SEARCHING

What is searching?

Searching is the process finding an item with specified properties. The item may be stored in data base, array, trees or text files etc.

Why searching?

Searching is one of the computer science algorithms. All of us know, any information gathering process or retrieval process is basically a searching process. We already know that searching is more efficient if the data available in a sorted manner. Sorting is the one of the techniques for making the elements ordered.

Types of searching

The following are the types of searching techniques, which will be used to find out an item stored in memory.

1. Linear search
2. Binary Search

Linear search

Linear search is also called as ***sequential search***. This is a simple searching method. This doesn't require the list to be sorted also. The key, which is to be searched, is compared with each element of the list one by one. If match exists, the search is terminated. The search will be failed if no match exists in the list.

C Program:

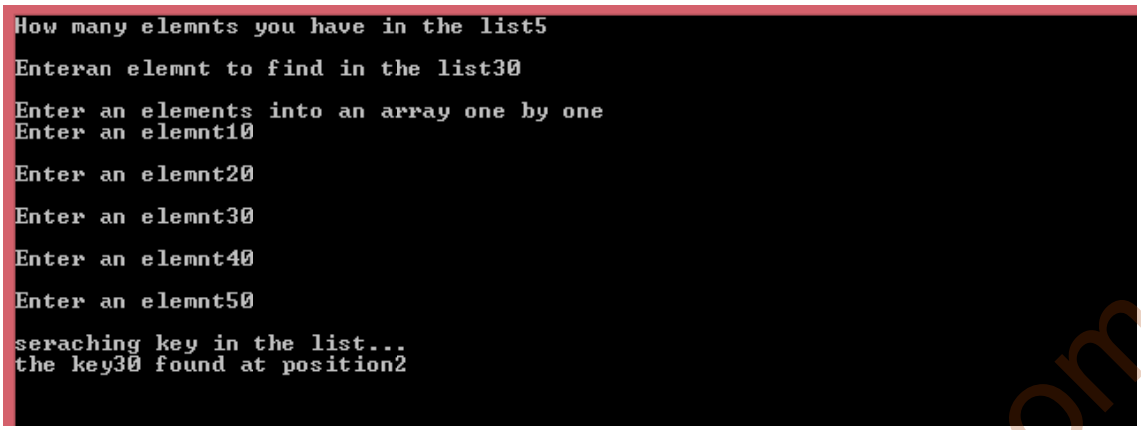
```
#include<stdio.h>
#include<stdlib.h>
#defineMaxSize 10

void main(){
```

```
        intarr[MaxSize],key,max;
        inti,pos;
        printf("How many elements you have in the list");
        scanf("%d",&max);
        printf("\nEnter an element to find in the list");
        scanf("%d",&key);
        printf("\nEnter an elements into an array one by one");
        for(i=0;i<max;i++){
                printf("\nEnter an element");
                scanf("%d",&arr[i]);
        }
        printf("\nseraching key in the list... ");
        for(i=0;i<max-1;i++){
                if(key==arr[i]){
                        pos=i;
                        printf("\nthekey%d found at position%d",key,pos);
                        return;
                }
        }

        printf("\nThegivenkey%d not found in the list",key);
        exit(0);
}
```

Output:



