Coding Insertion Operation in Array in Data Structures in C language

codewithharry.com/videos/data-structures-and-algorithms-in-hindi-10

In the last tutorial, we discussed all the primary operators and the concepts behind each. Today, we will learn how to code their algorithms. But before that, let's give ourselves a quick revision.

We talked about four operations-basically, **traversal**, **insertion**, **deletion,** and **searching**. As already mentioned, traversal is not any big a deal. It can just be achieved by using a *for* loop. Our main objective today would be to implement insertion. So, let's slide our chairs to our coding arena. I have attached the code snippet below.

Understanding code snippet 1:

1. We will start by declaring an array *arr* of length 100. Initialize this array with some 4-5 elements. This will be our used memory.
2. We'll create a void *display* function using the method of traversal. Pass this array to the display function by value or by reference. And print the elements. Printing the elements of an array has already been covered in my C playlist. Visit now if you haven't yet.
3. We'll now create an integer function *indInsertion* (integer, just to check if the operation succeeds). Before that, create an integer variable *size* to store the used size of the array. Pass into this void function the array and its used size, the element to be inserted and the total size, and the index where it is inserted.

   ```
   indInsertion(arr, size, element, 100, index);
   ```

4. In the *indInsertion* function, write the case of validity. Here, we'll check if the index is within the range [0,100]. We'll continue if it's valid; otherwise, return -1.
5. Create a *for* loop to shift the elements from the index to the last element to their adjacent right. This way, we'll create a void at the index we want to insert in.

6. Insert the element in the index. Return 1 on completion.

```
#include<stdio.h>


void display(int arr[], int n){
    // Code for Traversal
    for (int i = 0; i < n; i++)
    {
        printf("%d ", arr[i]);
    }
    printf("\n");
}

int indInsertion(int arr[], int size, int element, int capacity, int index){
    // code for Insertion
    if(size>=capacity){
        return -1;
    }
    for (int i = size-1; i >=index; i--)
    {
        arr[i+1] = arr[i];
    }
    arr[index] = element;
    return 1;
}

int main(){
    int arr[100] = {7, 8, 12, 27, 88};
    int size = 5, element = 45, index=1;
    display(arr, size);
    indInsertion(arr, size, element, 100, index);
    size +=1;
    display(arr, size);
    return 0;
}
```

Code Snippet 1: Insertion Operation Algorithm

Output of the above program:

```
7 8 12 27 88
7 45 8 12 27 88
```

So, as you can see, element 45 got inserted at index 1, and the rest of the elements from this index to the last shifted to their right. And this is how we do an insertion in an array. We may come across a lot of variations to insert into an array, but we'll go slow for now.

Quiz: Modify this program to display the array only after a successful insertion, otherwise print *Insertion failed*.

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. 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 the next operation, which is **deletion** in C language. Till then, keep learning.