



Tutorial Link <https://codequotient.com/tutorials/Searching an Element - Linear Search/5a12edf146765b2b63e3476b>

TUTORIAL

Searching an Element - Linear Search

Chapter

1. Searching an Element - Linear Search

Topics

1.2 Linear Search

1.5 Recursive implementation of linear

The process of identifying or finding a particular record is called Searching. You often spend time in searching for any desired item. If the data is kept properly in sorted order, then searching becomes very easy and efficient. Any search is said to be successful or unsuccessful depending upon whether the element that is being searched is found or not. In this article you will get to know the basic concepts of searching in sorted and unsorted arrays that is used in data structures. Search can be done popularly in two ways: -

Linear Search

If we start from the first element of list, and compare each element with the element we are searching, it is called linear search. This method can be performed on a sorted or an unsorted list (usually arrays). In both cases, the search will start from array index 0 and each time it match the searched element and the element at current index. If they match it will return the index otherwise, it will move to next index. If the whole array is passed to a function and the searched element does not match then it will return a negative index. The time complexity of Linear search is $O(n)$ as it may have to search for all the elements in worst case. In case of an array the general algorithm for linear search is as follows: -

```
X = searched_Element
index = 0
While(index < length_of_array)
    If: X == array[index] then RETURN index
    Else: index = index + 1
End
End
RETURN -1
```

Following is the iterative implementation of linear search: -

```
1 function linear_search(arr,n,x){
2     let i;
3     for (i=0; i<n; i++)
4         if (arr[i] == x)    // Check each element of
the array.
5             return i;      // if found return the
position
6     return -1;    // otherwise return -1
7 }
8
9 function main(){
10     let arr = [10,11,12,13,14,25,26,37,48,29]
11     const x=25;    // Searched Element.
12
13     let loc=linear_search(arr, arr.length, x);    //
Call the search function
14
15     if(loc != -1)
16         console.log(`Element found at location :
${loc}`);
17     else
18         console.log(`Element not present in the
array.`);
19     return 0;
20 }
21
22 main()
```

JavaScript

```
1 #include<stdio.h>
2
3 int linear_search(int arr[], int n, int x)
```

C

```
4 {
5     int i;
6     for (i=0; i<n; i++)
7         if (arr[i] == x)    // Check each element of the
array.
8             return i;      // if found return the position
9     return -1;    // otherwise return -1
10 }
11
12 int main()
13 {
14     int loc,x,array[]={10,11,12,13,14,25,26,37,48,29};
15
16     x=25;    // Searched Element.
17
18     loc=linear_search(array, 10, x);    // Call the
search function
19
20     if(loc != -1)
21         printf("Element found at location : %d",loc);
22     else
23         printf("Element not present in the array.");
24     return 0;
25 }
26
```

```
1 import java.util.Scanner;
2 // Other imports go here
3 // Do NOT change the class name
4 class Main{
5     static int linear_search(int arr[], int n, int x)
6     {
7         int i;
8         for (i=0; i<n; i++)
9             if (arr[i] == x)    // Check each element of the
array.
10                 return i;      // if found return the position
11     return -1;    // otherwise return -1
12     }
13     public static void main(String[] args)
14     {
15         int loc,x,array[]={10,11,12,13,14,25,26,37,48,29};
```

Java

```

16
17     x=25;    // Searched Element.
18
19     loc=linear_search(array, 10, x);    // Call the
search function
20
21     if(loc != -1)
22         System.out.print("Element found at location : " +
loc);
23     else
24         System.out.print("Element not present in the
array.");
25     }
26 }

```

```

1  def linear_search(array,n,x):
2      for i in range(n):
3          if array[i] == x: # Check every element of the
array
4              return i      # If found , return the
position
5          return -1         # Otherwise return -1
6
7
8  if __name__ == '__main__':
9      array = [10,11,12,13,14,25,26,37,48,29]
10     x = 25 # Element to be searched
11     loc = linear_search(array,len(array),x) # Calling
the function
12     if(loc !=-1):
13         print('Element found at location :',loc)
14     else:
15         print('Element not present in array.')

```

Python 3

```

1  #include<iostream>
2  #include<cstdio>
3  #include<cmath>
4  using namespace std;
5
6  int linear_search(int arr[], int n, int x) {
7      int i;
8      for (i=0; i<n; i++)
9          if (arr[i] == x)    // Check each element of
the array.

