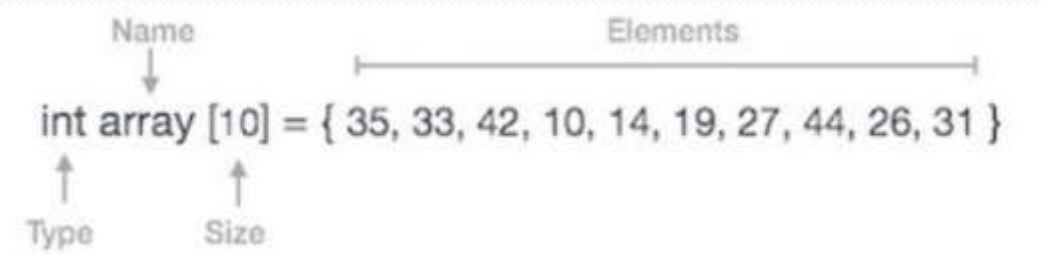
Array

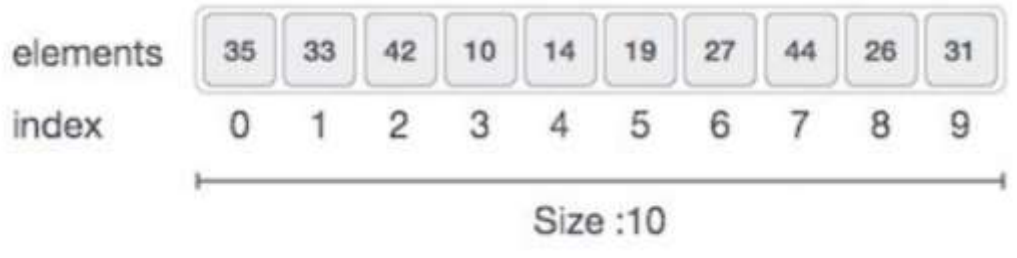
Introduction

- An array is a group of consective memory locations with same name and data type.
- Simple variable is a single memory location with unique name and a type. But an Array is collection of different adjacent memory locations. All these memory locations have one collective name and type.
- The memory locations in the array are known as elements of array. The total number of elements in the array is called length.
- The elements of array is accessed with reference to its position in array, that is called index or subscript.

Array Representation

Arrays can be declared in various ways in different languages. For illustration, let's take C array declaration.





As per the above illustration, following are the important points to be considered.

- Index starts with 0.
- Array length is 10 which means it can store 10 elements.
- Each element can be accessed via its index. For example, we can fetch an element at index 6 as 9.

Types of array

One dimensional Array :

A one-dimensional array is also called a single dimensional array where the elements will be accessed in sequential order. This type of array will be accessed by the subscript of either a column or row index.

Multi-dimensional Array :

When the number of dimensions specified is more than one, then it is called as a multi-dimensional array. Multidimensional arrays include *2D arrays* and *3D arrays.* A *two-dimensional* array will be accessed by using the subscript of row and column index. For traversing the two-dimensional array, the value of the rows and columns will be considered. In the two-dimensional array face [3] [4], the first index specifies the number of rows and the second index specifies the number of columns and the array can hold 12 elements (3 * 4).

Declaration of an array

We know that all the variables are declared before they are used in the program. Similarly, an array must be declared before it is used. During declaration, the size of the array has to be specified. The size used during declaration of the array informs the compiler to allocate and reserve the specified memory locations.

Syntax:- data_type array_name[n];

where, n is the number of data items (or) index(or) dimension.

