SNS COLLEGE OF TECHNOLOGY

Coimbatore – 35

An Autonomous Institution

Accredited by NBA – AICTE and Accredited by NAAC – UGC with 'A++' Grade
Approved by AICTE, New Delhi & Affiliated to Anna University, Chennai

DEPARTMENT OF INFORMATION TECHNOLOGY

23ITT101 – PROGRAMMING IN C & DS

UNIT III – ARRAYS AND INTRODUCTION TO DATA STRUCTURES

ONE AND TWO DIMENSIONAL ARRAYS

INTRODUCTION

Array is a data structure that is used to store variables that are of similar data types at contiguous locations. The main advantage of the array is random access and cache friendliness. There are mainly three types of the array:

- **One Dimensional (1D) Array**
- **Two Dimension (2D) Array**
- **Multidimensional Array**

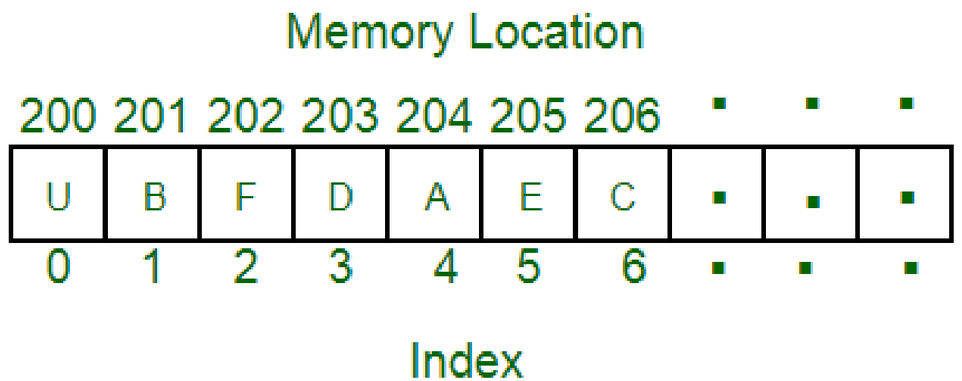
ONE – DIMENSIONAL ARRAYS

A one-dimensional array is a type of array in which the elements are arranged in a single row. Each element in the array can be accessed using a unique index or position in the array. Here are some key points about one-dimensional arrays:

- ✓ One-dimensional arrays are also known as linear arrays or vectors.
- ✓ In most programming languages, the index of the first element in a one-dimensional array is 0.
- ✓ The elements in a one-dimensional array can be of any data type, such as integers, floats, characters, or strings.
- ✓ One-dimensional arrays are often used to store and manipulate lists of data, such as scores, temperatures, or stock prices.
- ✓ The size of the array is fixed.

REPRESENTATION OF 1-D ARRAY

char alphabets[5] = {‘U’, ‘B’, ‘F’, ‘D’, ‘A’, ‘E’, ‘C’};



In this example, we create a one-dimensional array called alphabets that contains five elements. The first element in the array is ‘U’, the second element is ‘B’, and so on. We can access the elements of the array using their index. For example, alphabets[0] returns the value ‘U’, alphabets[3] returns the value ‘D’, and so on.

EXAMPLE

```
// C Program to illustrate the use of 1D array
#include <stdio.h>
int main()
{
        int arr[5];             // 1d array declaration
        for (int i = 0; i < 5; i++) {           // 1d array initialization using for loop
                arr[i] = i * i - 2 * i + 1;
        }
        printf("Elements of Array: ");
        for (int i = 0; i < 5; i++) {           // printing 1d array by traversing using for loop
                printf("%d ", arr[i]);
        }
        return 0;
}
```

```
Elements of Array: 1 0 1 4 9
```

TWO DIMENSIONAL ARRAYS

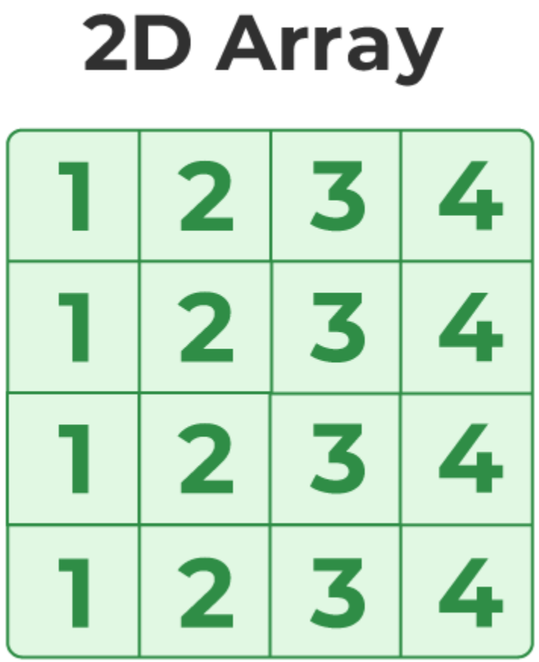
A Two-Dimensional array or 2D array in C is an array that has exactly two dimensions. They can be visualized in the form of rows and columns organized in a two-dimensional plane.

