Files in C++

File

- File is used for persistent data storage. It resides on non-volatile secondary storage devices to retain the data for longer time
- Data in files gets lost only when someone erases the data or when hardware storage media fails due to some reasons
- Each file has End-Of-File marker and it varies as per operating system. Actually all files are managed by operating system.

Need of file

- Terminal I/O using streams cin and cout stores the data in iostreams temporarily
- Data in these streams are lost when the execution of an application gets terminated or when power fails
- To have the data available even after program termination, it should be stored in files on non-volatile memory devices like disks
- Basically There are two types of files
  - 1) Text Files
  - 2) Binary Files

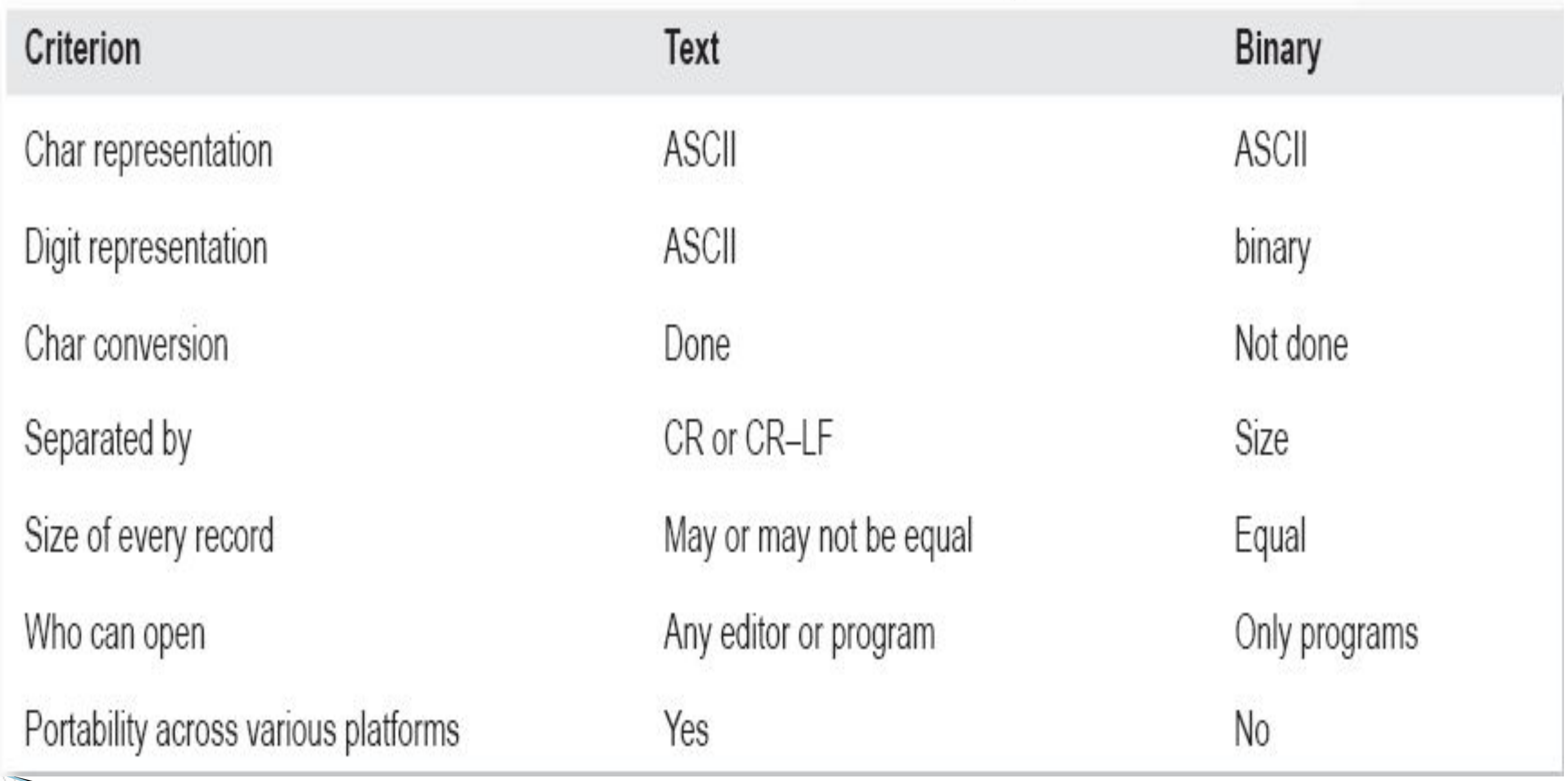
TEXT Files

- There are two different types of streams, namely, text streams and binary streams.
- The text stream accepts data in ASCII form. If 0 is typed, the ASCII value of 0, that is,48, would be inserted.
- If the <Enter> key is pressed, two characters, that is, CR and LF (carriage return and line feed) are inserted in the Windows environment. This is known as *conversion*. When one reads back, conversion is again needed to convert from CR–LF to the visual interpretation of the <Enter> key.

BINARY Files

- Binary streams are pure binary streams; when 0 is typed, binary zeroes are inserted in the stream. When one writes to binary streams, no conversion takes place.
- For example, if the user enters 15 in a binary file, the binary value equivalent to 15 (00000111) is entered in the stream.
- If the user presses the <Enter> key, only the value 13 is sent, unlike two values in the case of a text file.
- The meaning of this is that when one reads, no conversion is needed in binary.

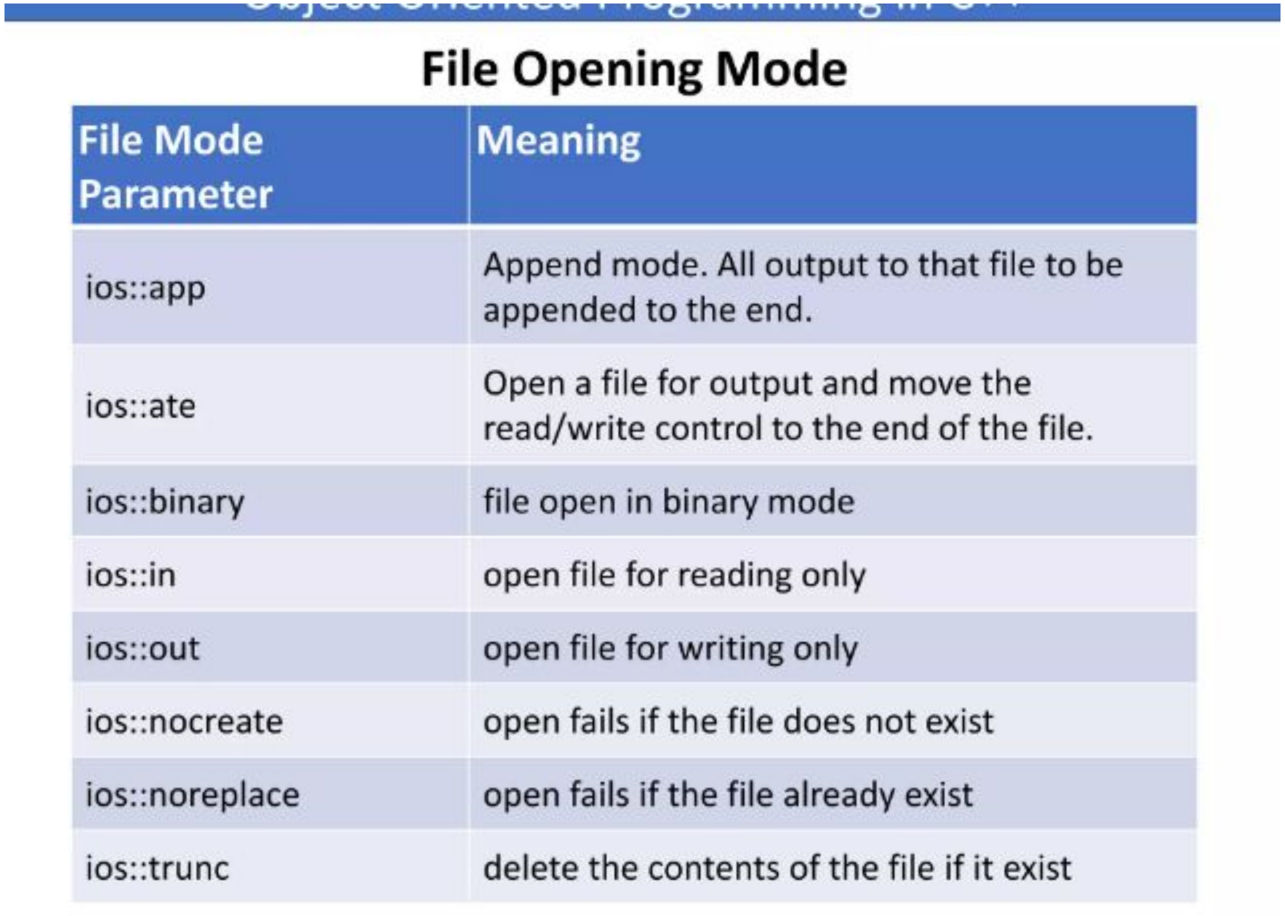
TEXT AND BINARY STREAMS


Differences between text and binary stream files

Criterion	Text	Binary
Char representation	ASCII	ASCII
Digit representation	ASCII	binary
Char conversion	Done	Not done
Separated by	CR or CR–LF	Size
Size of every record	May or may not be equal	Equal
Who can open	Any editor or program	Only programs
Portability across various platforms	Yes	No

