Unit V – Part 02
(Array in C)

Lecture Slide

AS2023

2D Array & Multi-Dimensional Array

Two Dimensional Array

- Consider the following data table which shows the students' scores for three subjects

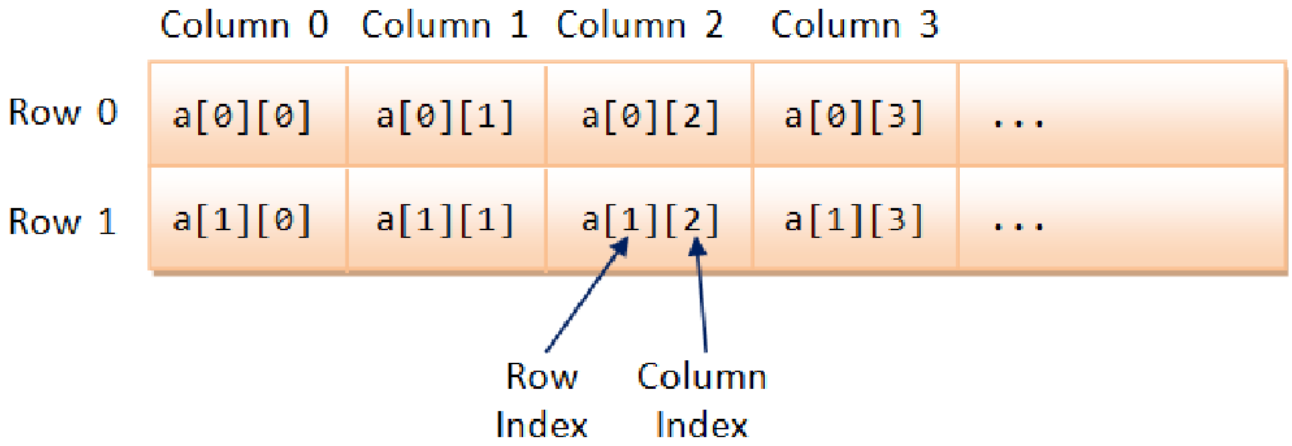
	Maths	Dzongkha	English
Namkha	56	78	50
Amrita	60	40	65
Birkha Daju	59	35	39
Sumdho	89	78	50

- Such table of items can be represented by using 2-D array
- Generally declared as:

```
type array_name[row_size][column_size]
```

2-D Array cont..

- Elements are represented as shown below:



- As with the 1-D array, each dimension of the array is indexed from ***0*** to ***n-1***
- The first index selects the row and the second index selects the column within the row

Initializing 2-D Array

- Initialization can be done in the following ways:

```
int table[2][3] = {0, 0, 0, 1, 1, 1};
```

(OR)

```
int table[2][3] = {{0, 0, 0},{ 1, 1, 1}};
```

(OR)

```
int table[2][3] = {
                    {0, 0, 0},
                    {1, 1, 1}
                  };
```

- The missing values will be automatically set to zero for integer and float and NULL for char type

```
int table [2][3] = {{1, 1}, {2} };
```

Demonstration

- WAP to compute the sum of two 2x2 matrices
- Consider the following data table:

	Maths	Dzongkha	English
Namkha	56	78	50
Amrita	60	40	65
Birkha Daju	59	35	39
Sumdho	89	78	50

WAP to:

1. Compute total marks scored by each student
2. Average mark of each student

Multi-Dimensional Arrays

- C allows arrays of three or more dimensions
- General form are written as:

```
type array_name[s1][s2].....[sm]
```

- The number of subscripts determines the dimension of an array.
- Example

```
int calender[12][4][30];
int table[5][4][2][10];
```

Dynamic Arrays

- Creation of array at compile time follows *static memory allocations*
- An array that receives static memory allocation are called as ***static array***
- This approach works as long as we know exactly what our data requirements are.
- And such array can't be modified during run time
- Thus, C allows dynamic memory allocation and array created during run time is called as *dynamic array*

Demonstration: WAP to read MxN matrix and display the highest value in each row

Passing Array to Function

Passing 1 –D Array to function

- To pass 1-D array to a function it is sufficient to list the name of array, *without any subscripts*, and the size of the array as arguments

```
function_name(array_name, size);
```

- And the called function expecting this function must be defined appropriately defined as

```
type function_name(type array_name[], int size)
```

Demonstration: WAP to pass the 1D array {7,3,4,7,8,1} to function and find the sum of all array values

Passing Array to function

- Rules to pass an array to a function
  1. The function must be called by passing only the name of the array
  2. In the function definition, the formal parameters must be an array type; the size of the array does not need to be specified
  3. The function prototype must show that the argument is array
- When function changes the value of the elements, the change will be updated in the original array that is passed to a function
- But this is not applicable for individual element passed as an argument

Passing 2–D Array to function

- Rules
  1. The function must be called by passing only the array name
  2. In the function definition, we must indicate that the array has two dimensions by including two sets of backets
  3. The size of the second dimension must be specified
  4. The prototype declaration should be similar to function header

Demonstration: WAP to compute the average of 2-D array.

Thank you