ULAB
UNIVERSITY OF LIBERAL ARTS BANGLADESH

Report No: 01

Report Name: Arrays

CSE 2104
Object Oriented Progrmming Lab

Thursday, February 12, 2025.

NAZIFA TASNIM HIA

Lecturer, Department of CSE at University of Liberal Arts Bangladesh (ULAB)

NAME: Shohan Hossain Alfaz

ID: 242014154
Department: CSE

Title: Arrays

Objectives:

- To understand the concept of arrays and their applications.
- To learn how to declare and initialize arrays in C.
- To implement various array operations, including traversal, insertion, deletion, and merging.
- To develop problem-solving skills using arrays.

Introduction:

Arrays are fundamental data structures used to store multiple elements of the same data type in contiguous memory locations. They allow efficient data manipulation and retrieval. This lab focuses on understanding the implementation and operations of one-dimensional arrays in C.

Background Theory:

An array is a collection of elements identified by index or key. It provides a systematic way of storing and accessing data efficiently. Arrays can be classified into one-dimensional and multi-dimensional arrays. In this task, we focus on one-dimensional arrays and their operations, including:

1. Traversing an array
2. Inserting an element
3. Deleting an element
4. Merging two arrays

Arrays in C are declared using the syntax:

type array_name[size];

For example:

int marks[10];

This declares an integer array named marks with 10 elements.

Problem Understanding:

The task requires implementing basic array operations, specifically:

1. Reading and displaying elements of an array.
2. Inserting an element at a specific position.
3. Understanding array memory representation.

Algorithm Design:

Algorithm for Traversing an Array:

1. Start.
2. Declare an array and input its size.
3. Use a loop to input array elements.
4. Use another loop to print array elements.
5. Stop.

Algorithm for Inserting an Element in an Array:

1. Start.
2. Declare an array and input its size.
3. Take input for array elements.

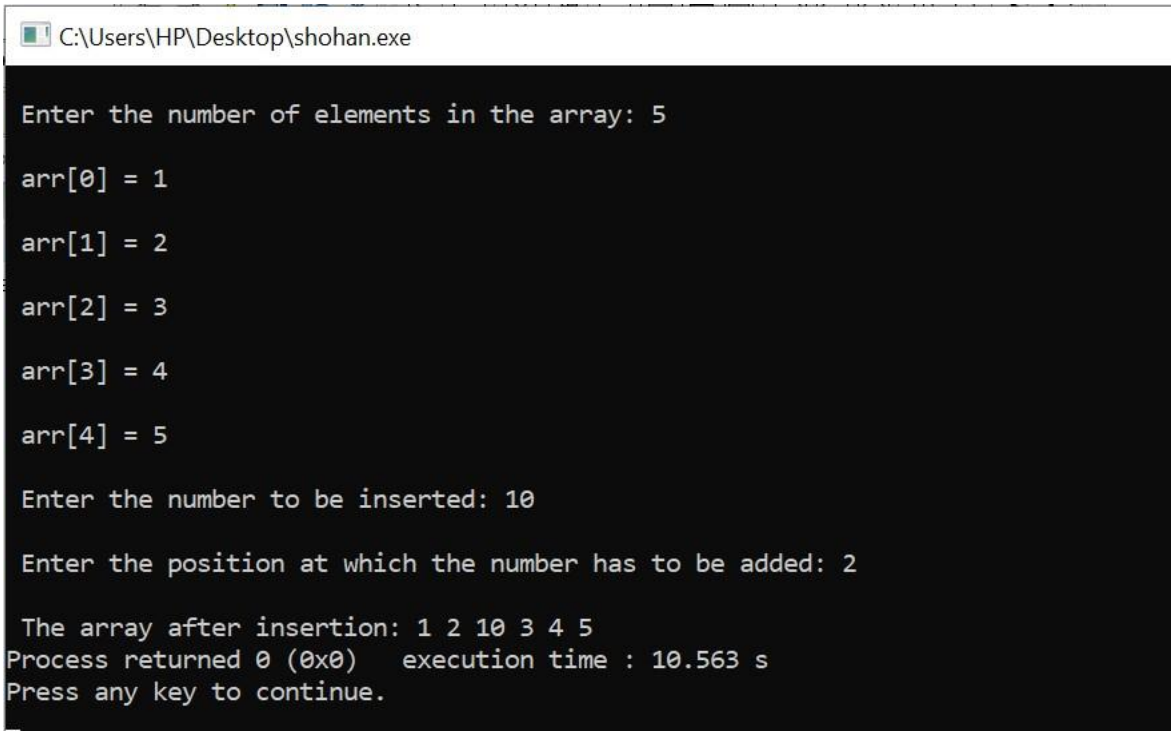
4. Ask for the element and position where it should be inserted.
5. Shift elements from the specified position to the right.
6. Insert the new element at the specified position.
7. Print the updated array.
8. Stop.

Code:

shohan.c

```c
#include <stdio.h>

int main() {

    int i, n, num, pos, arr[10];

    printf("\n Enter the number of elements in the array: ");
    scanf("%d", &n);

    for(i = 0; i < n; i++) {
        printf("\n arr[%d] = ", i);
        scanf("%d", &arr[i]);
    }

    printf("\n Enter the number to be inserted: ");
    scanf("%d", &num);

    printf("\n Enter the position at which the number has to be added: ");
    scanf("%d", &pos);

    for(i = n; i >= pos; i--) {
        arr[i] = arr[i - 1];
    }

    arr[pos] = num;
    n++;

    printf("\n The array after insertion: ");
    for(i = 0; i < n; i++) {
        printf("%d ", arr[i]);
    }

    return 0;
}
```

Code Output:



Conclusion:

This lab helped in understanding the fundamentals of arrays and their operations. We successfully implemented traversal and insertion operations in a one-dimensional array. This understanding is essential for solving complex problems in data structures and algorithms efficiently.