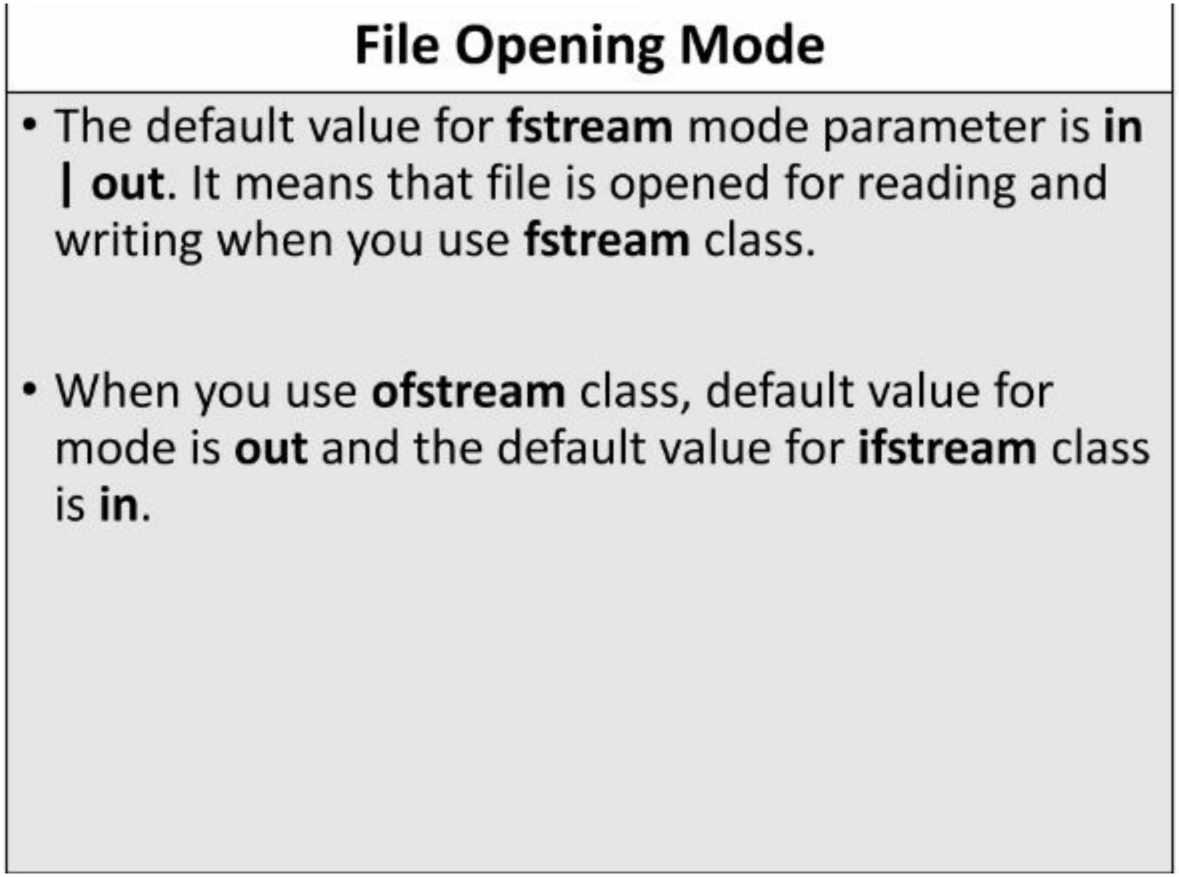
Text Files

- Defining Files :
  - 1. ifstream <filename> (input file stream)
  - 2. ofstream <filename> (output file stream)
  - 3. fstream <filename> (I/O file stream)



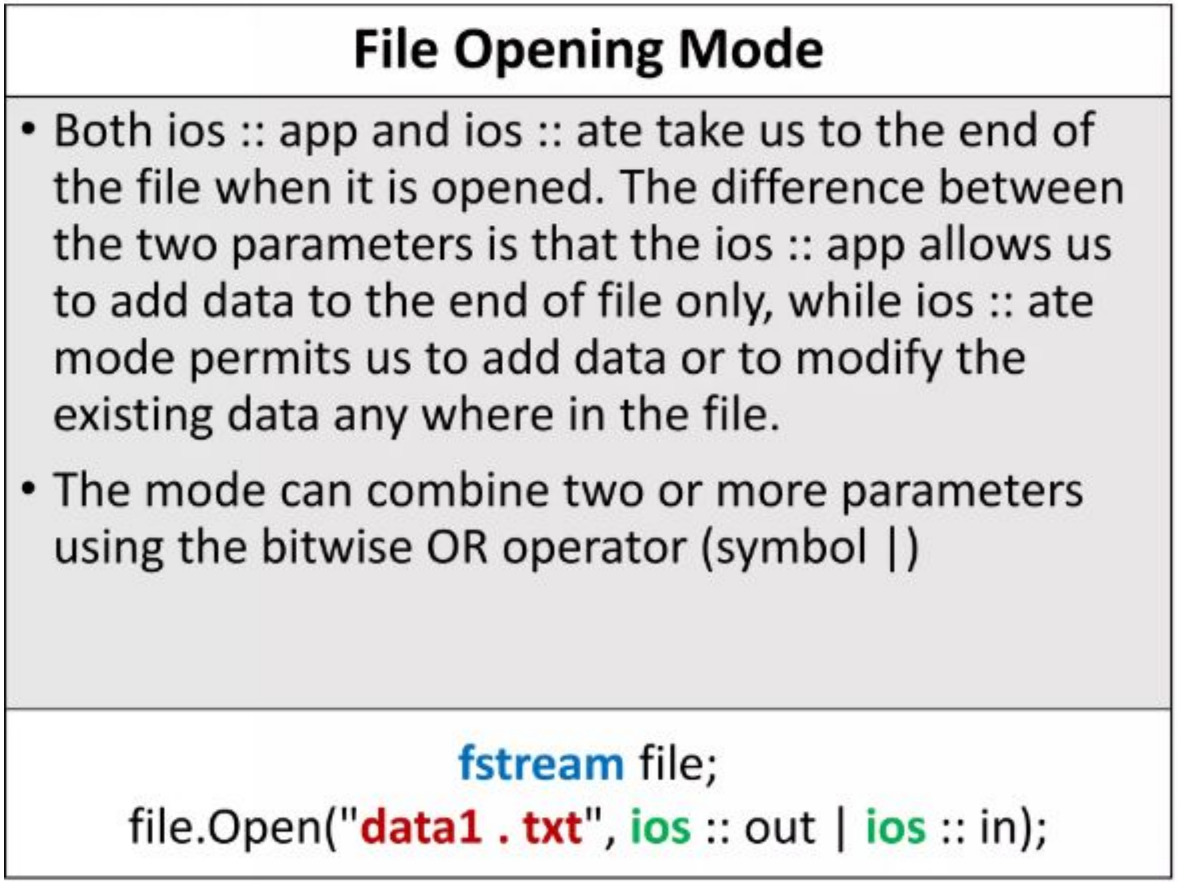
File Opening Mode

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

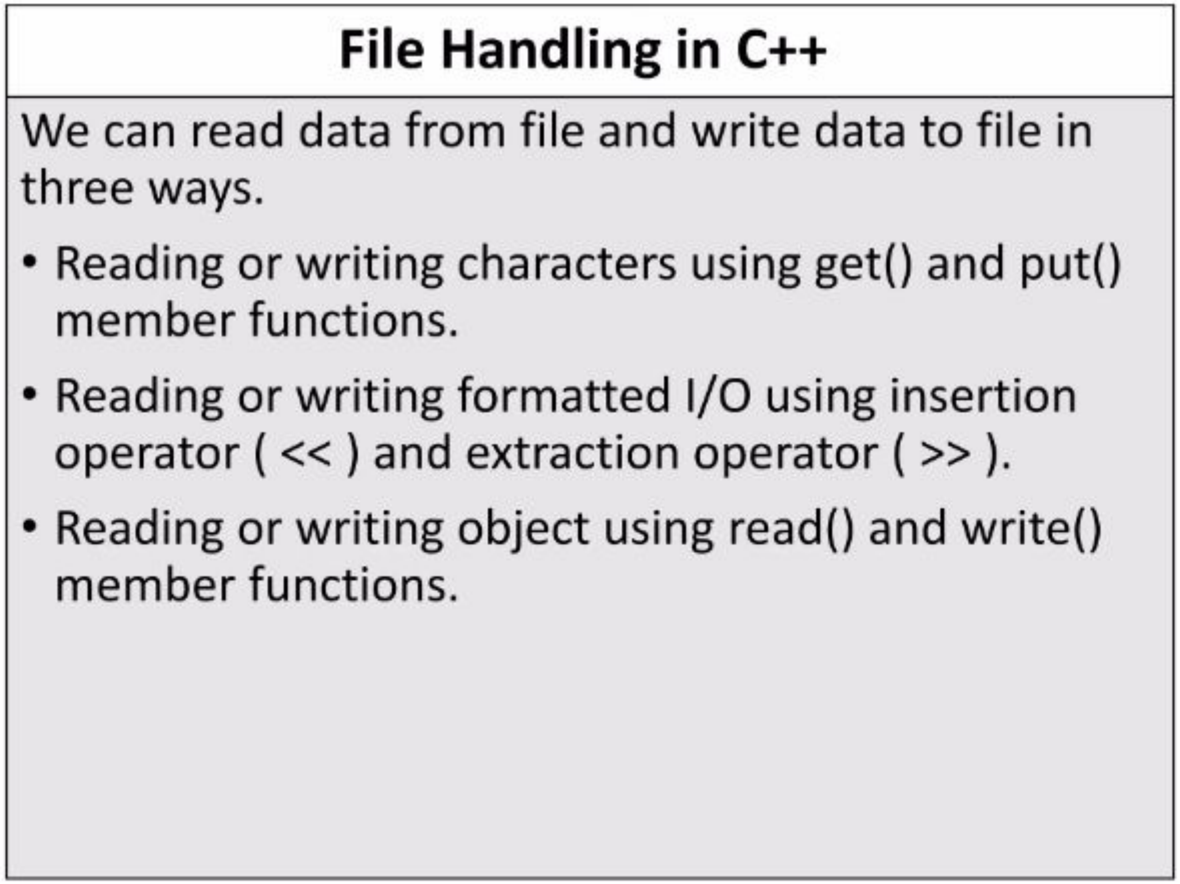
- The default value for **fstream** mode parameter is **in | out**. It means that file is opened for reading and writing when you use **fstream** class.
- When you use **ofstream** class, default value for mode is **out** and the default value for **ifstream** class is **in**.



