Unit 2

Program for Linear Search

Aim:

- To implement Linear search algorithm

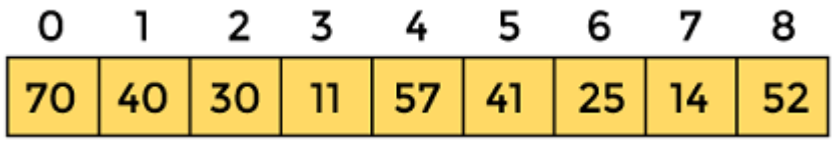
Theory:

- Linear search is also called as sequential search algorithm.
- It is the simplest searching algorithm.
- In Linear search, we simply traverse the list completely and match each element of the list with the item whose location is to be found.
- If the match is found, then the location of the item is returned; otherwise, the algorithm returns NULL.

- It is widely used to search an element from the unordered list,
- i.e., the list in which items are not sorted.
- The worst-case time complexity of linear search is O(n).

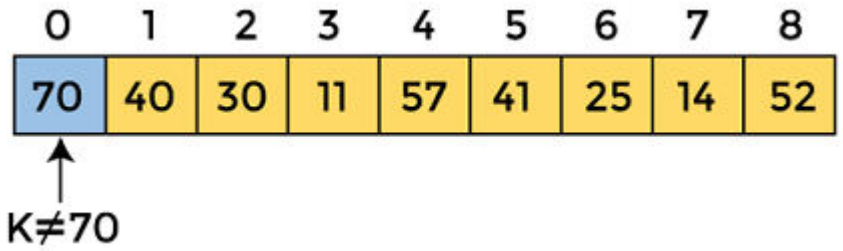
Working of Linear search

- To understand the working of linear search algorithm we will take an unsorted array.
- Let the elements of array are –

