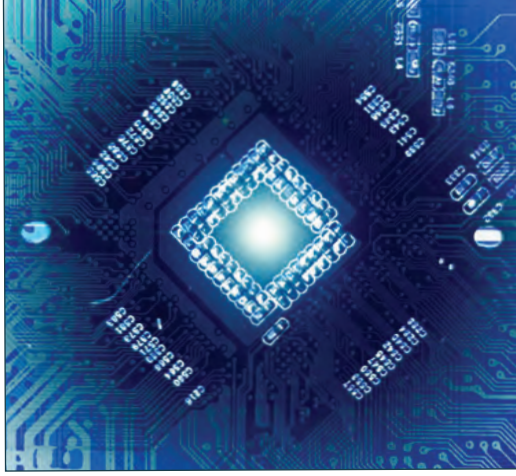
CHAPTER 7



Files

Objectives

After completing this chapter, the student should understand and be familiar with the following:

- Batch files
- Creating a batch file
- Reading from a batch file
- Writing to a batch file
- Appending to a batch file
- Formatting the output
- Binary file
- Creating a binary file
- Reading from a binary file
- Writing to a binary files
- Type casting
- User-Defined-type

7.1 Batch Files

The purpose of writing a program is to get input, process input, and produce output. There are three ways to acquire inputs:

- Assign input to a variable or object
- Get or read input from standard files
- Get or read input from batch files

Until now, you've been dealing with standard files. **Batch files** are text files created by a programmer. The batch file can be created manually, or it can be created through a program. You can read from a batch file, write to a batch file, or append to a batch file using **"fstream"** data type. To read from a batch file, you need to create a text file and store it in the same directory as your C++ project.

"C++ has two basic classes to handle files: ifstream and ofstream.

To use them, include the header file fstream."[1]

7.2 Finding the Exact Location of a Text File

Write a simple program to create a text file, and then write or append text to the created text file. Look in your directory to see where this file is stored, and then you can create your own text file and store it in the same directory and start manipulating it.

In Microsoft Visual Studio Platform, this text file is stored in the same folder as the one you created through your C++ program. For example, for this program a folder called *PFile* is created under project folder. Under *PFile* folder, there will be another folder, also called *PFile*, that stores the text file under PFile directory, which can be accessed through your program using ***fstream class***. The **text file** is written in notepad.

Read / write / append manipulation

1. To read from a file
   - Create a text file.
   - Declare a file name, i.e., MyFile, type of fstream.
   - Open the file with Read Mode.
   - Read information from the file and store it in variable(s) or another batch file.
   - Close the file.
2. To write to a file
   - Declare a file name, i.e., MyFile, type of fstream.
   - Open the file with Write Mode.
   - Get information.
   - Write information or process information to file.
   - Close file.
3. To append a file
   - Declare a file name, i.e., MyFile, type of fstream.
   - Open the file with Append Mode.
   - Get information.
   - Write information or process information to file.
   - Close the file.

1. Alex Allain, "Lesson 10: C__ File I/O," *Cprogramming.com*, accessed January 10, 2015. http://www.cprogramming.com/tutorial/lesson10.html

7.3 Creating a Text File and Writing to It

To write to a file, you need to open the file as write mode. To do so, include ***fstream*** header file to your program. Create an object of type fstream. Open the file as a write mode, i.e., MyFile.open ("TEST.txt", ios::out);

Program 7a

Write a program to ask user to enter a list of names and write that list to a batch file. Your program should terminate when a dollar sign, $, is read.

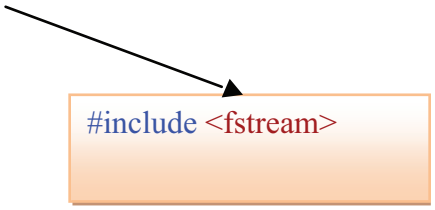
```
#include <iostream>
#include <fstream>
#include <string>
#include <iomanip>
using namespace std;
int main(void)
{

// Declare variables.

        bool Flag = true;
        string Info;
        fstream MyFile;

   MyFile.open ("Test.txt", ios::out);
   cout <<"\n\t enter a name\n";
        cin >> Info;

        while (Flag)
        {
                if (Info ==  "$")
                        Flag = false;

                else if (MyFile)       // to make sure the file is open

                  {
                        MyFile <<setw(15) <<left<< Info;

                        MyFile<<endl;
                        cout <<"\n\t enter a name\n";
                        cin >> Info;
                  }
                else
                        cout<<"\n\t File cannot be opened\n";
        }
                MyFile.close();
  cout << endl;
  system ("pause");
  return 0;
}
```

```
          enter a name
Mohsen

          enter a name
Changis

          enter a name
Borzo

          enter a name
Bakhshi

          enter a name
$

Press any key to continue . . .
```

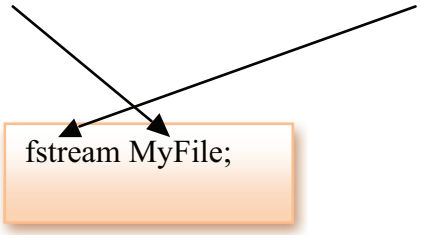
Output of the program

Explanation of Program 7a

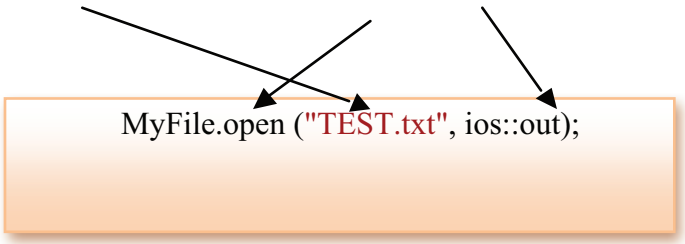
Fstream file is included to the program.