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 |)

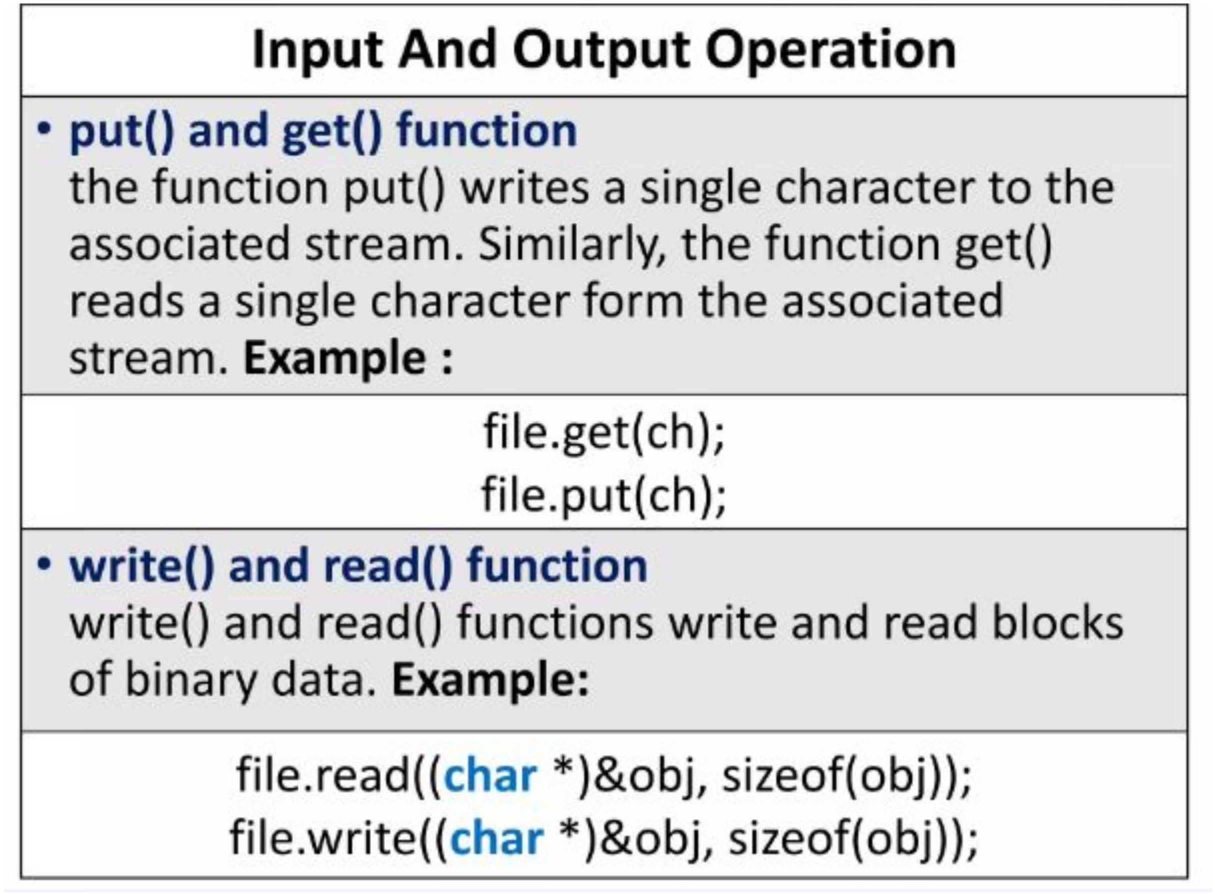
```
fstream file;
file.Open("data1 . txt", ios :: out | ios :: in);
```



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.

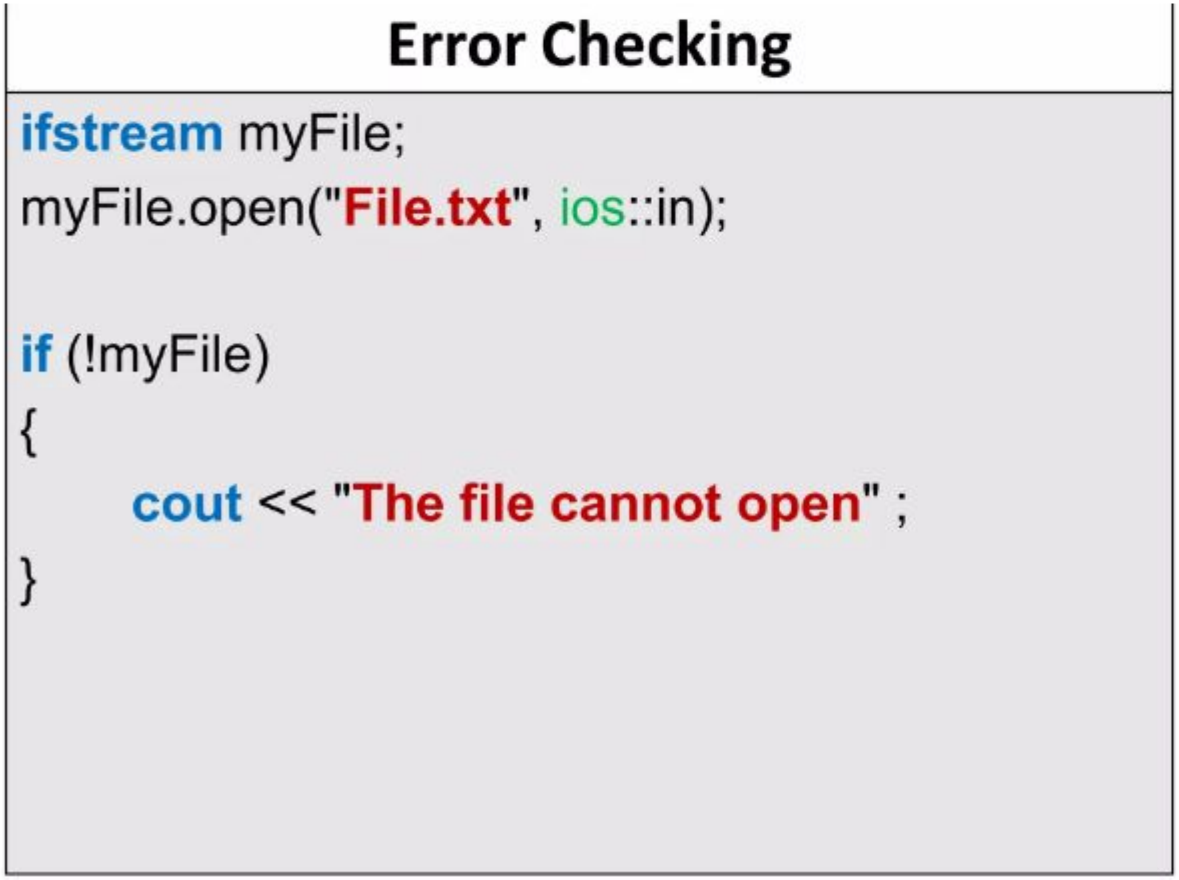


Input And Output Operation
• **put() and get() function** the function put() writes a single character to the associated stream. Similarly, the function get() reads a single character form the associated stream. **Example :**

file.get(ch);
file.put(ch);

