Array

Introduction to array

An array is a collection of similar type of element storable together in a single contagious memory location. The important thing to remember is the all the data elements stored in an array should be of similar data type. Array is a reference type which uses the Heap Area of memory to store the same type of data items and uses continues memory locations to store every data item of array.

All data items in array are stored in a certain order in memory and for each element to identify uniquely, a Zero Based Index Number is specified with it. Also Array is a fixed Data Structure. So once the size of the Array is defined, then the size of array can not be modify.

- Whatever the data type of the array can be.
- The variables of array are stored on their near memory location.
- The initialization of array starts with '0'.

Creating C array:-

Whenever you create an array, you define the name of the array and how many values it is going to store in it. As if you want to store 4 numbers, you can create an array for that. The general structure of creating arrays in C is given below.

```
data_type array-name[size];
```

By size, you define how many values you want to store.

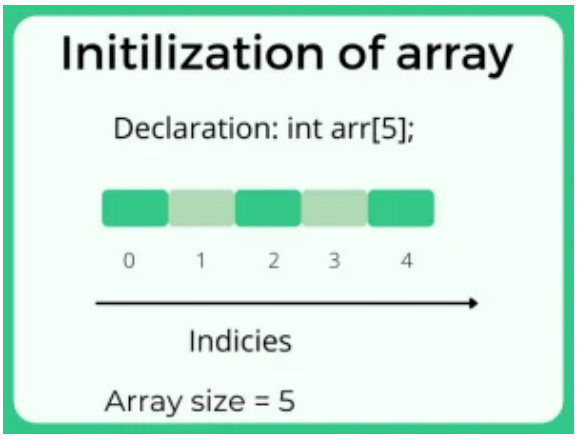
```
int sum[5]
```

In this you can store any 5 integer values in it.

Initializing C array

We can initialize each element of array by using index. Here's the example:-

```
sum[0]=30;  // initialization of array
sum[1]=50;
sum[2]=24;
sum[3]=60;
sum[4]=78;
```



Types of arrays

There are two types of array :

- Single Dimensional Array
- Multiple Dimensional Array

Single Dimensional Array

In an array we don't need to declare each and every element we only need to declare the array, the type of data in the array and the size of the array. Let's see an example for declaring and saving element in the array and finally printing the elements saved.

```
#include<stdio.h>
int main()
{
  int arr[5];       //declaring the array
  int i,a;
  for(i=0;i<5;i++)    //loop to take input in the array from user
  {
     scanf("%d",&arr[i]);
  }
  for(i=0;i<5;i++)    // loop to print the elements of the array
  {
     printf("%d\n",arr[i]);
  }
```

```
    return 0;
}
```

Note : An array of size 5 can store 5 element but the 1st element is stored in the zeroth index value and then 1st, 2nd, 3rd, and last at 4th. And this goes for every array.

Initializing one dimensional array:-

There are a bunch of way to store data in an array, like

```
int marks[5];
arr[0]=19;
arr[1]=11;
arr[2]=14;
arr[3]=8;
arr[4]=45;
```

OR

```
int marks[5]={19,11,14,8,45};
```

Both of these are the same array with the same elements.This type of liner array is called 1-D array

Multi Dimensional Array

A multi-dimensional array or a 2-D array is a similar to 1-D array but the element stored in a 2-D array are stored in a matrix format rather than a linear manner. The elements are stored in rows and columns like a matrix.

The simplest form of multi-dimensional array is 2-D array.

Initializing of 2-D Array:-

Let's see an example to understand.

```
int arr[3][4]={1,2,3,4,2,3,4,5,3,4,5,6}; or        //this array has 3 rows and 4 c
olumns

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

Here's the array will be like:-

```
1 2 3 4
2 3 4 5
3 4 5 6
```

Example of Multi-dimensional array:-

```
#include<stdio.h>
int main()
{
   int arr[3][4]={{1,2,3,4},{2,3,4,5},{3,4,5,6}};
   int i,j;
   for(i=0;i<3;i++)
   {
       for(j=0;j<4;j++)
       {
           printf("[%d][%d]=%d\n",i,j,arr[i][j]);
       }
   }
   return 0;
}
```

Some Important :

Update Array Element

We can update the value of array elements at the given index i in a similar way to accessing an element by using the array **square brackets []** and **assignment operator (=)**.

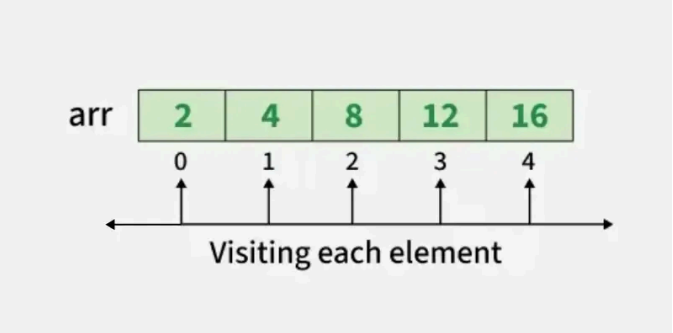
`array_name[i] = new_value;`

```
#include <stdio.h>
int main() {
    int arr[5] = {2, 4, 8, 12, 16};

    // Update the first value
    // of the array
    arr[0] = 1;
    printf("%d", arr[0]);
    return 0;
}
```

C Array Traversal

Array Traversal is the process in which we visit every element of the array in a specific order. For C array traversal, we use loops to iterate through each element of the array.




```
#include <stdio.h>
int main() {
    int arr[5] = {2, 4, 8, 12, 16};

    // Print each element of
    // array using loop
    printf("Printing Array Elements\n");
    for(int i = 0; i < 5; i++){
```

```
        printf("%d ", arr[i]);
    }
    printf("\n");
    // Printing array element in reverse
    printf("Printing Array Elements in Reverse\n");
    for(int i = 4; i>=0; i--){
        printf("%d ", arr[i]);
    }
    return 0;
}
```

Size of Array

The size of the array refers to the number of elements that can be stored in the array. The array does not contain the information about its size but we can extract the size using sizeof() operator.

```
#include <stdio.h>
int main() {
    int arr[5] = {2, 4, 8, 12, 16};

    // Size of the array
    int size = sizeof(arr)/sizeof(arr[0]);
    printf("%d", size);
    return 0;
}
```

The sizeof() operator returns the size in bytes. sizeof(arr) returns the total number of bytes of the array. In an array, each element is of type int, which is 4 bytes. Therefore, we can calculate the size of the array by dividing the total number of bytes by the byte size of one element.

Properties of C Arrays

C arrays have the following distinguishing properties:

1. Fixed Size Collection
2. Homogeneous Elements
3. Indexing in Array
4. Dimensions of Array
5. Contiguous Storage
6. Random Access
7. Array name relation with pointer
8. Bound Checking
9. Array Decay

Advantages of Array in C

The following are the main advantages of an array:

1. Random and fast access of elements using the array index.
2. Use of fewer lines of code as it creates a single array of multiple elements.
3. Traversal through the array becomes easy using a single loop.
4. Sorting becomes easy as it can be accomplished by writing fewer lines of code.

Disadvantages of Array in C

1. C Arrays are not dynamic they only allow a fixed number of elements to be entered which is decided at the time of declaration.
2. Insertion and deletion of elements can be costly since the elements are needed to be rearranged after insertion and deletion.