Operations on Arrays in Data Structures: Traversal, Insertion, Deletion and Searching

codewithharry.com/videos/data-structures-and-algorithms-in-hindi-9

In the last tutorial, we discussed implementing our abstract data type array and its set of values. In today's lesson, we'll explore how we can operate on these arrays. For example: traversing through the array, sorting the array, and many more. We'll start with the primary ones.

Operations on an Array:

While there are many operations that can be implemented and studied, we only need to be familiar with the primary ones at this point. An array supports the following operations:

- Traversal
- Insertion
- Deletion
- Search

Other operations include sorting ascending, sorting descending, etc. Let's follow up on these individually.

Traversal:

Visiting every element of an array once is known as **traversing** the array.

Why Traversal?

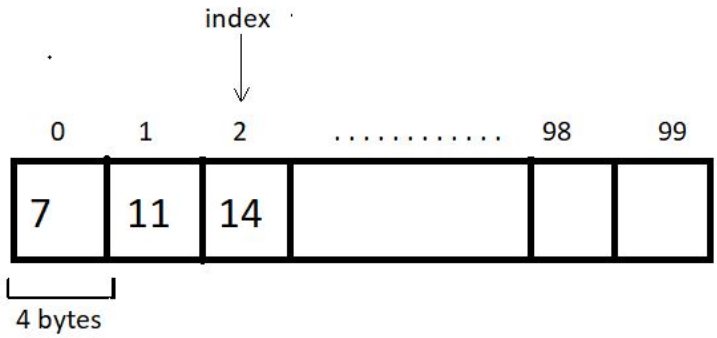
For use cases like:

- Storing all elements – Using scanf()
- Printing all elements – Using printf()
- Updating elements.

An array can easily be traversed using a *for* loop in C language.

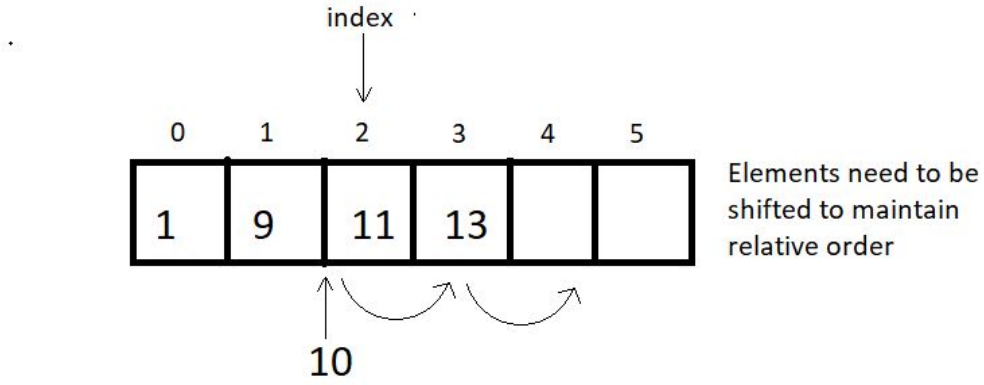
An important note on Arrays:

If we create an array of length 100 using a[100] in C language, we need not use all the elements. It is possible for a program to use just 60 elements out of these 100. (But we cannot go beyond 100 elements).



Insertion:

An element can be inserted in an array at a specific position. For this operation to succeed, the array must have enough capacity. Suppose we want to add an element 10 at index 2 in the below-illustrated array, then the elements after index 1 must get shifted to their adjacent right to make way for a new element.