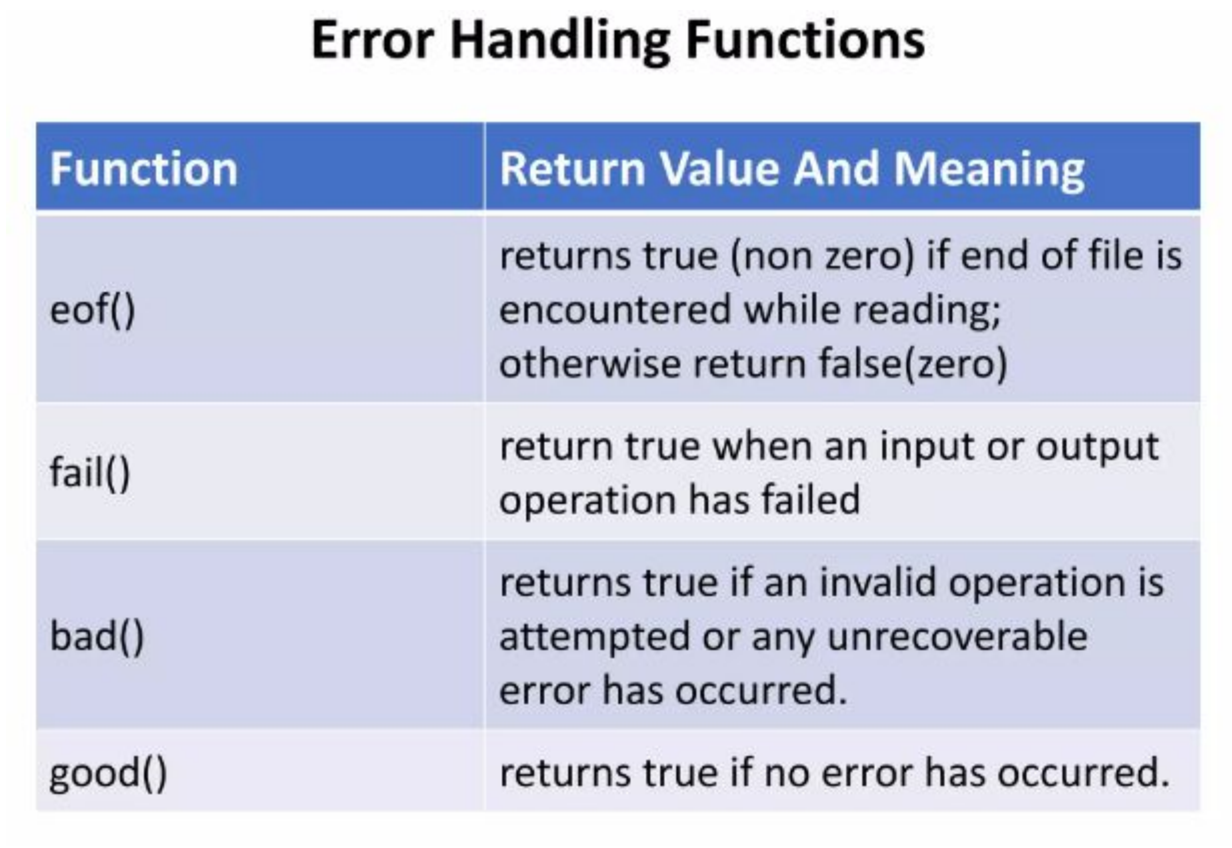
• **write() and read() function** write() and read() functions write and read blocks of binary data. **Example:**

file.read((char *)&obj, sizeof(obj));
file.write((char *)&obj, sizeof(obj));



Error Checking

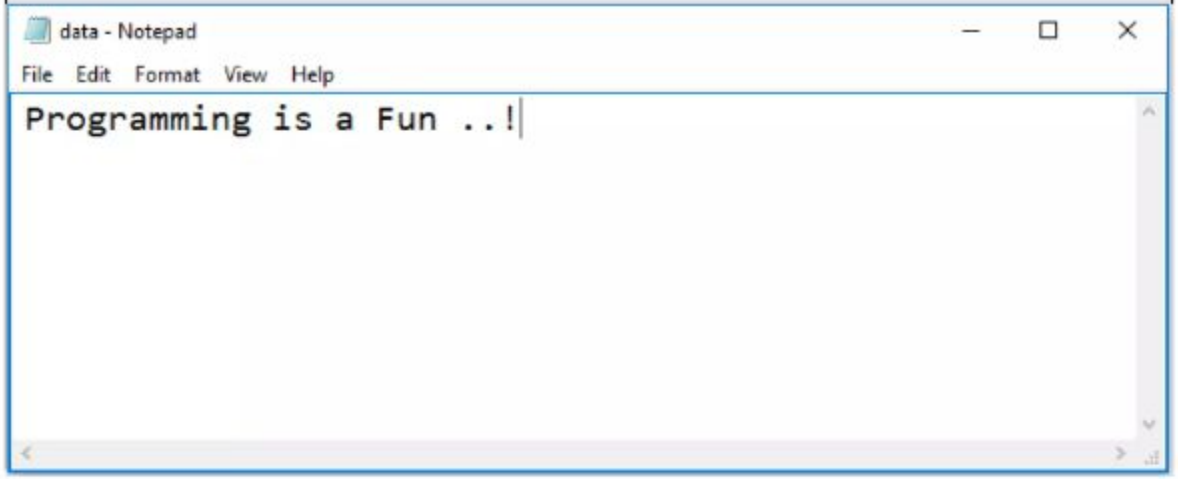
```
ifstream myFile;
myFile.open("File.txt", ios::in);

if (!myFile)
{
      cout << "The file cannot open" ;
}
```



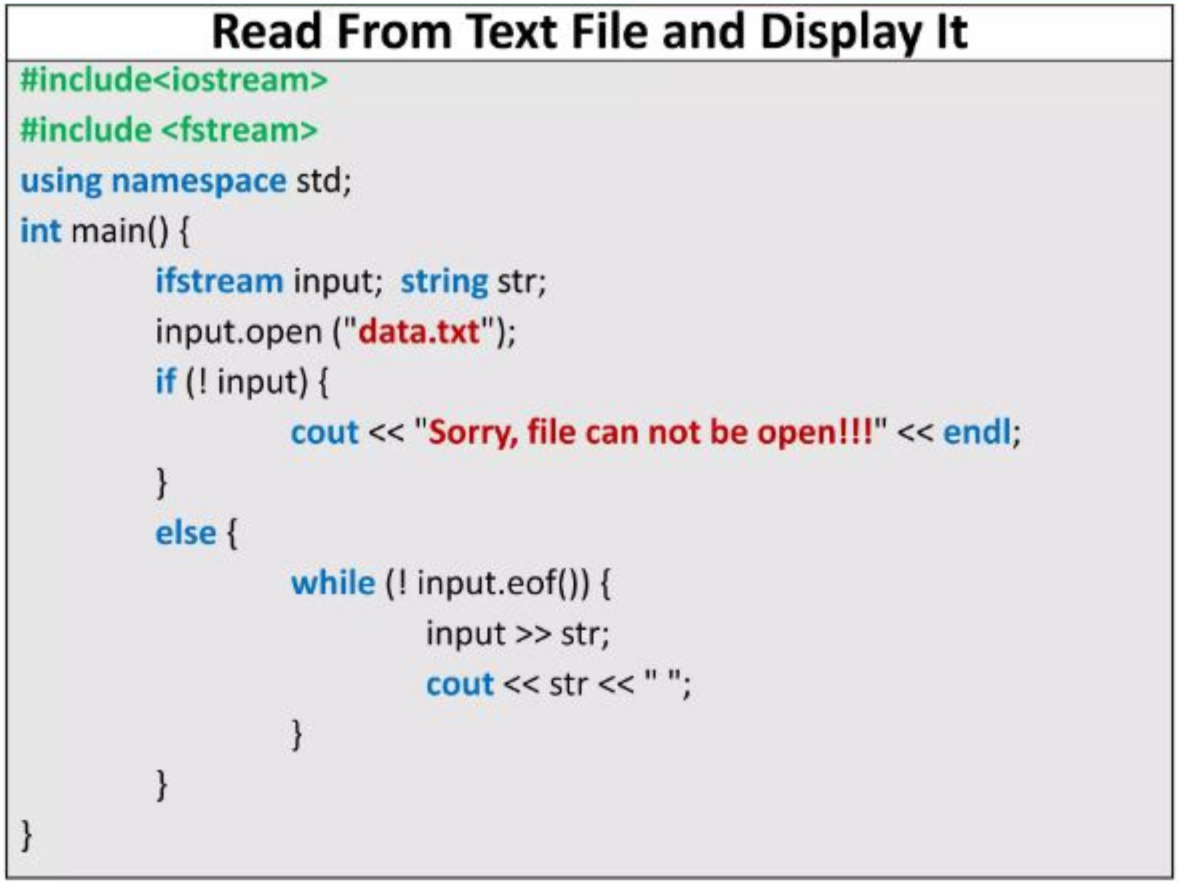
Error Handling Functions

Function	Return Value And Meaning
eof()	returns true (non zero) if end of file is encountered while reading; otherwise return false(zero)
fail()	return true when an input or output operation has failed
bad()	returns true if an invalid operation is attempted or any unrecoverable error has occurred.
good()	returns true if no error has occurred.

Read the Following File .
File Name is “data.txt”.



