**Aim:** To write a C program that finds the location of a given element in a sorted array using the Binary Search technique.

**Apparatus / Software Required:**

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

**Theory:** Binary Search is an efficient algorithm used to find the position of a target element in a sorted list. It works by repeatedly dividing the search range in half until the element is found or the range becomes empty.  
  
**Steps:**  
1. Compare the target element with the middle element of the array.  
2. If equal, return the position.  
3. If smaller, search the left half.  
4. If greater, search the right half.  
5. Repeat until found or no elements remain.  
  
**Advantages:**  
1. Faster than linear search for large datasets.  
2. Works in O(log n) time in the worst case.  
  
**Limitation:**3. Requires the array to be sorted before searching.

**Algorithm:**

1. Start  
2. Input the size of the array (n)  
3. Input n sorted elements into the array  
4. Input the element to search  
5. Initialize first = 0, last = n-1  
6. Repeat until first <= last:  
 - Find middle = (first + last) / 2  
 - If arr[middle] == search, print location and stop  
 - Else if arr[middle] < search, set first = middle + 1  
 - Else set last = middle - 1  
7. If element not found, display message  
8. Stop

**Program in C with Comments:**

#include <stdio.h> // Include standard input-output header file  
  
int main() { // Start of main function  
 int arr[100], n, i, search, first, last, middle; // Declare variables  
  
 // Step 1: Input size of array  
 printf("Enter the number of elements: "); // Ask user for total elements  
 scanf("%d", &n); // Read number of elements from user  
  
 // Step 2: Input array elements (must be sorted)  
 printf("Enter %d sorted elements:\n", n); // Prompt for sorted input  
 for(i = 0; i < n; i++) { // Loop from 0 to n-1  
 scanf("%d", &arr[i]); // Read each element and store in array  
 }  
  
 // Step 3: Input element to search  
 printf("Enter the element to search: "); // Ask user for the target element  
 scanf("%d", &search); // Read target element  
  
 // Step 4: Initialize search boundaries  
 first = 0; // Start index of array  
 last = n - 1; // End index of array  
 middle = (first + last) / 2; // Calculate middle index  
  
 // Step 5: Binary Search loop  
 while(first <= last) { // Continue until start index is less than or equal to end index  
 if(arr[middle] == search) { // If middle element matches search  
 printf("%d found at location %d.\n", search, middle + 1); // Print found message (1-based position)  
 return 0; // Exit program as element is found  
 }  
 else if(arr[middle] < search) { // If middle element is less than search element  
 first = middle + 1; // Move search range to right half  
 }  
 else { // If middle element is greater than search element  
 last = middle - 1; // Move search range to left half  
 }  
 middle = (first + last) / 2; // Update middle index for next iteration  
 }  
  
 // Step 6: If element is not found  
 printf("Element not found in the list.\n"); // Display not found message  
  
 return 0; // End of program  
}

**Sample Output:**

Enter the number of elements: 5  
Enter 5 sorted elements:  
10  
20  
30  
40  
50  
Enter the element to search: 40  
40 found at location 4.  
  
Enter the number of elements: 4  
Enter 4 sorted elements:  
5  
10  
15  
20  
Enter the element to search: 12  
Element not found in the list.

**Conclusion:** The Binary Search algorithm successfully finds the position of an element in a sorted list by repeatedly dividing the search range. It is more efficient than Linear Search for large datasets but requires the data to be sorted beforehand.