



5.1.1. Understanding Linear search

01:52

Linear search is a searching technique in which it **sequentially** checks each element of the list for the **target value** until a match is found (or) until all the elements have been searched.

The working procedure for **linear search** is as follows:

1. Let us consider an array of **n** elements and a **key element** which is going to be search in the list of elements.
2. Compare the **key element** with the first element **a[0]**, if it is **matched** then stop the process and print the **Index** of the key element where it is found, otherwise **repeat** the same process with **a[1]**.
3. Compare the **key element** with the second element **a[1]**, if it is **matched** then stop the process and print the **Index** of the key element where it is found, otherwise **repeat** the same process with **a[2]**.
4. Continue this process until a match is found (or) until all the elements have been searched.

Let us consider an example of array numbers "50 20 40 10 80", and the key element is to find is 10.

```

Search - 1 :
Compare 10 with value of a[0] i.e., 50, both are not eq
Search - 2 :
Compare 10 with value of a[1] i.e., 20, both are not eq
Search - 3 :
Compare 10 with value of a[2] i.e., 40, both are not eq
Search - 4 :
Compare 10 with value of a[3] i.e., 10, both are equal

```

Fill in the missing code so that it produces the desired output.

Sample Test Cases

Question Hints

LinearSear...

Submit

```

1 #include<stdio.h>
2 void main() {
3     int a[20], i, n, key, flag = 0, pos;
4     printf("Enter value of n: ");
5     scanf("%d", &n);
6     for (i=0; i<n; i++) { //Complete the code in for
7         printf("Enter element for a[%d]: ", i);
8         scanf("%d", &a[i]);
9     }
10    printf("Enter key element: ");
11    scanf("%d", &key);
12    for (i=0; i<n; i++) { //Complete the code in for
13        if (key==a[i]) { //Write the condition part
14            flag = 1; // Complete the statement
15            pos = i; // Complete the statement
16            break;
17        }
18    }
19    if (flag==1) { //Write the condition part
20        printf("The key element %d is found at the position %d\n", key, pos);
21    } else {
22        printf("The key element %d is not found in the array\n", key);
23    }
24 }

```

Debugger

Plots

Prev

Reset

Submit

Next



5.1.2. Write a C program to Search a Key ele...
04:36

Write a program to search a **key element** with in the given array of elements using **linear search** process.

At the time of execution, the program should print the message on the console as:

Enter value of n :

For example, if the user gives the **Input** as:

Enter value of n : 3

Next, the program should print the messages one by one on the console as:

Enter element for a[0] :
Enter element for a[1] :
Enter element for a[2] :

if the user gives the **input** as:

Enter element for a[0] : 89
Enter element for a[1] : 33
Enter element for a[2] : 56

Next, the program should print the message on the console as:

Enter key element :

if the user gives the **input** as:

Enter key element : 56

then the program should **print** the result as:

The key element 56 is found at the position 2

Similarly if the key element is given as **25** for the above one dimensional array elements then the program should print the output as **"The key element 25 is not found in the array"**.

Fill in the missing code so that it produces the desired result.

LinearSear...

Submit

```

1 #include<stdio.h>
2 void main() {
3     int a[20], i, n, key, flag = 0,
      pos;
4     printf("Enter value of n : ");
5     scanf("%d", &n);
6     // Write code to read array
      elements
7     for(i=0;i<n;i++){
8         printf("Enter element for a[%d]
          : ",i);
9         scanf("%d",&a[i]);}
10
11
12     printf("Enter key element : ");
13     scanf("%d", &key);
14     // Write code for linear search
      process
15     for(i=0;i<n;i++){
16         if(key==a[i]){
17             flag=1;
18             pos=i;
19             break;
20         }
21     }
22
23     if (flag==1) { //Write the
      condition part
24         printf("The key element %d
          is found at the position
          %d\n",key,pos); //Complete the
          statement
25     } else {
26         printf("The key element %d
          is not found in the array\n",key);
          //Complete the statement
27     }
28 }

```

Debugger

Sample Test Cases

+


< Prev




Reset

Submit

Next >



**GGU**
GUJARATI GURUKUL
UNIVERSITY

5.1.3. Write a Program to Search an element ... 09:37   



Write a program to search the given element from a list of elements with linear search technique using recursion.