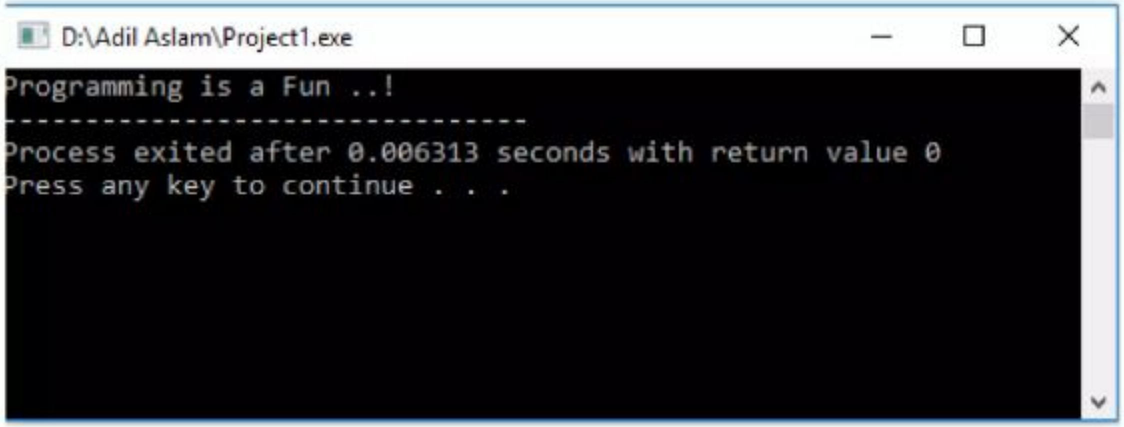
```
Programming is a Fun ..!
```



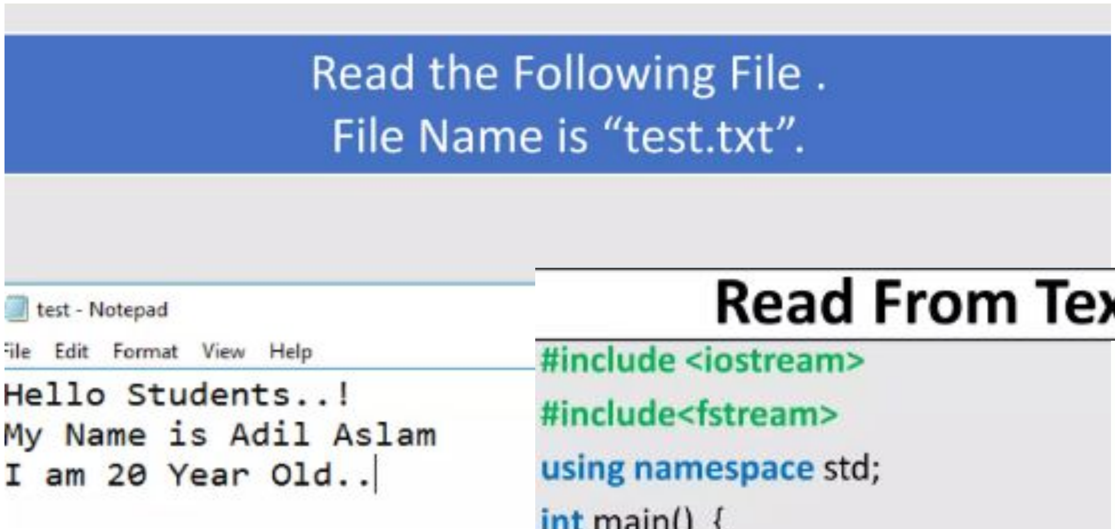
Read From Text File and Display It

```
#include<iostream>
#include <fstream>
using namespace std;
int main() {
        ifstream input;  string str;
        input.open ("data.txt");
        if (! input) {
                cout << "Sorry, file can not be open!!!" << endl;
        }
        else {
                while (! input.eof()) {
                        input >> str;
                        cout << str << " ";
                }
        }
}
```

Output of the Previous Program is :

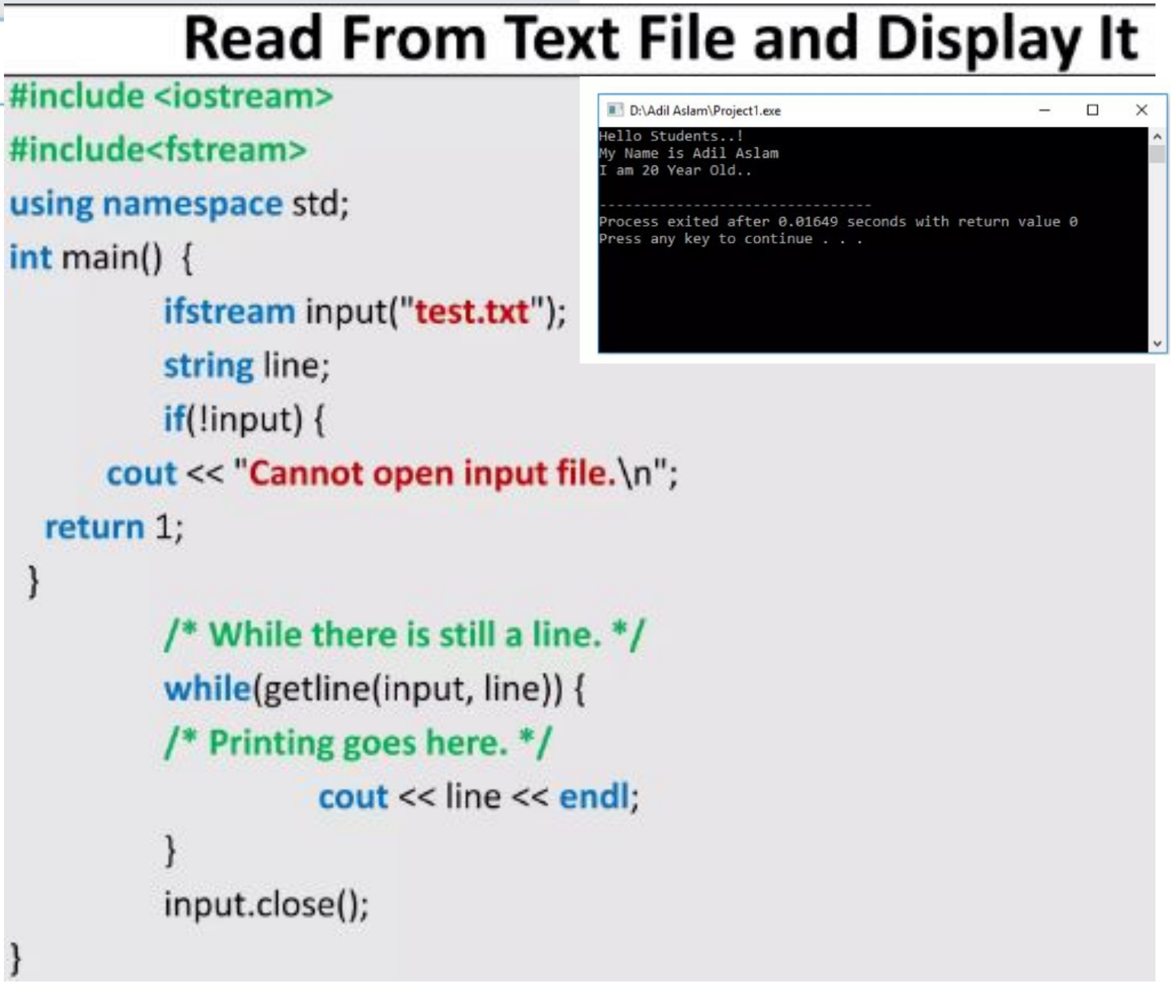


```
Programming is a Fun ..!
--------------------------------
Process exited after 0.006313 seconds with return value 0
Press any key to continue . . .
```



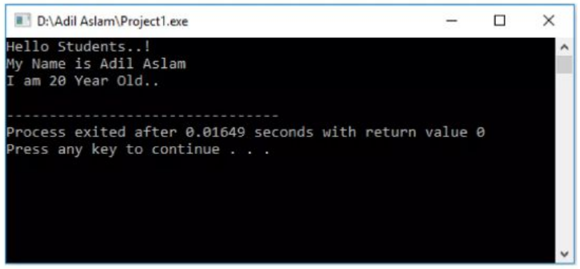
Read the Following File .
File Name is “test.txt”.

```
Hello Students..!
My Name is Adil Aslam
I am 20 Year Old..
```

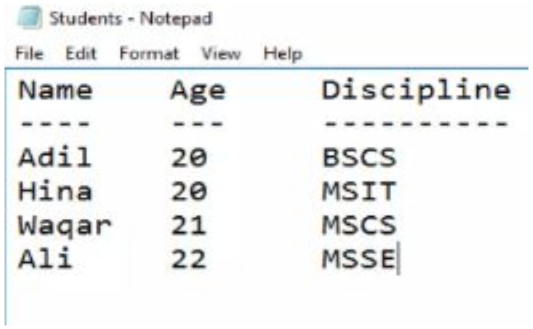


Read From Text File and Display It

```
#include <iostream>
#include<fstream>
using namespace std;
int main()  {
            ifstream input("test.txt");
            string line;
            if(!input) {
        cout << "Cannot open input file.\n";
   return 1;
  }
            /* While there is still a line. */
            while(getline(input, line)) {
            /* Printing goes here. */
                        cout << line << endl;
            }
            input.close();
}
```

