Date
Page

Assignment No 7

Name : Sameer Manoj Bramhecha
Roll No : 21115
Batch : E-1
Performance Date : 9/11/2021
Submission Date : 13/11/2021

Title :→ Demonstrate various file operations using C++.

Objectives:- 1) To learn and understand streams and files in object oriented paradigm.
2.) To demonstrate file operations like create open, read, write and close a file.

Problem Statement:→ Write a C++ program that creates an output file, writes information to it, closes the file and open it again as an input file and read the information from the file.

Learning Outcomes:- After completion of this assignment, students ~~s~~ will be able to implement file handling using C++.

~~Theo~~ S/W or H/W requirement:-
1) 64-bit open Source Linux.
2.) Open source C++ programming tool like G++/GCC.

Theory:-
The iostream library is an object-oriented library that provides input and output functionality using streams. A stream is an abstraction that represents a device on which input and output operations are performed. A stream can basically be represented as a source or destination of characters

of indefinite length.
streams are generally associated to a physical source or destination of characters, like a disk file, the keyboard, or the console, so the characters gotten or written to/from our abstraction ~~cot~~ called streams are physically input/output to the physical device.
For example, file streams are C++ objects to manipulate and interact with files. Once a file stream is used to open a file, any input or output operation performed on that stream is physically reflected in the file.
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. This requires another standard C++ library called fstream, which defines 3 new data types:
1.] ofstream:- This data type represents the output file stream and is used to create files & to write ~~in~~ information to files.
2.] ifstream:- This data type represents the input file stream and is used to read information from files.
3.] fstream:- 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. These classes are derived directly or indirectly from the classes istream and ostream. We have already used objects whose types were these classes:

cin is an object of class istream and cout is an object of class ostream. Therefore, we have already been using classes that are related to our file streams. And infact, we can use our file streams the same way we are already used to use cin and cout, with the only difference that we have to associate these streams with physical files.
let's see an example:

```
#include <iostream>
# include <fstream>
using namespace std;
int main() {
ofstream myfile;
myfile.open ("example.txt");
myfile <<"writing this to a file. \n";
myfile. close ();
return 0;
}
```

Opening a file :→ A file much be opened before you can read it or write to it. Either the ofstream or fstream object may be used to open a file for writing and ifstream object is used to open a file for reading purpose only. Syntax for open():→
void open(const char ~~*~~ *filename, ios:: openmode);
Here, the 1st arguement specifies the name and location of the file to be opened and the second arguement of the open() member function defines the mode in which the file should be opened.

1.) ios:: in → open a file for reading.
2.) ios :: out → open a file for writing.
3.) ios:: ate → open a file for output and move the read/write control to the end of the file.
4.) ios :app → Append mode. All output to that file to be appended to the end.

Date
Page

5) ios:: trunc → If the file already exists, its content will be truncated before opening the file.
You can combine 2 or more of these values by ORing them together.

Closing a file :- When a C++ program terminates it automatically closes flushes all the streams, release all the allocated memory and close all the opened files.
But it is always a good practise that a programmer should close all the opened files before program termination.
Following is the standard syntax for close function:
void close();

Writing to a file :→ While doing C++ programming, you write information to a file from your program using the stream insertion operator (<<) just as you use an ofstream or fstream object instead of the cout object.

Ex:
```
# include <iostream>
  # include < fstream >
  using namespace std;
 int main() {
 ofstream myfile ("example.txt");
 if (myfile.is_open())
 { my file<< "Line1" << endl;
   myfile <<"Line 2" << endl;
   myfile.close();
 } else cout << "Unable to open file";
 return 0;
 }
```

Date
Page

Reading from a file :→ You read information from a file into your program using the stream extraction operator (>>) just as you use that operator to input information from the keyboard. The only difference is that you use an ifstream or fstream object instead of the cin object.

```
//reading a text file.
#include <iostream>
#include <fstream>
#include <string>
using namespace std;
int main () {
string line;
ifstream myfile ("example.txt");
if (myfile.is_open ()) {
while (getline (myfile, line)) {
cout << line << "\n"; }
myfile.close (); }
else cout << "unable to open file";
return 0;
}
```

Text files :→ Text files streams are those where the ios::binary flag is not included in their opening mode. These files are designed to store text and thus all values that are input or output from/to them can suffer some formatting ~~trast~~ transformations, which do not necessarily correspond to their literal binary value.

File Pointers and Manipulators :-

• Each file has 2 pointers known as file pointers, one is called the input pointer and the other is called output pointer.

• The input pointer is used for reading the contents of a given file location and the output pointer is used

Date
Page

for writing to a given file location.
• Each time an input or output operation takes place, the appropiate pointer is automatically advanced.

Functions for Manipulations of File pointer:
• All the actions on the file pointers takes place by default.
• For controlling the movement of file pointers file stream classes support the following functions:-
① seekg() → Moves get pointer to a specified location.
~~② putg()~~ → ② seekp() → Moves the put pointer to a speified location.
③ tellg() → Give the current position of the get pointer.
④ tellp() :-> Give the current posn of the put pointer.

Specifying the offset :→
• 'Seek' functions seekg() & seekp() can also be used with 2 arguements as follows:
seekg (offset, refposition);
seekp (offset, refposition);
The parameter offset represents the number of bytes the file pointer is to be moved from the location specified by the parameter refposition.
The refposition takes one of the following 3 constants defined in the ios class:

1) ios::beg → Start of file
2) ios:: cur → Current position of the pointer
3) ios:: end → End of file.

Algorithm :→

1) Start
2) Include iostream and fstream header files
3) In main(), read name ~~and age from~~ of file from user.
4.) Read name and age of student.
5.) Open file using ofstream for writing.
6.) If file is not present, a error message is displayed. And if file is present, it is opened.
7.) close the file
8.) Open the same file for reading.
9.) In while loop, start reading from file line by line and display output on terminal.
10.) If end of file is reached, exit loop.
11.) Stop.

Test Cases:
Test Case 1:-

```
Enter File Name to open:
Assignment7.txt
Enter Name
Sameer
Enter Age
 19
Enter Name
Ram
Enter Age
 18
Enter Name
Shyam
Enter Age.
 20
```

Expected Output:

```
-------DATABASE-------
        Sameer          19
        Ram             18
        Shyam           20
```

Actual Output:

```
-------DATABASE-------
        Sameer          19
        Ram             18
        Shyam           20
```

Test case 1 :-> Pass.

Test Case 2:-

Enter file Name to open:

my.txt.

Expected Output:-

```
Error opening the file
```

Actual Output:-

```
Error opening the file.
```

Test Case 2 - Pass

Conclusion:-

Hence, we have studied and learnt various file handling operations and methodes available in fstream header file.