Programming Techniques
Self-Study for Project
File processing

Objectives

By the end of this lab, the student should be able to:

- Use C++ text file I/O.
- Define C++ structure and use its members

File Processing

C++ provides two classes to deal with files, ***ifstream*** for reading from files and ***ofstream*** for writing to files. In order to use them, **<fstream>** should be included.

In this lab, we will only be concerned with text I/O. Another common file access mode is binary I/O.To access a text file, the following steps are performed:

- **Declaring an ifstream/ofstream object:**
  Simply, an instance of type ifstream/ofstream is declared just as an instance of any variable type:

```
ofstream outputFile;
ifstream inputFile;
```

- **Opening the file:**
  This step makes the object ready to read from/write into the file. It is made by invoking the *open* function.

```
outputFile.open("test.txt", ios::out);
```

  We pass two parameters to the function "open":

  - The first parameter is **File Name**, which is the *absolute* or *relative path* to the file.
  - The second is **Access Mode**, which is a combination of flags that determine how the program will access the file. We will only use in this lab the three flags shown in the Table 3-1.

ios::in	Opens the file for input. This is the default mode for ifstream (i.e. there is no need to explicitly specify it).
ios:out	Opens the file for output. This is the default mode for ofstream. If the file does not exist, it is created. If it exists, it is overwritten.
ios::app	Opens the file for appending. This can be used with ofstream, causing the output to be appended to the end of an existing file.

Table 3-1: Some file access flags

To open a file, the program must have enough privileges to open or create the file and the file should not be in use. The `ifstream/ofstream` object acts as a handle on the text file resource indicating that the file is in use.

It is possible to combine steps 1 and 2 above in a single statement:

```
ifstream inputFile("input.txt", ios::in);
```

To check if the file was found and opened successfully we have to call:

```
inputFile.is_open()
```

It returns true if the file was found and opened successfully.

- **I/O Operations:**

Once the file is opened, the stream object can be used just like *cin* and *cout* using the >> and << operators respectively. For example, to write "Hello Files !" in a file simply write:

```
outputFile << "Hello Files !";
```

To read an integer x from a file:

```
inputFile >> x;
```

To read multiple values from a text files separated by spaces or tabs:

```
inputFile >> x >> y;
```

To read a full line starting from the last read position, we can use the function *getline:*

```
inputFile.getline(name_str,len,'\n')
```

The three parameters passed to *getline:*

- The first parameter is of type char* , it should be a pointer to an array of characters where the read string will be stored.
- The second parameter is the streamsize with integer value indicating the maximum number of characters to read.
- The third parameter is optional, it is of type char. Reading successive characters stops when this delimeter is encountered.

It is often needed to test whether the end of file has been reached. This is done using *eof* function. The ***eof*** function returns true if the end of file has been read.

```
ifstream f("numbers.txt");
int x;
f >> x;
while(!f.eof())
{
   cout << x << endl;
   f >> x;
}
```

Code 3-1 : Reading a file that contains a list of integers and displaying its contents

- **Closing the file:**

After finishing with the file, it should be closed. When a file is closed:

- The buffers are flushed, making sure that all data has been already written into the file.
- The OS is informed that the file is no longer being used by the program.

- To close a file, use function close
  ```
  outputFile.close();
  ```

Open the "Examples" solution and read the provided examples: Example1 and Example2

➔ **Example1: illustrates how to deal with files in general**

➔ **Example2: illustrates how to pass an ifstream/ofstream folder by reference to a function. The example opens the file once in the main and read/write lines (or part of the file) line by line using a separate function that is called for each line.**

Exercise [Important]

Write a Program that reads inputs from "input.txt" and outputs the results to "outputs.txt".
Firstly it reads the name of the user and then the number of elements of two arrays (the two arrays have the same size).
The program is required to multiply the values of the same index in both arrays.
The output file should contain the results and a "Thanks" to the user.

- **Input File Format**

User_Name
Number_Of_Elements
Array1_Value1 Array2_Value1
Array1_Value2 Array2_Value2
……..
……..
Array1_ValueN Array2_ValueN

- **Example:**

Jack
4
1 2
3 4
5 6
7 8

- **Output File Format**

Multiplication_Of_Value1_in_array1_2
Multiplication_Of_Value2_in_array1_2
……..
……..
Thanks, User_Name

- **Example:**

2
12
30
56
Thanks Jack