Points to remember :

- An array is a collection of similar types of data placed in contiguous memory locations
- An array can also be taken as multiple homogeneous type of data referred by one name
- Array can be 1 dimensional (1-D), 2-D,… N-D
- 1-D arrays are also known as List, Vector, Tuple, Set etc.
- 2-D arrays are also known as Table, Matrix, Set/collection of 1-D arrays etc.
- N-D arrays are also know as set/collection of (N-1)-D arrays
- Arrays are finite/static, means size is mandatory in definition/declaration. In C/C++ the size must be a constant/literal, in java the size can also be a user input. In both the cases, array is static in nature because this size cannot be altered in run time
- Data stored in array can be accessed randomly
- Arrays in C/C++ are only collection/data structures ie. a technology, but in java/c#/.. it is an object
- Arrays sometimes called subscripted type; because we use [] to have array, which is known as subscript operator
- Each items in the collection is called an array element, each array element has an index
- Each index (except VB) must be a +ve integer in range 0 to n-1, 0 is called Lower Bound, n-1 is Upper Bound. In VB lower bound can be 0 or 1
- In C, although array index is bounded between 0 to n-1, but in some compiler you can use index > n-1
- Each array element has an unique address (memory address)
- Main advantage of array –
    - We can use looping on it
    - Single name
    - Consecutiveness
    - In arrays, the elements can be accessed randomly by using the index number
- Main limitations of array –
    - It is static in nature, size could not be increased/decreased in run time hence wastage/shortage of memory/space may occur
    - Due to consecutiveness large sized array may not be possible although having enuf memory
    - Insertion in array is complex
    - Deletion is not possible, over write is possible
- Irrespective of all the limitations or array, it is very useful and is also used vividly in coding
- Array can be used as internal data structure of some other data structures like – stack, queue, binary tree etc.
- The name of the array in C is a const type pointer, contains the address of 0th element ie., &a[0], known as ‘Base Address’
- int a[5]; should be read as “a is an <array of 5 integers>” rather saying <integer array>
- char s[10]; here ‘s’ is an array of 10 characters. It can also be used to represent a string in C if a null character (\0) ascii value of which is 0 (zero) is put into it at end
- C-string : it is an array of character/a set of characters that ends with null character. If we remove the null char from the end, it will become a simple array of chars but not a string; the null character acts like the sentinel

Array declaration

C & C++)
```
int  arr[5];         //1-D
int  arr[3][5];      //2-D
int  arr[2][3][5]; //3-D
int  arr[2][3][5]....[n'th dim size];//N-D
```

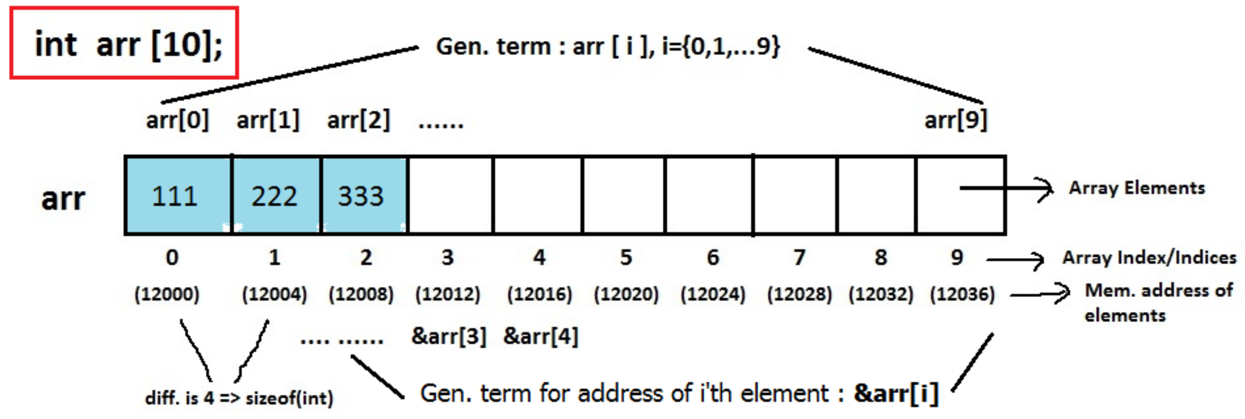
Java)
```
int  arr[]  =  new  int[5];
int  []arr  =  new  int[5];

int  arr[][] = new  int[3][5];         2-D
.....
```

Array Initialization

C, C++)
```
double balance[5] = {1000.0, 2.0, 3.4, 7.0, 50.0};
double balance[] = {1000.0, 2.0, 3.4, 7.0, 50.0};
int num[5] =  { };              // num = [0, 0, 0, 0, 0]
int num[5] =  { 0 };            // num = [0, 0, 0, 0, 0]
char name1[ ] = { 'J', 'a', 'n' };
char name2[ ] = { "Jan" };
char name3[4] = "Jan";
int month_days[2][12] =
{
   31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31,
   31, 29, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31
};
int month_days[2][12] =
{
   {31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31},
   {31, 29, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31}
};
int grid[3] [4] = { [2][0] = 8, [2][1] = 6, [2][2] = 4, [2][3] = 1 };
int grid[] [4] = { [2][0] = 8, [2][1] = 6, [2][2] = 4, [2][3] = 1 };
```

Array visualization [1-D]



