**Programming Fundamentals**

|  |  |
| --- | --- |
| Lab 02 | |
| **Topic** | C-String and Basic File Writing |
| **Objective** | * Array   + Integer, Character, C-String   + Different input methods   + Problem solving using Arrays. * File I/O (Text File Writing)   + Need and importance of File Output.   + Different modes of output file stream (out, app).   + Opening/Closing a text file stream.   + Writing formatted/un-formatted data to file.   + Problem solving involving files (ofstream). |

**Lab Description:**

This lab is basically design for the revision of character array and C-String. You will also learn importance of file output and how to write into a text file.

In previous lab we already discuss integer array, let’s start with character array.

**Character array:**

An array is a collection of a fixed number of components (also called elements) all of the same data type and in contiguous (that is, adjacent) memory space. An array whose components are of type char.

**Declaration statement of an array:**

char list[10];

Above statement is used for creating an array. But as we discussed above when a memory location is reserved for this array it holds some values which are not assigned by user. So those values are considered as garbage value. In order to avoid garbage value, it is a good practice to assign values at the time of creation of array.

**Initialization statement of an array:**

Assigning value at the time of declaring of array or variable is called initialization statement. There are multiple ways of initialize an array.

**Initialize an array with null character:**

char list[5]={}; OR char arr[5]{};

**Full array initialization with different elements:**

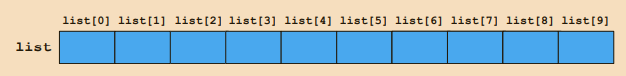
char list[5]={‘a’,’b’,’c’,’d’,’e’};

**Partial array initialization with different elements:**

char list[5]={‘a’,’b’};

on remaining indexes null will be assigned as initial value in case of partial array initialization.

**Accessing array components:**



**Generic way of accessing an array is:** IdentifierName[index#];

char list[10]; //list[0] is use for accessing first element of an array.

**Array input:**

We can take input from user into an array. We can take input index wise one by one and we can also take input at specific index number.

**Input at specific index:**

cin>> list[0]; //taking input on index 0 which is the first element of an array.

**Input in whole array:**

We use repetition statement(loops) for taking input in whole array. We can take input one by one as mention above but it is not considered as a good practice.

for(int i=0;i<size;i++)// i is used as index number and size is the number element you want to enter.

{

cin>>list[i]; // taking input at specific index i. After every iteration value of i will be updated.

}

**Array output:**

We can display the values of array on console as output to user. We can display index wise one by one and we can also take input at specific index number.

**Output of specific index:**

cout<< list[0]; //display the value of index 0 which is the first element of an array.

**Output of whole array:**

We use repetition statement(loops) for display whole array. We can display elements one by one as mention above but it is not considered as a good practice.

for(int i=0;i<size;i++)// i is used as index number and size is the number element you want to display.

{

cout<<list[i]; // display specific index i. After every iteration value of i will be updated.

}

**Array index out of bound:**

Index of an array is in bounds if index is between 0 and ARRAY\_ SIZE - 1, that is, 0 <= index <= ARRAY\_SIZE - 1. If index is negative or index is greater than ARRAY\_SIZE - 1, then we say that the index is out of bounds.

**C-String:**

A character array which is terminated at null is called C-String. C-String is quite simmilar with character array but, some of the working is a bit different. We can declare a C-String same as we declare a character array. But initialization is a bit different.

**Initialization statement of a C-String:**

char list[5]={“abcd”};

In case of character array we write character in single qoute which are comma seprated in case of character array. In case of C-String we write a string in double qoutes. One more differecne is we need an extra index for storing null in case of c-string. Because a null character will be placed at the end of the c-string.

**Acessing a c-string:**

We can access the element of a c-string same as we did in case of character array. But input and output of a c-string is a bit different than a character array.

**Input a c-string:**

We can take the input into c-string same as we discuss in character array. But there is an issue if we need to read a full name using this method it falis. Because, extraction operator, >>, skips all leading whitespace characters and stops reading data into the current variable as soon as it finds the first whitespace character. So we need something else in order to take complete input (with whitespaces) into a c-string.