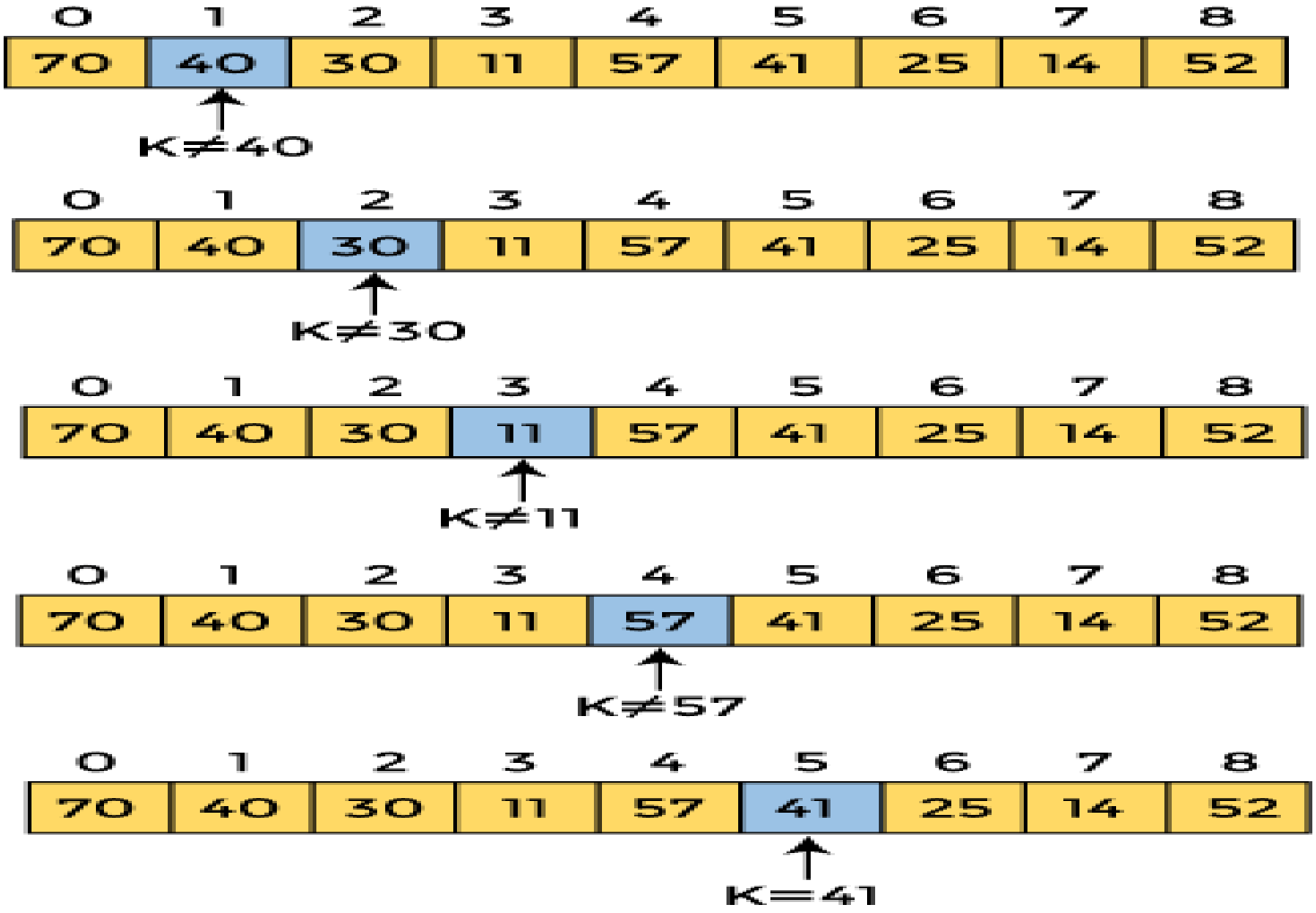


- If the element to be searched is K = 41

- start from the first element and compare K with each element of the array.



- The value of K, i.e., 41, is not matched with the first element of the array. So, move to the next element. And follow the same process until the respective element is found.


0 1 2 3 4 5 6 7 8
70 40 30 11 57 41 25 14 52
K≠40
0 1 2 3 4 5 6 7 8
70 40 30 11 57 41 25 14 52
K≠30
0 1 2 3 4 5 6 7 8
70 40 30 11 57 41 25 14 52
K≠11
0 1 2 3 4 5 6 7 8
70 40 30 11 57 41 25 14 52
K≠57
0 1 2 3 4 5 6 7 8
70 40 30 11 57 41 25 14 52
K=41

- Now, the element to be searched is found.
- So algorithm will return the index of the element matched

Time Complexity

- Best Case Complexity –
- In Linear search, best case occurs when the element we are finding is at the first position of the array.
- The best-case time complexity of linear search is O(1).

- Worst Case Complexity - In Linear search, the worst case occurs when the element we are looking is present at the end of the array.
- The worst-case in linear search could be when the target element is not present in the given array, and we have to traverse the entire array.
- The worst-case time complexity of linear search is O(n).

Space Complexity

- In Linear Search, we are creating a boolean variable to store if the element to be searched is present or not.
- The variable is initialized to false and if the element is found, the variable is set to true.
- This variable can be used in other processes or returned by the function.

- As the amount of extra data in Linear Search is fixed, the Space Complexity is O(1).

- Therefore, Space Complexity of Linear Search is O(1).

Procedure:

simple approach to implement a linear search is

- Step1 : Begin with the leftmost element of arr[] and one by one compare x with each element.
- Step 2:If x matches with an element then return the index.
- Step 3: If x does not match with any of the elements then return -1.

