**\***[**\*\*\*Visit**](https://github.com/panchalbhavya2210)

**Q - 1: WHAT IS ARRAY?**

ANS -> An array is a collection of items of same data type stored at contiguous memory locations.

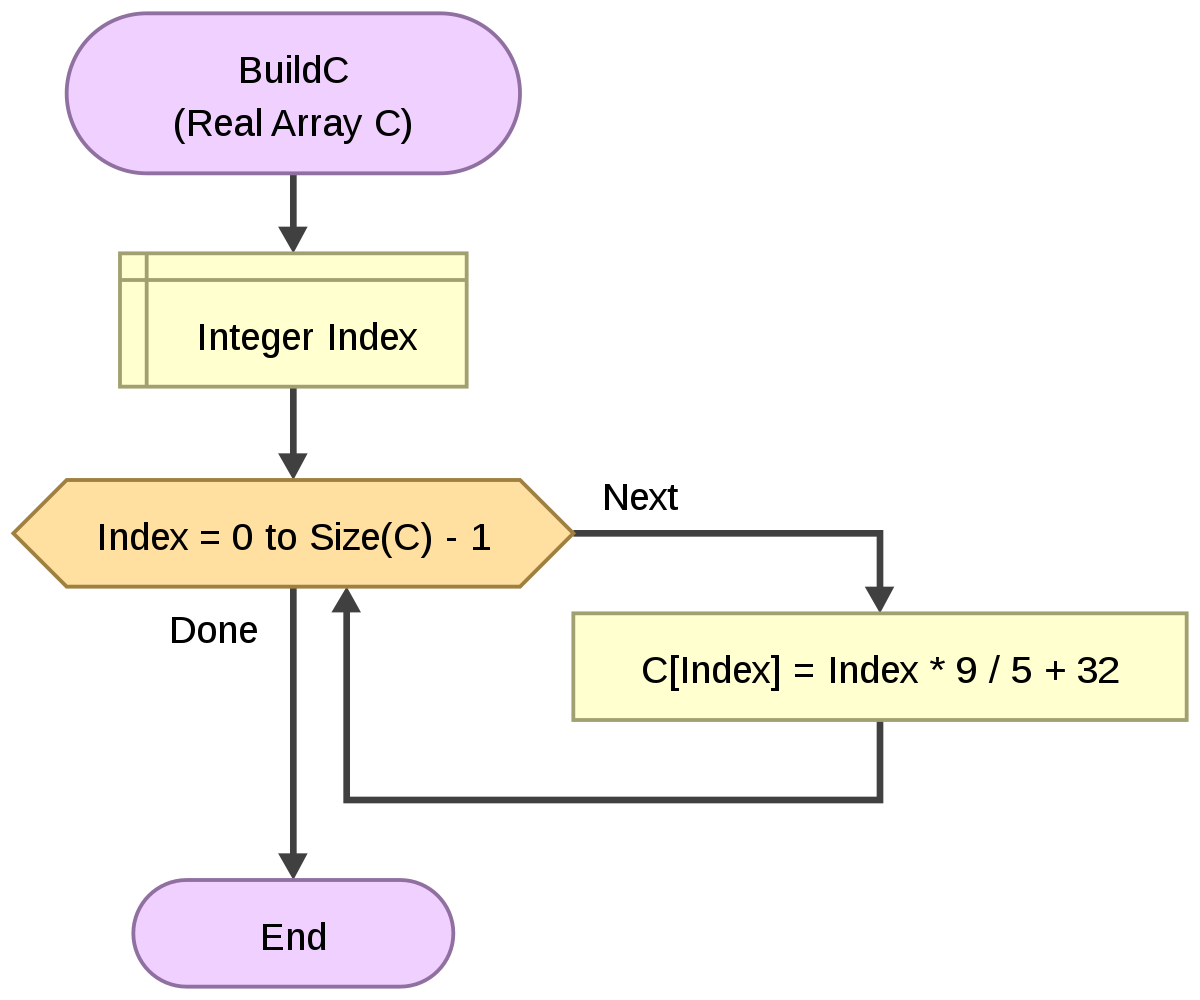
This makes it easier to calculate the position of each element by simply adding an offset to a base value, i.e., the memory location of the first element of the array (generally denoted by the name of the array). The base value is index 0 and the difference between the two indexes is the offset.