C Program: Linear Search to find multiple Occurrences of given element

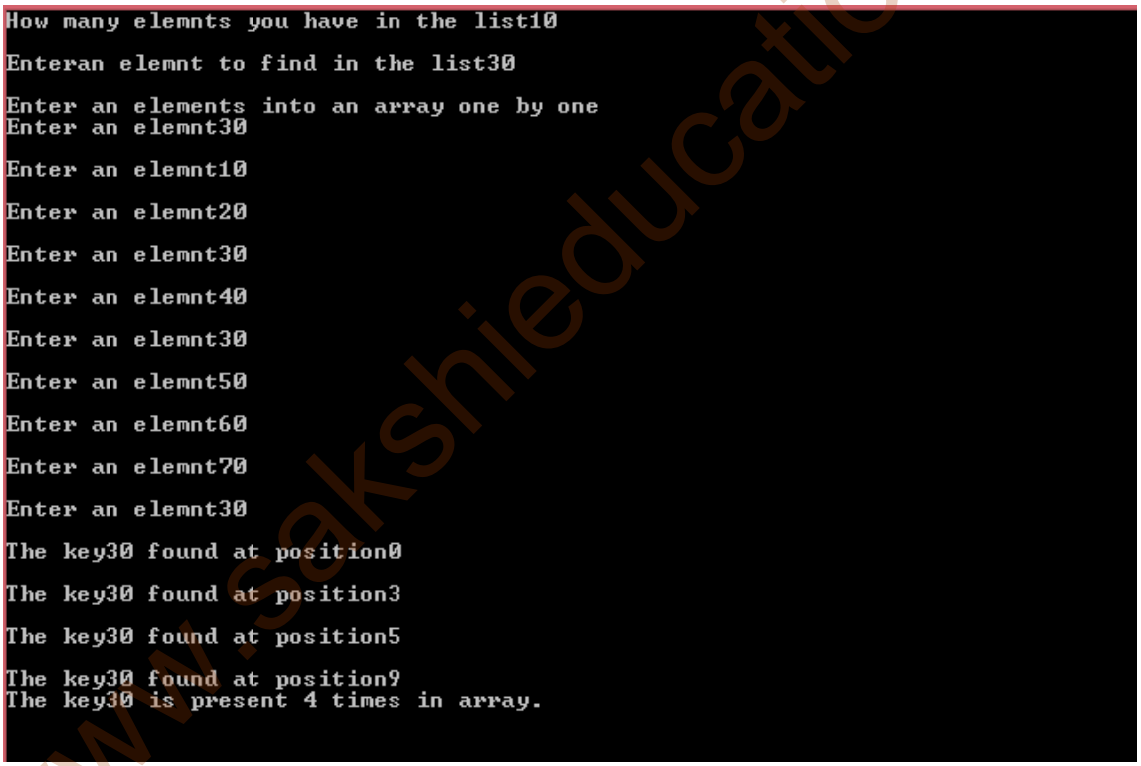
```
#include<stdio.h>
#include<stdlib.h>
#defineMaxSize 20

void main()
{
intarr[MaxSize],key,max,count=0;
inti,pos;
printf("How many elements you have in the list");
scanf("%d",&max);
printf("\nEnter an element to find in the list");
scanf("%d",&key);
printf("\nEnter an elements into an array one by one");
for(i=0;i<max;i++){
      printf("\nEnter an element");
      scanf("%d",&arr[i]);
}
for ( i = 0 ; i < max ; i++ ){
if ( arr[i] == key ){
            printf("\nThekey%d found at position%d\n", key, i);
```

```
            count++;
        }
    }
if ( count == 0 )
printf("%d key is not present in array.\n", key);
else
printf("The key%d is present %d times in array.\n", key, count);

}
```

Output:



C Program: Linear search using Function

```
#include<stdio.h>
#include<stdlib.h>
#defineMaxSize 20
```

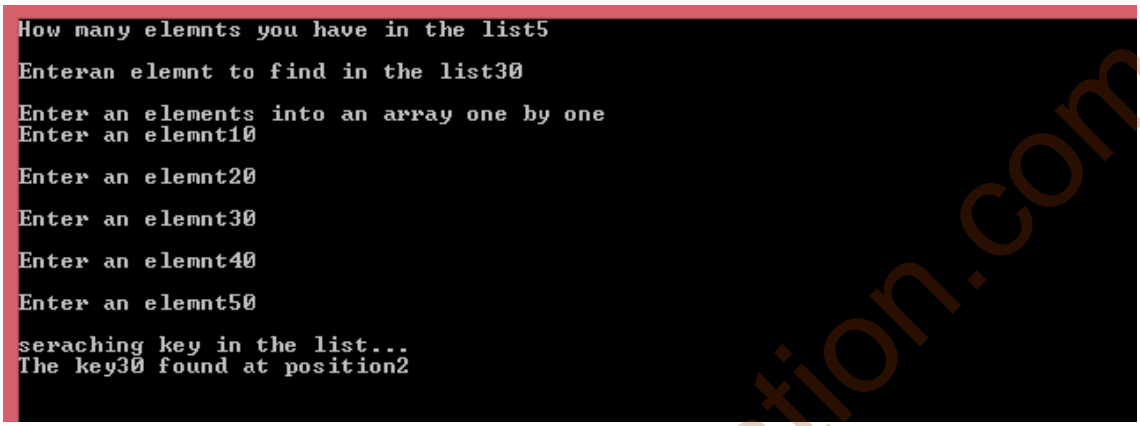
```
intlinearsearch(int elements[],intmax_size,int key);
void main(){
        intarr[MaxSize],key,max;
        inti,pos;
        printf("How many elements you have in the list");
        scanf("%d",&max);
        printf("\nEnter an element to find in the list");
        scanf("%d",&key);
        printf("\nEnter an element into an array one by one");
        for(i=0;i<max;i++){
                printf("\nEnter an element");
                scanf("%d",&arr[i]);
        }
        printf("\nseraching key in the list... ");
        pos=linearsearch(arr,max,key);
        if(pos!=-1)
                printf("\nThekey%d found at position%d\n", key, pos);
        else
                printf("Key not found in the list");

}
//function or linear search
intlinearsearch(int elements[],intmax_size,int key){
        int i=0;
        for(i=0;i<max_size-1;i++){
                if(key==elements[i])

                        return i;
```

```
        }
        return -1;
}
```