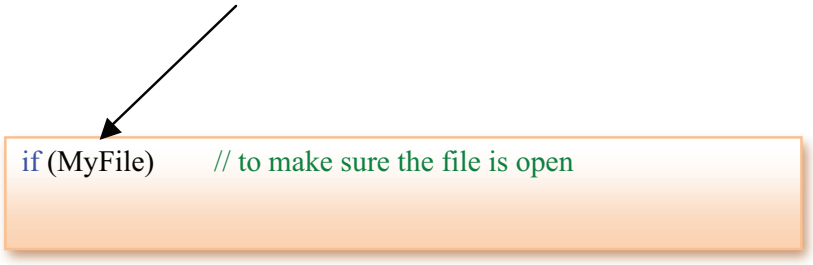
Object MyFile is declared as type of fstream.



Create TEST.txt file, and open as "write only."

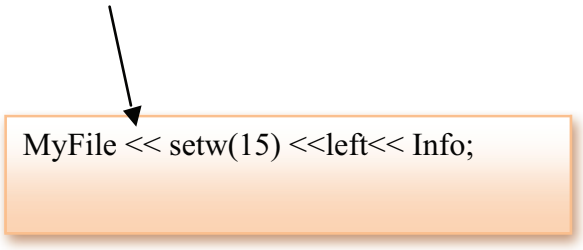


To check if the file is open



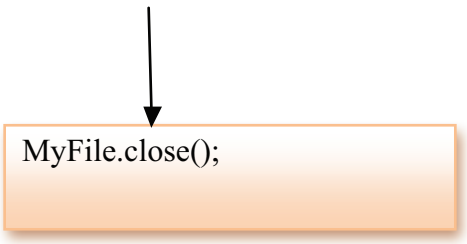
```
if (MyFile)        // to make sure the file is open
```

To write to a batch file



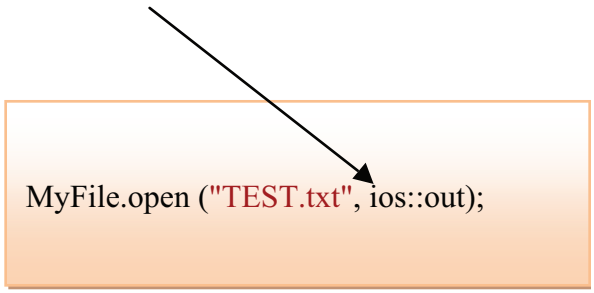
```
MyFile << setw(15) <<left<< Info;
```

To close a file



```
MyFile.close();
```

namespace is used to prevent both internal and external linkage issues.
ios like fstream, cout, and cin is part of ***std namespace***.



```
MyFile.open ("TEST.txt", ios::out);
```

7.4 Reading from a File

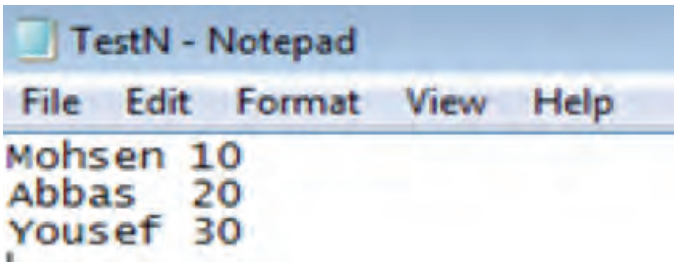
Beside including the ***fstream*** header file to your program, there are three points to remember when reading from a batch file:

- First, you make sure that there is an existing file.
- Second, make sure that the existing file is not empty.
- Third, open the file as a read mode, i.e., MyFile.open ("Test.txt", ios::in);

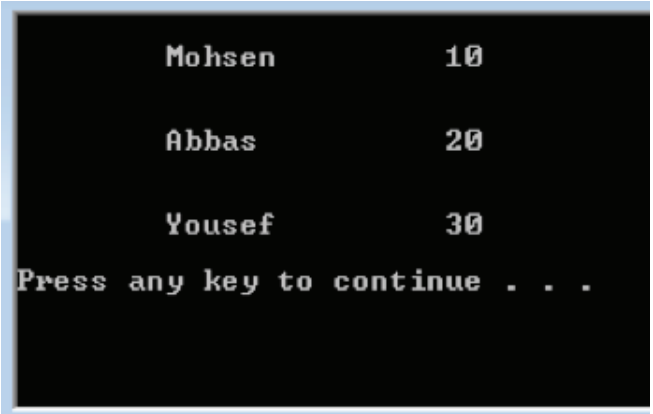
Program 7b

Write a program to read a list of names and IDs from a batch file and display them.

