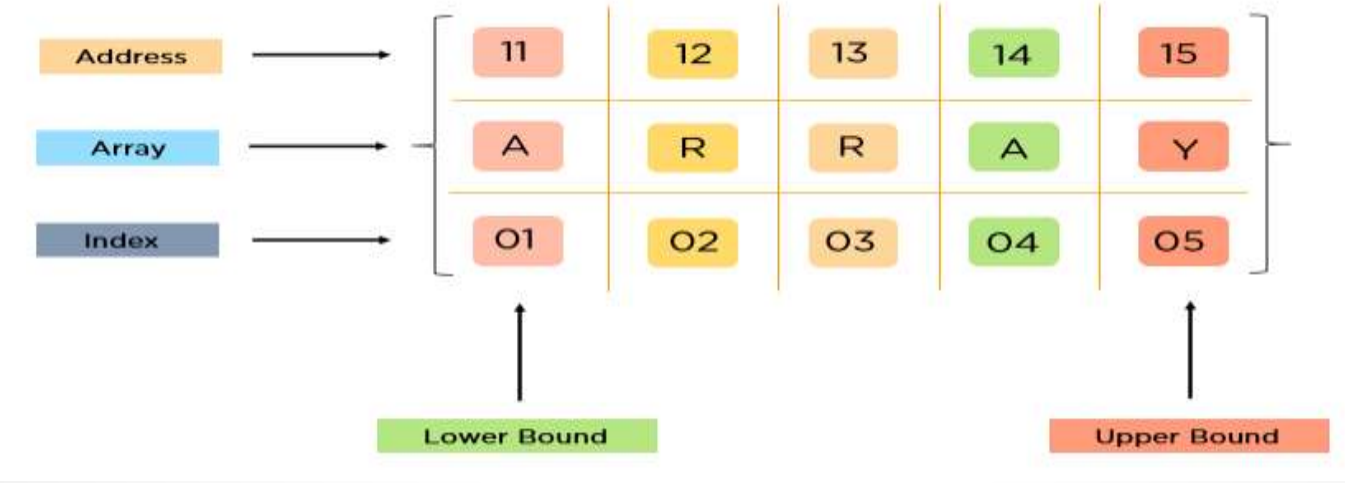
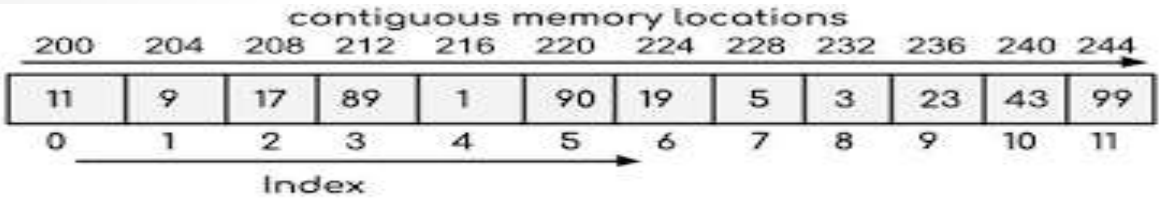
What is an Array?

- An array is a fixed size sequential collection of elements of the same data type that share a common name
- An array is a derived data type.
- Array is a type of linear data structure .
- An array is a collection of items of same data type stored at contiguous memory locations.



Array in memory



An array is used to represent a list of numbers, or a list of names
For Example:

- List of employees in an organization.
- Test scores of a class of students.
- List of customers and their telephone numbers.
- List of students in the college

Why do we need an array ?

Consider the following issue:

"We have a list of 1000 students' marks of an integer type.

