

# Programming Fundamental

## Lab#09

### Table of Contents

File Handling: .....	2
Opening a file .....	2
Writing to a File.....	3
Reading from a File .....	4
Closing of a file.....	7

## File Handling:

So far, we have been using the iostream standard library, which provides cin and cout methods for reading from standard input and writing to standard output respectively. In this lab will teach you how to read and write from a file. This requires another standard C++ library called fstream, which defines three new data types

Sr.No	Data Type & Description
1	<b>ofstream</b> This data type represents the output file stream and is used to create files and to write information to files.
2	<b>ifstream</b> This data type represents the input file stream and is used to read information from files.
3	<b>fstream</b> 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.

To perform file processing in C++, header files <iostream> and <fstream> must be included in your C++ source file

## Opening a file

The first operation generally performed on an object of one of these classes to use a file is the procedure known as to opening a file. An open file is represented within a program by a stream and any input or output task performed on this stream will be applied to the physical file associated with it. The syntax of opening a file in C++ is:

```
Object.open (filename, mode);
```

Here, the first argument specifies the name and location of the file to be opened and the second argument of the **open()** member function defines the mode in which the file should be opened.

Sr.No	Mode Flag & Description
-------	-------------------------

1	<b>ios::app</b> Append mode. All output to that file to be appended to the end.
2	<b>ios::ate</b> Open a file for output and move the read/write control to the end of the file.
3	<b>ios::in</b> Open a file for reading.
4	<b>ios::out</b> Open a file for writing.
5	<b>ios::trunc</b> If the file already exists, it's contents will be truncated before opening the file.

You can combine two or more of these values by **ORing** them together. For example if you want to open a file in write mode and want to truncate it in case that already exists, following will be the syntax –

```
ofstream outfile;
outfile.open("file.dat", ios::out | ios::trunc );
```

Similar way, you can open a file for reading and writing purpose as follows –

```
fstream afile;
afile.open("file.dat", ios::out | ios::in );
```

## Writing to a File

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

int main()
{
    fstream st; // step 1: Creating object of ofstream class
    st.open("abc.txt",ios::out); // Step 2: open a file

    string ch="ProgramingLab";

    st <<ch; // Step 4: writing on a file

    st.close(); // Step 5: Closing file
```

```

    system("pause");
    return 0;
}

```

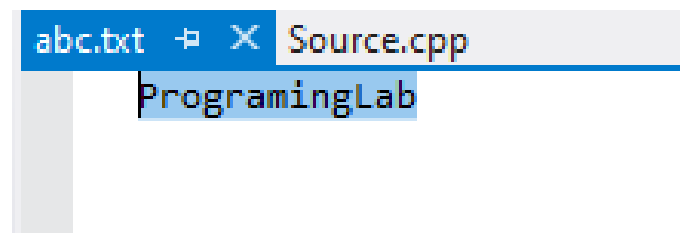
## Another way is:

```

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

int main()
{
    ofstream st; // step 1: Creating object of ofstream class
    st.open("abc.txt"); // Step 2: open a file
    string ch="ProgramingLab";
    st <<ch; // Step 4: writing on a file
    st.close(); // Step 5: Closing file
    system("pause");
    return 0;
}

```



## Reading from a File

```

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

int main()
{
    string ch;
    ifstream st; // step 1: Creating object of ifstream class
    st.open("abc.txt", ios::in); // Step 2: open a file

    st >>ch; // Step 4: Reading from a file
    st.close(); // Step 5: Closing file

    cout<<ch<<endl; //Print on console
    system("pause");
    return 0;
}

```


## Another way is:

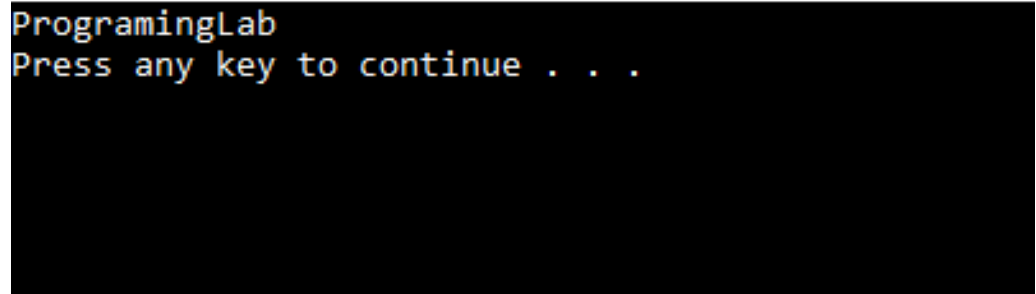
```
#include<iostream>
#include<string>
#include<fstream>
using namespace std;