**Q - 2: EXPLAIN TYPE NAME AND SIZE IN ARRAY.**

ANS ->

<https://www.cs.fsu.edu/~myers/c++/notes/arrays.html>

**Q - 3: EXPLAIN THE FLOW DIAGRAM OF ARRAY.**

ANS -> 

**Q - 4: EXPLAIN LINEAR**

ANS -> **Linear Data Structure**

It is a type of data structure where the arrangement of the data follows a linear trend. The data elements are arranged linearly such that the element is directly linked to its previous and the next elements. As the elements are stored linearly, the structure supports single-level storage of data. And hence, traversal of the data is achieved through a single run only.

**Characteristics**

* It is a type of data structure where data is stored and managed in a linear sequence.
* Data elements in the sequence are linked to one after the other.
* Implementation of the linear structure of data in a computer’s memory is easy as the data is organized sequentially.
* Array, queue. Stack, linked list, etc. are examples of this type of structure.
* The data elements stored in the data structure have only one relationship.
* Traversal of the data elements can be carried out in a single run as the data elements are stored in a single level.
* There is poor utilization of the computer memory if a structure storing data linearly is implemented.
* With the increase in the size of the data structure, the time complexity of the structure increases.

**Q - 5: EXPLAIN STACK IN DETAIL.**

ANS -> A Stack is a linear data structure that follows the LIFO (Last-In-First-Out) principle. Stack has one end, whereas the Queue has two ends (front and rear). It contains only one pointer top pointer pointing to the topmost element of the stack. Whenever an element is added in the stack, it is added on the top of the stack, and the element can be deleted only from the stack. In other words, a stack can be defined as a container in which insertion and deletion can be done from the one end known as the top of the stack.

**Some key points related to stack**

* It is called as stack because it behaves like a real-world stack, piles of books, etc.
* A Stack is an abstract data type with a pre-defined capacity, which means that it can store the elements of a limited size.
* It is a data structure that follows some order to insert and delete the elements, and that order can be LIFO or FILO.

**Standard Stack Operations**

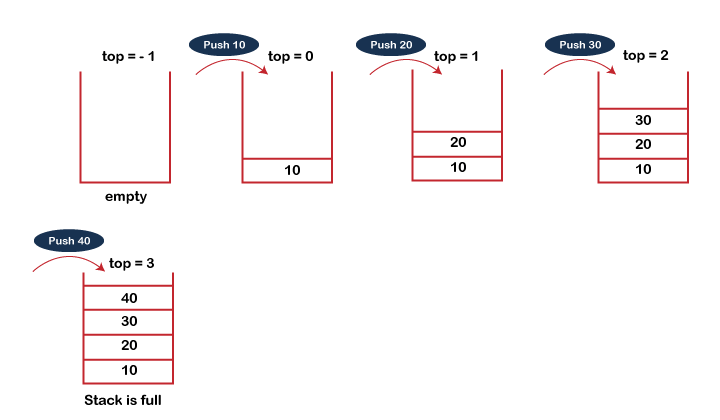
The following are some common operations implemented on the stack:

* push(): When we insert an element in a stack then the operation is known as a push. If the stack is full then the overflow condition occurs.
* pop(): When we delete an element from the stack, the operation is known as a pop. If the stack is empty means that no element exists in the stack, this state is known as an underflow state.
* isEmpty(): It determines whether the stack is empty or not.
* isFull(): It determines whether the stack is full or not.'
* peek(): It returns the element at the given position.
* count(): It returns the total number of elements available in a stack.
* change(): It changes the element at the given position.
* display(): It prints all the elements available in the stack.

**PUSH operation**

The steps involved in the PUSH operation is given below:

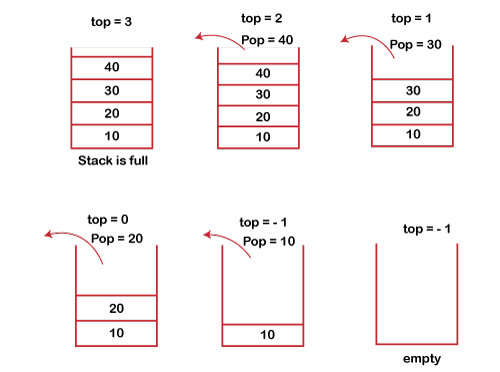
* Before inserting an element in a stack, we check whether the stack is full.
* If we try to insert the element in a stack, and the stack is full, then the overflow condition occurs.
* When we initialize a stack, we set the value of top as -1 to check that the stack is empty.
* When the new element is pushed in a stack, first, the value of the top gets incremented, i.e., top=top+1, and the element will be placed at the new position of the top.
* The elements will be inserted until we reach the max size of the stack.