We can solve this issue using cin.get() method.

**cin.get():**

Using this method, we can take whitespaces as input from user. This method helps us to take input as character by character and as a whole string as well.

**Taking input character by character:**

cin.get(ch); // ch is a character variable.

**Taking input as a string:**

cin.get(list,10) //list is a c-string and 10 is representing the input buffer size.

**Output of c-string:**

we can display a c-string using insertion operator, <<.

cout<<list; //list is a c-string.

**What if we display a character array like above mention method?**

cout<<arr; //arr is a character array.

There is a limitation with insertion operator, <<, it displays the data till null character. So if we are displays character array with this method it may shows some garbage values after the data.

**File Write:**

You want to take admission in any university. You need to fill the admission form. Let suppose after getting admission in university they ask your information no daily basis because, they did not store your information anywhere.

Is it a good practice to ask your students about their information on daily basis?

Obviously not, it’s not a good practice. So, what’s the solution of this problem?

Simple solution is you must store the data somewhere in order to perform different operations. We can create a text file in order to store our data. An area in secondary storage used to hold information is called file. The standard I/O header file, iostream, contains data types and variables that are used only for input from the standard input device and output to the standard output device. In addition, C++ provides a header file called fstream, which is used for file output. Among other things, the fstream header file contains the definitions of a data type ofstream, which means output file stream and is similar to ostream. The variables cin and cout are already defined and associated with the standard I/O devices. In addition, <<, setfill, and so on can be used with cout. These same operators and functions are also available for file output, but the header file fstream does not declare variables to use them. You must declare variables called file stream variables, which include ofstream variables for output. You then use these variables together with <<, or other functions for output. Remember that C++ does not automatically initialize user-defined variables. Once you declare the fstream variables, you must associate these file variables with the output source.

**File Output is a six-step process:**

1. Include the header file fstream in the program. // #include<fstream>

2. Declare output file stream variables. //ofstream fout;

3. Associate the output file stream variables with the output sources (text file). //fout.open(“fileName.txt”);

4. Verification of file opening. //fout.is\_open()

5. Use the output file stream variables with <<, or other output functions. //fout<<”Hello World”<<endl;

6. Close the file. // fout.close();

**Sample code:**

#include<fstream>

//Add additional header files you use

using namespace std;

int main() {

//Declare output file stream variables such as the following

ofstream fout;

//Open the files

fout.open("data.txt"); //open the output file

if(fout.is\_open())// this function return true if file is open and return false if file is not open.

{

//Code for file writing.

fout<< statements;

}

fout.close(); //Close files

return 0;

}

**Output file modes:**

There are different file opening modes. For output purpose we will discuss two modes app and out.

**Output(out):**

While we open a file for writing purpose default mode of file opening is out. In this mode pervious content of file will be overwritten with new content.

**Syntax:**

ofstream fout;

fout.open(“fileName.txt”,ios::out); or fout.open(“fileName.txt”);

**Append(app):**

This mode will be helpful if you want to add new data into a file. In this mode control of writing will be transferred at the end of file. All new data will be added at the end of all previously added records.

**Syntax:**

ofstream fout;

fout.open(“fileName.txt”,ios::app);

**Lab Tasks**

**Create File and add data**

Perform the following task:

* Create an empty file **“marks.txt”.**
* Ask user to enter marks of 10 students one by one on console.
* Now write those marks to the “marks.txt” file using file handling.

**Read file**

Read marks of the students from file **“marks.txt”** and display on console.

**Update File data**

Read marks of the students from file **“marks.txt”** and add bonus 5 marks in each student marks, then store the updates marks in new file **“updated marks.txt”.**

**Perform operation on file**

Program that reads an array of 5 integer numbers from a file **“integer.txt”,** and display sum and average on console as well as store sum and average in file **“result.txt”.**

**Display character array from file**

Read UCP vision and mission from file “University Data.txt” and display it on console.

**Read and display character separated data**

Program that read character separated data form file **“sample data.txt”** and display it on console.

**Case conversion**

Program that read file “My data.txt” and convert upper case letters into lower case and lower-case letters into upper case. Then display it on console as well as write in new file “converted data.txt.