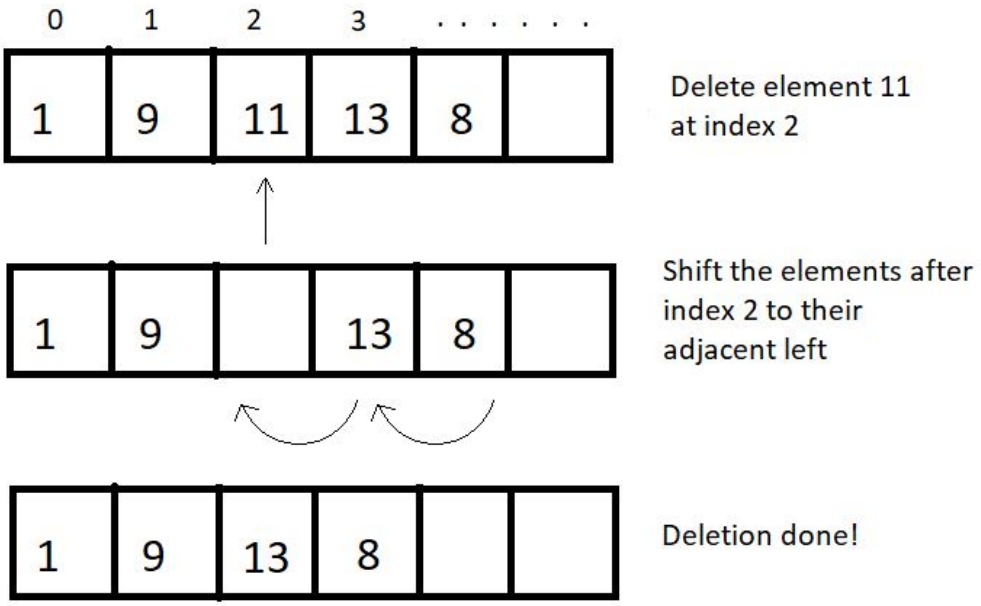


When no position is specified, it's best to insert the element at the end to avoid shifting, and this is when we achieve the best runtime O(1).

Deletion:

An element at a specified position can be deleted, creating a void that needs to be fixed by shifting all the elements to their adjacent left, as illustrated in the figure below.

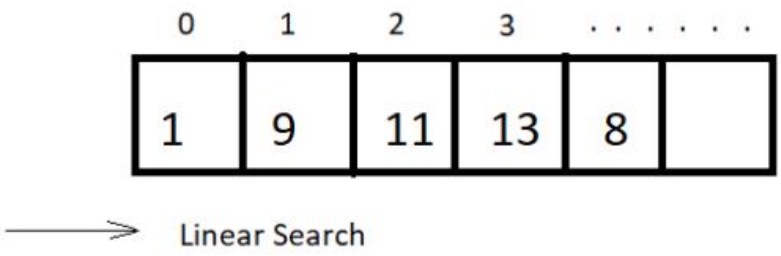
We can also bring the last element of the array to fill the void if the relative ordering is not important. :)



Quick Quiz: What is the best and the worst runtime for a delete operation?

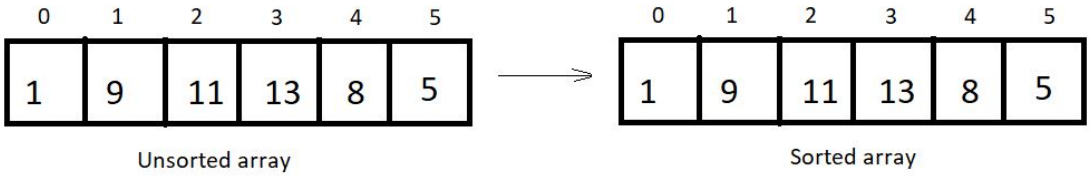
Searching:

Searching can be done by traversing the array until the element to be searched is found.O(n) There is still a better method. As you may remember, we talked about binary search in some previous tutorials. Don't forget to look it up if you missed it. We had analyzed both linear and binary search. This search method is only applicable for sorted arrays. Therefore, for sorted arrays, the time taken to search is much less than an unsorted array. O(logn)



Sorting:

Sorting means arranging an array in an orderly fashion (ascending or descending). We have different algorithms to sort arrays. We'll see various sorting techniques later in the course.



So, these were few primary operators for an abstract data type array. Today, we just had their introduction and visualization. We can now move on to code them in our editors and see how these algorithms work.

Thank you for being with me throughout. I hope you enjoyed the tutorial. If you appreciate my work, please let your friends know about this course too. Make sure to download the notes linked below. If you haven't checked out the whole playlist yet, move on to codewithharry.com or my YouTube channel to access it. See you all in the next tutorial, where we'll learn to code these operations in C language. Till then, keep learning.

Download pdf Notes here