SLIIT ACADEMY

Higher Diploma in Information Technology
Year 1, Semester 1

Introduction to Programming(C++)

Lecture 07 : Advanced Programming Techniques - Array Processing in C++ - I

Intended Learning Outcomes

End of this lecture you will be able to learn ,

LO1:To introduce arrays and the uses of arrays.

LO2:To understand the typical operations perform on arrays with use of C++

LO3:Identification of string functions to manipulate character arrays

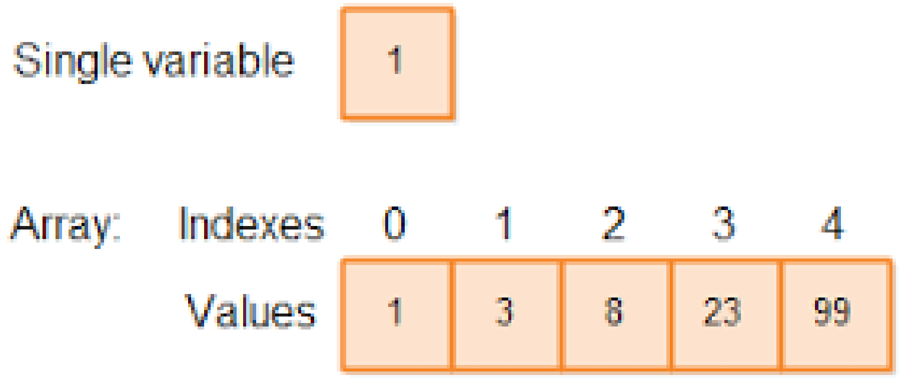
LO4:To illustrate the manipulation of one and two-dimensional arrays.

Motivation

- The variables we have been using so far are called simple variables; They hold one value.

  Example : Employee_Name, counter

- There are situations in which we wish to store a group of related items and to be able to refer to them by a common name.
- The array variable allows us to do this.



Array Processing

- Arrays are one of the most powerful programming tools available which provides the programmer with a way of organizing a collection of homogeneous data items **(items that have the same type and same length)** into a single data structure.
- An array is a data structure that is made up of several variables all of which have the same data type and maintained under a common name.

Advantages of Arrays

- No matter how many values are stored, the program only needs one name to refer to the values.
- Program logic can be shorter and, in many cases, simpler
  - Instead of using separate field names, loops can be used to refer the items in an array.
  - During each pass of the loop, the subscript changes.

SLIIT ACADEMY

Typical Operations Perform on Arrays

- Describe or Define the array
  - Reserve storage
  - Specify the type of data the array contains.
- Initialize the array
  - Put values in array
- Access or lookup entries
  - Find entries in array

Types of Arrays in C++

There are 2 types of an array in C++ :

- One-dimensional array
- Multidimensional array

C++ Array Declaration

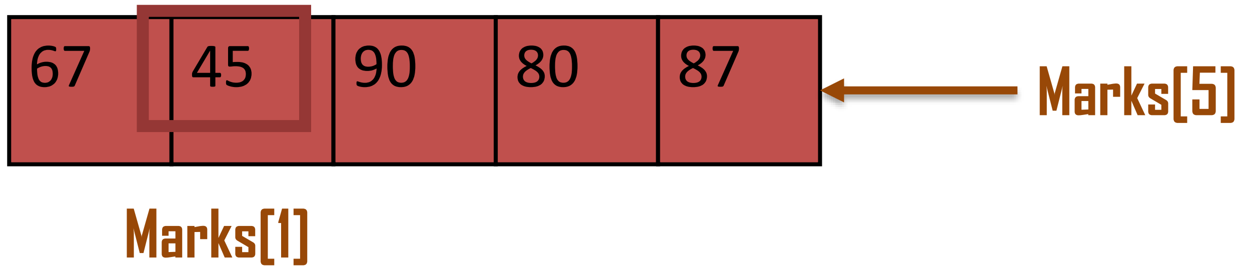
To declare an array in C++, the programmer needs to consider the followings:

- The data type of the values which will be stored in the array
- The name of the array (an identifier that will be used to access and update the array values)
- The dimensionality of the array:
  - One dimensional (list of values),
  - Two-dimension array (a matrix or a table), etc.
- The size of each dimension

```
dataType arrayName[arraySize];
```