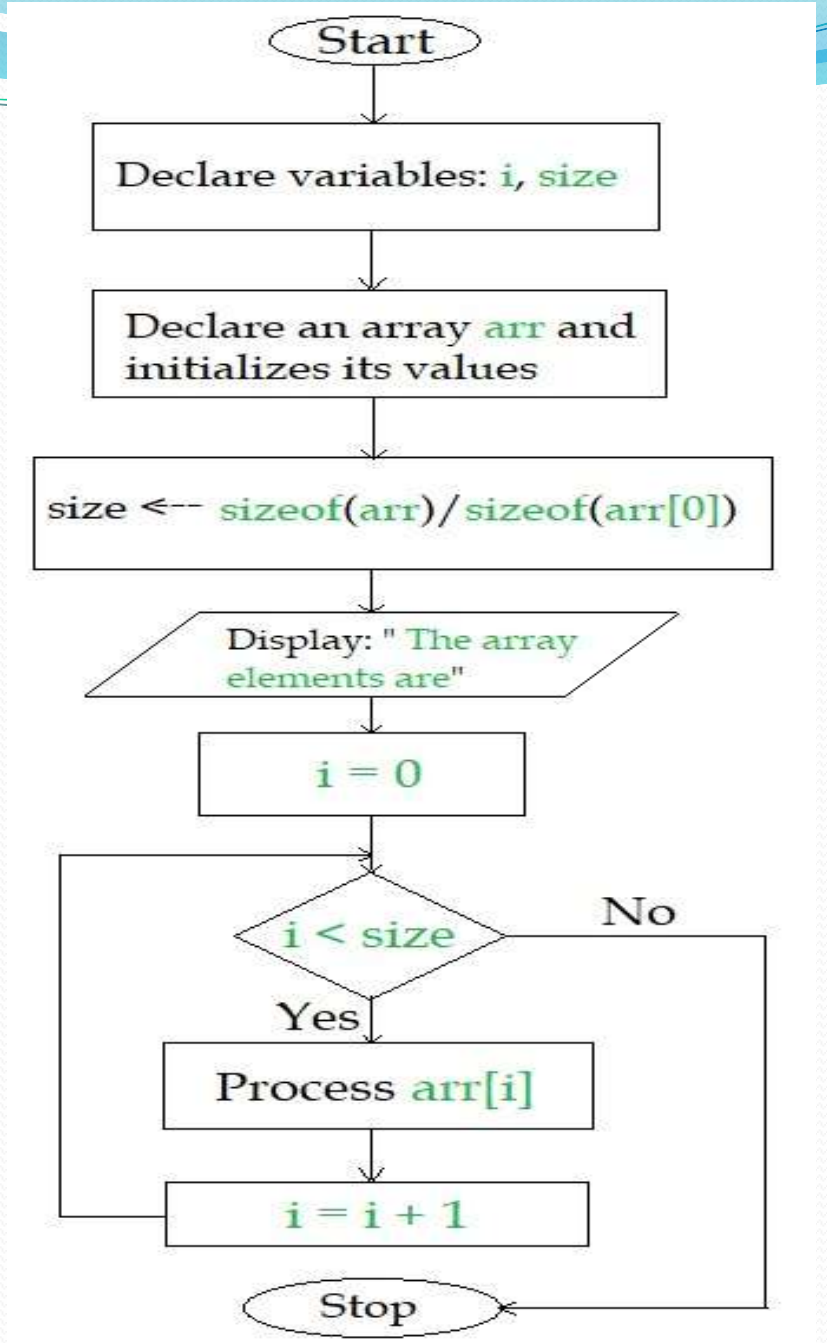
0 to (n-1) is the range of array.

Ex: int a[5];

float x[10];

Traversing Operation

This operation is used to visit all elements in an array



```
#include <iostream>
using namespace std;
int main()
{
    int i, size;
    int arr[]={53, 99, -11, 5, 102};    //declaring
and initializing array
    size=sizeof(arr)/sizeof(arr[0]);
    cout << “The array elements are: “;
    for(i=0;i<size;i++)
        cout << “\n” << “arr[“ << i << “]= “ <<
arr[i];
    return 0;
}
```

```
The array elements are:
arr[0]= 53
arr[1]= 99
arr[2]= -11
arr[3]= 5
arr[4]= 102
```

Sorting

Sorting is a concept in which the elements of an array are rearranged in a logical order. This order can be from lowest to highest or highest to lowest.

```
#include <iostream>
using namespace std;
int main() {
   // Write C++ code here
  // cout << "Hello world!";
  int n,i,j;
  cin>>n;
  int arr[n];

  for(i=0;i<n;i++){
     cin>>arr[i];
  }
```

```
  for(i=1;i<n-1;i++){
     for(j=i+1;j<n;j++){
        if(arr[j]<arr[i]){
           int temp= arr[j];
           arr[j]=arr[i];
           arr[i]=temp;
        }
     }
  }for(i=0;i<n;i++){
     cout<<arr[i]<<" ";
  }cout<<endl;

   return 0;
}
```

Output

```
3 6 8 9 5
3 5 6 8 9
```

Searching

This operation is applied to search an element of interest in array.

```
#include <iostream>
using namespace std;

int search(int arr[], int n, int x)
{
    int i;
    for (i = 0; i < n; i++)
        if (arr[i] == x)
            return i;
    return -1;
}

// Driver code
int main(void)
{
    int arr[] = { 2, 3, 4, 10, 40 };
    int x = 10;
    int n = sizeof(arr) / sizeof(arr[0]);

    // Function call
    int result = search(arr, n, x);
    (result == -1)
        ? cout << "Element is not present
in array"
        : cout << "Element is present at
index " << result;
return 0;
}
```

