

**VIVEKANANDA COLLEGE
THAKURPUKUR
KOLKATA-700063**

NAAC ACCREDITED 'A' GRADE



Topic: Searching

Course Title: Data Structure

Paper: CMS-A-CC-2-3-TH

Unit: Searching and Sorting

Semester: 2nd Semester

Name of the Teacher: Amitava Biswas

Name of the Department: Computer Science

What is Search?

Search is a process of finding a value in a list of values. In other words, searching is the process of locating given value position in a list of values.

Linear Search Algorithm (Sequential Search Algorithm)

Linear search algorithm finds a given element in a list of elements with **$O(n)$** time complexity where **n** is total number of elements in the list. This search process starts comparing search element with the first element in the list. If both are matched then result is element found otherwise search element is compared with the next element in the list. Repeat the same until search element is compared with the last element in the list, if that last element also doesn't match, then the result is "Element not found in the list". That means, the search element is compared with element by element in the list.

Linear search is implemented using following steps...

- **Step 1** - Read the search element from the user.
- **Step 2** - Compare the search element with the first element in the list.
- **Step 3** - If both are matched, then display "Given element is found!!!" and terminate the function
- **Step 4** - If both are not matched, then compare search element with the next element in the list.
- **Step 5** - Repeat steps 3 and 4 until search element is compared with last element in the list.
- **Step 6** - If last element in the list also doesn't match, then display "Element is not found!!!" and terminate the function.

Example

Consider the following list of elements and the element to be searched...

list

0	1	2	3	4	5	6	7
65	20	10	55	32	12	50	99

search element **12**

Step 1:

search element (12) is compared with first element (65)

list

0	1	2	3	4	5	6	7
65	20	10	55	32	12	50	99

12

Both are not matching. So move to next element

Step 2:

search element (12) is compared with next element (20)

list

0	1	2	3	4	5	6	7
65	20	10	55	32	12	50	99

12

Both are not matching. So move to next element

Step 3:

search element (12) is compared with next element (10)

list

0	1	2	3	4	5	6	7
65	20	10	55	32	12	50	99

12

Both are not matching. So move to next element

Step 4:

search element (12) is compared with next element (55)

list

0	1	2	3	4	5	6	7
65	20	10	55	32	12	50	99

12

Both are not matching. So move to next element

Step 5:

search element (12) is compared with next element (32)

list

0	1	2	3	4	5	6	7
65	20	10	55	32	12	50	99

12

Both are not matching. So move to next element

Step 6:

search element (12) is compared with next element (12)

list

0	1	2	3	4	5	6	7
65	20	10	55	32	12	50	99

12

Both are matching. So we stop comparing and display element found at index 5.

Implementation of Linear Search Algorithm using C Programming Language

```
#include<stdio.h>
#include<conio.h>

void main(){
    int list[20],size,i,sElement;

    printf("Enter size of the list: ");
    scanf("%d",&size);

    printf("Enter any %d integer values: ",size);
    for(i = 0; i < size; i++)
        scanf("%d",&list[i]);

    printf("Enter the element to be Search: ");
    scanf("%d",&sElement);

    // Linear Search Logic
    for(i = 0; i < size; i++)
    {
        if(sElement == list[i])
        {
            printf("Element is found at %d index", i);
            break;
        }
    }
    if(i == size)
        printf("Given element is not found in the list!!!");
    getch();
}
```

```
"C:\Users\User\Desktop\New folder\Search\bin\Debug\Search.exe"
Enter size of the list: 5
Enter any 5 integer values: 3 1 5 7 4
Enter the element to be Search: 7
Element is found at 3 index
Process returned 0 (0x0) execution time : 77.741 s
Press any key to continue.
```

Binary Search Algorithm

Binary search algorithm finds a given element in a list of elements with $O(\log n)$ time complexity where n is total number of elements in the list. The binary search algorithm can be used with only a sorted list of elements. That means the binary search is used only with a list of elements that are already arranged in an order. The binary search can not be used for a list of elements arranged in random order. This search process starts comparing the search element with the middle element in the list. If both are matched, then the result is "element found". Otherwise, we check whether the search element is smaller or larger than the middle element in the list. If the search element is smaller, then we repeat the same process for the left sublist of the middle element. If the search element is larger, then we repeat the same process for the right sublist of the middle element. We repeat this process until we find the search element in the list or until we left with a sublist of only one element. And if that element also doesn't match with the search element, then the result is "Element not found in the list".

Binary search is implemented using following steps...

Step 1 - Read the search element from the user.

Step 2 - Find the middle element in the sorted list.

Step 3 - Compare the search element with the middle element in the sorted list.

Step 4 - If both are matched, then display "Given element is found!!!" and terminate the function.

Step 5 - If both are not matched, then check whether the search element is smaller or larger than the middle element.

Step 6 - If the search element is smaller than middle element, repeat steps 2, 3, 4 and 5 for the left sublist of the middle element.

Step 7 - If the search element is larger than middle element, repeat steps 2, 3, 4 and 5 for the right sublist of the middle element.

Step 8 - Repeat the same process until we find the search element in the list or until sublist contains only one element.

Step 9 - If that element also doesn't match with the search element, then display "Element is not found in the list!!!" and terminate the function.

