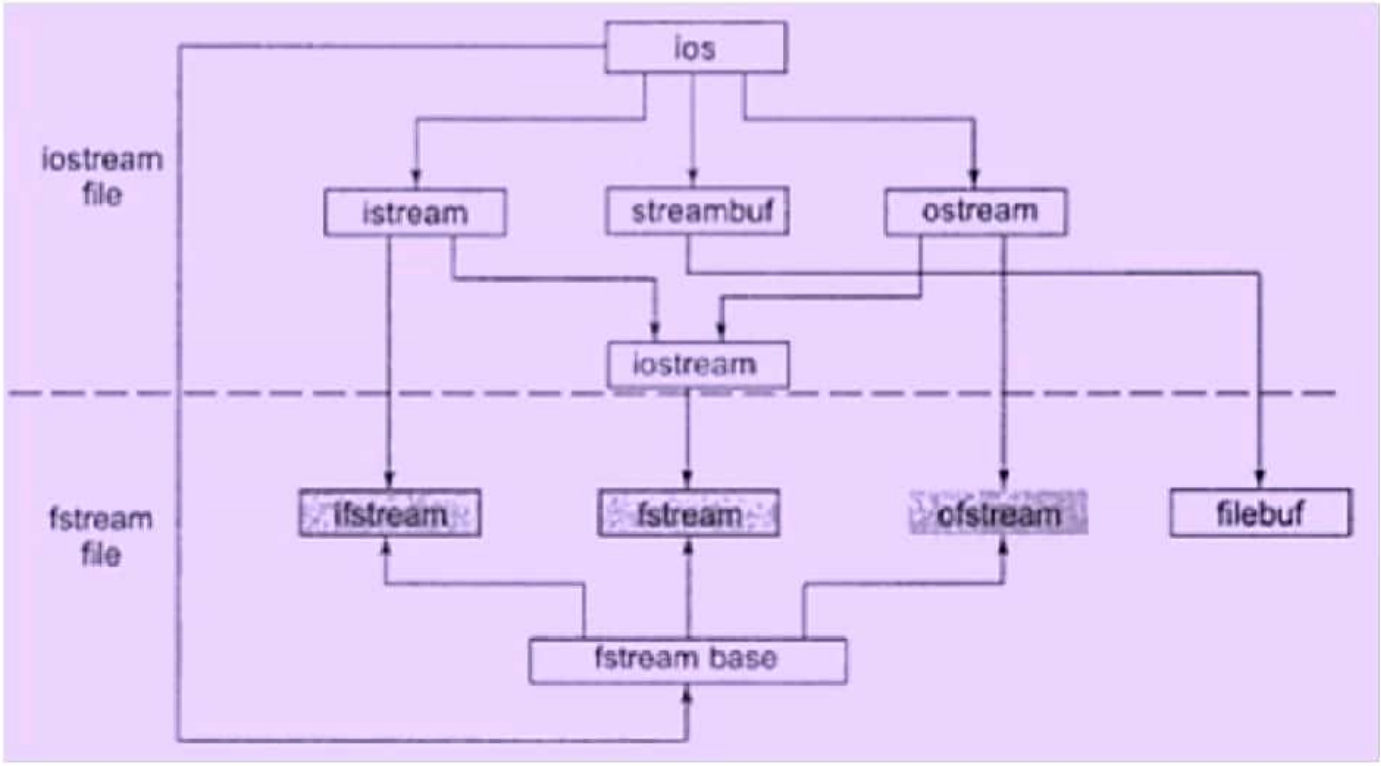
Introduction



Introduction



Classes for File stream operations



Classes for File stream operations

- Includes *ifsteam*, *ofstream* and *fstream*.
- These classes are derived from *fstreambase* and corrosponding *iostream* class.
- Are declared in *fstream*.

Details of file stream classes

Class	Contents
filebuf	Purpose is to set file buffers to read and write. Contains close() and open().
fstreambase	Provides operations common to the file streams. Serves as base for *ifsteam*, *ofstream* and *fstream*. Contains close() and open().
ifstream	Provides input operations Contains open() Inherits get(), getline(), read(), seekg(), tellg() from *istream*.
ofstream	Provides output operations Contains close() Inherits put(), write(), seekp(), tellp() from *ostream*.
fstream	Provides I/O operations Inherits al functions from *istream* and *ostream* from *iostream*.

Files (Streams)

- Files are used to store data in a relatively permanent form, on floppy disk, hard disk, tape or other form of secondary storage. Files can hold huge amounts of data if need be.

- Ordinary variables (even records and arrays) are kept in main memory which is temporary and rather limited in size. The following is a comparison of the two types of storage:

File Handling in C++

- **File Handling** concept in C++ language is used for store a data permanently in computer. Using file handling we can store our data in Secondary memory (Hard disk).
- **Why use File Handling**
  - Memory is **volatile**
  - Any data that you key in by keyboard while a program is running is also **volatile**
  - For **permanent storage**.
  - The transfer of input - data or output - data from one computer to another can be easily done by using files.