int main()
{
    string ch;
    ifstream st; // step 1: Creating object of ifstream class
    st.open("abc.txt"); // Step 2: open a file

    st >>ch; // Step 4: Reading from a file
    st.close(); // Step 5: Closing file

    cout<<ch<<endl; //Print on console
    system("pause");
    return 0;
}
```

 C:\Users\This Pc\documents\visual studio 2012\Projects\Project1\Debi



```
ProgramingLab
Press any key to continue . . .
```

## Reading till end of file

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

int main()
{
    string ch;
    ifstream st; // step 1: Creating object of ifstream class
    st.open("abc.txt",ios::in); // Step 2: open a file

    while(!st.eof())
    {
        st >>ch; // Step 4: Reading from a file
        cout<<ch<<endl; //Print on console
    }
    st.close(); // Step 5: Closing file

    system("pause");
}
```

```

    return 0;
}

```

abc.txt X Source.cpp

```

line1
line2
line3
line4

```

C:\Users\This Pc\documents\visual studio 2012\Projects\Project1\Debug\Proje

```

line1
line2
line3
line4
Press any key to continue . . . _

```

```

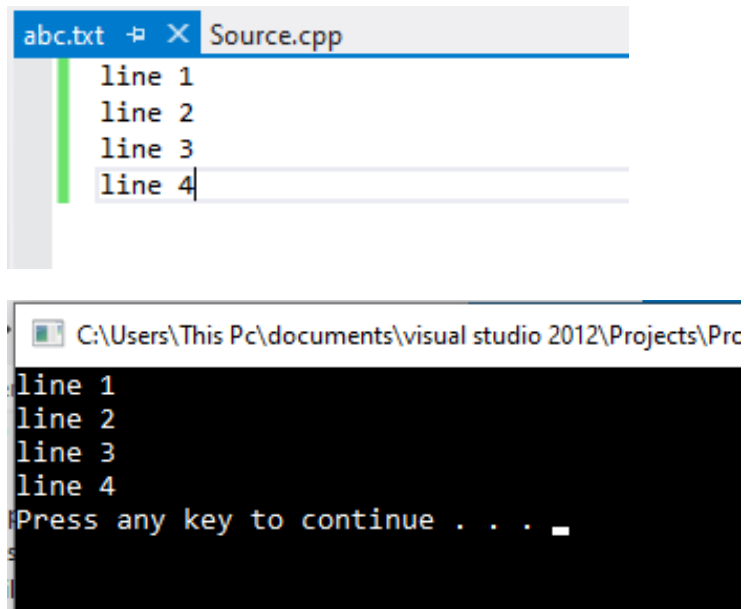
#include<iostream>
#include<string>
#include<fstream>
using namespace std;

int main()
{
    string ch;
    fstream st; // step 1: Creating object of ifstream class
    st.open("abc.txt",ios::in); // Step 2: open a file

    while(!st.eof())
    {
        getline(st,ch); // Step 4: Reading from a file
        //st>>ch;
        cout<<ch<<endl; //Print on console
    }
    st.close(); // Step 5: Closing file

    system("pause");
    return 0;
}

```



```
abc.txt  X Source.cpp
line 1
line 2
line 3
line 4

C:\Users\This Pc\documents\visual studio 2012\Projects\Pro
line 1
line 2
line 3
line 4
Press any key to continue . . . _
```

## Closing of a file

When a C++ program terminates it automatically flushes all the streams, release all the allocated memory and close all the opened files. But it is always a good practice that a programmer should close all the opened files before program termination.

Following is the standard syntax for `close()` function, which is a member of `fstream`, `ifstream`, and `ofstream` objects.

```
Object.close();
```