Example

Consider the following list of elements and the element to be searched...

list

0	1	2	3	4	5	6	7	8
10	12	20	32	50	55	65	80	99

search element **12**

Step 1:

search element (12) is compared with middle element (50)

list

0	1	2	3	4	5	6	7	8
10	12	20	32	50	55	65	80	99

12

Both are not matching. And 12 is smaller than 50. So we search only in the left sublist (i.e. 10, 12, 20 & 32).

list

0	1	2	3	4	5	6	7	8
10	12	20	32	50	55	65	80	99

Step 2:

search element (12) is compared with middle element (12)

list

0	1	2	3	4	5	6	7	8
10	12	20	32	50	55	65	80	99

12

Both are matching. So the result is "Element found at index 1"

search element **80**

Step 1:

search element (80) is compared with middle element (50)

list

0	1	2	3	4	5	6	7	8
10	12	20	32	50	55	65	80	99

80

Both are not matching. And 80 is larger than 50. So we search only in the right sublist (i.e. 55, 65, 80 & 99).

list

0	1	2	3	4	5	6	7	8
10	12	20	32	50	55	65	80	99

Step 2:

search element (80) is compared with middle element (65)

list

0	1	2	3	4	5	6	7	8
10	12	20	32	50	55	65	80	99

80

Both are not matching. And 80 is larger than 65. So we search only in the right sublist (i.e. 80 & 99).

list

0	1	2	3	4	5	6	7	8
10	12	20	32	50	55	65	80	99

Step 3:

search element (80) is compared with middle element (80)

list

0	1	2	3	4	5	6	7	8
10	12	20	32	50	55	65	80	99

80

Both are not matching. So the result is "Element found at index 7"

Implementation of Binary Search Algorithm using C Programming Language

```
#include<stdio.h>
#include<conio.h>

void main()
{
    int first, last, middle, size, i, sElement, list[100];
    clrscr();

    printf("Enter the size of the list: ");
    scanf("%d",&size);

    printf("Enter %d integer values in Assending order\n", size);

    for (i = 0; i < size; i++)
        scanf("%d",&list[i]);

    printf("Enter value to be search: ");
    scanf("%d", &sElement);

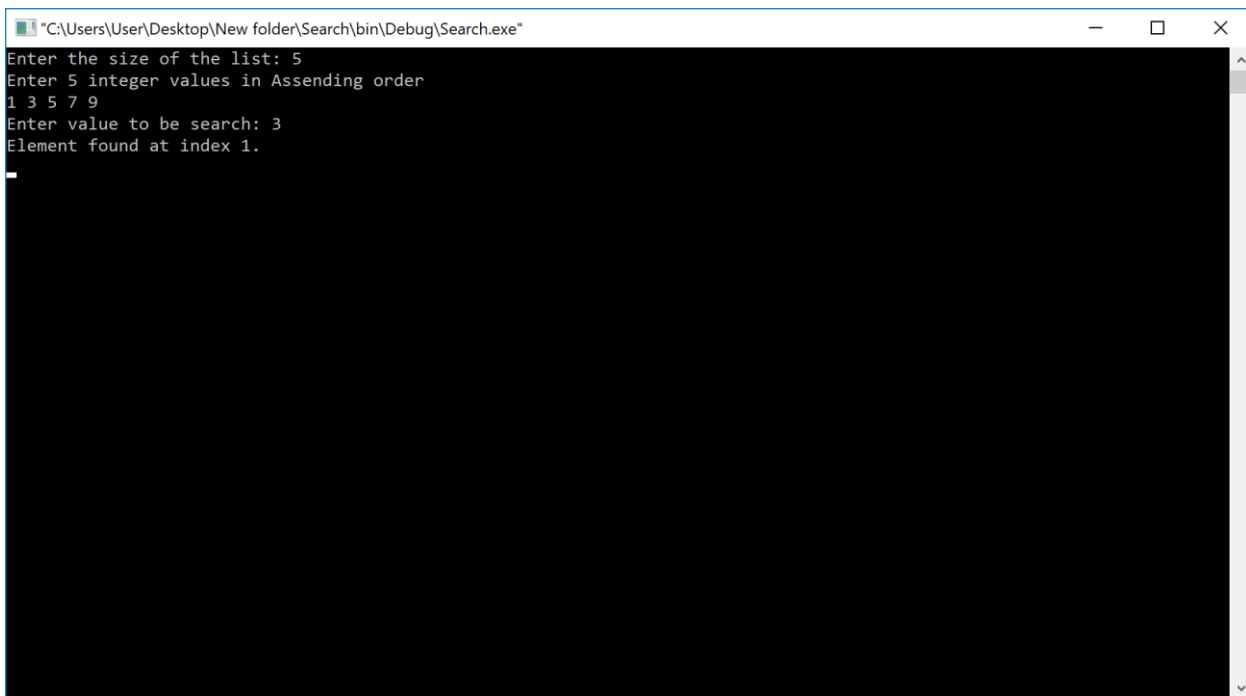
    first = 0;
    last = size - 1;
    middle = (first+last)/2;

    while (first <= last) {
        if (list[middle] < sElement)
            first = middle + 1;
        else if (list[middle] == sElement) {
            printf("Element found at index %d.\n",middle);
            break;
        }
        else
```

```
        last = middle - 1;

        middle = (first + last)/2;
    }
    if (first > last)
        printf("Element Not found in the list.");
    getch();
}
```

Output



```
"C:\Users\User\Desktop\New folder\Search\bin\Debug\Search.exe"
Enter the size of the list: 5
Enter 5 integer values in Assending order
1 3 5 7 9
Enter value to be search: 3
Element found at index 1.
```