FoC2020-Lab

Lab-7 :C Programming on sorting and searching techniques using one dimensional array

---

1. Write and execute C program to read an array of ‘n’ elements and key element using keyboard, search a key element in an array *using linear search* and display whether a search is successful or unsuccessful.

Linear Search: A linear search, also known as a sequential search, is a method of finding an element within a list. It checks each element of the list sequentially (from index 0 to n-1) until a match is found or till end the of List reached.

Linear Search: **Steps:**

note : Use a `for/while` loop to Traverse in the array.

1) In every iteration of the loop,
   - **compare the `X(key)`** value with the current valuea[i] of the array.
   - If the **values match**, **return the current index** of the array.
   - If the values do not match, **move on to the next array element**

2) **.If no match is found, return `-1`.**

```
#include <stdio.h>
void main()
{
  int num;
  int i,  keynum, found = 0;
  printf("Enter the number of elements ");
  scanf("%d", &num);
  int array[num];
  printf("Enter the elements one by one \n");
    for (i = 0; i < num; i++)
    {
        scanf("%d", &array[i]);
    }
  printf("Enter the element to be searched ");
  scanf("%d", &keynum);
   for (i = 0; i < num ; i++)
    {
        if (keynum == array[i] )
        {
```

```
                  found = 1;
                  break;
              }
          }
        if (found == 1)
         printf("Element is present in the array at position %d",i+1);
        else
         printf("Element is not present in the array\n");
    }
```

---

2. Write and execute C program to read an array of elements and a key element using keyboard, search for a key element in an array using binary search and display the position of search element if key is found or unsuccessful search if key element is not foundboard . Display the smallest number and its position in the one dimensional array.

Binary search :

Points to Note: Binary search method requires the list of elements to be in a **sorted order.** To search an element we compare it with the element present at the center((n-1/2) of the list. If it matches the search is successful. - Else, the list is divided into two halves from 0th element to center element (first half) and another from center to final element (second half). The search element 'x' is further estimated to be in one of these halves comparing with the center element.

Consider the *middle* element *m* in *A[]*

1. If *k=m* , we have found the key *k* so we can stop immediately.

2. If *k<m*, then the key *k* can appear only in *the first half* of the sequence.

3. If *k>m*, then the key *k* can appear only in *the second half* of the sequence.

In the latter two cases, we *recursively continue in the selected half* of the sequence until either.

For example : Steps of binary search for the **key 19**

list **A [1,4,4,7,7,8,11,19,21,23,24,30]** (sorted array) .

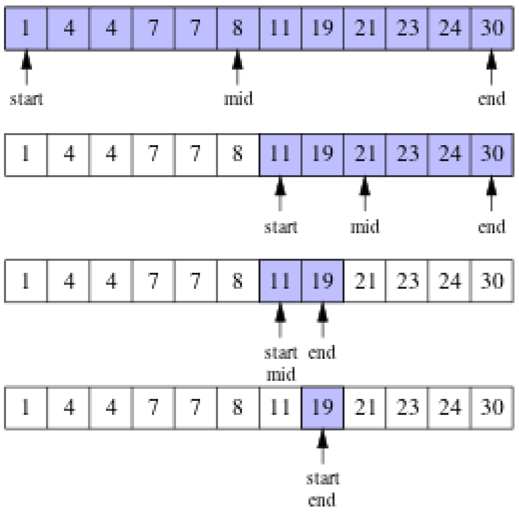
1. Firrst consider its middle element 8 (m)

2.As the sequence is sorted and 8 is less than the key 19 hene19 can occur only in the second half of the sequence(11 to 30)

3. We then repeat the same with the subsequence consisting of the second half

- Its middle element is 21, which is greater than the key 19. consider the first half of *this subsequenc*(11- 21) whose middle element is 11.