OutPut:



Efficiency

To find the number of key comparisons for successful match, we can add number required for each comparison and divided by total number of elements in the list. This can be write as

$$(1+2+3+...n)/n = n(n+1)/2n$$

The time taken to search element is **O(n).** To improve the performance of linear search we will go for indexed searching method.

Binary Search

Binary search is another simple searching method. This requires list to be sorted before searching an element. You can use any sorting algorithm before finding an item in a list.

Algorithm

After sorting the list, divide the given list into two halves. The given key is compared with middle element of the list. Now three situations may occur.

1. The middle element matches with the key- the search will be end peacefully.

2. The middle element is greater than the key- then the search will be happened in the first half of the list.
3. The middle element is less than the key- then the search will be happened in the second half of the list.

This process will be repeated until we find the key or search fails if key doesn't exist.

Efficiency: The time taken to search any element O (logn).

Iterative Binary search C Program

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

#defineMaxSize 20

intBinarySearch(int elements[],intmax_size,int key);
voidbubblesort(int elements[],intmax_size);

void main(){
        intarr[MaxSize],key,max;
        inti,pos;
        printf("How many elements you have in the list");
        scanf("%d",&max);
        printf("\nEnter an elements into an array one by one");
        for(i=0;i<max;i++){
                printf("\nEnter an element");
                scanf("%d",&arr[i]);
        }
        printf("\nsorting elements in the list... ");
        bubblesort(arr,max);
        for(i=0;i<max;i++){
```

```
            //printf("\nEnter an element");
            printf("%d",arr[i]);
            printf(" ");
        }
        printf("\nEnter an element to find in the list");
        scanf("%d",&key);

        printf("\nseraching key in the list... ");
        pos=BinarySearch(arr,max,key);
        if(pos!=-1)
            printf("\nThekey%d found at position%d\n", key, pos+1);
        else
            printf("Key not found in the list");

}
//function or Binarysearch
intBinarySearch(int elements[],intmax_size,int key){
        int first, last, middle;
        first=0;
        last=max_size-1;
        middle=(first+last)/2;
        while( first <= last ){
if ( elements[middle] < key )
first = middle + 1;
elseif ( elements[middle] == key ){
return middle;
}
else
last = middle - 1;
```

```
        middle = (first + last)/2;
}
        if ( first > last )
        return -1;
}
//bubble sort
voidbubblesort(int elements[],intmax_size){
        inti,j,temp;
        for(i=0;i<max_size;i++){
                for(j=0;j<max_size-i-1;j++){
                        if(elements[j]>elements[j+1]){
                                temp      = elements[j];
                                elements[j]   = elements[j+1];
                                elements[j+1] = temp;
                        }

                }
        }
}
```

Output:

```
Enter an elemnt4

Enter an elemnt6

Enter an elemnt10

Enter an elemnt20

Enter an elemnt2

Enter an elemnt7

Enter an elemnt9

Enter an elemnt1

Enter an elemnt15

Enter an elemnt5

sorting elemnets in the list... 1 2 4 5 6 7 9 10 15 20
Enteran elemnt to find in the list4