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


int main(void)
{

        // Declare variables.

        bool End_Of_File = false;
        string Info;
        int Number;

        // Declare file name.
        fstream MyFile;

        // Open file as a read mode.
        MyFile.open ("Test.txt", ios::in);


        while (!End_Of_File)
        {
                if (MyFile.eof())
                        End_Of_File = true;
                else
                {
                        MyFile >> Info;
                        MyFile >> Number;
                        cout <<"\n\t"<< setw(15) <<left<<Info;
                        cout <<setw(6)<<Number;
                        cout <<endl<<endl;
                }

        }
                MyFile.close();

  system ("pause");
  return 0;
}
```



Given the above input from a batch file, the output is displayed on a standard file as follows:



Open the text file as "read only."

```
MyFile.open ("Test.txt", ios::in);
```

Check for end of file flag.

```
MyFile.eof())
```

To read from a text file

```
MyFile >> Info;
```

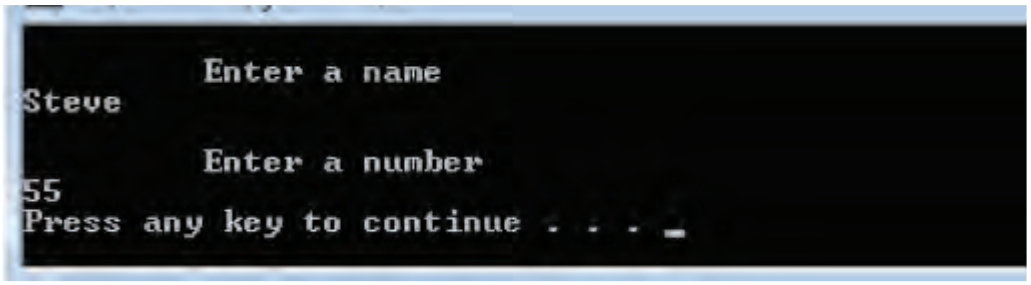
7.5 Appending Text to an Existing File

Open an existing file as **append mode**, which will append new information at the end of the file—if the file is not empty.

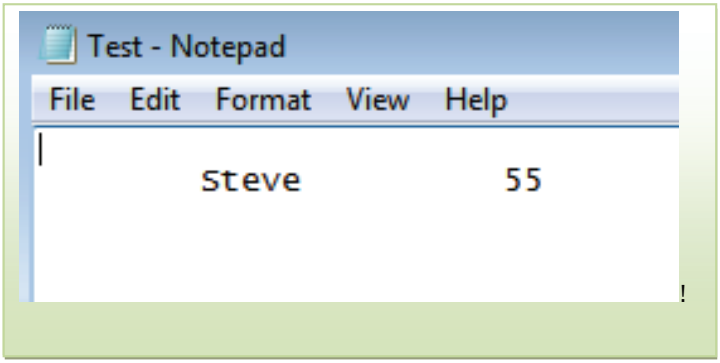
Program 7c

The following program appends a name to the end of the file.

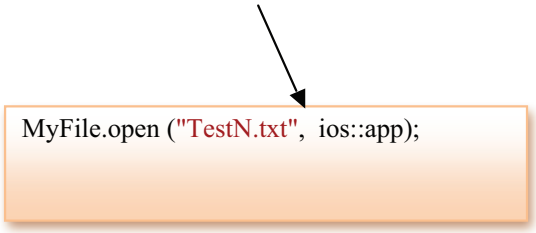
```
# include <fstream>
#include <string>
#include <iomanip>
#include <iostream>
using namespace std;
int main(void)
{
// Declare variables.
          bool End_Of_File = false;
          string Info;
          int Number;
          // Declare file name.
          fstream MyFile;
         MyFile.open ("Test.txt",  ios::app);
          cout<<"\n\t Enter a name\n";
          cin >> Info;
          cout<<"\n\t Enter a number\n";
          cin >> Number;
          MyFile <<"\n\t"<< setw(15) <<left<<Info;
          MyFile <<setw(6)<<Number;
          MyFile.close();
           system ("pause");
           return 0;
}
```



The new text file will be as follows:



To write at the end of the file or append information to the text file



```
MyFile.open ("TestN.txt",  ios::app);
```

7.6 Opening a File Both as Write Mode and Read Mode at the Same Time

There are two ways to open a file in both write and read mode:

1. Open the file as write mode. Close the file. Reopen it as read mode. Close the file again.

```
//Open file in write mode.
MyFile.open("Test.txt", ios::out);
// do operations
// Close the file.
MyFile.close();

//Reopen the file as read mode.
MyFile.open("Test.txt", ios::in);
```

2. To open the file both in write and read mode at the same time—and then do operation and then close the file:

```
//Open file in both write mode and read mode.
        MyFile.open("Test.txt", ios::out | ios::in );
        // Do operations.
        // Close the file.
        MyFile.close();
```

Declaring a file object—and creating and opening a file with both read and write mode with one statement:

fstream MyFile ("BatchFile.txt", ios::out | ios::in);

The above statement is equivalent to the following three statements:

- fstream MyFile;
- MyFile.open ("BatchFile.txt", ios::out);
- MyFile.open ("BatchFile.txt", ios::in);

fstream: An object of type ***fstream*** can be used for both read and write mode.

ifstream : An object of type ***ifstream*** can be used for read only mode.

ofstream: An object of type ***ofstream*** can be used for write only mode.

ifstream MyFile ("BatchFile.txt", ios::in)

ofstream MyFile ("BatchFile.txt", ios::out)

Objects of fstream, ifstream, and ofstream can create a file.

7.7 Member Functions of fstream

There are several member functions to consider:

1. Member function **istream** is used to read from a batch file.
2. Member function **ostream** is used to write from a batch file.
3. Member function **open** is used to open a batch file.
4. If a file is already open, then you cannot reopen the same file. For this reason the member function **is_open** can be used to check the status of the file.
5. Member function **close** is used to close a batch file. Any batch file that has been opened must be closed before the close of the program. A file may be opened and closed as many times as required.

7.8 Formatting the Output

Formatting the output is similar to formatting the output for standard file, as explained in Chapter Three. Following is the summary of formatting output explained in Chapter Three:

1. Make sure to include ***iomanip*** file as a header file.
2. Setw(X) allocates X spaces for output. If the size of input is less than X, then the output would be filled with leading blank.
3. Setprecision(X) would display X digits of inputs, which are number of digits before and after decimal point.
4. Setprecision(X) with showpoint display exactly X digits, which are the total number of digits before and after decimal point. If total number of digits before and after decimal point for input is Y, assuming Y < X, then the number of digits after decimal point for output would be Y digits and filled with trailing (X – Y) zeros.
5. Setprecision(X) with fixed manipulator would display X digits after decimal point. If number of digits after decimal point for input is Y, assuming Y < X, then the number of digits after decimal point for output would be Y digits and filled with trailing (X – Y) zeros.
6. Using left and setw(X) causes the output to be displayed with no leading blank, or the output would be left justified.
7. Using right and setw() causes the output to be displayed with leading blanks, or the output would be right justified.

7.9 Binary Files

> "A **binary file** is a file whose content must be interpreted by a program or a hardware processor that understands in advance exactly how it is formatted."[2]

Type casting from a text file to a binary file and vice versa[3]

MyFile.write (reinterpret_cast <char * > (variable or object), sizeof (variable or object))

MyFile.read (reinterpret_cast <char * > (variable or object), sizeof (variable or object))

Type casting of record from a text file to a binary file and vice versa[3]

MyFile.write (reinterpret_cast <char * > (&Variable or Object), sizeof (Variable or Object))

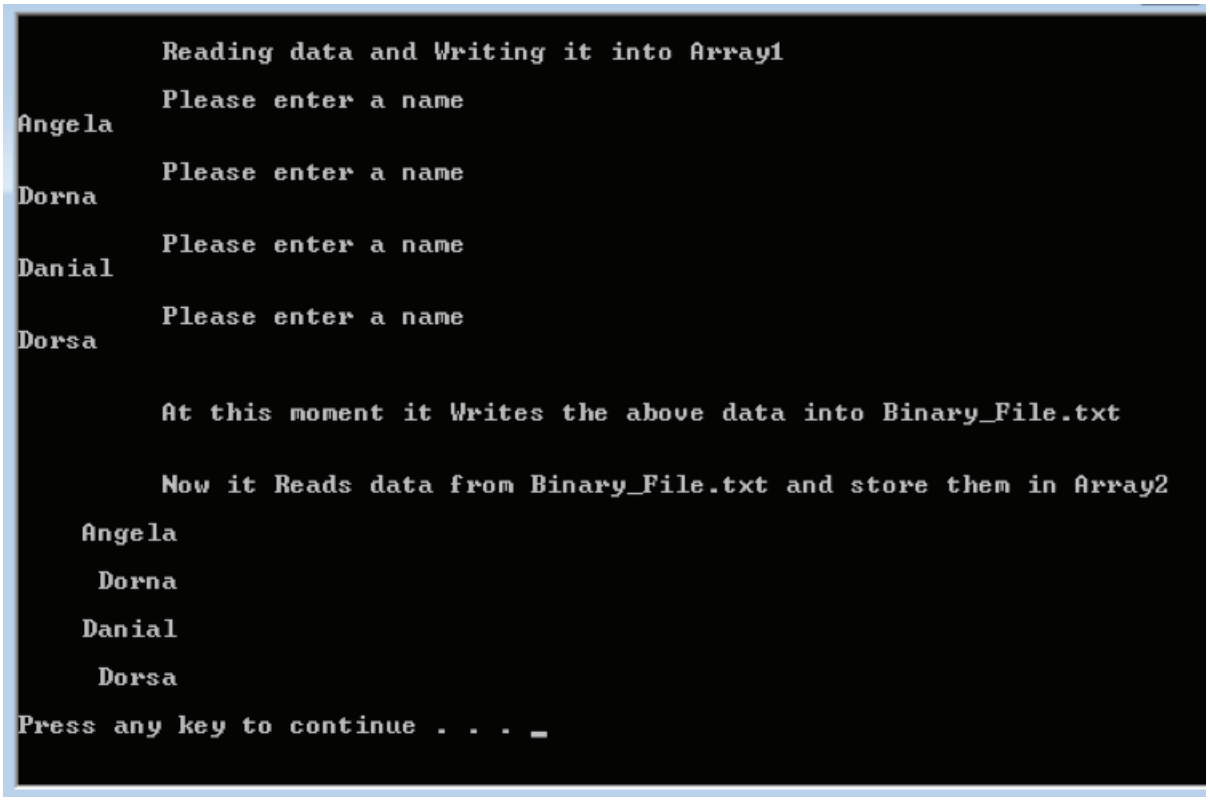
MyFile.read (reinterpret_cast <char * > (& Variable or Object), sizeof (Variable or Object))

2. Whatls.com. "Binary file: What is Binary File?" accessed February 3, 2015. http://whatis.techtarget.com/definition/binary-file
3. Tony Gaddis, *Starting Out With C++*, 691.

Program 7d

Write a program to read four names and store them in an Array1. Create a binary file. Write content of Array1 to the binary file. Write the content of binary file to another array called Array2. Display content of Array2. In case of record or class, the address (&) of the object is used.