Output: 3

Insertion

This operation is used to insert an element into an array.

```
// C Program to Insert an element
// at a specific position in an Array

#include <stdio.h>

int main()
{
 int arr[100] = { 0 };
 int i, x, pos, n = 10;

 // initial array of size 10
 for (i = 0; i < 10; i++)
  arr[i] = i + 1;

 // print the original array
 for (i = 0; i < n; i++)
  printf("%d ", arr[i]);
 printf("\n");

 // element to be inserted
 x = 50;

 // position at which element
 // is to be inserted
 pos = 5;

 // increase the size by 1
 n++;

 // shift elements forward
 for (i = n-1; i >= pos; i--)
  arr[i] = arr[i - 1];

 // insert x at pos
 arr[pos - 1] = x;

 // print the updated array
 for (i = 0; i < n; i++)
  printf("%d ", arr[i]);
 printf("\n");

 return 0;
}
```

output-1 2 3 4 5 6 7 8 9 10
1 2 3 4 50 5 6 7 8 9 10

Deletion

This operation is used to delete a particular element from an array.

```
// C++ program to remove a given
element from an array
#include<bits/stdc++.h>
using namespace std;

// This function removes an element
x from arr[] and
// returns new size after removal
(size is reduced only
// when x is present in arr[]
int deleteElement(int arr[], int n, int x)
{
// Search x in array
int i;
for (i=0; i<n; i++)
 if (arr[i] == x)
  break;
```

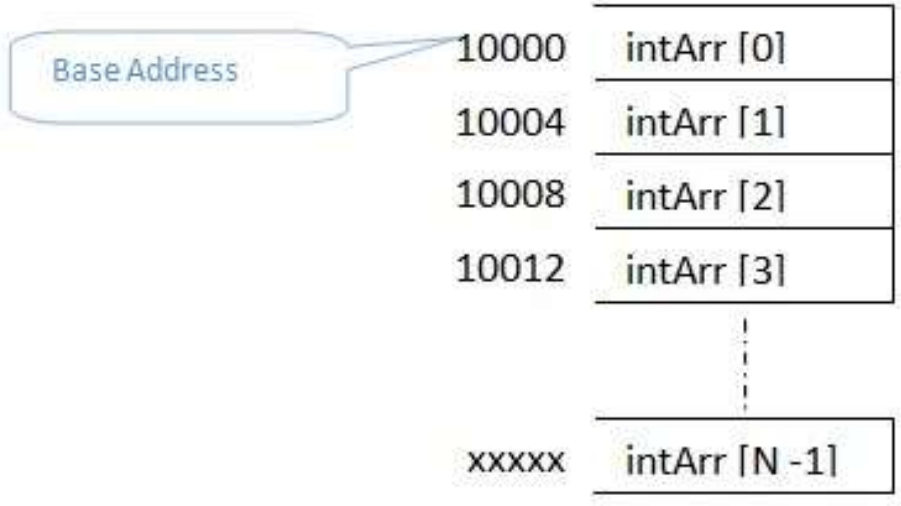
```
// If x found in array
if (i < n)
{
 // reduce size of array and move all
 // elements on space ahead
 n = n - 1;
 for (int j=i; j<n; j++)
  arr[j] = arr[j+1];
}

return n;
}

 /* Driver program to test above
 function */
 int main()
 {
  int arr[] = {11, 15, 6, 8, 9, 10};
  int n = sizeof(arr)/sizeof(arr[0]);
  int x = 6;
```

```
  // Delete x from arr[]
  n = deleteElement(arr, n, x);

  cout << "Modified array is \n";
  for (int i=0; i<n; i++)
  cout << arr[i] << " ";

  return 0;
 }
```

output-Modified array is
11 15 8 9 10

Memory Allocation of Array

Below diagram shows how memory is allocated to an integer array of N elements. Its base address – address of its first element is 10000. Since it is an integer array, each of its element will occupy 4 bytes of space. Hence first element occupies memory from 10000 to 10003. Second element of the array occupies immediate next memory address in the memory, i.e.; 10004 which requires another 4 bytes of space. Hence it occupies from 10004 to 10007. In this way all the N elements of the array occupies the memory space.



Applications

- Used in mathematical problems like matrices etc.
- They are used in the implementation of other data structures like linked lists etc.
- Database records are usually implemented as arrays.
- Used in lookup tables by computer.
- It effectively executes memory addressing logic wherein indices act as addresses to the one-dimensional array of memory.

Advantages

- Arrays store multiple data of similar types with the same name.
- It allows random access to elements.
- As the array is of fixed size and stored in contiguous memory locations there is no memory shortage or overflow.
- It is helpful to store any type of data with a fixed size.
- Since the elements in the array are stored at contiguous memory locations it is easy to iterate in this data structure and unit time is required to access an element if the index is known.