seraching key in the list...
The key4 found at position3
```

Recursive Binary Search C Program

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

#defineMaxSize 20

intBinarySearch(int elements[],intmax_size,intkey,int first, int last);
voidbubblesort(int elements[],intmax_size);

void main(){
        intarr[MaxSize],key,max;
        inti,pos;
        printf("How many elements you have in the list");
        scanf("%d",&max);
        printf("\nEnter an elements into an array one by one");
        for(i=0;i<max;i++){
                printf("\nEnter an element");
                scanf("%d",&arr[i]);
```

```
        }
        printf("\nsorting elements in the list... ");
        bubblesort(arr,max);
        for(i=0;i<max;i++){
                //printf("\nEnter an element");
                printf("%d",arr[i]);
                printf(" ");
        }
        printf("\nEnteran element to find in the list");
        scanf("%d",&key);

        printf("\nseraching key in the list... ");
        pos=BinarySearch(arr,max,key,0,max-1);
        if(pos!=-1)
                printf("\nThekey%d found at position%d\n", key, pos+1);
        else
                printf("Key not found in the list");

}
//function  BinarySearchuing recursion
intBinarySearch(int elements[],intmax_size,intkey,intfirst,int last){
        int mid=-1;
        if(first<=last){
                mid = (first + last)/2;
                if (elements[mid] == key)
                        return mid;
                elseif (elements[mid] < key)
                        mid=BinarySearch(elements,max_size, key, mid+1,
last);
```

```
            else// last possibility: a[mid] > x
                mid=BinarySearch(elements, max_size,key, first, mid-
1);
        }
        return mid;
}
//bubble sort
voidbubblesort(int elements[],intmax_size){
        inti,j,temp;
        for(i=0;i<max_size;i++){
            for(j=0;j<max_size-i-1;j++){
                if(elements[j]>elements[j+1]){
                    temp      = elements[j];
                    elements[j]   = elements[j+1];
                    elements[j+1] = temp;
                }

            }
        }
}
```

Output:

```
How many elemnts you have in the list10

Enter an elements into an array one by one
Enter an elemnt8

Enter an elemnt4

Enter an elemnt10

Enter an elemnt1

Enter an elemnt5

Enter an elemnt20

Enter an elemnt13

Enter an elemnt7

Enter an elemnt2

Enter an elemnt9

sorting elemnets in the list... 1 2 4 5 7 8 9 10 13 20
Enteran element to find in the list8

seraching key in the list...
The key8 found at position6
```

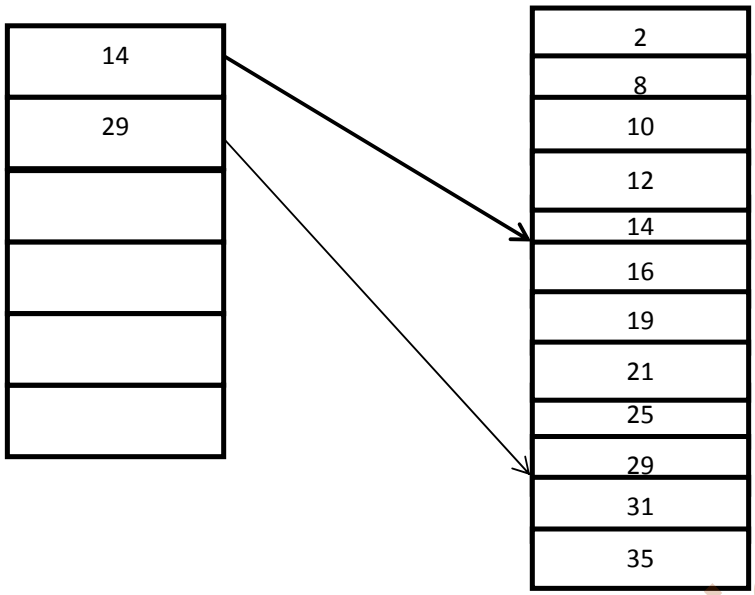
Comparing searching algorithms

Implementation	Search worst case	Search Avg case
Linear search	n	n/2
Binary search	logn	logn

Indexed sequential search

This is another searching method. This is widely used in file handling and database systems. This method based on the short representation of the list. This short representation is also called as index. There are many ways to form this index. This index will be formed by extract every fifth element from the list and make that as part of index.

Whenever, a particular key is to be searched, it is first searched in the index, the appropriate location from the index is extracted and that location is sequentially searched. Let us see an example.



Let us assume that we are searching for 16, the index is located first. If you don't get exact match for the key in the index, you will get only the range, which you may be used for searching the main list. In this example, we get the range that 16 are between 14 and 29. You may perform a sequential search in this range to get at the key value.