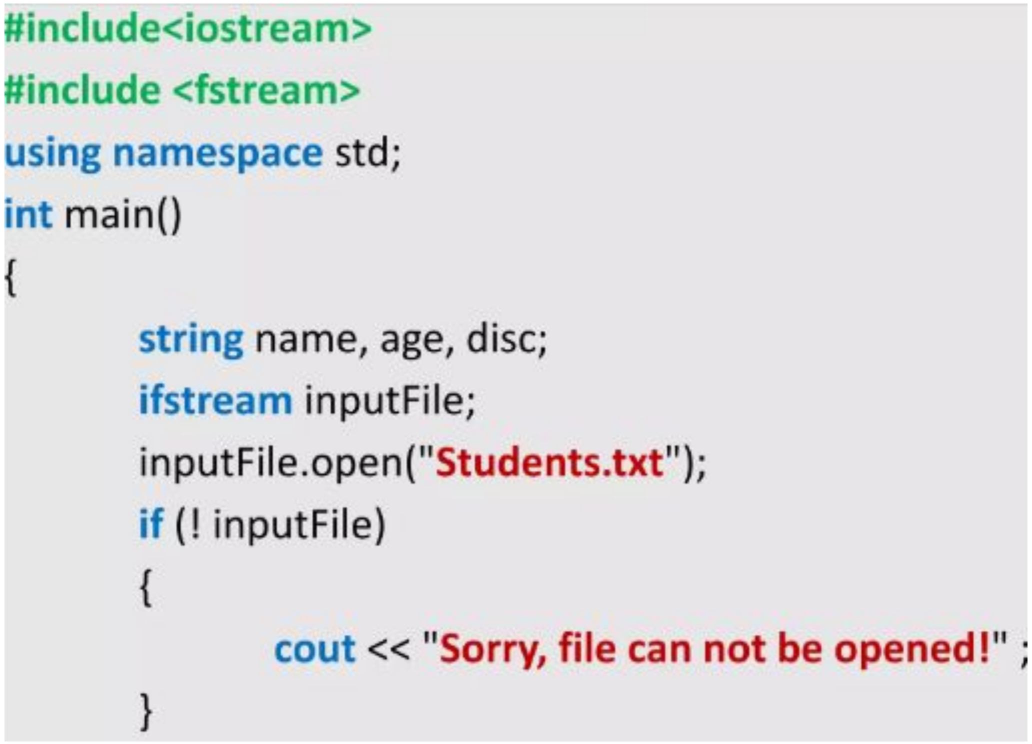


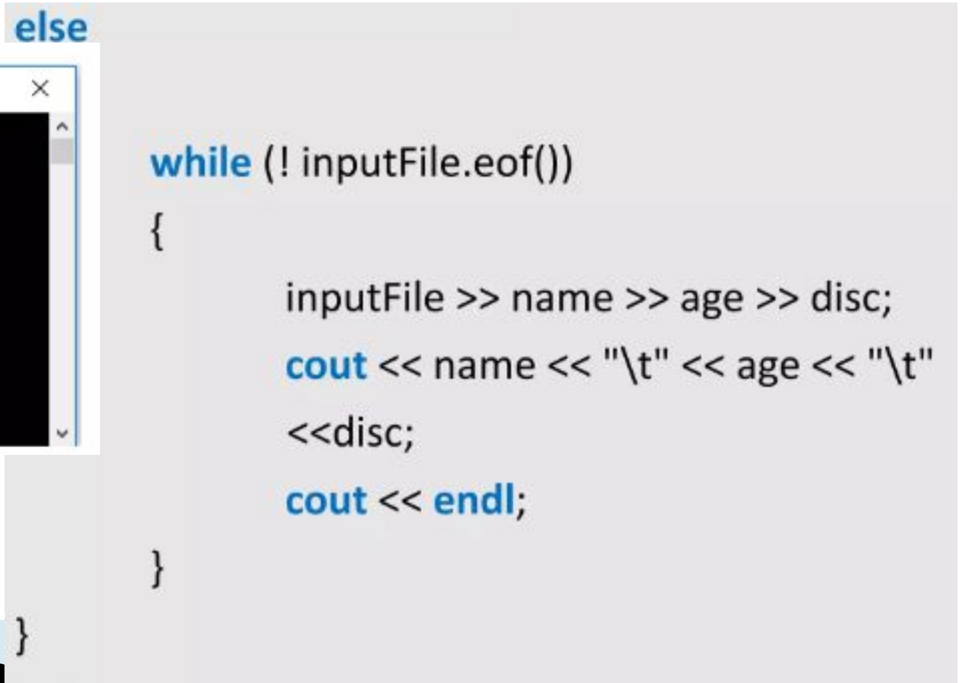
```
Hello Students..!
My Name is Adil Aslam
I am 20 Year Old..

--------------------------------
Process exited after 0.01649 seconds with return value 0
Press any key to continue . . .
```



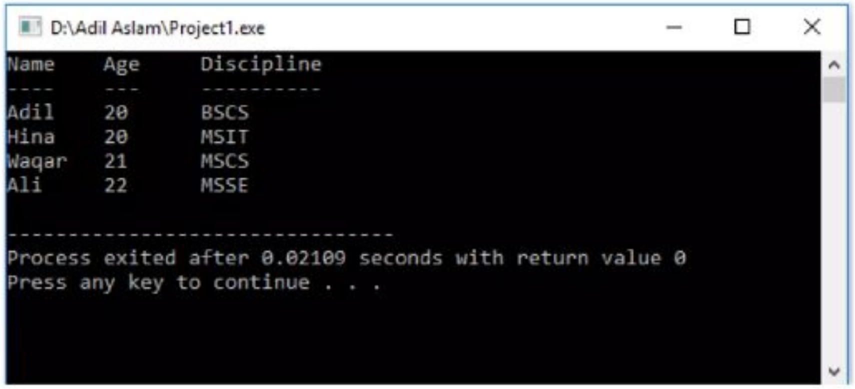
Students - Notepad

```
Name      Age      Discipline
----      ---      ----------
Adil      20       BSCS
Hina      20       MSIT
Waqar     21       MSCS
Ali       22       MSSE
```





```
#include<iostream>
#include <fstream>
using namespace std;
int main()
{
        string name, age, disc;
        ifstream inputFile;
        inputFile.open("Students.txt");
        if (! inputFile)
        {
                cout << "Sorry, file can not be opened!" ;
        }
else

        while (! inputFile.eof())
        {
                inputFile >> name >> age >> disc;
                cout << name << "\t" << age << "\t"
                <<disc;
                cout << endl;
        }
}
```



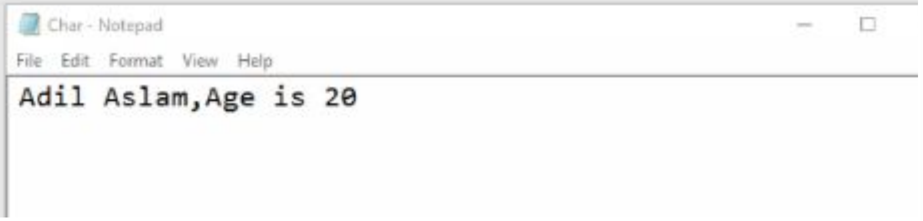
D:\Adil Aslam\Project1.exe

```
Name     Age     Discipline
----     ---     ----------
Adil     20      BSCS
Hina     20      MSIT
Waqar    21      MSCS
Ali      22      MSSE

--------------------------------
Process exited after 0.02109 seconds with return value 0
Press any key to continue . . .
```

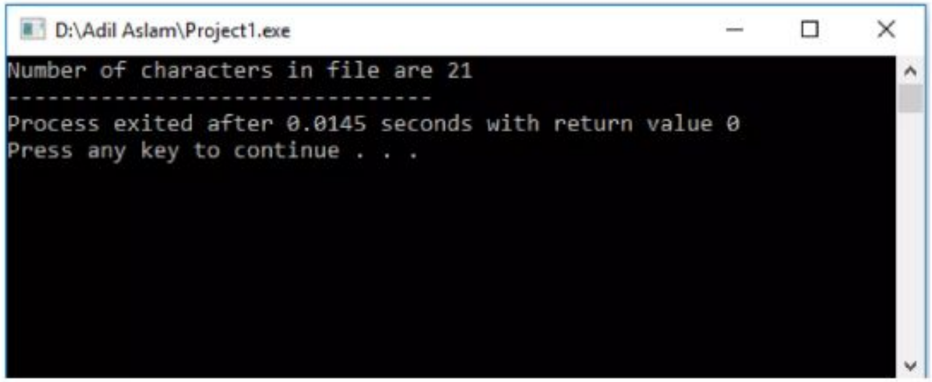
Example No. 4

Read the Following File .
File Name is “Char.txt”.

Count Number of Characters in this File



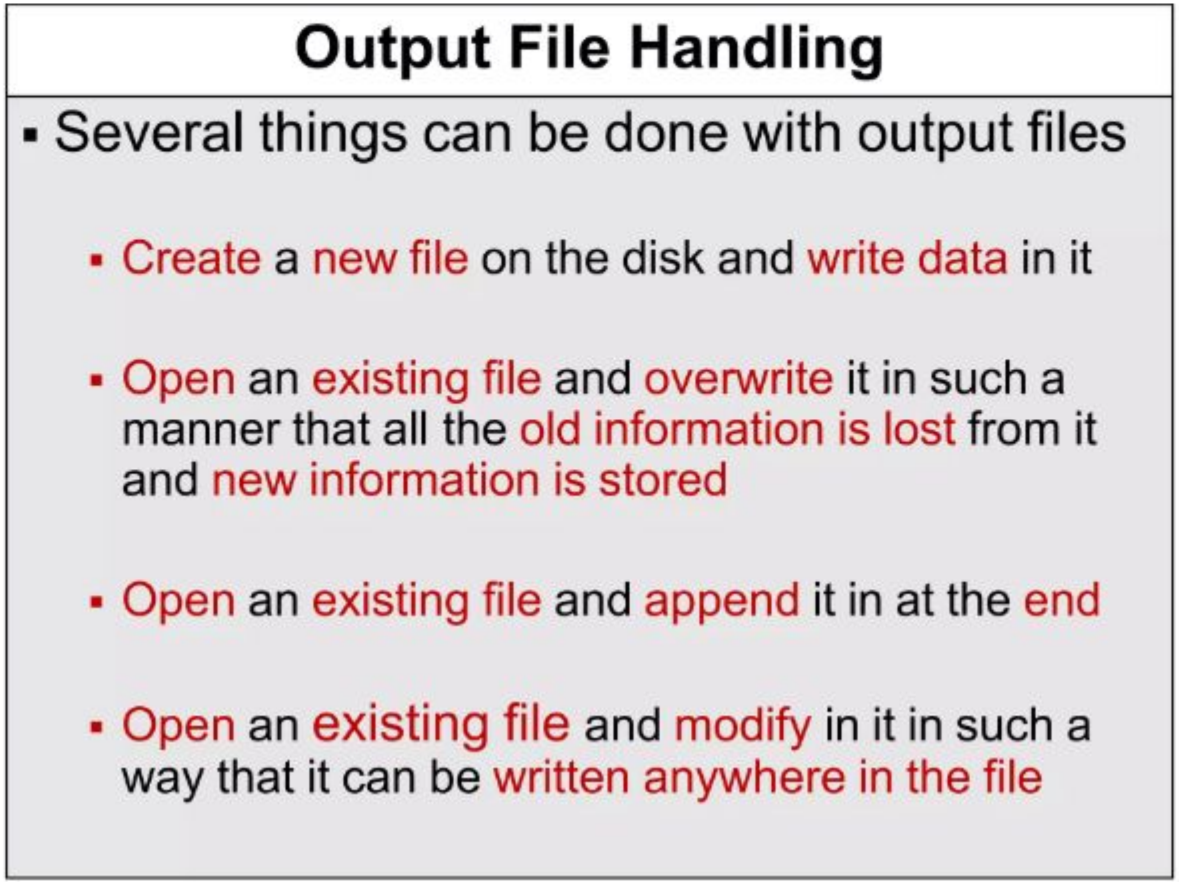
```
Adil Aslam,Age is 20
```



```
Number of characters in file are 21
--------------------------------
Process exited after 0.0145 seconds with return value 0
Press any key to continue . . .
```

