EXPERIMENT NO 10

File Handling in C++

Objectives:

In this lab students will learn:

- Read data from files.
- Write and append data on files.

Equipment required:

- Dev-C++/Eclipse/Visual Studio installed in PC/Windows

DISCUSSION

1. **Pre-Lab**

File Handling

A file is a collection of data that is usually stored on a computer's disk. Data can be saved to files and then later reused. The information/data stored under a specific name on a storage device, is called a file. Without file handling whatever the program does is temporary and once we stop the program all the memory vanishes. We can store it by writing our data on a file so we can reuse that data next time.
The following data types are used for file handling operation.

ifstream: Input File Stream: This data type can be used only to read data from files into memory.
ofstream: Output File Stream: This data type can be used to create files and write data to them.
fstream: File Stream: This data type can be used to create files, write data to them, and read data from them.
Using the fstream data type we define an fstream object just as we define objects of other data types. The following statement defines an fstream object named dataFile.

```
fstream dataFile;
```

Opening of the file

We use an fstream object's 'open' function to open a file. An fstream object's 'open' function requires two arguments. The first argument is a string containing the name of the file. The second argument is a file access flag that indicates the mode in which you wish to open the file. Here is an example.

```
dataFile.open("info.txt", ios::out);
```

The first argument in this function call is the name of the file, info.txt. The second argument is the file access flag ios::out . This tells C++ to open the file in output mode. Output mode allows data to be written to a file. The following statement uses the ios::in access flag to open a file in input mode, which allows data to be read from the file.

```
dataFile.open("info.txt", ios::in);
```

NOTE: When used by itself, the ios::out flag causes the file's contents to be deleted if the file already exists. When used with the ios::in flag, however, the file's existing contents are preserved. If the file does not already exist, it will be created.

Example
This program uses an fstream object to write data to a file.

Example

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

int main()
{
    fstream dataFile;

    cout << "Opening file...\n";
    dataFile.open("demofile.txt", ios::out);    // Open for output
    cout << "Now writing data to the file.\n";
    dataFile << "Jones\n";                      // Write line 1
    dataFile << "Smith\n";                      // Write line 2
    dataFile << "Willis\n";                     // Write line 3
    dataFile << "Davis\n";                      // Write line 4
    dataFile.close();                           // Close the file
    cout << "Done.\n";
    return 0;
}
```

OUTPUT

Program Output

```
Opening file...
Now writing data to the file.
Done.
```

Output to File demofile.txt

```
Jones
Smith
Willis
Davis
```

Example
This program uses an fstream object to read data from a file. Suppose the given data is already stored in the file.

J	a	y	n	e		M	u	r	p	h	y	\n	4	7
	J	o	n	e	s		C	i	r	c	l	e	\n	A
l	m	o	n	d	.		N	C			2	8	7	0
2	\n	<EOF>												

Example	OUTPUT

`int main()`
`{`
`  string input;  // To hold file input`
`  fstream nameFile; // File stream object`
 	
`  // Open the file in input mode.`
`  nameFile.open("murphy.txt", ios::in);`
 	
`  // If the file was successfully opened, continue`
`  if (nameFile)`
`  {`
`  // Read the file contents.`
`  while (nameFile >> input)`
`  {`
`  cout << input;`
`  }`
`  // Close the file.`
`  nameFile.close();`
`  }`
`  else`
`  {`
`  cout << "ERROR: Cannot open file.\n";`
`  }`
`  return 0;`
	`JayneMurphy47JonesCircleAlmond,NC28702`

The getline Function

The problem with above Program can be solved by using the getline function. The function reads a “line” of data, including whitespace characters. Here is an example of the function call:

getline(dataFile, str,'\n');

The three arguments in this statement are explained as follows:

dataFile:

This is the name of the file stream object. It specifies the stream object from which the data is to be read.

str:

This is the name of a string object. The data read from the file will be stored here.

' \ n':

This is a delimiter character of your choice. If this delimiter is encountered, it will cause the function to stop reading. (This argument is optional. If it’s left out, ' \ n' is the default.) The statement is an instruction to read a line of characters from the file. The function will read until it encounters a \n. The line of characters will be stored in the str object.

Example

This program demonstrates the working of getline function.

Example	OUTPUT
	Jayne Murphy 47 Jones Circle Almond, NC 28702

```
{
    string input;     // To hold file input
    fstream nameFile; // File stream object

    // Open the file in input mode.
    nameFile.open("murphy.txt", ios::in);

    // If the file was successfully opened, continue.
    if (nameFile)
    {
        // Read an item from the file.
        getline(nameFile, input);

        // While the last read operation
        // was successful, continue.
        while (nameFile)
        {
            // Display the last item read.
            cout << input << endl;

            // Read the next item.
            getline(nameFile, input);
        }

        // Close the file.
        nameFile.close();
    }
    else
    {
        cout << "ERROR: Cannot open file.\n";
    }
    return 0;
}
```

Example

This program uses an fstream object to write data to a file. The file is closed, and an end-of-file character is automatically written. When the file is reopened, the new output is appended to the end of the file.

J	o	n	e	s	\n	S	m	i	t	h	\n	<EOF>

J	o	n	e	s	\n	S	m	i	t	h	\n	W	i	l

l	i	s	\n	D	a	v	i	s	\n	<EOF>

Example	OUTPUT

`int main()`
`{`
`  ofstream dataFile;`
 	
`  cout << "Opening file...\n";`
`  // Open the file in output mode.`
`  dataFile.open("demofile.txt", ios::out);`
`  cout << "Now writing data to the file.\n";`
`  dataFile << "Jones\n"; // Write line 1`
`  dataFile << "Smith\n"; // Write line 2`
`  cout << "Now closing the file.\n";`
`  dataFile.close(); // Close the file`
 	
`  cout << "Opening the file again...\n";`
`  // Open the file in append mode.`
`  dataFile.open("demofile.txt", ios::out | ios::app);`
`  cout << "Writing more data to the file.\n";`
`  dataFile << "Willis\n"; // Write line 3`
`  dataFile << "Davis\n"; // Write line 4`
`  cout << "Now closing the file.\n";`
`  dataFile.close(); // Close the file`
 	
`  cout << "Done.\n";`
`  return 0;`
`}`
	`Jones`
	`Smith`
	`Willis`
	`Davis`

Checking for a File's Existence

Before Opening the file sometimes, we want to determine whether a file already exists before opening it for output. We can do this by first attempting to open the file for input. If the file does not exist, the open operation will fail. In that case, you can create the file by opening it for output. The following code gives an example.

Example

```
fstream dataFile;
dataFile.open("values.txt", ios::in);
if (dataFile.fail())
{
        // The file does not exist, so create it.
        dataFile.open("values.txt", ios::out);
        //
        // Continue to process the file...
        //
}
else    // The file already exists.
{
        dataFile.close();
        cout << "The file values.txt already exists.\n";
}
```

2. Post-Lab (Lab Tasks)

1. Write a code that reads the contents of one .txt file (created manually) in the code and write those in another .txt file.

2. Read a .csv file (created manually), and display its contents on console. The .csv file should contain 4 columns and 10 rows. A column for string, int, float, and character values generated randomly.

END