National University of Computer & Emerging Sciences, Karachi Computer Science Department

Course Code: CL-2001	Course : Data Structures Lab

Spring 2024, Lab Manual - 01

ARRAYS

An Array is a collection of fixed number of elements of same data type.

1-D ARRAY

- 1-D Array is a form of array in which elements are arranged in a form of List.
- To declare a 1D array you need to specify the data type, name and array size.

dataType arrayName [arraySize] ;

- Following is the declaration of a 1D array.

int numArray[5];

where;

Data Type: Integers (int)

Array Name: numArray

Array Size: 5

- To access array elements, you use the array name along with the index in subscript operator **“[ ]”**.

numArray[0], numArray[1], numArray[2], numArray[3], numArray[4]

where;

Index of the array starts with zero ‘0’.

Index of the last element is always ‘size - 1’ (in this case it is 4).

Example Code for 1-D Array

Program to read five numbers, find their sum, and print the numbers in reverse order.

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

int main() {
int item[5]; // Declare an array 'item' of five components
int sum = 0;
int counter;
cout << "Enter five numbers: ";

// Input five numbers into the array and calculate the sum
for (counter = 0; counter < 5; counter++) {
cin >> item[counter];
sum = sum + item[counter];
}

cout << endl;
cout << "The sum of the numbers is: " << sum << endl;

cout << "The numbers in reverse order are: ";

// Print the numbers in reverse order
for (counter = 4; counter >= 0; counter--)
cout << item[counter] << " ";

cout << endl;

return 0;
}
```

Sample Run: In this sample run, the user input is shaded.

```
Enter five numbers: 12 76 34 52 89
The sum of the numbers is: 263
The numbers in reverse order are: 89 52 34 76 12
```

2-D ARRAY

1. 2-D Array is a collection of fixed collection of elements arranged in rows and columns.
2. To declare a 2D array you need to specify the data type, name and no. of rows and columns.
   dataType arrayName [rowSize][columnSize] ;
3. Following is the declaration of a 2D array.

int numArray[5][5];

4. To access array element you use the array name along with the row Index and column Index in subscript operator **"[ ][ ]".**

   numArray[0][0], numArray[1][1], numArray[2][2], numArray[3][3], numArray[4][4].

   where; Index for the rows and columns of the array starts with zero '0'.

   Index of the last element in rows and columns is always **'sizeofRow - 1'** and **'sizeofColumn -1'** respectively (in this case it is 4).

Example Code for 2-D Array:

Program to read a 2D array of size 3x3 find the sum for each row, print the sum line by line.

```
#include <iostream>
using namespace std;  int main()
{
int item[3][3];      //Declare an array of size 3x3
int sum = 0;
int row, col;
cout << "Enter array elements: " << endl;
for (row = 0; row < 3; row++)
{
for (col = 0; col < 3; col++)
{
cin >> item[row][col];
sum = sum + item[row][col];
}
cout << "The sum of row " << i << " : " << sum << endl;
}
cout << endl; return 0;
}
```

Sample Run: In this sample run, the user input is shaded.

```
Enter array elements:
12 76 34
The sum of row 0 : 122 52 89 48
The sum of row 1 : 189 22 63 99
The sum of row 2 : 184
```

POINTERS

A Pointer is a variable whose content is a memory address.

SINGLE POINTERS

1. To declare a single pointer variable you need to specify the data type, an asterisk symbol (*) and the name of the pointer variable.

   dataType *ptrName;

2. Following is the declaration of a Pointer variable.

   int *ptr;

3. Pointer variable holds the memory address of the variable which is of same data type (integer in this case).
4. To assign the memory address of any variable to the pointer variable we use Address of Operator (&).

   int intVar
   = 5; ptr = &intVar;

   In this statement ptr now holds the memory address of an integer variable 'intVar'.

5. To access the value at the memory address (currently stored) in the variable we use Dereferencing Operator (*).
6. Do not confuse this with the symbol used for the declaration of a pointer.

   int intVar2 = *ptr;

   In this statement another integer variable 'intVar2' is now initialized with the value at the memory address which is stored in ptr (that is the value of intVar).

Example Code for Single Pointer

The following program illustrates how pointer variables work:

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

// Function to display the elements of a 1D array
void displayArray(int* arr, int size) {
cout << "Array elements: ";
for (int i = 0; i < size; i++) {
cout << arr[i] << " ";
}
cout << endl;
}

int main() {
// Declare and initialize an integer array with size 5
int numbers[] = {10, 20, 30, 40, 50};

// Display the original array
displayArray(numbers, 5);

// Modify an element of the array
numbers[2] = 100;

// Display the modified array
displayArray(numbers, 5);

// Pass the array to a function
displayArray(numbers, 5);

return 0;
}
```

2D Pointers:

In C++, a 2D array can be represented using pointers. A 2D pointer is essentially a pointer to an array of pointers, where each pointer points to a separate array representing a row in the 2D array.