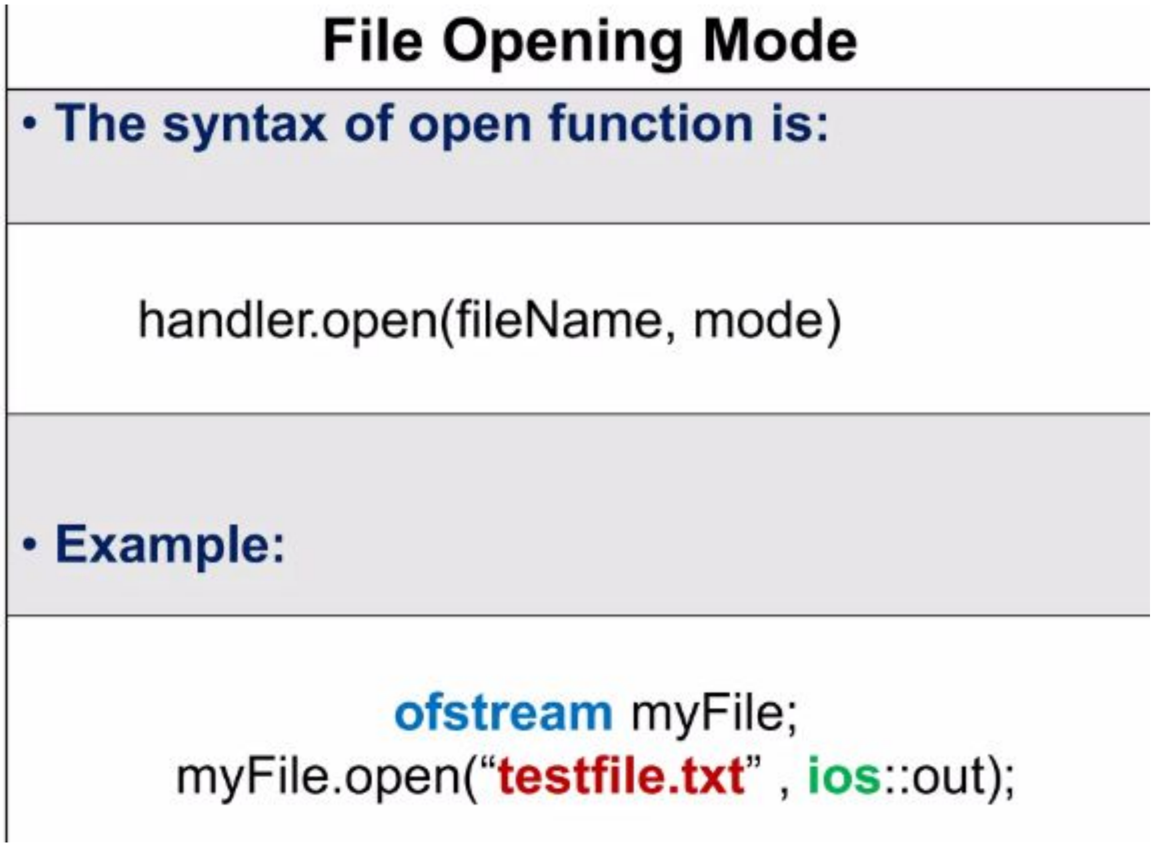
Program to Count Number of Characters-1

```
#include <iostream>
#include <fstream>
using namespace std;
int main()
{
   ifstream fin;
   fin.open("Char.txt");

   int count = 0;
   char ch;
         if (! fin)
         {
                  cout << "Sorry, file can not be opened!" ;
         }
        else
        {
                 while(!fin.eof())
                 {
                          fin.get(ch);
                          count++;
                 }
        cout << "Number of characters in file are " << count;
  }
  fin.close();
  return 0;
```



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);
```

- https://www.slideshare.net/AdilAslam4/file-handling-in-c-69352960

File operations Text Files

```
// basic file operations
#include <iostream>
#include <fstream>
Using namespace std;
int main ()
{
ofstream outfile;
outfile.open ("example.txt");
outfile<< "This is my first attempt to use a file\n";
outfile.close();
return 0;
}
```

Program for Working on a text file

```
#include <iostream>
#include <string>
#include <fstream>
using namespace std;
int main()
{
string InputLine, OutputLine;
ofstream EntryFile("FewLines.dat")
cout << "Input :" << endl;
while(true)
{
cin >> InputLine;
if(InputLine == "End") break;
EntryFile << InputLine << endl;
// Writing to EntryFile
}
EntryFile.close();
cout << "Output: " << endl;
ifstream DisplayFile("FewLines.dat");
while(IDisplayFile.eof())
{
DisplayFile >> OutputLine;
cout << OutputLine << "\n";
}
DisplayFile.close();
return 0;
}
```

Input

```
It was a fi ght
```

Output

```
It
was
a
fight
```

Using get() and put( ) in Text Files

cin.get(ch) cout.put(ch)

```
#include <iostream>
#include <string>
#include <fstream>
using namespace std;
#include <iomanip>
int main()
{
char ch;
ofstream EntryFile("FewLines.dat");
while(true)
{
cin.get(ch);
if(ch == '$') break;
EntryFile << ch;
}
EntryFile.close();
ifstream DisplayFile("FewLines.dat");
while(!DisplayFile.eof())
{
// Do not skip white space
DisplayFile.unsetf(ios::skipws);
DisplayFile >> ch;
cout << ch;
}
DisplayFile.close();
return 0;
}
```

Input

```
The battle
between One and Another
$
```

Output

```
The battle
between One and Another
```

Using getline( ) in Text Files

```
#include <iostream>
#include <cstring>
#include <fstream>
using namespace std;
int main()
{
char InputLine[80], OutputLine[80];
ofstream EntryFile("FewLines.dat");
while(true)
{
cin.getline(InputLine, 80);
if(!strcmp(InputLine, "End")) break;
EntryFile << InputLine << endl;
}
EntryFile.close();
ifstream DisplayFile("FewLines.dat");
while(!DisplayFile.eof())
{
DisplayFile.getline(OutputLine, 80);
cout << OutputLine << endl;
}
DisplayFile.close();
return 0;
}
```

Input

```
Imagination is
more important
than knowledge
End
```

Output

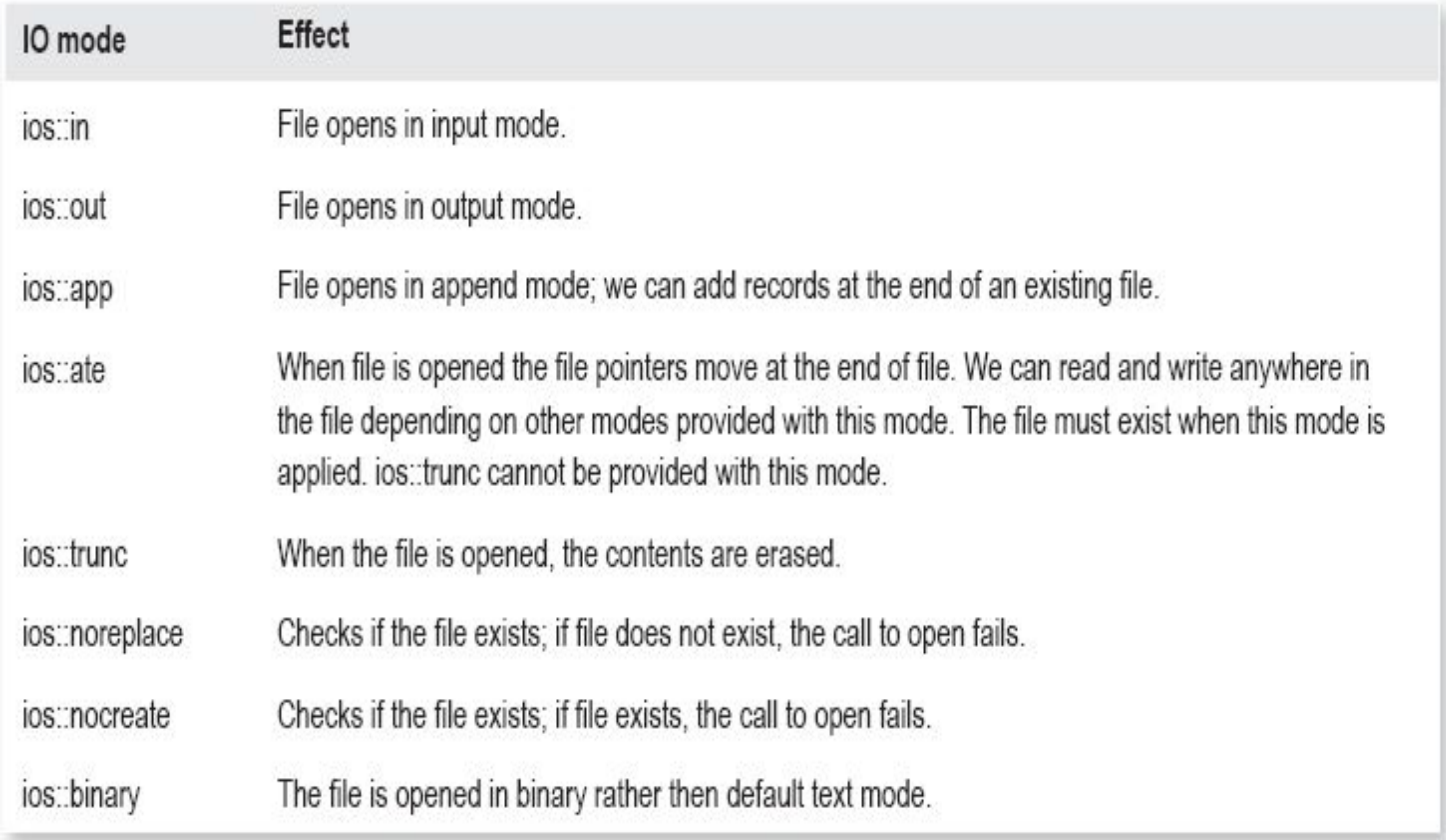
```
Imagination is more important than
knowledge
```

BINARY FILES

Opening a Binary File

- A binary file can be opened using a constructor. The constructors for ofstream and ifstream that we have seen so far are acceptable for text files.
- For binary files, another **constructor** with two arguments is needed. The first argument is the **name of the file** and the second one is the **file mode**.

I/O Modes



IO mode	Effect
ios::in	File opens in input mode.
ios::out	File opens in output mode.
ios::app	File opens in append mode; we can add records at the end of an existing file.
ios::ate	When file is opened the file pointers move at the end of file. We can read and write anywhere in the file depending on other modes provided with this mode. The file must exist when this mode is applied. ios::trunc cannot be provided with this mode.
ios::trunc	When the file is opened, the contents are erased.
ios::noreplace	Checks if the file exists; if file does not exist, the call to open fails.
ios::nocreate	Checks if the file exists; if file exists, the call to open fails.
ios::binary	The file is opened in binary rather then default text mode.

Opening a Binary File