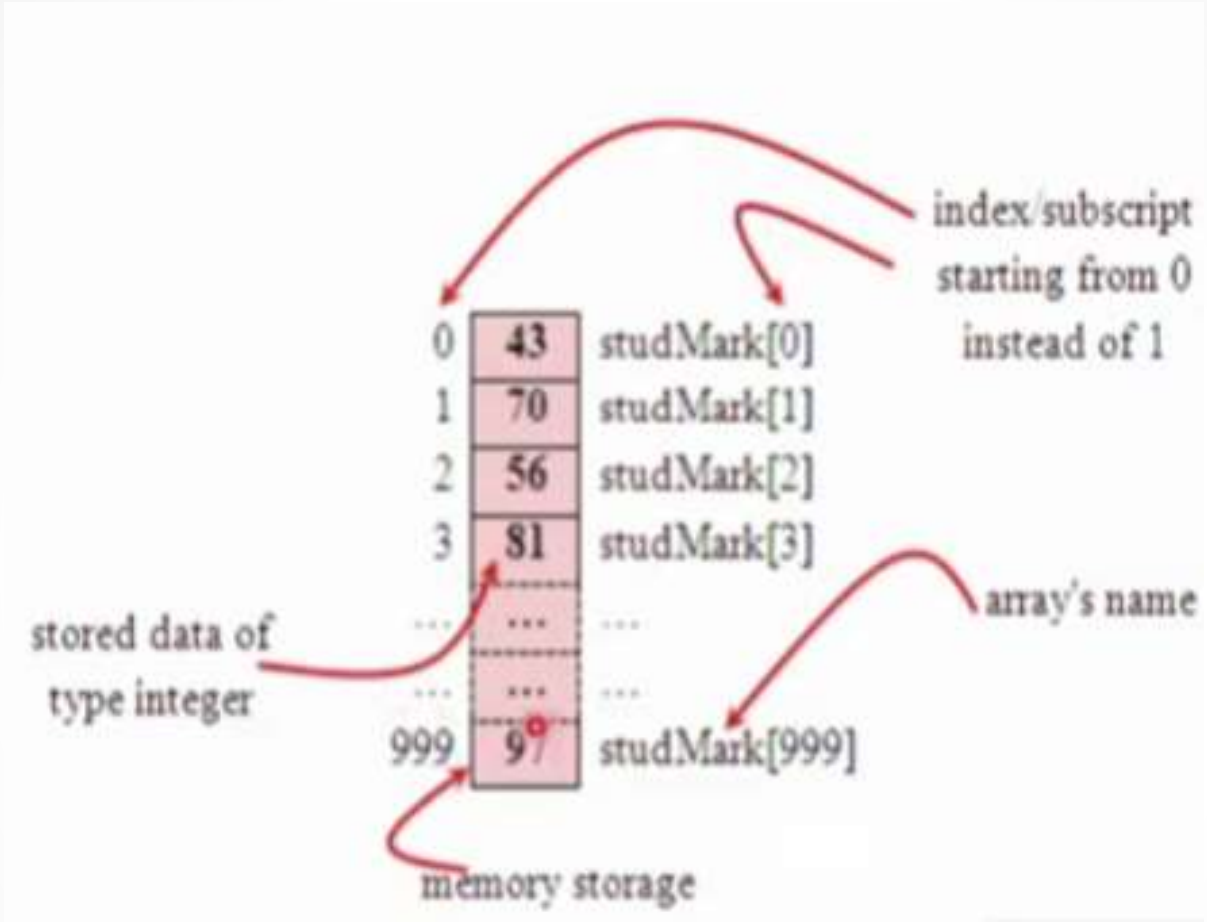
If using the basic data type (int), we will declare something like the following…

int studMark1, studMark2, studMark3, studMark4, …, studMark998, stuMark999, studMark 1000

By using an array, we just declare like this,

int studMark[1000];

This will reserve 1000 contiguous memory locations for storing the students' marks.


index/subscript
starting from 0
instead of 1
0
43
studMark[0]
1
70
studMark[1]
2
56
studMark[2]
3
81
studMark[3]
...
...
...
...
...
...
999
97
studMark[999]
stored data of
type integer
array's name
memory storage

Array Initialization.

An array may be initialized at the time of declaration,giving initial values to an array.