Title: Deleting an Element from an Array

Objectives:

- Understand the concept of deleting an element from an array.
- Learn how to shift elements to maintain the array structure after deletion.
- Implement deletion operation using C programming.
- Analyze the effect of deletion on array size and memory allocation.

Introduction:

Arrays are a fundamental data structure used to store multiple values of the same data type. One of the essential operations performed on arrays is deleting an element. When an element is deleted from an array, the remaining elements need to be shifted to fill the empty space to maintain the integrity of the data structure. This lab task focuses on understanding and implementing the deletion operation in C.

Background Theory:

Deleting an element from an array involves the following steps:

1. Identify the position (index) of the element to be deleted.
2. Shift all subsequent elements one position to the left to overwrite the deleted element.
3. Decrease the logical size of the array by one.
4. Print the updated array.

Pseudocode to delete an element from an array:

Step 1: [INITIALIZATION] SET I = POS
Step 2: Repeat Steps 3 and 4 while I < N - 1
Step 3: SET A[I] = A[I + 1]
Step 4: SET I = I + 1 [END OF LOOP]
Step 5: SET N = N - 1
Step 6: EXIT

Problem Understanding:

The problem requires us to delete an element from a given position in an array and shift the remaining elements to maintain the sequence.

Algorithm Design:

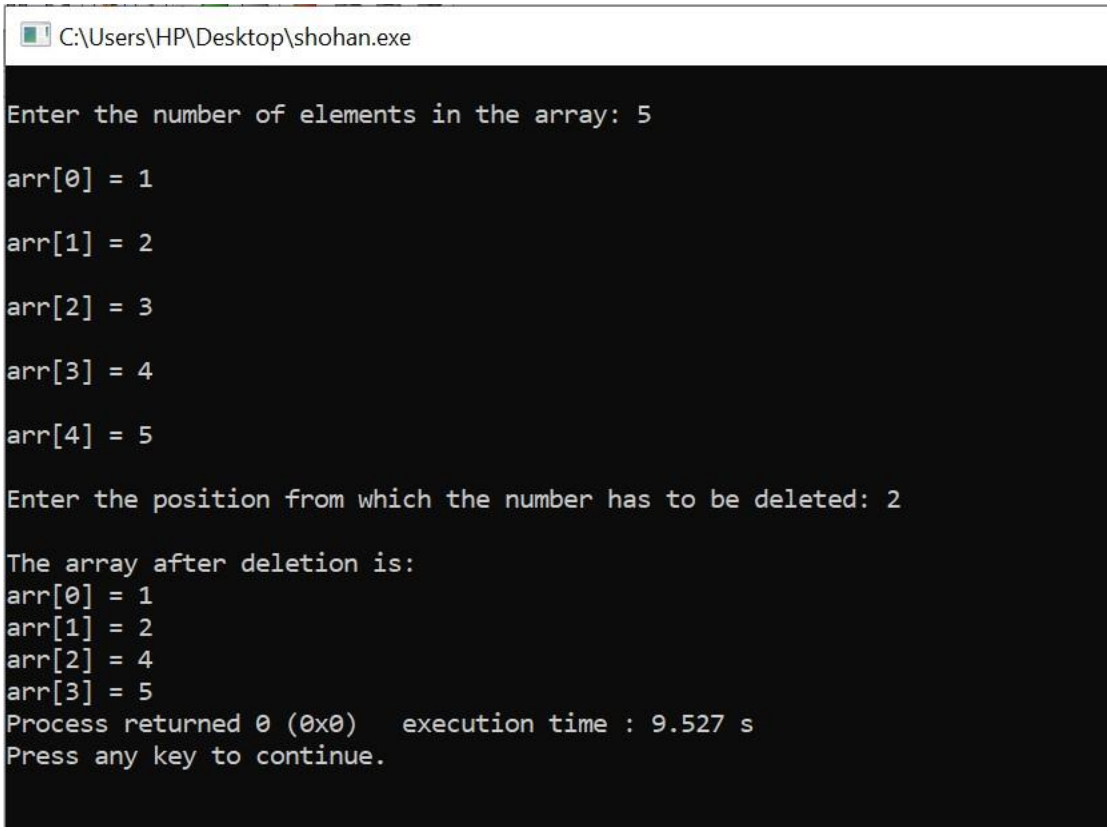
1. Start.
2. Declare an integer array with a predefined size.
3. Take user input for the number of elements in the array.
4. Accept array elements from the user.
5. Take user input for the position of the element to be deleted.
6. Shift all elements after the given position one step to the left.
7. Reduce the total element count by one.
8. Display the updated array.
9. Stop.

Code:

*shohan.c

```c
#include <stdio.h>

int main()
{
    int i, n, pos, arr[10];
    printf("\nEnter the number of elements in the array: ");
    scanf("%d", &n);
    for(i = 0; i < n; i++)
    {
        printf("\narr[%d] = ", i);
        scanf("%d", &arr[i]);
    }

    printf("\nEnter the position from which the number has to be deleted: ");
    scanf("%d", &pos);

    for(i = pos; i < n - 1; i++){
        arr[i] = arr[i + 1];
    }
    n--;

    printf("\nThe array after deletion is: ");
    for(i = 0; i < n; i++){
        printf("\narr[%d] = %d", i, arr[i]);
    }
    return 0;
}
```

Code Output:



Conclusion:

In this task, we successfully implemented and executed the deletion of an element from an array. By understanding the shifting operation, we ensured that the remaining elements retained their correct order. This lab task has enhanced our understanding of basic array operations and their practical implementation in C programming.