

Problem Statement

Write a C program to **traverse, insert, and delete** elements in a **1-D array** at different positions:

- **Insertion at the beginning, a specific position, and the end.**
 - **Deletion from the beginning, a specific position, and the end.**
- The program should provide a menu-based system for performing these operations.
-

2. Input & Output Description

- Input:
 - The number of elements (n), where $n \leq 100$.
 - The array elements entered by the user.
 - The operation choice (ch), which can be traverse, insert, or delete.
 - For insertion, the position and the element to be inserted.
 - For deletion, the position from which to delete.
 - Output:
 - The updated array after each operation.
 - If an invalid operation is attempted (e.g., out-of-bounds insertion/deletion), display an error message.
-

3. Algorithm for Binary Search

Algorithm for Traversal (traverse(n))

Step 1: Display "Array elements are:".

Step 2: Set $i = 0$.

Step 3: Repeat while $i < n$.

Step 4: Print $A[i]$.

Step 5: Increment i by 1.

Step 6: Print a new line.

Step 7: End.

Algorithm for Insertion (insert(pos, item, n))

Step 1: Check if $\text{pos} < 1$ or $\text{pos} > n + 1$.

Step 2: If true, display "Invalid position, insertion not possible" and exit.

Step 3: Set $i = n$.

Step 4: Repeat while $i \geq \text{pos}-1$.

Step 5: Assign $A[i] = A[i-1]$.

Step 6: Decrement i by 1.

Step 7: Assign $A[\text{pos}-1] = \text{item}$.

Step 8: Increment n by 1.

Step 9: End.

Algorithm for Deletion (delete(pos, n))

Step 1: Check if $\text{pos} < 1$ or $\text{pos} > n$.

Step 2: If true, display "Invalid position, deletion not possible" and exit.

Step 3: Set $i = \text{pos}-1$.

Step 4: Repeat while $i < n-1$.

Step 5: Assign $A[i] = A[i+1]$.

Step 6: Increment i by 1.

Step 7: Decrement n by 1.

Step 8: End.

Algorithm for Main (main())

Step 1: Display "Enter the number of elements:".

Step 2: Take input n .

Step 3: If $n > \text{MAX}$, display "Size exceeds limit" and exit.

Step 4: Set $i = 0$.

Step 5: Repeat while $i < n$.

Step 6: Take input $A[i]$.

Step 7: Increment i by 1.

Step 8: Call traverse(n).

Step 9: Repeat while true.

Step 10: Display "Menu: 1. Insert 2. Delete 3. Traverse 4. Exit".

Step 11: Take input ch.

Step 12: If ch == 1, go to Step 13.

Step 13: Take input pos and item.

Step 14: Call insert(pos, item, n).

Step 15: Call traverse(n).

Step 16: If ch == 2, go to Step 17.

Step 17: Take input pos.

Step 18: Call delete(pos, n).

Step 19: Call traverse(n).

Step 20: If ch == 3, call traverse(n).

Step 21: If ch == 4, exit.

Step 22: Display "Invalid choice, try again".

Step 23: End.

Source Code

```
#include <stdio.h>
#define MAX 100 // Maximum size of the array

int A[MAX]; // Array declaration

// Function for Traversal
void traverse(int n) {
    int i;
    printf("\nArray elements are: ");
    for (i = 0; i < n; i++) {
        printf("%d ", A[i]);
    }
    printf("\n");
}

// Function for Insertion
int insert(int pos, int item, int n) {
    int i;
    if (pos < 1 || pos > n + 1) {
        printf("\nInvalid position, Insertion not possible.\n");
        return n;
    }

    for (i = n; i >= pos - 1; i--) {
        A[i + 1] = A[i]; // Shifting elements to the right
    }

    A[pos - 1] = item; // Inserting the new element
```

```

        n++;
        return n;
    }

// Function for Deletion
int delete(int pos, int n) {
    int i;
    if (pos < 1 || pos > n) {
        printf("\nInvalid position, Deletion not possible.\n");
        return n;
    }

    for (i = pos - 1; i < n - 1; i++) {
        A[i] = A[i + 1]; // Shifting elements to the left
    }

    n--;
    return n;
}

// Main function
int main() {
    int n, ch, pos, item, i;

    // Taking input for number of elements
    printf("Enter the number of elements: ");
    scanf("%d", &n);

    if (n > MAX) {
        printf("\nSize exceeds the limit of %d. Cannot store elements.\n", MAX);
        return 1;
    }

    // Taking input for array elements
    printf("Enter %d elements:\n", n);
    for (i = 0; i < n; i++) {
        printf("A[%d]: ", i);
        scanf("%d", &A[i]);
    }

    traverse(n);

    // Menu-driven operations
    while (1) {
        printf("\n\nMenu of Operations");
        printf("\n1. Insert at beginning");
        printf("\n2. Insert at position");
        printf("\n3. Insert at end");
        printf("\n4. Delete from beginning");
        printf("\n5. Delete from position");
    }
}

```

```

printf("\n6. Delete from end");
printf("\n7. Traverse");
printf("\n8. Exit");
printf("\nEnter your choice: ");
scanf("%d", &ch);

switch (ch) {
    case 1: // Insert at beginning
        printf("\nEnter the element to insert: ");
        scanf("%d", &item);
        n = insert(1, item, n);
        traverse(n);
        break;

    case 2: // Insert at a given position
        printf("\nEnter the position and element to insert: ");
        scanf("%d%d", &pos, &item);
        n = insert(pos, item, n);
        traverse(n);
        break;

    case 3: // Insert at the end
        printf("\nEnter the element to insert: ");
        scanf("%d", &item);
        n = insert(n + 1, item, n);
        traverse(n);
        break;

    case 4: // Delete from beginning
        n = delete(1, n);
        traverse(n);
        break;

    case 5: // Delete from a given position
        printf("\nEnter the position to delete: ");
        scanf("%d", &pos);
        n = delete(pos, n);
        traverse(n);
        break;

    case 6: // Delete from end
        n = delete(n, n);
        traverse(n);
        break;

    case 7: // Traverse
        traverse(n);
        break;

    case 8: // Exit

```

```

        return 0;

    default:
        printf("\nInvalid choice! Try again.\n");
    }
}

return 0;
}

```

Output

```

Select Windows PowerShell
*****
*           Array Operations           *
*****
* 1. Traverse                         *
* 2. Insert at beginning              *
* 3. Insert at position               *
* 4. Insert at end                   *
* 5. Delete from beginning            *
* 6. Delete from position             *
* 7. Delete from end                 *
* 8. Exit                           *
*****
Enter your choice: 1
Array elements: 9 3 6 2 7
-----

```

default + del_beg, del_any: 2, del_end

```

Windows PowerShell
*****
*           Array Operations           *
*****
* 1. Traverse                         *
* 2. Insert at beginning              *
* 3. Insert at position               *
* 4. Insert at end                   *
* 5. Delete from beginning            *
* 6. Delete from position             *
* 7. Delete from end                 *
* 8. Exit                           *
*****
Enter your choice: 1
Array elements: 7 9 3 2 6 2 7 10
-----

```

Ins_beg: 7, ins_any: 3, 2, ins_end: 10

Discussion

The array must be checked whether it's empty or not before deleting elements. The memory should be carefully handled.

Teacher's signature