```
#include <iostream>
#include <fstream>
#include <string>
#include <iomanip>

using namespace std;
const int size = 4;
int main(void)
{
  // Declare variables.
 int LCV;
 string  Array1 [size], Array2 [size];
 // Reading data and writing it into Array1
 cout <<"\n\t Reading data and Writing it into Array1\n";
 for (LCV =0; LCV < size; LCV++)
  {
          cout << "\n\t Please enter a name\n";
          cin >> Array1 [LCV];
  }

 // Declare file name.
 fstream MyFile;
 // Open file as a write mode in binary.
 MyFile.open ("Binary_File.txt", ios::out | ios::binary);
 // Writing data from Array1 into binary file.
 cout <<"\n\n\t At this moment it Writes the above data into Binary_File.txt\n";
 MyFile.write (reinterpret_cast <char *> (Array1), sizeof(Array1));
 //Close binary file.
 MyFile.close();
 //Reopen binary file as reading mode.
 MyFile.open ("Binary_File.txt", ios::in | ios::binary);
 // Reading data from binary file and writing it into Array2.
 cout <<"\n\n\t Now it Reads data from Binary_File.txt and store them in Array2 \n\n";
 MyFile.read (reinterpret_cast <char *> (Array2), sizeof(Array2));
 //Displaying content of Array2
 for (int LCV =0; LCV < size; LCV++)
   {

