- C Arrays :-

* An array in C is a fixed-size collection of similar data items stored in contiguous memory locations.
* It can be used to store the collection of primitive data types such as int, char, float, etc and also derived and user-defined data types such as pointers, structures etc.

→ C Array declaration :-

* Syntax of Array declaration :-

```
data_type array_name [size];
```

(or).

```
data_type array_name[size1] [size2] ... [sizeN];
```

where N is the number of dimensions.

* The C arrays are static in nature i.e. they are allocated memory at the compile time.

```
#include<stdio.h>
int main()
{
// declaring array of integers.
```

```
    int arr_int [5];
    //declaring array of characters.
    char arr_char [5];
    return 0;
}
```

→ C array Initialization :-

* When the array is declared or allocated memory, the elements of the array contain some garbage value

1. Array initialization with declaration :-

* An initializer list is the list of values enclosed within braces { } separated by commas.

```
data-type array-name [size] = {value1, value2, ---,
                               valueN};
```

2. Array initialization with declaration without size :-

* The size of the array in these cases is equal to the number of elements present in the initializer list as the compiler can automatically deduce the size of the array.

```
data-type array-name [] = {1, 2, 3, 4, 5};
```

3. Array initialization after Declaration :-
(using loop)

```
for(int i=0; i<N; i++)
{
  array_name[i] = valuei;
}
```

```
#include<stdio.h>
int main()
{
  int arr[5] = {10,20,30,40,50};
  int arr1[] = {1,2,3,4,5};
  float arr2[5];
  for (int i=0; i<5; i++)
  {
    arr2[i] = (float)i + 2.1;
  }
  return 0;
}
```

→ Access array elements :-

* array_name [index] ;
* indexing in the array always starting with 0, i.e. the first element is at index 0 and the last element is at N-1 where N is the number of elements in the array.

```
#include<stdio.h>
int main()
{
  int arr[5] = {15, 25, 35, 45, 55};
```

```
printf(" Element at arr[2] : %d \n", arr[2]);
printf("Element at arr[4]: %d \n", arr[4]);
printf(" Element at arr[0]: %d ", arr[0]);
return 0;
}
```

Output:-

Element at arr[2] : 35
Element at arr[4] : 55
Element at arr[0] : 15

→ Update ~~arry~~ array element:-

* array_name[i] = new_value;

→ C Array traversal:-

* For c, array traversal, we use loops to iterate through each element of the array.

```
for (int i=0; i<N; i++)
{
  array_name[i];
}
```

*

```
//C Program to demonstrate the use of array.
#include<stdio.h>
int main()
{
  //array declaration and initialization
```

```
int arr[5] = {10, 20, 30, 40, 50};
// modifying element at index 2
arr[2] = 100;
// traversing array using for loop.
printf("Elements in Array :");
for (int i = 0; i<5; i++)
{
  printf ("%d", arr[i]);
}
return 0;
}
```

Output:-

Elements in Array : 10 20 ~~30~~ 100 40 50.

→ Types of Array in C:-

1). One dimensional Array in C:-

* 1D arrays in C are those have only one-dimension
* Syntax:-

```
array_name [size];
```

(*). // C program to illustrate the use of 1D array.

```
#include <stdio.h>
int main()
{
    // 1d array declaration
    int arr[5];
```

```
//1d Array initialization using for loop.
for (int i=0; i<5 ; i++)
{
  arr [i] = i*i  -2*i +1;
}
printf(" Elements of Array :");
//printing 1d array by traversing using for loop
for (int i=0 ; i<5 ; i++)
{
  printf(" %d" , arr [i]);
}
return 0;
}
```

Output :-

Element of Array : 1 0 1 4 9.

→ Arrays of characters (strings) :-

* A sequence of characters in the form of an array of characters terminated by a NULL character. These are called strings in C language

```
#include<stdio.h>
int main()
{
  //creating array of character
  char  arr[6] = {'G', 'e', 'e', 'k', 's', '\0'};

  //printing string.
```

```
int i=0;
while (arr[i]) {
    printf("%c", arr[i++]);
}
return 0;
}
```

Output :-

Geeks.

2. Multidimensional Array in C :-

A. Two- Dimensional Array in C :-

* They can be visualized in the form of rows and columns organized in a two-dimentional plane.

→ Syntax of 2D Array in C :-

✱. `array_name [size1][size2];`

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

```
#include<stdio.h>
int main()
{
  // declaring and initializing 2d array.
  int arr[2][3] = {10, 20, 30, 40, 50, 60};

  printf("2D Array :\n");

  // printing 2d Array
```

```
for (int i=0;  i<2 ; i++){
    for (int j=0; j<3 ; j++){
        printf ("\t%d ",  arr[i][j]);
    }
    printf ("\n");
}
return 0;
}
```

Output:-

```
2D Array:
10  20  30
40  50  60
```

By. Three-Dimensional Array in C:-

* array_name [size1] [size2] [size3];

```
#include<stdio.h>
int main()
{
  //3D array declaration
  int arr[2][2][2] = {10, 20, 30, 40, 50, 60};
  //printing elements.
  for (int i=0 ; i<2 ; i++){
    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;
}
```

Output:-

```
10  20
30  40
50  60
0   0
```

→ Relationship between Arrays and Pointers:-

```
#include <stdio.h>
int main()
{
    int arr[5] = {10,20,30,40,50};
    int *ptr = &arr[0];
    /*comparing address of first element and address
     stored */.
     // inside array name.
    printf("Address stored in Array name: %p\n
      Address of "." 1st Array element : %p\n",
      arr , &arr[0]);

     //printing array elements using pointers
      printf("Array elements using pointer :");
      for (int i=0; i<5; i++){
          printf(" %d", *ptr++);
      }
      return 0;
}
```

Output :-

Address stored in Array name: 0x7ffcab67d8e0

Address of 1st Array Element: 0x7ffcab67d8e0

Array elements using pointer : 10 20 30 40 50.

→ Passing an Array to a function in C :-

```
#include<stdio.h>
void printArray (int arr[])
{
  printf ("size of Array in functions : %d \n",
          sizeof (arr));
  printf ("Array elements : ");
  for (int i=0; i<5; i++) {
        printf ("%d", arr[i]);
  }
}
int main()
{
  int arr[5] = {10, 20, 30, 40, 50};
  printf ("size of Array in main() : %d \n",
      sizeof (arr));
  printArray (arr);
  return 0;
}
```

Output :-

size of Array in main() : 20

Size of Array in function : 8

Array Elements : 10 20 30 40 50

→ Return an Array from a function in C:-

```
#include<stdio.h>
//function
int *func()
{
  static int arr[5] = {1,2,3,4,5};
   return arr;
 }
int main()
{
  int *ptr = func();
  printf("Array Elements: ");
  for (int i=0; i<5 ; i++) {
       printf("%d", *ptr++);
   }
   return 0;
}
```

Output:-

Array elements : 1 2 3 4 5.

→ Examples of array in C:-

```
#include<stdio.h>
// C program to perform input and output on array
int main()
{
  // declaring an integer arry
int arr[5];
```

```
for (int i=0 ; i<5; i++){
    scanf ("%d", &arr[i]);
}

//printing array elements.
printf ("Array elements : ");
for (int i=0; i<5 ; i++){
    printf ("%d", arr[i]);
}
return 0;
}
```

input :- 5 7 9 1 4.

output :-

Array Elements : 5 7 9 1 4