14.2 LINEAR SEARCH

Linear search, also called as *sequential search*, is a very simple method used for searching an array for a particular value. It works by comparing the value to be searched with every element of the array one by one in a sequence until a match is found. Linear search is mostly used to search an unordered list of elements (array in which data elements are not sorted). For example, if an array A[] is declared and initialized as,

```
int A[] = {10, 8, 2, 7, 3, 4, 9, 1, 6, 5};
```



```
LINEAR_SEARCH(A, N, VAL)
Step 1: [INITIALIZE] SET POS = -1
Step 2: [INITIALIZE] SET I = 1
Step 3:      Repeat Step 4 while I<=N
Step 4:              IF A[I] = VAL
                            SET POS = I
                            PRINT POS
                            Go to Step 6
                     [END OF IF]
                      SET I = I + 1
             [END OF LOOP]
Step 5: IF POS = -1
        PRINT "VALUE IS NOT PRESENT
        IN THE ARRAY"
        [END OF IF]
Step 6: EXIT
```

Figure 14.1 Algorithm for linear search

and the value to be searched is VAL = 7, then searching means to find whether the value '7' is present in the array or not. If yes, then it returns the position of its occurrence. Here, POS = 3 (index starting from 0).

Figure 14.1 shows the algorithm for linear search.

In Steps 1 and 2 of the algorithm, we initialize the value of POS and I. In Step 3, a `while` loop is executed that would be executed till I is less than N (total number of elements in the array). In Step 4, a check is made to see if a match is found between the current array element and VAL. If a match is found, then the position of the array element is printed, else the value of I is incremented to match the next element with VAL. However, if all the array elements have been compared with VAL and no match is found, then it means that VAL is not present in the array.

Complexity of Linear Search Algorithm

Linear search executes in O(n) time where n is the number of elements in the array. Obviously, the best case of linear search is when VAL is equal to the first element of the array. In this case, only one comparison will be made. Likewise, the worst case will happen when either VAL is not present in the array or it is equal to the last element of the array. In both the cases, n comparisons will have to be made. However, the performance of the linear search algorithm can be improved by using a sorted array.

