**National University of Computer & Emerging Sciences (NUCES) Islamabad,**

Department of Computer Science

**Programming Fundamentals – Fall 2022**

**LAB 13**

A picture containing text, gear, metalware

Description automatically generated

**Learning Outcomes**

In this lab you are expected to learn the following:

* 1D arrays
* 2D arrays
* char arrays, strings
* passing arrays (1D, 2D) to functions

**ARRAYS-I (1-Dimensional)**

**Arrays:**

An array is a data structure for storing more than one data item that has a similar data type. Values are stored in adjacent memory locations.

Declare using [] operator:

**int tests[5];**

The definition: int tests[5]; allocates the following memory:

Graphical user interface, timeline

Description automatically generated

**Arrays Terminology:**

Graphical user interface, text, application

Description automatically generated

**Size Declarators:**

• Named constants are commonly used as size declarators.

**const int SIZE = 5;**

**int tests[SIZE];**

**Array Initialization:**

• Arrays can be initialized with an **initialization list**:

**const int SIZE = 5;**

**int tests[SIZE] = {79, 82, 91, 77, 84};**

• The values are stored in the array in the order in which they appear in the list.

• The initialization list cannot exceed the array size.

Graphical user interface, application, Teams

Description automatically generated

**Implicit Array Sizing:**

• Can determine array size by the size of the initialization list:

**int quizzes[]={12, 17, 15, 11};**

Graphical user interface, text, application

Description automatically generated

• Must use either array size declarator or initialization list at array definition

* Once the array is declared (it’s size is set for fixed size arrays), you cannot change the size again. That is, you cannot add more elements to it

**Accessing Array Elements:**

• Array elements can be used as regular variables:

**tests[0] = 79;**

**cout << tests[0];**

**cin >> tests[1];**

**tests[4] = tests[0] + tests[1];**

• Arrays must be accessed via individual elements: **cout << tests; // not legal, it only works with character arrays**

• Can access element with a constant or literal subscript: **cout << tests[3] << endl;**

• Can use integer expression as subscript:

**int i = 5;**

**cout << tests[i] << endl;**

**Looping over an array:**

Graphical user interface, text

Description automatically generated

Graphical user interface, text, application

Description automatically generated

**Printing out array using loops:**

Graphical user interface, text, application

Description automatically generated

**Processing Array Contents:**

• Array elements can be treated as ordinary variables of the same type as the array

• When using ++, -- operators, don’t confuse the element with the subscript:

**tests[i]++; // add 1 to tests[i]**

**tests[i++]; // increment i, no effect on tests**

• To copy one array to another, don’t try to assign one array to the other:

**newTests = tests; // Won't work**

• Instead, assign element-by-element:

**for (i = 0; i < ARRAY\_SIZE; i++)**

**newTests[i] = tests[i];**

**Comparing Arrays:**

• To compare two arrays, you must compare element-by-element

Graphical user interface, text

Description automatically generated

**Example:** The following example displays the sum of elements of array.

Graphical user interface, text

Description automatically generated

**Example:** The following example shows how to Find the Highest and Lowest Values in a Numeric Array

Graphical user interface, text, application, email

Description automatically generated

**Arrays as function argument:**

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application

Description automatically generated

**ARRAYS-II(Multi-dimensional and Char arrays)**

In C++, we can create an array of an array, known as a multidimensional array.

For example: **int x[3][4];**

Here, x is a two-dimensional array. It can hold a maximum of 12 elements. We can think of this array as a table with 3 rows and each row has 4 columns as shown below.

Table

Description automatically generated

**Initialization of two-dimensional array**

**int test[2][3] = {2, 4, 5, 9, 0, 19};**

The above method is not preferred. A better way to initialize this array with the same array elements is given below:

**int test[2][3] = { {2, 4, 5}, {9, 0, 19}};**

This array has 2 rows and 3 columns, which is why we have two rows of elements with 3 elements each.

Calendar

Description automatically generated



**Example: Two-Dimensional Array**

// C++ Program to display all elements

// of an initialized two-dimensional array

#include <iostream>

using namespace std;

int main() {

int test[3][2] = {{2, -5}, {4, 0}, {9, 1}};

// use of nested for loop

// access rows of the array

for (int i = 0; i < 3; ++i) {

// access columns of the array

for (int j = 0; j < 2; ++j) {

cout << "test[" << i << "][" << j << "] = " << test[i][j] << endl; }

}

return 0;

}

**Output**

test[0][0] = 2

test[0][1] = -5

test[1][0] = 4

test[1][1] = 0

test[2][0] = 9

test[2][1] = 1

**2D array as Function arguments:**

•Use array name as argument in function call:

**getExams(exams, 2);**

•Use empty [] for row, size declarator for column in prototype, header:

**const int COLS = 2;**

// Prototype

**void getExams(int [][COLS], int);**

// Header

**void getExams(int exams[][COLS], int rows)**

**Example:**

Graphical user interface, text, application, Word

Description automatically generated

Graphical user interface, text

Description automatically generated

**Example 1 D Char**

C++ program to display a string entered by user:

#include <iostream>

using namespace std;

int main()

{

char str[100];

cout << "Enter a string: ";

cin >> str;

cout << "You entered: " << str << endl;

return 0;

}

**Output**

Enter a string: C++

You entered: C++

The following declaration and initialization create a string consisting of the word "Hello". To hold the null character at the end of the array, the size of the character array containing the string is one more than the number of characters in the word "Hello."

**char greeting[6] = {'H', 'e', 'l', 'l', 'o', '\0'};**

Following is the memory representation of above defined string in C/C++.

Graphical user interface, text, application

Description automatically generated

Graphical user interface, application, Teams

Description automatically generated

**Example:**

Graphical user interface, application, Teams

Description automatically generated

**Output:**

Graphical user interface, application, Teams

Description automatically generated

**Passing char array to a function:**

Graphical user interface, text, application

Description automatically generated

Graphical user interface, application, Word

Description automatically generated

**Lab Tasks**

**Q1)** Write a C++ program to find and print all unique elements of a given array of integers. The main() function should take the values of the arrays as input and then pass the array to the function. The function should find the unique elements and display them. ARRAY\_SIZE=10.

**Sample Output:**

A screenshot of a computer

Description automatically generated

**Q2)** Write a C++ program to find the number of pairs of integers in a given array of integers whose sum is equal to a specified number taken as user input.

**Sample Output:**

Graphical user interface, text, application

Description automatically generated

**Q3)** Write a C++ program having a function to count all the vowels in a string taken as user input.

**Sample Output:**

Original string: hello number of vowels: 2

Original string: AeiOu number of vowels: 5

**Q4)** Write a program that declares a two-dimensional integer array by the name of matrix with rows=5, columns=5, stores the elements in the array matrix with the help of a nested for loop. Now you have to find individual sum of each row of the 2D array and store corresponding result in 1D array i.e. sum of all the elements of row 0 should be stored in 1st element of 1D array, similarly sum of all elements of the second row of 2D array should be stored at the second index of 1D array. Display the final sum array (i.e., 1D array). Think about the size of 1D array yourself.

**Example:**

**Array:**

2 3 5 3 1

4 5 1 2 1

4 7 3 2 0

2 1 1 5 1

1 7 8 9 0

**Sum array: 14 13 16 10 25**

**Q5)** Write a C++ program to take 3x3 integer array as input from the user using nested loops. Now, pass the 2D array to a function named Reverse. In the function, copy all the elements in another array but in reverse order and show the output.