- last step the subsequence consists of one element, and this element is both the middle element and is equal to the key 19



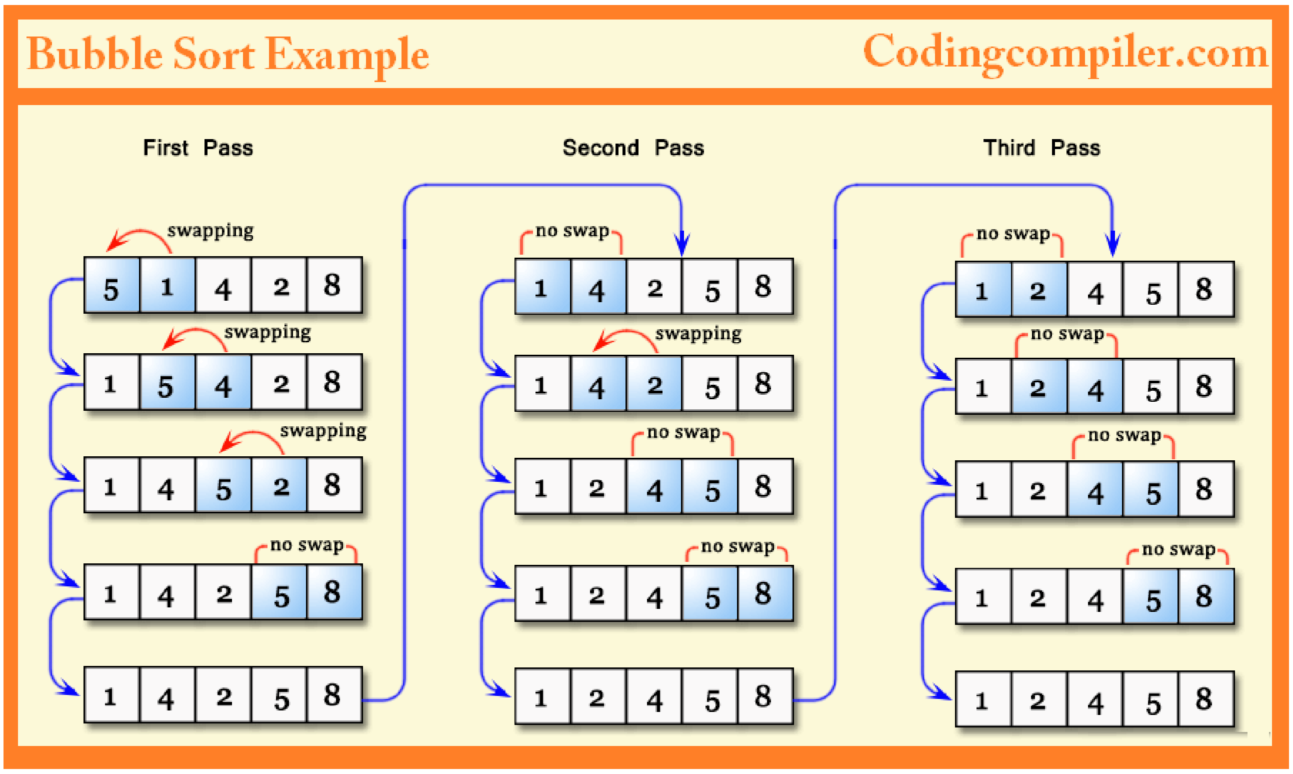
```
#include <stdio.h>
int main()
{
int c, first, last, middle, n, search, array[100];
printf("Enter number of elements\n");
scanf("%d", &n);
printf("Enter %d integers\n", n);
for (c = 0; c < n; c++)
scanf("%d", &array[c]);
printf("Enter value to find\n");
scanf("%d", &search);
first = 0;
last = n - 1;
middle = (first+last)/2;
while (first <= last)
{
if (array[middle] < search)
 first = middle + 1;
else if (array[middle] == search)
{
 printf("%d found at location %d.\n", search, middle+1);
 break;
}
else
last = middle - 1;
middle = (first + last)/2;
}
if (first > last)
printf("Not found! %d isn't present in the list.\n", search);
return 0;
}
```

----

3. Write and execute C program to read an array of elements and sort the elements in descending order using *bubble sort* and display the sorted elements

Bubble Sort is a simplesorting algorithmthat repeatedly steps through the list to be sorted, compares each pair of adjacent items andswapsthem if they are in the wrong order.