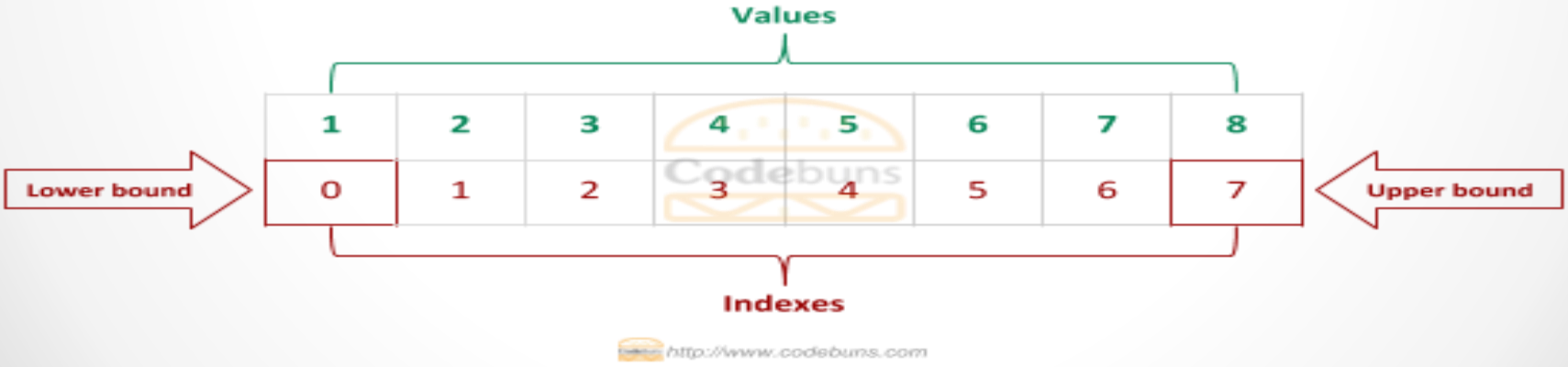
■ Initialization of an array may take the following form,

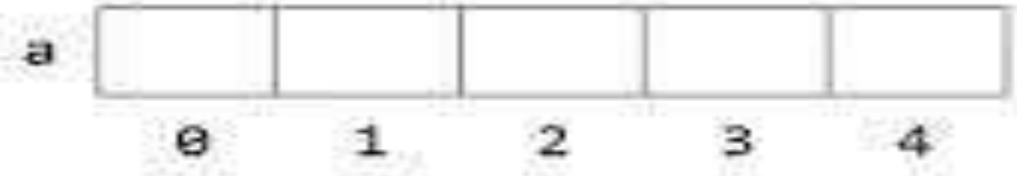
-type array_name[size] = {a_list_of_value};

```
int   id[7] = {1, 2, 3, 4, 5, 6, 7};

float   fFloatNum[5] = {5.6, 5.7, 5.8, 5.9, 6.1};

char     chVowel[6] = {'a', 'e', 'i', 'o', 'u', '\0'};
```

Lower bound & Upper bound of an array

LB refers to the lower bound of an array (first index element of an array) and UB refers to the upper bound of an array, that is the last index of an array. Array - [1,2,3,4,5,6,7,8], where 1 is the lower bound and 8 is the upper bound element.






```
Size of the Array  = 5
Index of the Array = 0 1 2 3 4

First index = 0
Last index  = 4

First index is called lower bound
Last index is called upper bound
```

Memory Address Calculation in an Array

Address Calculation in single (one) Dimension Array:

Actual Address of the **1st** element of the array is known as **Base Address (B)**
Here it is 1100

Memory space acquired by every element in the Array is called **Width (W)**
Here it is 4 bytes

Actual Address in the Memory	1100	1104	1108	1112	1116	1120
Elements	**15**	**7**	**11**	**44**	**93**	**20**
Address with respect to the Array (Subscript)	0	1	2	3	4	5

Lower Limit/Bound of Subscript **(LB)**

Calculating the address of any element In the 1-D array:

Address of A[I] = B + W * (I – LB)

I = Subset of element whose address to be found,

B = Base address,

W = Storage size of one element store in any array(in byte),

LB = Lower Limit/Lower Bound of subscript(If not specified assume zero).