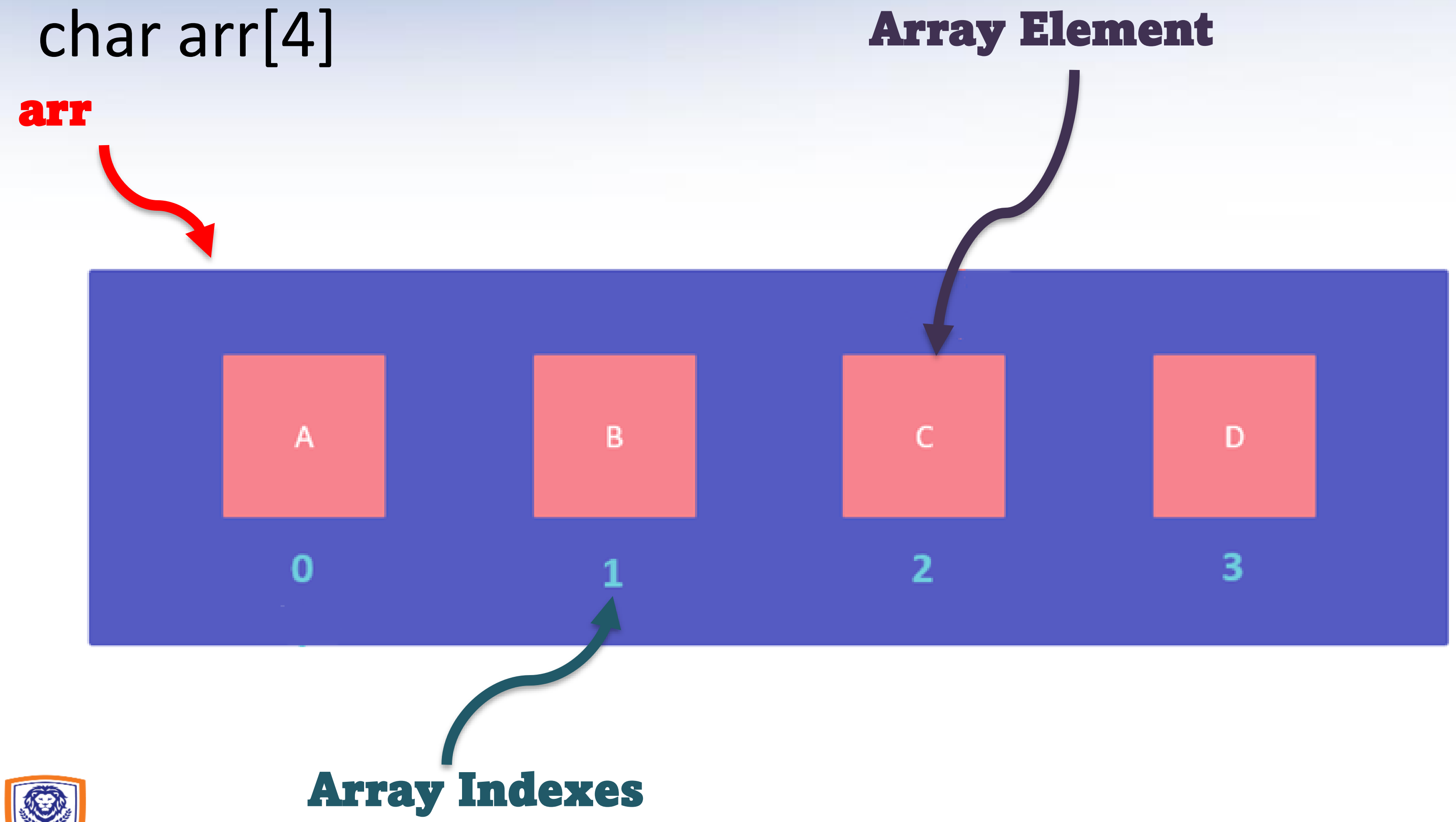
One Dimensional Arrays

- One dimensional array is a sequence of data items where each item or element can be referenced by the position it occupies in the sequence.
- The array element can be simple or complex data type but they all must be the same.