```
// Using open methods
ofstream  MSC_StudFile_Out;
MSC_StudFile_Out.open("MSC.dat", ios::out | ios::binary | ios::trunc);

// Using constructor
ifstream MSC_StudFile_In("MSC.dat", ios::in | ios::binary);
```

OR

```
ofstream MSC_StudFile_Out("MSC.dat", ios::out | ios::binary | ios::trunc);

ifstream MSC_StudFile_In;
MSC_StudFile_In.open("MSC.dat", ios::in | ios::binary);
```

Reading from and Writing to Binary Files

- Two member functions for ifstream and ofstream objects are useful in reading and writing.

- ofstreamFileObject.write((char *) &<the object>, sizeof(<the same object>))
- ifStreamFileObject.read((char *) &<the object>, sizeof(<the same object>))

- Binary file read and write is performed *objectwise* and not *elementwise*.
- **Closing Binary Files**
  - FileObject.close()

Write to Binary File

```
#include <iostream>
#include <fstream>
using namespace std;
class student
{
int RollNo;
char Name[30];
char Address[40];
public:
void ReadStudent();
};
void student::ReadStudent()
{
cout << "\n Enter roll no.: ";
cin >>RollNo;
cout << "\n Enter name: ";
cin >>Name;
cout << "\n Enter address: ";
cin >>Address;
cout << "\n";
}
int main()
{
student MSC_Student_Out;
ofstream MSC_StudFile_Out;
MSC_StudFile_Out.open("MSC.dat", ios::out |
ios::binary | ios::trunc);
if(!MSC_StudFile_Out.is_open())
cout << "File cannot be opened \n";
char Continue = 'y';
do
{
MSC_Student_Out.ReadStudent();
MSC_StudFile_Out.write((char*)
&MSC_Student_Out,
sizeof(MSC_Student_Out));

if(MSC_StudFile_Out.fail())
cout << "File write failed";
cout << "Do you want to continue? (y/n): ";
cin >> Continue;
} while(Continue != 'n');
MSC_StudFile_Out.close();
return 0;
}
```

Input

```
Enter roll no.: 1
Enter name: akash
Enter address: Ahmedabad
Do you want to continue? (y/n): y
Enter roll no.: 2
Enter name: Ratan
Enter address: Vadodara
Do you want to continue? (y/n): n
```

Read from Binary File

```
#include <iostream>
#include <fstream>
#include <string>
using namespace std;
class student
{
int RollNo;
char Name[30];
char Address[40];
public:
void WriteStudent();
};
void student:: WriteStudent()
{
cout << "\n The roll no.: ";
cout << RollNo;
cout << "\n The name: ";
cout << Name;
cout << "\n The address: ";
cout << Address;
cout << "\n";
}
int main()
{
student MSC_Student _In;
ifstream MSC_StudFile_In("MSC.dat",
ios::in | ios::binary);
while(!MSC_StudFile_In.eof())
{
MSC_StudFile_In.read((char*)
&MSC_Student_In,
sizeof(MSC_Student_In));

if(MSC_StudFile_In.fail())
break;
MSC_Student _In.WriteStudent();
}
MSC_StudFile_In.close();
return 0;
}
```

Output

```
The roll no.: 1
The name: Akash
The address: Ahmedabad
The roll no.: 2
The name: Ratan
The address Vadodara
```

RANDOM ACCESS USING seekg() and seekp()

- seekg() is a function to move the get or read pointer of the file.
- seekp() is a function to move the put or write pointer of the file.
- The function takes the following two arguments:
  - 1. Number of bytes to skip
  - 2. From where to skip

Cont…(seekg() and seekp())

- Important points related to the arguments of these two functions:

- The first argument can be positive as well as negative.

- For the first argument, the data type is integer. For the second argument, it is an enumeration containing the following values:
  - (a) ios::beg Beginning of the file
  - (b) ios::end End of file
  - (c) ios::cur   Current position of the fi le

Read from Binary File seekp()

```
#include <iostream>
#include <fstream>
using namespace std;
class student
{
int RollNo;
char Name[30];
char Address[40];
public:
void ReadStudent();
};
void student::ReadStudent()
{
cout << "\n Enter roll no.: ";
cin >>RollNo;
cout << "\n Enter name: ";
cin >>Name;
cout << "\n Enter address: ";
cin >>Address;
cout << "\n";
}
int main()
{
student MSC_Student_Out;
ofstream MSC_StudFile_Out;
MSC_StudFile_Out.open("MSC.dat", ios::out |
ios::binary | ios::trunc);
if(!MSC_StudFile_Out.is_open())
cout << "File cannot be opened \n";
MSC_Student_Out.ReadStudent();

MSC_StudFile_Out.write((char*)
&MSC_Student_Out,
sizeof(MSC_Student_Out));

MSC_Student_Out.ReadStudent();

MSC_StudFile_Out.seekp(0, ios::beg);

MSC_StudFile_Out.write((char*)&MSC_Stude
nt_Out, sizeof(MSC_Student_Out));

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

Input

```
Enter roll no.: 1
Enter name: akash
Enter address: Ahmedabad
Do you want to continue? (y/n): y
Enter roll no.: 2
Enter name: Ratan
Enter address: Vadodara
Do you want to continue? (y/n): n
```

Read from Binary File using seekg()

```
#include <iostream>
#include <fstream>
#include <string>
using namespace std;
class student
{
int RollNo;
char Name[30];
char Address[40];
public:
void WriteStudent();
};
void student:: WriteStudent()
{
cout << "\n The roll no.: ";
cout << RollNo;
cout << "\n The name: ";
cout << Name;
cout << "\n The address: ";
cout << Address;
cout << "\n";
}
```

```
int main()
{
student MSC_Student _In;
ifstream MSC_StudFile_In("MSC.dat",
ios::in | ios::binary);
MSC_StudFile_In.seekg(1 *
sizeof(student), ios::beg);
while(!MSC_StudFile_In.eof())
{
MSC_StudFile_In.read((char*)
&MSC_Student_In,
sizeof(MSC_Student_In));

if(MSC_StudFile_In.fail())
break;
MSC_Student _In.WriteStudent();
}
MSC_StudFile_In.close();
return 0;
}
```

Output

```
The roll no.: 2
The name: Ratan
The address Vadodara
```

tellg() and tellp()

- tellg() and tellp() are functions to find where the read and write pointers of a file are pointing to in terms of bytes from the beginning

```
#include <fstream>
#include <iostream>
using namespace std;
int main()
{
long FilePosition;
ofstream OutputFile;
OutputFile.open("FewLines.txt");
OutputFile.write("Oxford University Press", 23);
FilePosition = OutputFile.tellp();
cout<<FilePosition;
OutputFile.seekp(FilePosition-5);
OutputFile.write("India", 5);
OutputFile.close();
return 0;
}
```

O/P :

```
23
FewLines.txt
Oxford University India
```

I/O ERRORS

- If the path is not provided to retrieve the file in different situations, the same program running fine at one place will not be able to work at another place.
- While working with multi-user OS such as Linux or Windows, it is also important to know that files can be created or read only where there is a permission to write or read, respectively. The same program running perfectly on one machine or one account might just not work on another because of permission restrictions for different files and folders to different users.
- I/O error Check :
- if(filename.is_open())

I/O ERRORS

- While reading or writing a file:
- Both read() and write(), when successful, return the stream, and return zero otherwise.

```
if(MSC_StudFile_In.read((char*) &MSC_Student_In,
sizeof(struct student)))
{
// Actions to be taken
}
else
{
// Actions to be taken when error occurs
}
```

I/O ERRORS

- After executing a read() or write() statement, one can check whether it has failed by calling the function fail()

```
MSC_StudFile_In.read((char*) &MSC_Student_In, sizeof(struct student));

if(MSC_StudFile_In.fail())
break;
```

- The fail() function is quite generic. One can call fail() after any file-related operation to check whether the operation was successful or not.