Array of an element of an array say "A[I]" is calculated using the following formula:

Address of A [I] = B + W * (I – LB)

Where,
B = Base address
W = Storage Size of one element stored in the array (in byte)
I = Subscript of element whose address is to be found
LB = Lower limit / Lower Bound of subscript, if not specified assume 0 (zero)

Example:
Given the base address of an array **B[1300.....1900]** as 1020 and size of each element is 2 bytes in the memory. Find the address of **B[1700]**.
Solution:
The given values are: B = 1020, LB = 1300, W = 2, I = 1700

Address of A [I] = B + W * (I – LB)
= 1020 + 2 * (1700 – 1300)
= 1020 + 2 * 400
= 1020 + 800
= 1820 **[Ans]**

Types of Array

Based on dimension there are 3 types of arrays

1.One-dimensional arrays

2. Two-dimensional arrays

3. Multidimensional array

1.One-dimensional arrays

- A one-dimensional array is a group of elements having same data type and same name.

Or

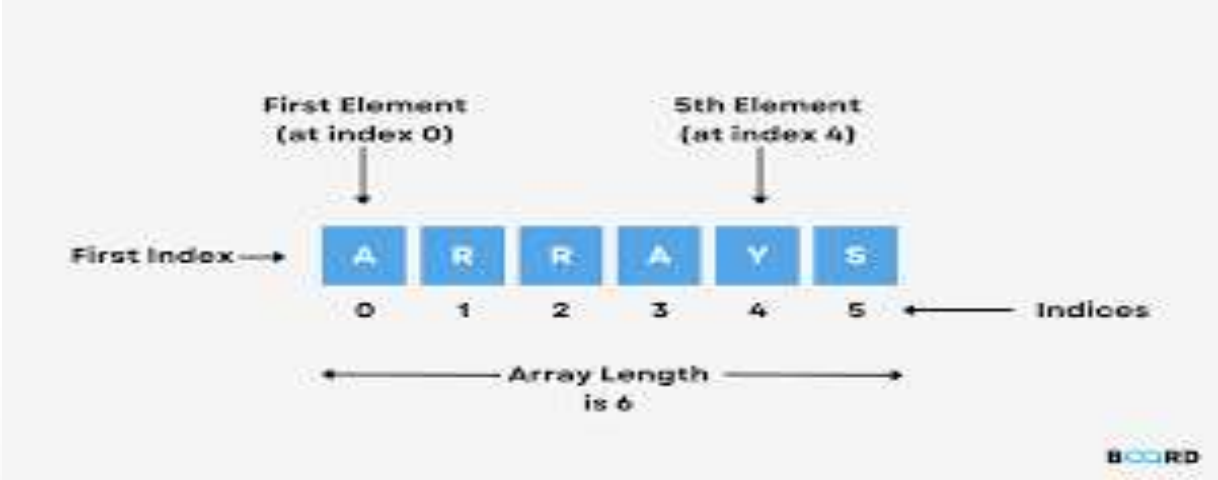
- A One-Dimensional Array is the simplest form of an Array in which the elements are stored linearly and can be accessed individually by specifying the index value of each element stored in the array.

- Use single subscript

- single row and multiple columns

Syntax: datatype Arrayname[size];

Example: int A[10];