          cout << setw(10) << Array2[LCV]<<endl;
     cout <<endl;
   }

 MyFile.close();
 system ("pause");
 return 0;
}
```

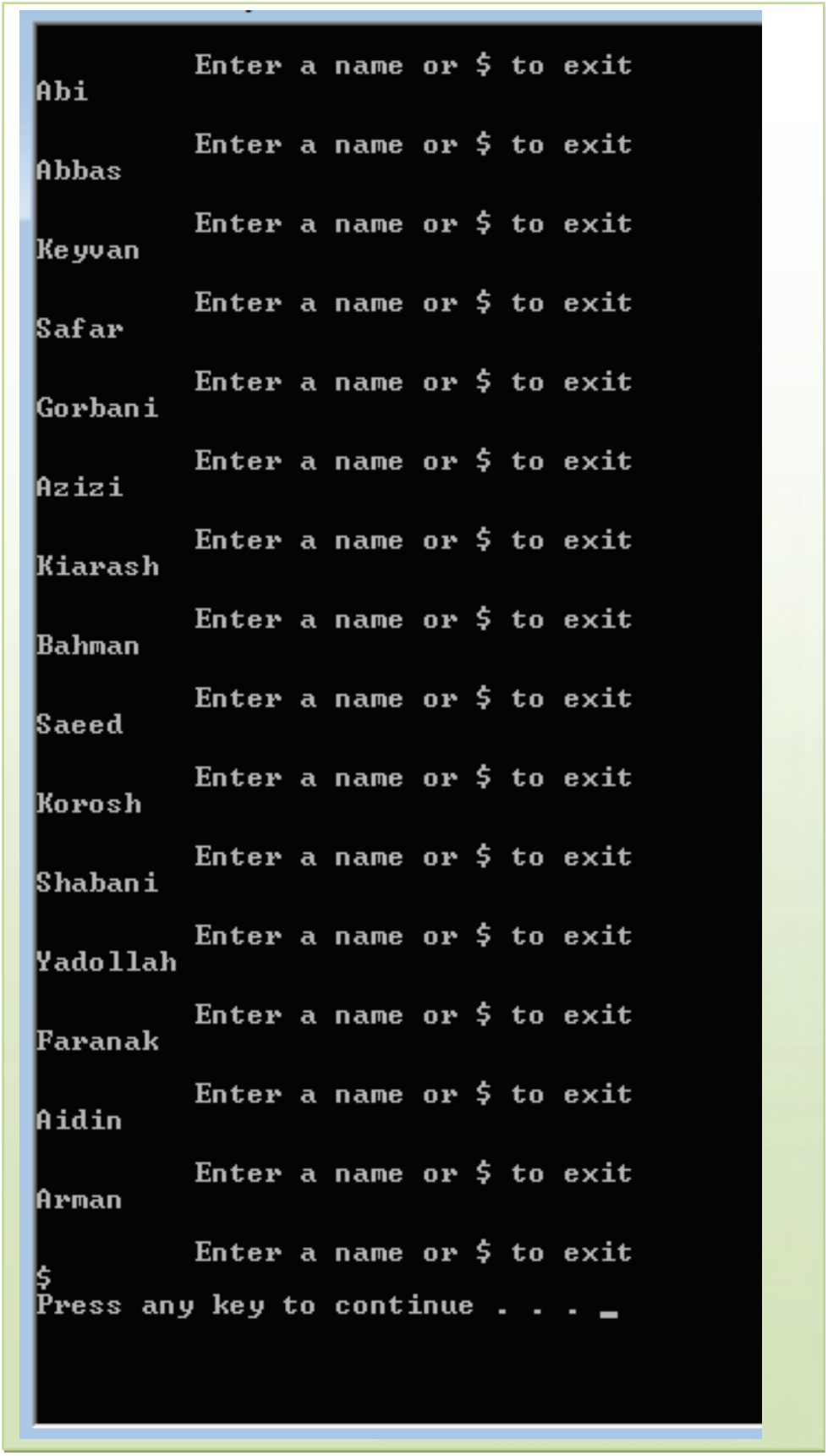


Output of the program

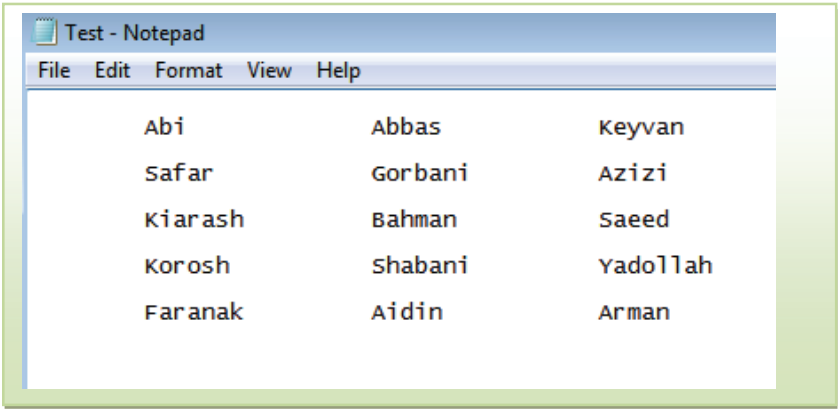
Program 7e

Write a program to read peoples' names and ages from a standard file and write them in a batch file. Your program should terminate when a dollar sign, $, is read.

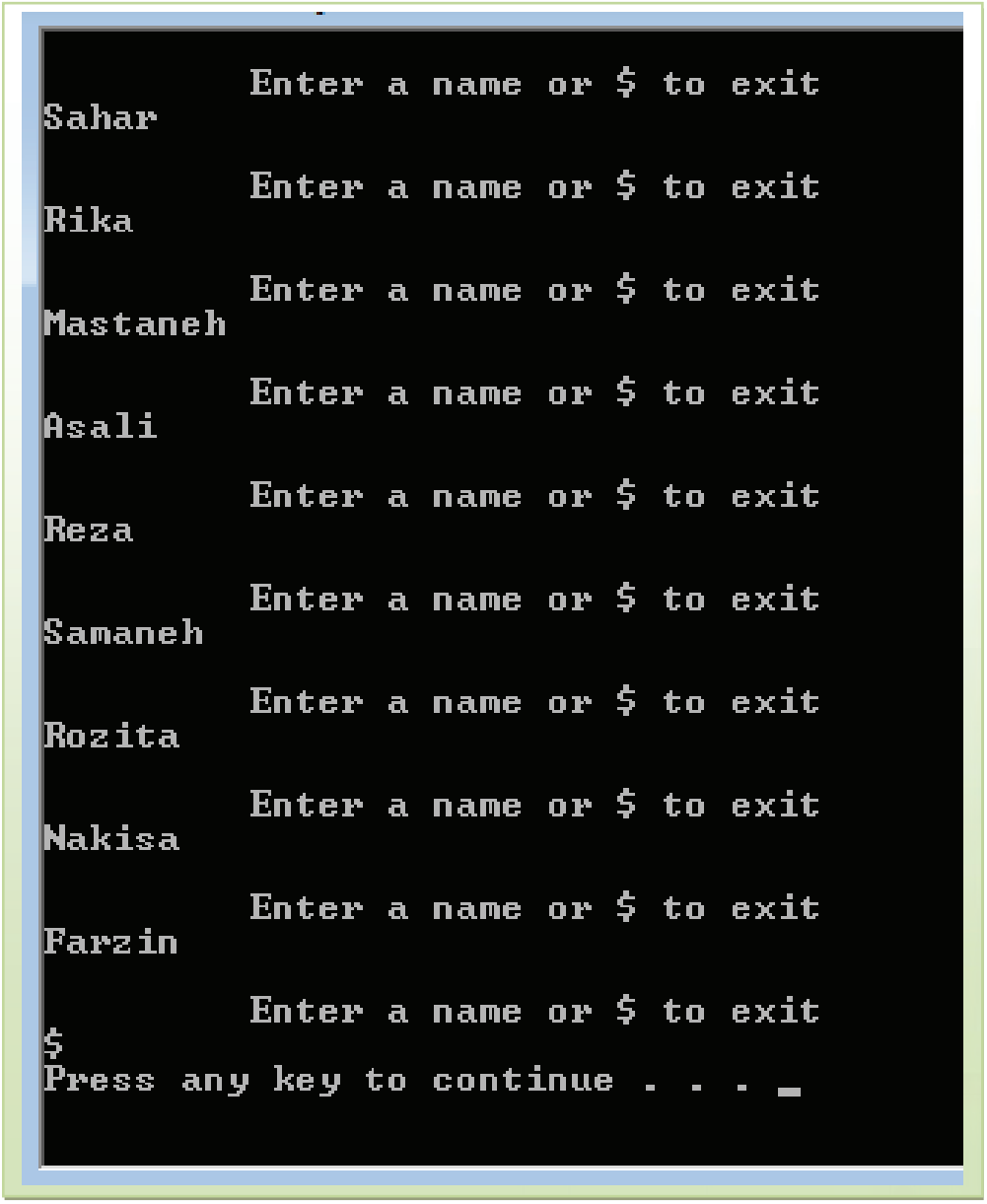
```
# include <fstream>
#include <string>
#include <iomanip>
#include <iostream>
using namespace std;

int main(void)
{
// Declare Variables
        bool End_Of_File = false;
        string Info;
        int counter =0;
        bool Flag = true;
        // Declare file name
        fstream MyFile;