Example Code for 2D Pointers:

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

// Function to display the elements of a 2D array
void displayArray(int** arr, int rows, int cols) {
cout << "Array elements:\n";
for (int i = 0; i < rows; i++) {
for (int j = 0; j < cols; j++) {
cout << arr[i][j] << " ";
}
cout << endl;
}
}

int main() {
const int rows = 3;
const int cols = 4;

// Declare a 2D array using pointers
int** matrix = new int*[rows];
for (int i = 0; i < rows; i++) {
matrix[i] = new int[cols];
}

// Initialize the elements of the 2D array
int value = 1;
for (int i = 0; i < rows; i++) {
for (int j = 0; j < cols; j++) {
matrix[i][j] = value++;
}
}

// Display the 2D array
displayArray(matrix, rows, cols);
// Modify an element of the array
matrix[1][2] = 100;

// Display the modified array
displayArray(matrix, rows, cols);

// Deallocate the memory
for (int i = 0; i < rows; i++) {
delete[] matrix[i];
}
delete[] matrix;

return 0;
}
```

Lab Task:

1- Given an array of integers nums and an integer target, return indices of the two numbers such that they add up to target.

You may assume that each input would have exactly one solution, and you may not use the same element twice.You can return the answer in any order.

Input: nums = [2,7,11,15], target = 9

Output: [0,1]

Output: Because nums[0] + nums[1] == 9, we return [0, 1].

2- Write a program that uses pointers to find the maximum and minimum elements in a 1D array without using array indices.

3-Q4:Given a list of integers, find out the number that has the highest frequency. If there are one or more such numbers, output the smaller one.

Input: size = 6
4 6 8 5 3 2
Output: 2

Input: size = 8
1 2 2 1 1 2 1 2
Output: 1

4- Given an array of integers, implement an algorithm to sort the array using only pointers and without using additional data structures.

5- Implement a function that transposes a given 2D array using pointers. The function should modify the original array in-place.

6- Given a two dimensional array print its mirror image if the mirror is placed along one of the sides of the array.

Horizontal image

Original Array:	Mirror Image:
1 2 3	3 2 1
4 5 6	6 5 4
7 8 9	9 8 7

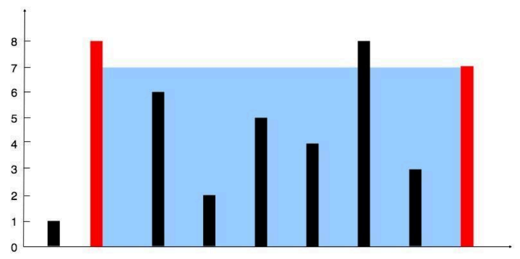
Vertical image

Original Array:	Mirror Image:
1 2 3 4	4 3 2 1
5 6 7 8	8 7 6 5
9 10 11 12	12 11 10 9

7 -You are given an integer array height of length n. There are n vertical lines drawn such that the two endpoints of the ith line are (i, 0) and (i, height[i]).

Find two lines that together with the x-axis form a container, such that the container contains the most water. Return the maximum amount of water a container can store. Notice that you may not slant the container.

Input: height = [1,8,6,2,5,4,8,3,7]
Output: 49
Explanation: The above vertical lines are represented by array [1,8,6,2,5,4,8,3,7]. In this case, the max area of water (blue section) the container can contain is 49.



8- Write a program that optimally searches for a specific value in a 2D array and returns its position (row and column) using pointers. Assume the array is sorted in both rows and columns.

9- Create a program that uses an array of pointers to strings. Allow the user to input several strings, and then display them in reverse order.

10- Write a program that performs a spiral traversal of a 2D array (matrix) using pointers. The program should print the elements in a clockwise spiral order.
Input

```
1  2  3  4
5  6  7  8
9 10 11 12
13 14 15 16
```

Output

```
1  2  3  4  8 12 16 15 14 13 9 5 6 7 11 10
```

11- Given an array of integers arr[] of size N and an integer, the task is to rotate the array elements to the left by d positions using pointers.
Examples:
Input: arr[] = {1, 2, 3, 4, 5, 6, 7}, d = 2
Output: 3 4 5 6 7 1 2
Input: arr[] = {3, 4, 5, 6, 7, 1, 2}, d=2
Output: 5 6 7 1 2 3 4

12- Print all the substrings of a given string using recursions.