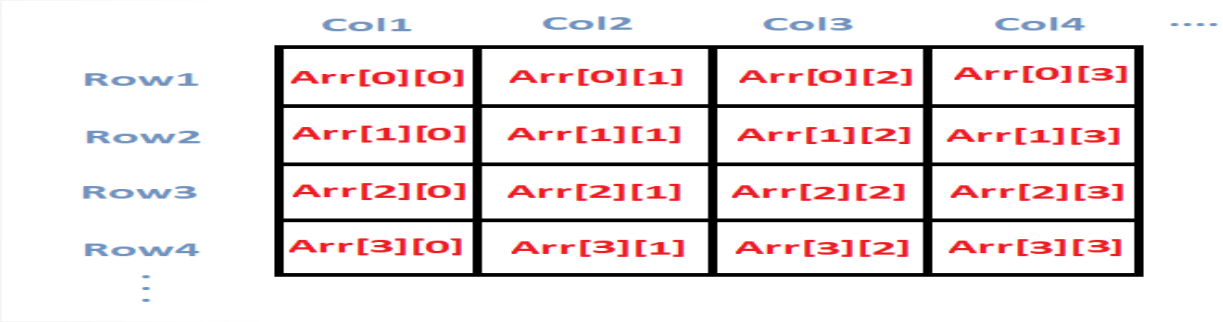
Program to take 5 values from the user and store them in an array and Print the elements stored in the array

```
#include <stdio.h>
int main()
{
int  arr[5];

printf("Enter 5 integers: \n");

for(int i=0; i<5; ++i)
            {
            scanf("%d", &values[i]);
             }
printf("\nDisplaying integers: ");

for(int i = 0; i < 5; ++i)
            {
            printf("%d\t", arr[i]);
            }
return 0;

}
```

*** Write a C program to append 2 arrays?

Two-dimensional array

- A **two-dimensional array** in C is the simplest form of a multi-dimensional array.
- It can be visualized as an array of arrays.
- 2D array is a Collection of 1D arrays



- A two-dimensional array is also called a **matrix**. (a table of rows and columns).

For example matrix of size 3 x 4 should display like this:

2	11	7	12
5	2	9	15
8	3	10	42

- It can be of any type like integer, character, float, etc. depending on the initialization.

Declaration of two dimensional Array in C

The syntax to declare the 2D array is given below.

```
data_type   array_name[rows][columns];
int matrix[4][3];
```

Here, 4 is the number of rows, and 3 is the number of columns.

Initialization of 2D Array in C

```
int arr[4][2] = {1234, 56, 1212, 33, 1434, 80, 1312, 78};
```

Or

```
int arr[4][2] =         { {1234, 56},
                          {1212, 33},
                          {1434, 80},
                          {1312, 78}
                        } ;
```

……..

```
int matrix[2][3] = { {1, 4, 2}, {3, 6, 8} };

printf("%d", matrix[0][2]);                    // Outputs 2
```

- Use two subscripts, one for the row and one for the column.
- A nested for loop is used to input elements in a two dimensional array.
- Nested loop is used to print the rows and columns in row and column order

```
for(i=0; i<2; i++)
        {
        for(j=0;j<3;j++) ----------Nested loop
                {
                 printf("Enter value for disp[%d][%d]:", i, j);
                scanf("%d", &disp[i][j]);
                }
        }
```

Two dimensional array in c provide a structured form to handle data in a matrix-like format, useful for various applications such as

- game boards,
- spreadsheets,
- image processing.

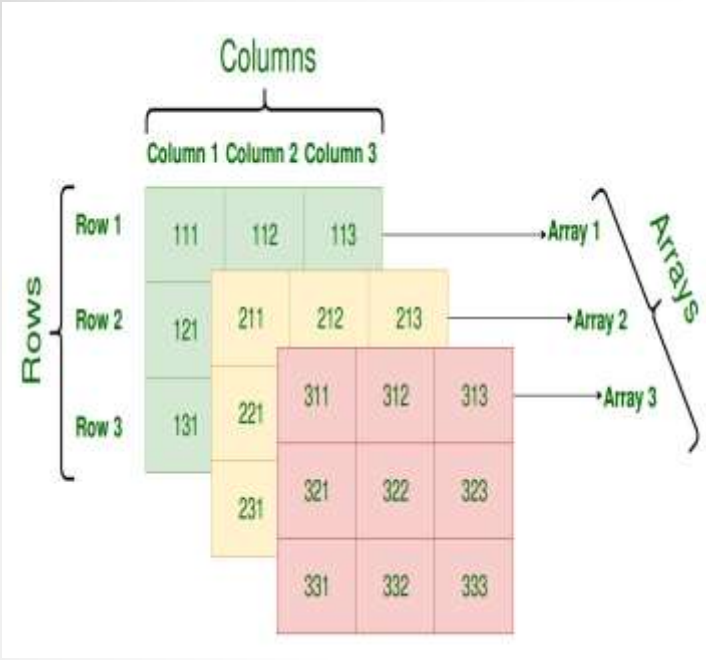
Write a C Program to read and print Matrix elements.

```c
#include <stdio.h>

int main()
{
    int i, j, m, n;
    int matrix[10][20];

    printf("Enter number of rows : ");
    scanf("%d", &m);
    printf("Enter number of columns : ");
    scanf("%d", &n);

    /* Input data in matrix */
    for (i = 0; i < m; i++)
    {
        for (j = 0; j < n; j++)
        {
            printf("Enter data in [%d][%d]: ", i, j);
            scanf("%d", &matrix[i][j]);
        }
    }
```

```
    /* Display the matrix */
    for (i = 0; i < m; i++)
    {
        for (j = 0; j < n; j++)
        {
            printf("%d\t", matrix[i][j]);
        }
        printf("\n");
    }

    return 0;
}
```