        // Open the file as a read mode

       MyFile.open ("Test.txt",  ios::app);

        while (Flag)
        {

                cout<<"\n\t Enter a name or $ to exit\n";
                cin >> Info;
                if (Info == "$")
                                Flag = false;
                else
                {
                        counter++;
                        MyFile <<"\t"<< setw(15) <<left<<Info;
                        if (counter % 3 ==0)
                                MyFile <<"\n"<<endl;
                }
        }

// Close the file

   MyFile.close();

   system ("pause");
   return 0;
}
```


Enter a name or $ to exit
Abi
Enter a name or $ to exit
Abbas
Enter a name or $ to exit
Keyvan
Enter a name or $ to exit
Safar
Enter a name or $ to exit
Gorbani
Enter a name or $ to exit
Azizi
Enter a name or $ to exit
Kiarash
Enter a name or $ to exit
Bahman
Enter a name or $ to exit
Saeed
Enter a name or $ to exit
Korosh
Enter a name or $ to exit
Shabani
Enter a name or $ to exit
Yadollah
Enter a name or $ to exit
Faranak
Enter a name or $ to exit
Aidin
Enter a name or $ to exit
Arman
Enter a name or $ to exit
$
Press any key to continue . . .

The Output of Program 7e:



If you run Program 7e for the second time, then the new information will be appended to the end of the file. Let's add nine more names.



The output of Program 7e after the second run, given the above input, will be as follows:

The purpose of using ***"counter % 3"*** in program 7e was to display every three inputs or names on the same line and advance the *write pointer* by using *"\n"* and /or *"endl"*. The *"endl"* or *"\n"* advances the *write pointer* to the next line. The reason that both of them are used here is to have space between every three names or lines.

```
if (counter % 3 == 0)
    MyFile <<"\n"<<endl;
```

7.10 Including User-Defined-File

The purpose of this section is to reuse some of the user defined functions in multiple files or programs to reduce the size of program. As a software engineer being in charge of a large project, you will find this technique would save you time and money. Let's compare two different programs:

1. Write a program to get each employee's name, address, social security number, employee ID number, department name, manager name, supervisor name, Rank, number of dependent, weekly hours, rate per hour and so on.
2. Write a program to get each student's name, address, student ID, major, GPA, List of courses registered and so on.

For the first case, you have to get information such as name is type of "STRING", Employee's ID is type of "INTEGER", Weekly hour and rate per hour are type of "REAL or FLOAT or DOUBLE". You have three different types which means you need three different functions to get information and additional functions to perform other tasks to fulfill the requirement of the project. In addition, you need three different functions to display information for three different types which are Display (int Data), Display (Double Data), and Display (string Data).

For the second case, student's name is type of "STRING", and Student's ID is type of "INTEGER", GPA is type of "DOUBLE", and letter grade is type of "CHARACTER".

In both cases, you have three functions types of integer, string, and double to get information. Notice that you can reuse three functions of first case to get information and three display functions of first case to display information. *Function will be addressed in chapter 9.*

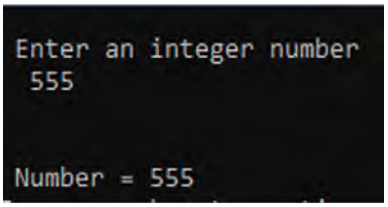
The following header file has a function called " get_Data()" to read an integer number. Any program can call this header file to use "get_Data()" function. Use define directive to define the header file.

```
#ifndef  Test_File_H

#define  Test_File_H

#include  <iostream>


using  namespace  std;

int  get_Data();

int  get_Data()
        {
                int  Data;

                cout <<  "\n\ Enter an integer number\n  ";

                cin >>  Data;

                return  Data;
        }
```

The following program is a "Driver" which calls the "get_data()" function in the header file. You need to include the header file in your program and it is safe to put exact location of header file. The preprocessor searches for the header file in the directory specified by include directive. Use double quote to include user-define-file.

```
#include <iostream>

#include "C:\Users\Javadian\source\repos\Test_File\Test_File\Header.h"

using namespace std;

int main()

        {
                int Number;

                Number = get_Data();

                cout << "\n\n Number = " << Number << endl;

                system("pause");