How to Achieve File Handling

- **Create** a **Handler** Object
- **Naming** a **file**
- **Opening** a **file**
- **Reading** data from **file**
- **Writing** data into **file**
- **Closing** a **file**

Reading a File

```
#include<fstream>

ifstream inputFile ;

inputFile.open("Record.txt");

c:\myProg\Record.txt

inputFile.close();
```

Handler

Opening the File

Closing the File

Functions use in File Handling

Function	Operation
open()	To create a file
close()	To close an existing file
get()	Read a single character from a file
put()	write a single character in file.
read()	Read data from file
write()	Write data into file.

Defining and Opening a File

- The function open() can be used to open multiple files that use the same stream object.

- **Syntax**

```
file-stream-class   stream-object;
stream-object.open ("filename");
```

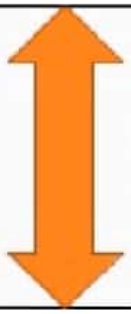
- **Example**

```
ofstream  inputFile;     // create stream
inputFile . open ("data1.txt"); // connect stream to data1
```

Defining and Opening a File

```
ofstream  inputFile;     // create stream
inputFile . open ("data1.txt"); // connect stream to data1
```

- **Opening File Using Constructor**



Both are Same: Creating and writing data in file

```
ofstream inputFile ("data1.txt");
```

Closing a File

- A file must be close after completion of all operation related to file. For closing file we need **close()** function.

```
inputFile.close();
```

Opening and Closing File

Process of Opening `inputFile.open("data1.txt");`

Process of Closing `inputFile.close("data1.txt");`

File Opening Mode

File Mode Parameter	Meaning
ios::app	Append mode. All output to that file to be appended to the end.
ios::ate	Open a file for output and move the read/write control to the end of the file.
ios::binary	file open in binary mode
ios::in	open file for reading only
ios::out	open file for writing only
ios::nocreate	open fails if the file does not exist
ios::noreplace	open fails if the file already exist
ios::trunc	delete the contents of the file if it exist

File Opening Mode

- The default value for **fstream** mode parameter is **in | out**. It means that file is opened for reading and writing when you use **fstream** class.

- When you use **ofstream** class, default value for mode is **out** and the default value for **ifstream** class is **in**.

File Opening Mode

- Both ios :: app and ios :: ate take us to the end of the file when it is opened. The difference between the two parameters is that the ios :: app allows us to add data to the end of file only, while ios :: ate mode permits us to add data or to modify the existing data any where in the file.
- The mode can combine two or more parameters using the bitwise OR operator (symbol |)

```
fstream file;
file.Open("data1 . txt", ios :: out | ios :: in);
```

File Handling in C++

We can read data from file and write data to file in three ways.

- Reading or writing characters using get() and put() member functions.
- Reading or writing formatted I/O using insertion operator (<<) and extraction operator (>>).
- Reading or writing object using read() and write() member functions.

Output File Handling

- Several things can be done with output files
  - Create a new file on the disk and write data in it
  - Open an existing file and overwrite it in such a manner that all the old information is lost from it and new information is stored
  - Open an existing file and append it in at the end
  - Open an existing file and modify in it in such a way that it can be written anywhere in the file

File Opening Mode

- **The syntax of open function is:**

```
handler.open(fileName, mode)
```

- **Example:**

```
ofstream myFile;
myFile.open("testfile.txt" , ios::out);
```

Open a File for Writing

```
#include<iostream>
#include<fstream>
using namespace std;
int main() {
ofstream ofile;            // declaring an object of class
ofile.open("Myfile.txt");  // open "file.txt" for writing data
/* write to a file */
ofile << "This is a line in a file" << endl;
ofile << "This is another line" << endl;
  /* write to a console */
cout << "Data written to file" << endl;
ofile.close();     // close the file
return 0;
}
```

Open a file and Append data to the end of the File-1

```
#include<iostream>
#include<fstream>
using namespace std;
int main() {
  char line[100];
  fstream file;   // declare an object of fstream class
  // open file in append mode
  file.open("file.txt", ios :: out | ios :: app);
  if (file.fail()) {    // check if file is opened successfully
    // file opening failed
    cout << "Error Opening file ... " << endl;
  }
```

Open a file and Append data to the end of the File-2

```
else {
    // proceed with further operations
    cout << " Enter Your Name : ";
    cin.getline(line, 100);
    file << line << endl;     // Append the line to the file
    cout << "Enter Your RollNo : ";
    cin.getline(line, 100);
    file << line << endl;
    cout << "Enter Your Age : ";
    cin.getline(line, 100);
    file << line << endl;
    cout << "Line written into the file" << endl;
    }
  return 0;
}
```

File Handling

- C++'s standard library called **fstream**, defines the following classes to support file handling.

- **ofstream class :** Provides methods for writing data into file. Such as, open(), put(), write(), seekp(), tellp(), close(), etc.
- **ifstream class :** Provides methods for reading data from file. Such as, open(),get(), read(), seekg(), tellg(), close(), etc.
- **fstream class :** Provides methods for both writing and reading data from file. The fstream class includes all the methods of ifstream and ofstream class.

Read Write Object using read() and write() Function

- There are member functions **read( )** and **write( )** in the **fstream** class which allows reading and writing of class objects.
- These functions can also be used to write array elements into the file.
- The write() function is used to write object or record (sequence of bytes) to the file. A record may be an array, structure or class.
- The read() function is used to read object (sequence of bytes) to the file.