By, Sourish Dutta. *Ref. : Personal & Web*
WhatsApp group link : https://chat.whatsapp.com/G3JFxzxLvMf0mOr2cVtd9W

Traversing 1-D array LB=Lower Bound, UB=Upper Bound

Forward :

```
for( i=LB ; i<=UB; i++)
{
        // perform action on arr[i]
}
```

Reverse :

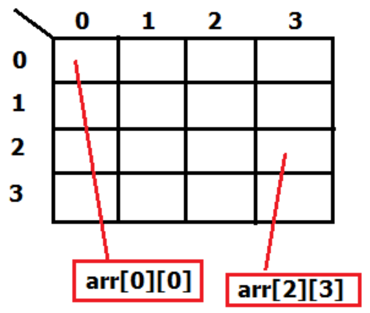
```
for( i=UB ; i>=LB; i--)
{
        // perform action on arr[i]
}
```

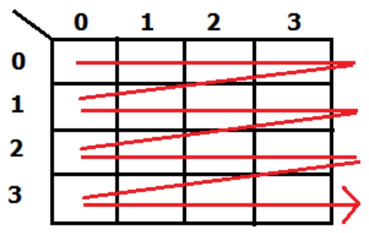
Random :

```
for( i= j ; i<=k; i++)        where, LB<=j<=i<=k<=UB
{
        // perform action on arr[i]
}
```

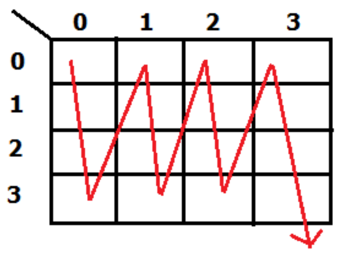
2-D Array representation

int arr[3][3];





Row-Major Representation



Colom-Major Representation

Traversing 2-D array : int arr[Row][Col]

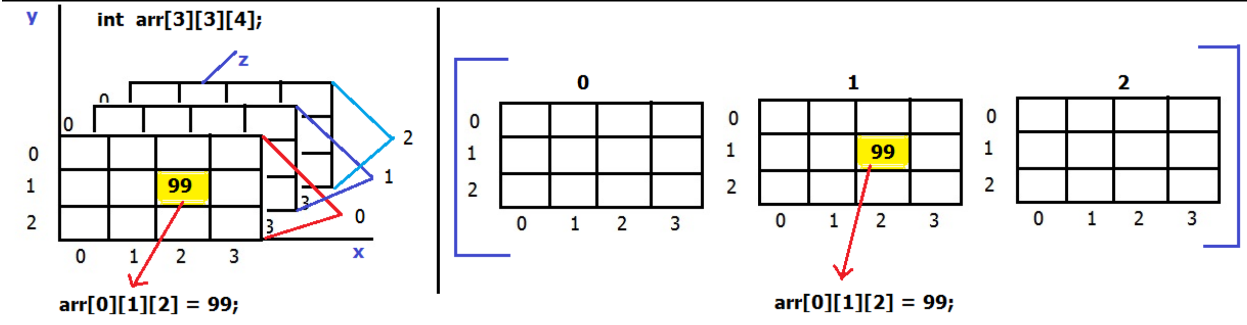
Row Major :

```
for( i=0;  i<Row; i++)
{
        for(j=0; j<Col; j++)
        {
                //perform action on arr[i][j]
        }
}
```

Colum Major :

```
for(j=0; j<Col; j++)
{
        for(i=0;  i<Row; i++)
        {
                //perform action on arr[i][j]
        }
}
```

3-D Array representation



Array Programming Exercise

Vector Programming excercise

Write a C program to find sum of all array elements
Write a C program to find maximum and minimum element in an array
Write a C program to find second largest element in an array
Write a C program to count total number of even and odd elements in an array
Write a C program to copy all elements from an array to another array
Write a C program to insert an element in an array
Write a C program to delete an element from an array at specified position
Write a C program to count frequency of each element in an array
Write a C program to print all unique elements in the array.
Write a C program to count total number of duplicate elements in an array.
Write a C program to delete all duplicate elements from an array.
Write a C program to merge two array to third array
Write a C program to find reverse of an array
Write a C program to put even and odd elements of array in two separate array
Write a C program to search an element in an array (linear,binary)
Write a C program to sort array elements in ascending or descending order (selection, bubble)
Write a C program to sort even and odd elements of array separately
Write a C program to left rotate an array
Write a C program to right rotate an array
Testing whether the contents of one array are a subset of the contents of another

Matrix programming exercises

Write a C program to add two matrices.
Write a C program to subtract two matrices.
Write a C program to multiply two matrices.
Write a C program to check whether two matrices are equal or not.
Write a C program to find sum of main diagonal elements of a matrix.
Write a C program to find sum of minor diagonal elements of a matrix.
Write a C program to find sum of each row and column of a matrix.
Write a C program to interchange diagonals of a matrix.
Write a C program to find upper triangular matrix.
Write a C program to find lower triangular matrix.

Write a C program to find sum of upper triangular matrix.
Write a C program to find sum of lower triangular matrix.
Write a C program to find transpose of a matrix.
Write a C program to find determinant of a matrix.
Write a C program to check Identity matrix.
Write a C program to check Sparse matrix.
Write a C program to check Symmetric matrix.