                return 0;
        }
```

The preprocessor should know the exact location of "Header.h" file in the driver program.



Output of the above program

Summary

Batch files are text files created by programmer.

The **text file** is written in **notepad**.

Creating a text file and writing to it by using fstream: To write to a file, you need to open the file as write mode. To do so, include ***fstream*** header file to your program. Create an object of type fstream. Open the file as a write mode, i.e., MyFile.open ("TEST.txt", ios::out);

Reading from a text file using fstream object: Beside including the fstream header file to your program, there are three points to remember to read from a batch file. First you make sure that there is an exit a file. Second, make sure that the existing file is not empty. Third, open the file as a read mode, i.e., MyFile.open ("Test.txt", ios::in);

To append text to the existing file open an existing file as append mode which will append new information at the end of the file—if the file is not empty, i.e., MyFile.open ("Test.txt", ios::app);

Formatting the output: Include ***iomanip*** file. Use setw(X), left and setw(X), right and setw(), setprecision(X), setprecision(9) with fixed, and setprecision(X) with showpoint.

Binary Files: Binary files are readable only by a computer. A user would not be able to read a binary file.

Type casting from a text file to a binary file and vice versa.

MyFile.write (reinterpret_cast <char * > (variable or object), sizeof (variable or object))

MyFile.read (reinterpret_cast <char * > (variable or object), sizeof (variable or object))

Type casting of record from a text file to a binary file and vice versa.

MyFile.write (reinterpret_cast <char * > (&Variable or Object), sizeof (Variable or Object))

MyFile.read (reinterpret_cast <char * > (& Variable or Object), sizeof (Variable or Object))

Exercises

1. What is a batch file?
2. How do you create a batch file? Give an example.
3. Suppose a batch file name is ***"BatchFile.txt"***. What command do you use to open the file as write mode?
4. For question # 3, what command do you use to open the file as append mode?
5. For question # 3, what command do you use to open the file as read mode?
6. Explain set(X) with fixed manipulator.
7. Explain setprecision(X) with showpoint manipulator.
8. Explain set(X) with left manipulator.
9. Explain set(X) with right manipulator.
10. What is difference between fstream and ifstream?
11. What is difference between fstream and ofstream?
12. What is difference between ofstream and ifstream?
13. What is a binary file?
14. Suppose you have written integer numbers into in binary file. Write the type-casting command to read it from binary file and store it in an array of integers.
15. How do you create a binary file?
16. How do you open a binary file?
17. How do you write to a binary file?
18. How do you read from a binary file?
19. What is difference between text file and binary file?
20. What is function for detecting *end of file?*
21. What are three things you should do to read data from a file?

Programming Practices

1. Create a batch file and store twenty integer numbers. Write a C++ program to open this file in read mode. Display the content of the file.
2. Write a program to read the records of twenty students. Each record has student's name, number of courses, and major; store them in an array size 20. There are three exams per course. Compute average grade for each course and find the GPA of each student. Create a batch file to store the twenty students' records.
3. Write a program to create a binary file. Read twenty real numbers type of double and write them in the binary file. Display the content of binary file using type casting method.
4. Write a C++ program to create a file as write and read mode. Ask user to enter twenty names and write them to the file. Display the content of the file.

Project 1: Employees

5. Write a C++ program to create a batch file for twenty employees. Read name, ID, weekly hours, and rate per hour of twenty employees. Compute gross salary. If gross salary exceed $2,500.00, then deduct 33% tax; otherwise, deduct 24% tax. Compute tax and net salary. Write weekly hours, rate per hour, gross salary, tax, and net salary of all employees in another batch file. Display all information for all employees. Make sure your program does the following:
   - Creates a file called ***Original.txt***
   - Enters twenty employees' information and saves the file
   - Opens the Original file as a read mode
   - Creates another file through your program and calls it ***Copy.txt***
   - Opens the Copy file as a write mode
   - Reads employee's information from Original file, one at a time
   - Calculates gross salary, tax, and net salary
   - Writes employee's information in the Copy file
   - Once all records are read and written, then opens the Copy file as read mode
   - Reads information from Copy file and displays them.
   - Closes Original file
   - Closes Copy file

Project 2: GPA revisited

6. Write a program to create a file and store each student's information as follows:

Table 7.1

Name	EX1	EX2	EX3	EX1	EX2	EX3	EX1	EX2	EX3	EX1	EX2	EX3
	For CS 240			For Math 246			EX3 For CS 243			EX3 For English 131		
Jack	88	58	47	81	91	61	71	92	76	73	94	69
Susan	73	94	69	82	84	37	28	58	87	81	91	61
Michael	98	89	100	96	98	97	92	91	90	78	88	67
Elizabeth	23	45	67	15	34	67	38	68	67	88	58	47

Expand the list for twenty students. Each student is taking four courses. There are three exams per course. Find average grade for each course. Find letter grade for each course. Find GPA for each student, including all four courses.

Letter Grade > 90	A
80 <= Letter Grade <90	B
70 <= Letter Grade <80	C
60 <= Letter Grade <70	D
Letter Grade <60	F

Project 3: String and File

7. Write a program to create a file. Your file should contain the following text:

   Batch Files are text files created by programmer.

   The text file is written in notepad.

   Creating a text file and writing to it by using fstream: To write to a file, you need to open the file as write mode. To do so, include fstream header file to your program. Create an object of type fstream. Open the file as a write mode.

   Reading from a text file using fstream object: Besides including the fstream header file to your program, there are three points to remember to read from a batch file. First you make sure that there exit a file. Second, make sure that the existing file is not empty. Third, open the file as a read mode.

   To append text to the existing file: Open an existing file as append mode which will append new information at the end of the file if the file is not empty.

   Binary Files: Binary files are readable only by computer. A user would not be able to read a binary file.

Your program should read the text file and produce the following output:

1. Find number of words in each line and display.
2. Find number of lines and display it.
3. Find total number of words and display it
4. Find number of words starts with capital letter in each line and display it.
5. Find total number of words starts with capital letter in each line and display it.
6. Find how many times the word "*file*" is repeated and display the number of "*file*" written in the above text.