Note: Write the functions `read1()` and `linearSearch()` in Program911a.c

Sample Test Cases +

Program911a.c

```
1 #include <stdio.h>
2 #include "Program911a.c"
3 void main() {
4     int a[20], n, pos, key;
5     printf("Enter n value : ");
6     scanf("%d", &n);
7     read1(a, n);
8     printf("Enter a key element : ");
9     scanf("%d", &key);
10    pos = linearSearch(a, 0, n - 1, key);
11    if (pos == -1) {
12        printf("The key element %d is not found\n", key);
13    } else {
14        printf("The key element %d is found at position : %d\n", key, pos);
15    }
16 }
```

< Prev

Reset

Submit

Next >



5.1.4. Understanding Binary search

08:30

Binary search is faster than linear search, as it uses divide and conquer technique and it works on the sorted list either in ascending or descending order.

Binary search compares the key element to the middle element of the array; if they are unequal, the half in which the key element cannot lie is eliminated and the search continues on the remaining half until it is successful.

The working procedure for binary search is as follows:

1. Let us consider an array of n elements and a key element which is going to be search in the list of elements.
2. The main principle of binary search is first divide the list of elements into two halves.
3. Compare the key element with the middle element.
4. If the comparison result is true the print the Index position where the key element has found and stop the process.
5. If the key element is greater than the middle element then search the key element in the second half.
6. If the key element is less than the middle element then search the key element in the first half.
7. Repeat the same process for the sub lists depending upon whether key is in the first half or second half of the list until a match is found (or) until all the elements in that half have been searched.

Let us consider an example of array numbers "50 20 40 10 80", and the key element is to find is 10.

Search - 1 :
First Sort the given array elements by using any one of After sorting the elements in the array are 10 20 40 50

Search - 2 :
Compare 10 with middle element i.e., $(low + high) / 2 =$ Here $10 < 40$ so search the element in the left half of

Search - 3 :
Compare 10 with middle element i.e., $(low + high) / 2 =$ Here $10 == 10$ so print the index 0 where the element h

Click on [Live Demo](#) to understand the working of binary search process.

Fill in the missing code so that it produces the desired output.

Sample Test Cases

BinarySear...

Submit

```

1 #include<stdio.h>
2 void main() {
3     int a[20], i, j, n, key, flag = 0, low, high, mid, temp;
4     printf("Enter value of n : ");
5     scanf("%d", &n);
6     for (i=0; i<n; i++) { // Complete the code in for
7         printf("Enter element for a[%d] : ", i);
8         scanf("%d", &a[i]); // Complete the statement
9     }
10    printf("Enter key element : ");
11    scanf("%d", &key);
12    // Bubble sort process
13
14    for (i=0; i<n; i++) { // Complete the code in for
15        for (j=0; j<(n-i-1); j++) { // Complete the code in for
16            if (a[j]>a[j+1]) { // Write the condition part
17                temp = a[j]; // Complete the statement
18                a[j] = a[j+1]; // Complete the statement
19                a[j+1] = temp; // Complete the statement
20            }
21        }
22    }
23    printf("After sorting the elements in the array are\n");
24    for (i=0; i<n; i++) { // Complete the code in for
25        printf("Value of a[%d] = %d\n", i, a[i]);
26    }
27    low = 0; // Complete the statement
28    high = n-1; // Complete the statement
29    while (low <= high) { // Complete the condition part in while
30        mid = (low+high)/2; // Complete the statement
31        if (a[mid] == key) { // Write the condition part
32            flag = 1; // Complete the statement
33            break;
34        } else if (a[mid] < key) { // Write the condition part
35            low = mid+1; // Complete the statement
36        } else if (a[mid] > key) { // Write the condition part
37            high = mid-1; // Complete the statement
38        }
39    }
40    if (flag == 1) { // Write the condition part
41        printf("The key element %d is found at the position %d\n", key, mid);
42    } else {
43        printf("The Key element %d is not found in the array\n", key);
44    }
45 }

```

Prev

Reset

Submit

Next

5.1.5. Write a C program to Search a Key ele...

Write a program to **search** a key element in the given array of elements using **binary search**.

At the time of execution, the program should print the message on the console as:

Enter value of n :

For example, if the user gives the **input** as:

Enter value of n : 3

Next, the program should print the messages one by one on the console as:

```
Enter element for a[0] :
Enter element for a[1] :
Enter element for a[2] :
```

if the user gives the **input** as:

```
Enter element for a[0] : 89
Enter element for a[1] : 33
Enter element for a[2] : 56
```

Next, the program should print the message on the console as:

Enter key element :

if the user gives the **input** as:

Enter key element : 56

then the program should **print** the result as:

```
After sorting the elements in the array are
Value of a[0] = 33
Value of a[1] = 56
Value of a[2] = 89
The key element 56 is found at the position 1
```

Similarly if the key element is given as **25** for the above one dimensional array elements then the program should print the output as **"The Key element 25 Is not found in the array"**.