**POP operation**

The steps involved in the POP operation is given below:

* Before deleting the element from the stack, we check whether the stack is empty.
* If we try to delete the element from the empty stack, then the underflow condition occurs.
* If the stack is not empty, we first access the element which is pointed by the top
* Once the pop operation is performed, the top is decremented by 1, i.e., top=top-1.



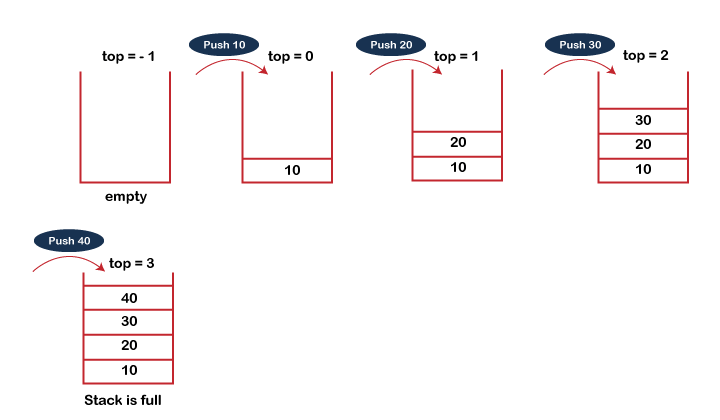
**Q - 6 : EXPLAIN PUSH AND POP WITH EXAMPLE.**

ANS ->

**PUSH operation**

The steps involved in the PUSH operation is given below:

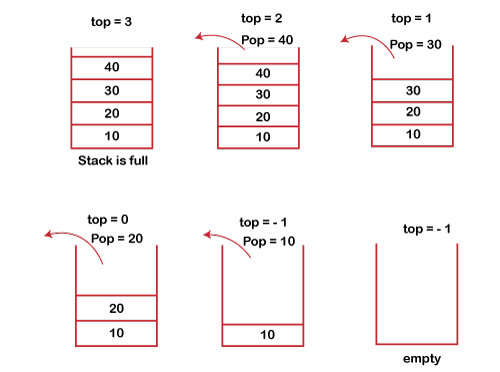
* Before inserting an element in a stack, we check whether the stack is full.
* If we try to insert the element in a stack, and the stack is full, then the overflow condition occurs.
* When we initialize a stack, we set the value of top as -1 to check that the stack is empty.
* When the new element is pushed in a stack, first, the value of the top gets incremented, i.e., top=top+1, and the element will be placed at the new position of the top.
* The elements will be inserted until we reach the max size of the stack.



**POP operation**

The steps involved in the POP operation is given below:

* Before deleting the element from the stack, we check whether the stack is empty.
* If we try to delete the element from the empty stack, then the underflow condition occurs.
* If the stack is not empty, we first access the element which is pointed by the top
* Once the pop operation is performed, the top is decremented by 1, i.e., top=top-1.



**Q - 8 : EXPLAIN WITH EXAMPLE.**

* **INFIX**
* **POSTFIX**
* **PREFIX**

ANS-><https://www.tutorialspoint.com/prefix-and-postfix-expressions-in-data-structure>

**Q - 9 : EXPLAIN CIRCULAR QUE.**

Ans -> <https://www.programiz.com/dsa/circular-queue>

**Q - 10 : DIFFERENCE BETWEEN ASCENDING QUEUE AND DESCENDING QUE**

ANS ->

**Ascending Priority Queue:** Element can be inserted arbitrarily but only smallest element can be removed. For example, suppose there is an array having elements 4, 2, 8 in the same order. So, while inserting the elements, the insertion will be in the same sequence but while deleting, the order will be 2, 4, 8.

**Descending priority Queue:** Element can be inserted arbitrarily but only the largest element can be removed first from the given Queue. For example, suppose there is an array having elements 4, 2, 8 in the same order. So, while inserting the elements, the insertion will be in the same sequence but while deleting, the order will be 8, 4, 2.