Concept Diagram of 1D Arrays

char arr[4]



Array size = 4

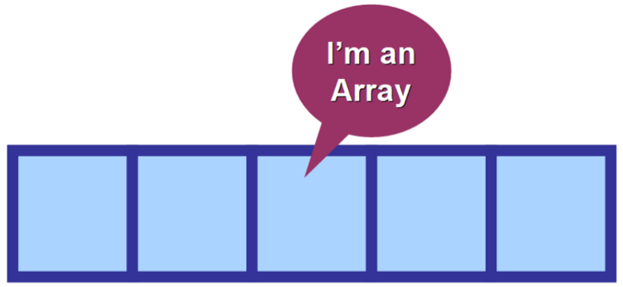
No of Elements = 4

First Index = 0

Last Index = 3

Array Processing

- The individual data items that make up the array are referred to as the **element of the array.**
- Elements in the array are distinguished from one another by using **an index or subscript**, enclosed in parentheses.
- Subscript can be a number (0,1,2,3,4…)

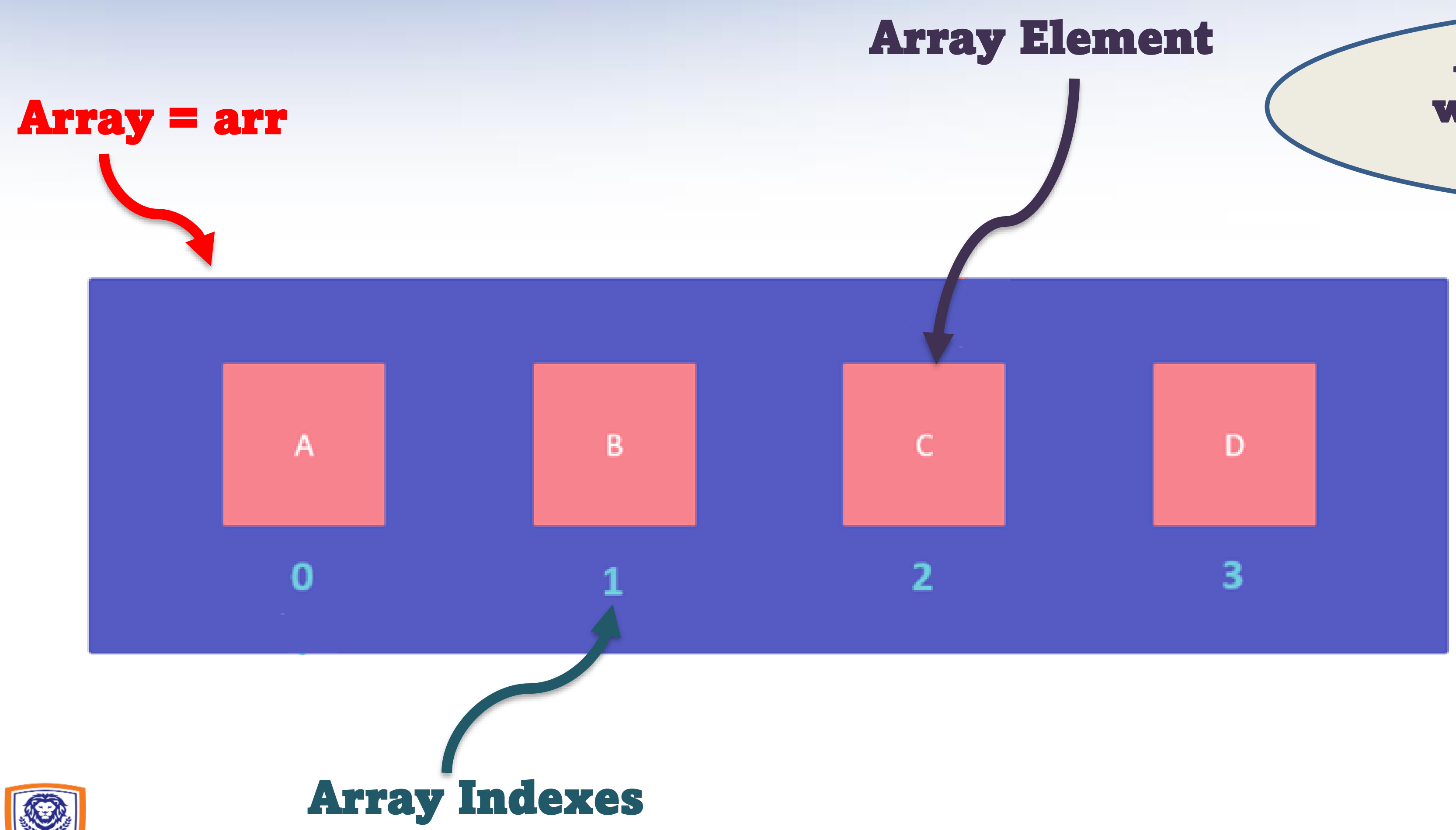


Access Elements in C++ Array

- In C++, each element in an array is associated with a number. The number is known as an **array index**.
- We can access elements of an array by using those indices.

```
arrayName[index];
```

