**Aim:** To write a C program that inserts a new element at the end as well as at a given position in an array.

**Apparatus / Software Required:**

1. Computer System
2. GCC Compiler / Turbo C
3. Text Editor or IDE (Code::Blocks, Dev-C++, VS Code, etc.)

**Theory:**  
In arrays, elements are stored in contiguous memory locations.

1. To insert an element at the **end**, simply place the new element after the last existing element.
2. To insert an element at a **specific position**, all elements after that position need to be shifted one step to the right to make space for the new element.

**Steps:**

1. Input the size of the array.
2. Input array elements.
3. Input the new element to insert at the end.
4. Place the element at the end (index = n).
5. Input another new element and its desired position.
6. Shift elements from that position to the right.
7. Insert the element at the given position.
8. Display the final array.

**Advantages:**

1. Easy way to insert elements when array has free space.
2. Simple and direct implementation.

**Limitations:**

1. Requires shifting of elements when inserting at a specific position (time complexity O(n)).
2. Fixed size array may not allow further insertion if already full.

**Algorithm:**

1. Start
2. Input size of the array (n)
3. Input n elements into the array
4. Input new element for insertion at the end
5. Place it at index n and increment n
6. Input new element and position for insertion
7. Shift all elements from (position-1) to the right by 1
8. Place the new element at (position-1)
9. Display final array
10. Stop

**Program:**

#include <stdio.h> // Standard input-output header

int main() {

int arr[100], n, i, element, pos;

// Step 1: Input size of array

printf("Enter the number of elements: ");

scanf("%d", &n);

// Step 2: Input array elements

printf("Enter %d elements:\n", n);

for(i = 0; i < n; i++) {

scanf("%d", &arr[i]);

}

// Step 3: Insert element at the end

printf("Enter element to insert at the end: ");

scanf("%d", &element);

arr[n] = element; // Place element at index n

n++; // Increase size

printf("Array after inserting at the end:\n");

for(i = 0; i < n; i++) {

printf("%d ", arr[i]);

}

printf("\n");

// Step 4: Insert element at a given position

printf("Enter element to insert: ");

scanf("%d", &element);

printf("Enter position (1 to %d): ", n+1);

scanf("%d", &pos);

if(pos < 1 || pos > n+1) {

printf("Invalid position!\n");

} else {

// Shift elements to the right

for(i = n; i >= pos; i--) {

arr[i] = arr[i-1];

}

arr[pos-1] = element; // Insert at given position

n++; // Increase size

printf("Array after inserting at position %d:\n", pos);

for(i = 0; i < n; i++) {

printf("%d ", arr[i]);

}

printf("\n");

}

return 0; // End program

}

**Sample Output:**

Enter the number of elements: 4

Enter 4 elements:

10

20

30

40

Enter element to insert at the end: 50

Array after inserting at the end:

10 20 30 40 50

Enter element to insert: 25

Enter position (1 to 6): 3

Array after inserting at position 3:

10 20 25 30 40 50

Enter the number of elements: 3

Enter 3 elements:

5

15

25

Enter element to insert at the end: 35

Array after inserting at the end:

5 15 25 35

Enter element to insert: 10

Enter position (1 to 5): 2

Array after inserting at position 2:

5 10 15 25 35

**Conclusion:**  
The program successfully demonstrates how to insert a new element at the **end** and at a **specific position** in an array. This method is simple but involves shifting elements during position-based insertion, which can be inefficient for large arrays.