1. loop: Starting from the first index, compare the first and the second elements.
   a) If the first element is greaterthan the second element, they are swapped.
   b) Then compare the second and the third elements. Swap them if they are not in order.
   c) The above process goes on until the last element.
2. After each iteration, the largest element among the unsorted elements is placed at the end.
   Array is sorted when loop reach end of iteration.

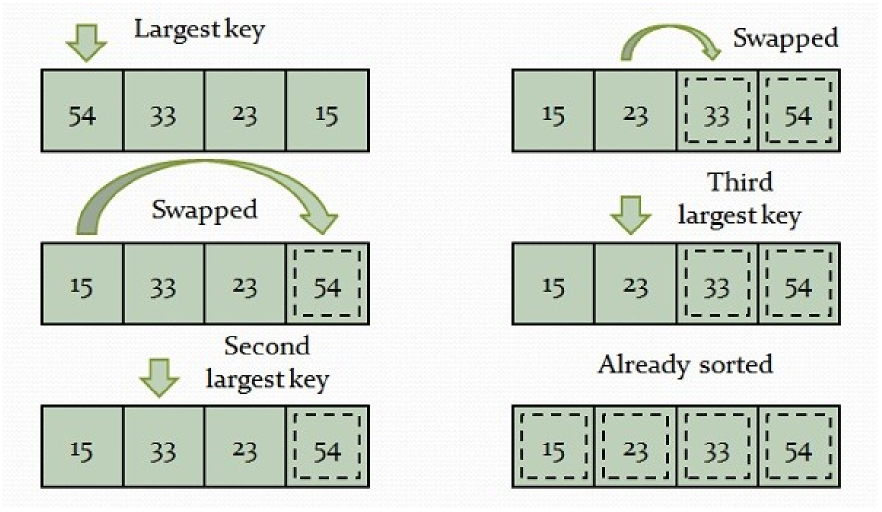


```
#include<stdio.h>
int main(){
int array[50], n, i, j, temp;
printf("Enter number of elements\n");
scanf("%d", &n);
printf("Enter %d integers\n", n);
for(i = 0; i < n; i++)
scanf("%d", &array[i]);
for (i = 0 ; i < ( n - 1 ); i++){
        for (j= 0 ; j < n - i - 1; j++){
                if(array[j] < array[j+1]){
                temp=array[j];
                array[j]   = array[j+1];
                array[j+1] = temp;
                }
        }
}
printf("Sorted list in descending order:\n");
for ( i = 0 ; i < n ; i++ )
        printf("%d\n", array[i]);
return 0;
}
```

4. Write and execute C program to read an array of elements and sort the elements in ascending order using *selection sort* and display the sorted elements.

Selection sort is another simple sorting algorithm. It is a stable sorting algorithm. In bubble sort, we find that in every pass, if adjacent elements are not in order, they are swapped.

Selection sort provides an improvement over bubble sort, with one swapping in every pass. In every pass, it finds out the largest or the smallest element and puts it in the right position



```
#include <stdio.h>
int main()
{
int array[100], n, c, d, position, t;

printf("Enter number of elements\n");
scanf("%d", &n);

printf("Enter %d integers\n", n);

for (c = 0; c < n; c++)
scanf("%d", &array[c]);

for (c = 0; c < (n - 1); c++)

{
  position = c;

 for (d = c + 1; d < n; d++)
  {
   if (array[position] > array[d])
    position = d;
  }
if (position != c)
  {
   t = array[c];
   array[c] = array[position];
   array[position] = t;
  }
 }

printf("Sorted list in ascending order:\n");

 for (c = 0; c < n; c++)
  printf("%d\n", array[c]);

 return 0;
}
```