Access an Array Element



An array is a shorthand way of naming a bunch of variables.

arr[0] = 'A'

arr[1] = 'B'

arr[2] = 'C'

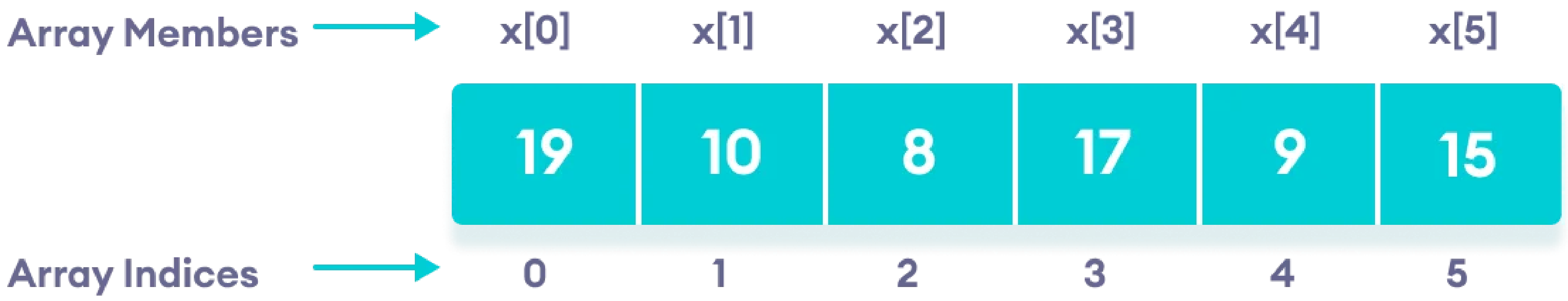
arr[3] = 'D'

C++ Array Initialization

Method 01:

In C++, it's possible to initialize an array during declaration.

```
int arr[6] = {19,10,8,17,9,15};
```



C++ Array Initialization

Method 02:

Another method to initialize array during declaration.

```
int arr[] = {19,10,8,17,9,15};
```

Here, we have not mentioned the size of the array. In such cases, the compiler automatically computes the size.

C++ Array Initialization

Method 03: Use the assignment Operator

```
int arr[6];
arr[0] = 19;
arr[1] = 10;
arr[2] = 08;
arr[3] = 17;
arr[4] = 09;
arr[5] = 15;
```

Method 04: Read input values into the array from the keyboard.

```
int arr[6];
for(int i=0; i<6; i++)
{
    cin>>arr[i];
}
```

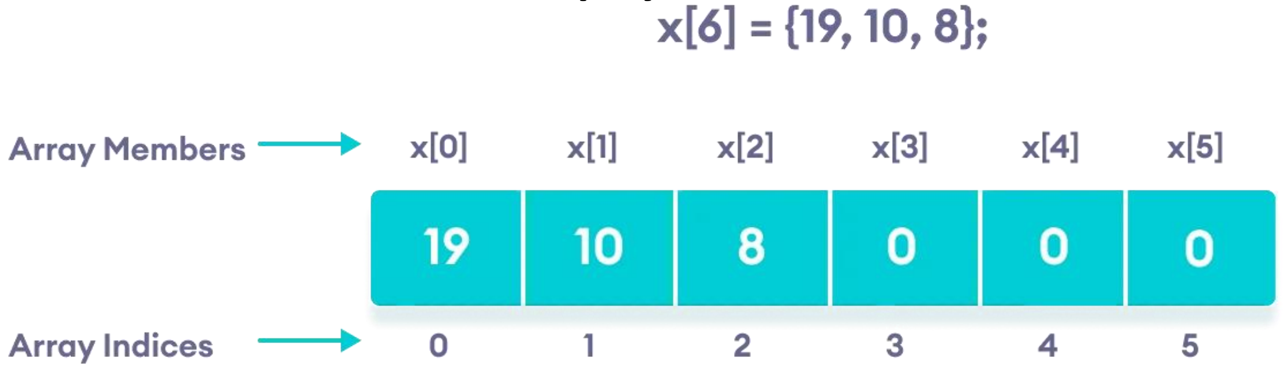
SLIIT ACADEMY

C++ Array With Empty Members

- In C++, if an array has a size n, can store up to n number of elements in the array. However, what will happen if store less than n number of elements.

```
int arr[6] = {19,10,8};
```