3.Multi-dimensional array.

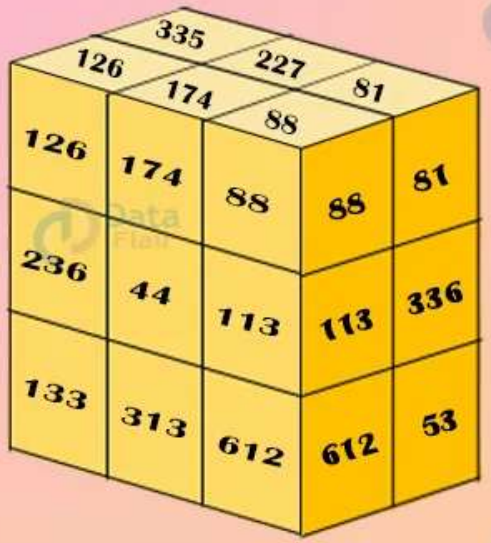
- A multi-dimensional array can be termed as an array of arrays
- The most commonly used forms of the multidimensional array are:
    - Two Dimensional Array
    - Three Dimensional Array
- A **two-dimensional array** or **2D array** in C is the simplest form of the multidimensional array.
- We can visualize a two-dimensional array as an array of one-dimensional arrays



2D Arrays

	Column 1	Column 2	Column 3	Column 4
Row 1	Matrix[0][0]	Matrix[0][1]	Matrix[0][2]	Matrix[0][3]
Row 2	Matrix[1][0]	Matrix[1][1]	Matrix[1][2]	Matrix[1][3]
Row 3	Matrix[2][0]	Matrix[2][1]	Matrix[2][2]	Matrix[2][3]

3D Arrays



Advantages of Arrays

- Arrays represent multiple data items of the same type using a single name.
- In arrays, the elements can be accessed randomly by using the index number.
- Arrays allocate memory in contiguous memory locations for all its elements.

Hence there is no chance of extra memory being allocated in the case of arrays. This avoids memory overflow or shortage of memory in arrays.

Applications of Arrays

- Array stores data elements of the same data type.
- Maintains multiple variable names using a single name. Arrays help to maintain large data under a single variable name. This avoids the confusion of using multiple variables.
- Arrays can be used for sorting data elements. Different sorting techniques like the Bubble sort, Insertion sort, Selection sort, etc
- Use arrays to store and sort elements easily.
- Arrays can be used for performing matrix operations. Many databases, small and large, consist of one-dimensional and two-dimensional arrays whose elements are records.
- Arrays can be used for CPU scheduling.
- Lastly, arrays are also used to implement other data structures like Stacks, Queues, Heaps, Hash tables, etc.

Disadvantages of Arrays

- The number of elements to be stored in an array should be known in advance.
- An array is a static structure (which means the array is of fixed size). Once declared the size of the array cannot be modified. The memory which is allocated to it cannot be increased or decreased.
- Insertion and deletion are quite difficult in an array as the elements are stored in consecutive memory locations and the shifting operation is costly.
- Allocating more memory than the requirement leads to a wastage of memory space and less allocation of memory also leads to a problem.