```

C++

```

10         return i;          // if found return the
    position
11     return -1;      // otherwise return -1
12 }
13
14 int main() {
15     int array[]={10,11,12,13,14,25,26,37,48,29};
16     int x=25;      // Searched Element.
17
18     int loc=linear_search(array, 10, x);    // Call the
    search function
19
20     if(loc != -1)
21         cout<<"Element found at location : "<<loc;
22     else
23         cout<<"Element not present in the array.";
24     return 0;
25 }

```

The output of above program is as below for different runs: -

```
Element found at location :5
```

Recursive implementation of linear

```

1  function rec_linear_search(arr,left,right,x) { Javascript
2      let result;
3      if (right < left)          // The array is exhausted
    so return -1
4          return -1;
5      if (arr[left] == x)        // If element found
    return position
6          return left;
7      // Call the function again with new subarray from
    next element.
8      result = rec_linear_search(arr, left+1, right, x);
9      return result;    // return the result to the
    calling function.
10 }
11
12 function main(){
13     let arr = [10,11,12,13,14,25,26,37,48,29]

```

```
14     const x=13;    // Searched Element.
15
16     let loc=rec_linear_search(arr,0, arr.length, x);
    // Call the search function
17
18     if(loc != -1)
19         console.log(`Element found at location :
    ${loc}`);
20     else
21         console.log(`Element not present in the
    array.`);
22     return 0;
23 }
24
25 main()
```

```
1  #include<stdio.h>
2
3  int rec_linear_search(int arr[], int left, int right,
    int x)
4  {
5      int result;
6      if (right < left)        // The array is exhausted so
    return -1
7      return -1;
8      if (arr[left] == x)      // If element found return
    position
9      return left;
10     // Call the function again with new subarray from
    next element.
11     result = rec_linear_search(arr, left+1, right, x);
12     return result;    // return the result to the calling
    function.
13 }
14
15 int main()
16 {
17     int loc,x,array[]={10,11,12,13,14,25,26,37,48,29};
18     x=13;                // element to be searched in the
    array
19     loc=rec_linear_search(array,0,10,x);
20     if(loc != -1)
21         printf("Element found at location : %d",loc);
22     else
        printf("Element not present in the array.");
```

```

23
24     return 0;
25 }
26

```

```

1  import java.util.Scanner;
2  // Other imports go here
3  // Do NOT change the class name
4  class Main{
5  static int rec_linear_search(int arr[], int left, int
right, int x)
6  {
7      int result;
8      if (right < left)          // The array is exhausted so
return -1
9      return -1;
10     if (arr[left] == x)        // If element found return
position
11     return left;
12     // Call the function again with new subarray from
next element.
13     result = rec_linear_search(arr, left+1, right, x);
14     return result;    // return the result to the calling
function.
15 }
16 public static void main(String[] args)
17 {
18     int loc,x,array[]={10,11,12,13,14,25,26,37,48,29};
19     x=13;    // Searched Element.
20     loc = rec_linear_search(array, 0, 10, x);    //
Call the search function
21     if(loc != -1)
22         System.out.print("Element found at location : " +
loc);
23     else
24         System.out.print("Element not present in the
array.");
25 }
26 }

```

Java

```

1  def rec_linear_search(array,left,right,x):
2      if (right < left):          # The array is exhausted
so return -1
3      return -1;
4      if (array[left] == x):      # If element found

```

Python 3

```

return position
5     return left
6     # Call the function again with new subarray from
next element.
7     result = rec_linear_search(array, left+1, right, x)

8     return result    # return the result to the calling
function.
9
10
11 if __name__ == '__main__':
12     array = [10,11,12,13,14,25,26,37,48,29]
13     x = 13 # Element to be searched
14     loc = rec_linear_search(array,0,len(array),x) #
Calling the function
15     if(loc !=-1):
16         print('Element found at location :',loc)
17     else:
18         print('Element not present in array.')

```

```

1  #include<iostream>
2  #include<cstdio>
3  #include<cmath>
4  using namespace std;
5
6  int rec_linear_search(int arr[], int left, int right,
int x) {
7      int result;
8      if (right < left)          // The array is exhausted
so return -1
9          return -1;
10     if (arr[left] == x)        // If element found
return position
11         return left;
12     // Call the function again with new subarray from
next element.
13     result = rec_linear_search(arr, left+1, right, x);
14     return result;    // return the result to the
calling function.
15 }
16
17 int main() {
18     int array[]={10,11,12,13,14,25,26,37,48,29};
19     int x=13;    // Searched Element.
20     int n = sizeof(array)/sizeof(array[0]);

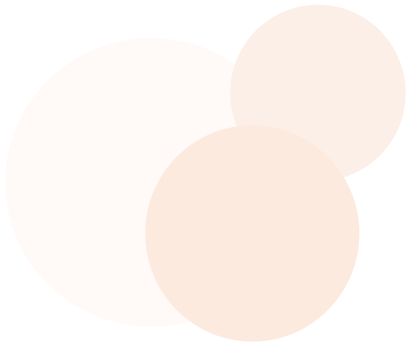
```

C++


```
21     int loc=rec_linear_search(array,0,n, x);    // Call
the search function
22
23     if(loc != -1)
24         cout<<"Element found at location : "<<loc;
25     else
26         cout<<"Element not present in the array.";
27     return 0;
28 }
```

The output of above program is as below for different runs: -

```
Element found at location :3
```



Tutorial by codequotient.com | All rights reserved, CodeQuotient 2020