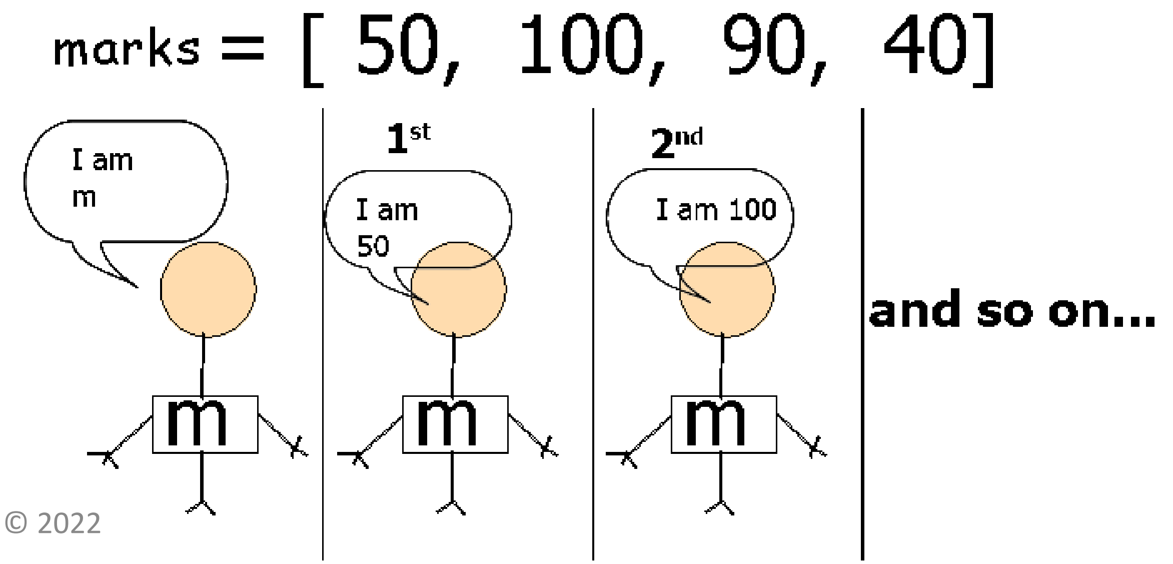
- In such cases, the compiler assigns random values to the remaining places. Oftentimes, this random value is simply 0.



Practice Question 01

Write a C++ program that will prompt for receive 10 examination scores from a mathematics test, compute the class average, and display all the scores and the class average to the screen.

Hint: To store marks use an array.



The C-Style Character String(Cstrings)

- CStrings are similar to normal strings, but they **contain an extra null character at the end of the string** that makes them Cstrings. This null character corresponds to the end of the string.
- The C string is a **one-dimensional array of characters** which is terminated by a null character '\0'.

Example: 'A' is the character A

"A" is the C-string A : "A" represents two characters, 'A' and '\0'

The C-Style Character String

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

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."

If you follow the rule of array initialization, then can write the above statement as follows

```
char greeting[] = “Hello”
```

Following is the memory presentation of above defined string in C++ :

you do not place the null character at the end of a string constant. The C++ compiler automatically places the '\0' at the end of the string when it initializes the array.

Index	0	1	2	3	4	5
Variable	H	e	l	l	o	\0
Address	0x23451	0x23452	0x23453	0x23454	0x23455	0x23456

SLIIT ACADEMY

Cstring Manipulation Functions

All the functions are defined in the cstring header file.

strlen() function

The strlen() function in C++ returns the length of the given C-string.

strcpy() function:

The strcpy() function in C++ copies a character string from source to destination.

Two Dimensional Arrays

- Two dimensional arrays have two subscripts to identify an element of the array.
- They can be thought of as the rows and columns in a table.
- The convention in this case is to use two subscripts.