PROGRAMMING EXAMPLE

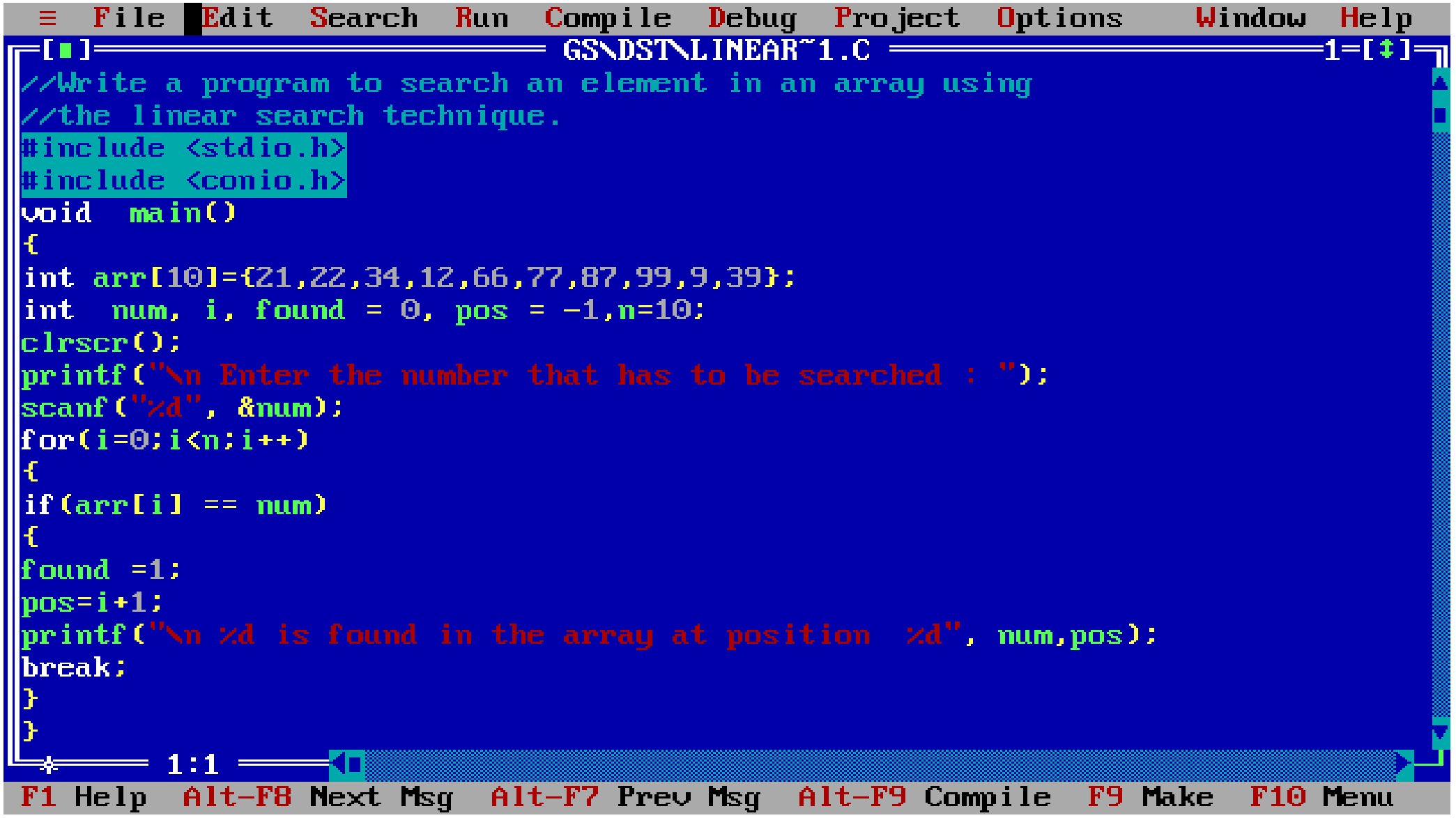
1. Write a program to search an element in an array using the linear search technique.

```
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#define size 20 // Added so the size of the array can be altered more easily
int main(int argc, char *argv[]) {
        int arr[size], num, i, n, found = 0, pos = -1;
        printf("\n Enter the number of elements in the array : ");
        scanf("%d", &n);
        printf("\n Enter the elements: ");
        for(i=0;i<n;i++)
        {
                scanf("%d", &arr[i]);
        }
        printf("\n Enter the number that has to be searched : ");
        scanf("%d", &num);
        for(i=0;i<n;i++)
        {
                if(arr[i] == num)
                {
                        found =1;
                        pos=i;
                        printf("\n %d is found in the array at position= %d", num,i+1);
                        /* +1 added in line 23 so that it would display the number in
      the first place in the array as in position 1 instead of 0 */
                        break;
                }
        }
        if (found == 0)
```



```
                printf("\n %d does not exist in the array", num);
                return 0;
        }
```

Program


File Edit Search Run Compile Debug Project Options Window Help
GS\DST\LINEAR~1.C
//Write a program to search an element in an array using
//the linear search technique.
#include <stdio.h>
#include <conio.h>
void main()
{
int arr[10]={21,22,34,12,66,77,87,99,9,39};
int num, i, found = 0, pos = -1,n=10;
clrscr();
printf("\n Enter the number that has to be searched : ");
scanf("%d", &num);
for(i=0;i<n;i++)
{
if(arr[i] == num)
{
found =1;
pos=i+1;
printf("\n %d is found in the array at position %d", num,pos);
break;
}
}
1:1
F1 Help Alt-F8 Next Msg Alt-F7 Prev Msg Alt-F9 Compile F9 Make F10 Menu

```
if (found == 0)
printf("\n %d does not exist in the array", num);
getch();
}
```

Output

```
The array elements are
21
22
34
12
66
77
87
99
9
39

 Enter the number that has to be searched : 39

 39 is found in the array at position  10_
```

```
The array elements are
21
22
34
12
66
77
87
99
9
39

 Enter the number that has to be searched : 100

 100 does not exist in the array_
```