- It is a list of lists of the variable of the same data type.
- It also allows random access and all the elements can be accessed with the help of their index.
- It can also be seen as a collection of 1D arrays.
- It is also known as the Matrix.
- Its dimension can be increased from 2 to 3 and 4 so on.
- They all are referred to as a multi-dimensional array.
- The most common multidimensional array is a 2D array.

REPRESENTATION OF 2D ARRAYS

array_name[size1] [size2];

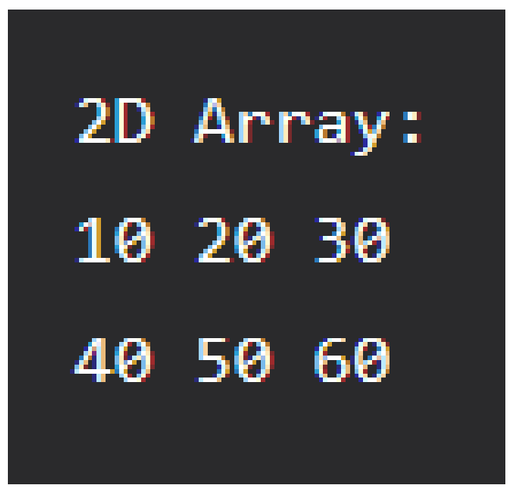
Here,
size1: Size of the first dimension.
size2: Size of the second dimension.




```
int  arr[2][3] = { 10, 20, 30, 40, 50, 60 };
```

EXAMPLE

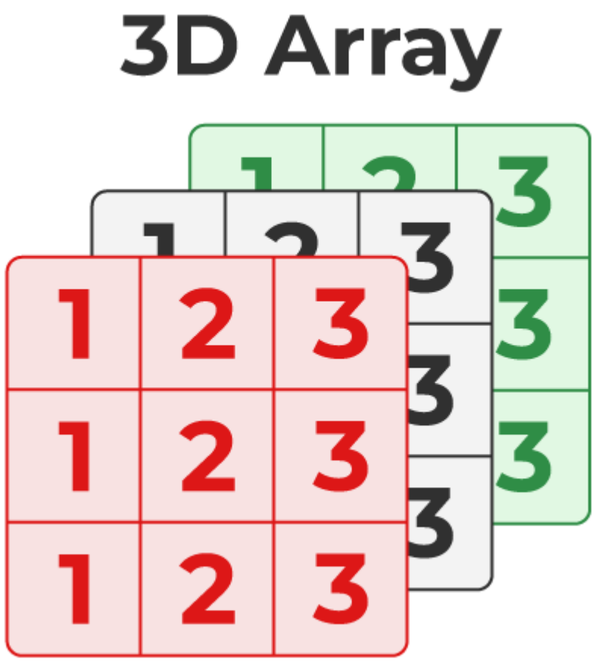
```
// C Program to illustrate 2d array
#include <stdio.h>
int main()
{
        int arr[2][3] = { 10, 20, 30, 40, 50, 60 };          // declaring and initializing 2d array
printf("2D Array:\n");
        for (int i = 0; i < 2; i++) {                  // printing 2d array
                for (int j = 0; j < 3; j++) {
                        printf("%d ",arr[i][j]);
                }
                printf("\n");
        }
        return 0;
}
```



MULTI DIMENSIONAL ARRAYS

Another popular form of a multi-dimensional array is Three Dimensional Array or 3D Array. A 3D array has exactly three dimensions. It can be visualized as a collection of 2D arrays stacked on top of each other to create the third dimension.

array_name [size1] [size2] [size3];



int arr[2][2][2] = { 10, 20, 30, 40, 50, 60 };

EXAMPLE

```
// C Program to illustrate the 3d array
#include <stdio.h>
int main()
{
        int arr[2][2][2] = { 10, 20, 30, 40, 50, 60 };          // 3D array declaration
        for (int i = 0; i < 2; i++) {                           // printing elements
                for (int j = 0; j < 2; j++) {
                        for (int k = 0; k < 2; k++) {
                                printf("%d ", arr[i][j][k]); }
                        printf("\n");    }
                printf("\n \n");    }
        return 0;
}
```

```
10 20
30 40


50 60
0 0
```

DIFFERENCE BETWEEN 1D AND 2D ARRAYS

Basis	One Dimension Array	Two Dimension Array
Definition	Store a single list of the element of a similar data type.	Store a ‘list of lists’ of the element of a similar data type.
Representation	Represent multiple data items as a list.	Represent multiple data items as a table consisting of rows and columns.
Declaration	The declaration varies for different programming language: 1.For C++, ***datatype variable_name[row]*** 2.For Java, ***datatype [] variable_name= new datatype[row]***	The declaration varies for different programming language: 1.For C++, ***datatype variable_name[row][column]*** 2.For Java, ***datatype [][] variable_name= new datatype[row][column]***
Dimension	One	Two
Size(bytes)	size of(datatype of the variable of the array) * size of the array	size of(datatype of the variable of the array)* the number of rows* the number of columns.
Address calculation.	Address of a[index] is equal to (base Address+ Size of each element of array * index).	Address of a[i][j] can be calculated in two ways row-major and column-major **1.Column Major:** Base Address + Size of each element (number of rows(j-lower bound of the column)+(i-lower bound of the rows)) **2.Row Major:** Base Address + Size of each element (number of columns(i-lower bound of the row)+(j-lower bound of the column))