  Example: grades [num_students] [num_tests]

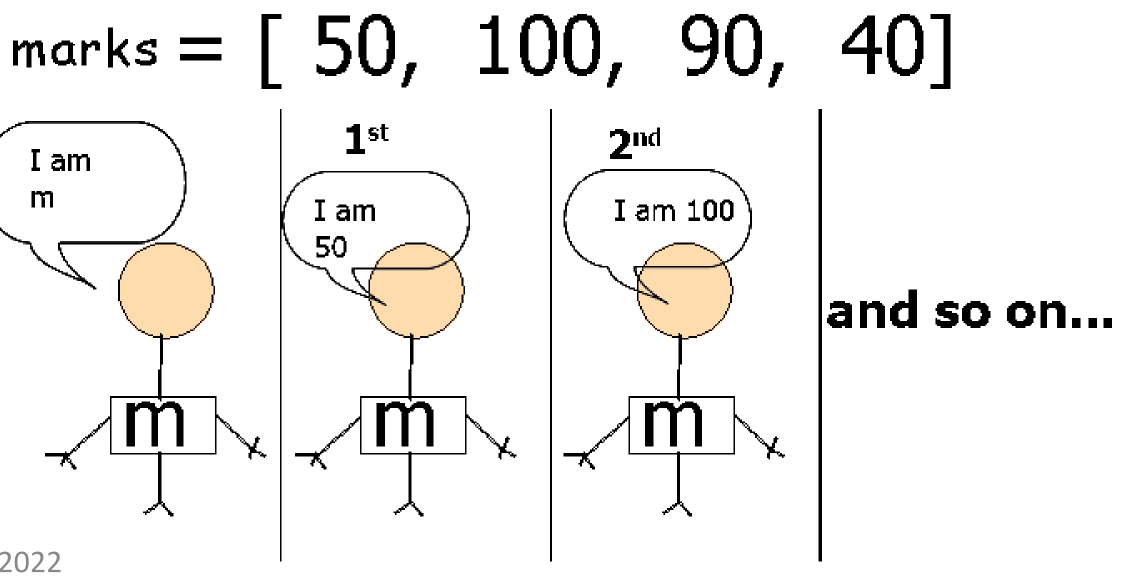
- By convention, the first subscript is understood to be for rows and the second subscript for columns.

Two Dimensional Arrays

- Two-dimensional array is loaded in columns within row order.
- All the columns for row one are loaded before moving to row two and loading the columns for that row, and so on .
- The reading of a series of values from a file into a two-dimensional array can be represented by a **NESTED FOR Loop or NESTED WHILE Loop.**

Practice Question 02

Design a program that will prompt to take marks of the 4 tests from the user. There are five students have attempted each exam. Also, program should display the marks of each exam obtained by the students.



C++ Two-Dimensional Array Declaration

Declaration of two-dimensional array with 5 rows and 4 columns.

```
int grades[5][4]
```

Initialization at declaration of the array.

```
int grades[5][4] = {{67,45,76,14},{48,59,38,90},
                    {79,49,80,69},{41,74,56,78},
                    {68,75,80,89}}
```

C++ Two-Dimensional Array Representation

```
int grades [5] [4]
```

Array Declaration

	Col 0	Col 1	Col 2	Col 3
Row 0	[0][0]	[0][1]	[0][2]	[0][3]
Row 1	[1][0]	[1][1]	[1][2]	[1][3]
Row 2	[2][0]	[2][1]	[2][2]	[2][3]
Row 3	[3][0]	[3][1]	[3][2]	[3][3]
Row 4	[4][0]	[4][1]	[4][2]	[4][2]

Sample Dataset

	exam1	exam2	exam3	exam4
Student1	67	45	76	14
Student2	48	59	38	90
Student3	79	49	80	69
Student4	41	74	56	78
Student5	68	75	80	89

SLIIT ACADEMY

Working with Two-Dimensional Arrays

Loading values to the Two-Dimensional Arrays

```
int grades [5][4];
for(int m=0;m<5;m++)
   {
        for(int n=0;n<4;n++)
        {
             cin>>grades[m][n];
        }
   }
```

Printing Values of Two-Dimensional Arrays

```
int grades [5][4];
for(int m=0;m<5;m++)
   {
        for(int n=0;n<4;n++)
        {
             cout<<grades[m][n];
        }
   }
```

Summary

- One dimensional Arrays
- Manipulation of Character arrays
- Two dimensional Arrays