Fill in the missing code so that it produces the desired result.

Sample Test Cases

+

 BinarySear...

Submit

er

```

1 #include<stdio.h>
2 void main() {
3     int a[20], i, j, n, key, flag =
4     0, low, high, mid, temp;
5     printf("Enter value of n : ");
6     scanf("%d", &n);
7     for(i=0; i<n; i++){
8         printf("Enter element for a[%d]
9         : ", i);
10        scanf("%d", &a[i]);
11    }
12    // Write the code to read an
13    array of elements
14
15    printf("Enter key element : ");
16    scanf("%d", &key);
17
18    // Write the code to sort the
19    elements using any sorting technique
20    for(i=0; i<n; i++){
21        for(j=0; j<(n-1-i); j++){
22            if(a[j]>a[j+1]){
23                temp=a[j];
24                a[j]=a[j+1];
25                a[j+1]=temp;
26            }
27        }
28    }
29    printf("After sorting the
30    elements in the array are\n");
31    // Write the code to display the
32    elements
33    for(i=0; i<n; i++){
34        printf("Value of a[%d] =
35        %d\n", i, a[i]);
36    }
37    low = 0; // Complete the
38    statement
39    high = n-1; // Complete the
40    statement
41    while(low<=high){
42        mid=(low+high)/2;
43        if(a[mid]==key){
44            flag=1;
45            temp=mid;
46            break;
47        }
48        else if(a[mid]>key){
49            high=mid-1;
50        }
51        else if(a[mid]<key){
52            low=mid+1;
53        }
54    }
55    // Write the code to search an
56    element using binary search process
57    if ( flag==1) { // Write the
58    condition part
59        printf("The key element %d
60        is found at the position
61        %d\n", key, temp); // Complete the
62        statement
63    } else {
64        printf("The Key element %d
65        is not found in the array\n", key
66        ); // Complete the statement
67    }
68 }

```

[← Prev](#)
[Reset](#)
[Submit](#)
[Next →](#)



2.5.1. Understanding Bubble sort

Bubble sort is an internal sorting technique in which adjacent elements are compared and exchanged if necessary.

The working procedure for bubble sort is as follows:

- Let us consider an array of n elements (i.e., $a[n]$) to be sorted.
- Compare the first two elements in the array i.e., $a[0]$ and $a[1]$, if $a[1]$ is less than $a[0]$ then interchange the two values.
- Next compare $a[1]$ and $a[2]$, if $a[2]$ is less than $a[1]$ then interchange the values.
- Continue this process till the last two elements are compared and interchanged.
- Repeat the above steps for $n-1$ passes.

Let us consider an example of array numbers "50 20 40 10 80", and sort the array from lowest number to greatest number using bubble sort.

In each step, elements written in **bold** are being compared. Number of elements in the array are 5, so 4 passes will be required.

Pass - 1 :

```
( 50 20 40 10 80 ) -> ( 20 50 40 10 80 ) // Compared the
( 20 50 40 10 80 ) -> ( 20 40 50 10 80 ) // Swap since
( 20 40 50 10 80 ) -> ( 20 40 10 50 80 ) // Swap since
( 20 40 10 50 80 ) -> ( 20 40 10 50 80 ) // Since the
```

Total number of elements in the given array are 5, so in Pass - 1 total numbers compared are 4. After completion of Pass - 1 the largest element is moved to the last position of the array.

Now, Pass - 2 can compare the elements of the array from first position to second last position.

Pass - 2 :

```
( 20 40 10 50 80 ) -> ( 20 40 10 50 80 ) // Since the
( 20 40 10 50 80 ) -> ( 20 10 40 50 80 ) // Swap since
( 20 10 40 50 80 ) -> ( 20 10 40 50 80 ) // Since the
```

In Pass - 2 total numbers compared are 3. After completion of Pass - 2 the second largest element is moved to the second last position of the array.

Now, Pass - 3 can compare the elements of the array from first position to third last position.

Pass - 3 :

```
( 20 10 40 50 80 ) -> ( 10 20 40 50 80 ) // Swap since
( 10 20 40 50 80 ) -> ( 10 20 40 50 80 ) // Since the
```

In Pass - 3 total numbers compared are 2. After completion of Pass - 3 the third largest element is moved to the third last position of the array.

Now, Pass - 4 can compare the first and second elements of the array.

Pass - 4 :

```
( 10 20 40 50 80 ) -> ( 10 20 40 50 80 ) // Since these
```

In Pass - 4 total numbers compared are 1. After completion of Pass - 4 all the elements of the array are sorted. So, the result is 10 20 40 50 80.

Click on [Live Demo](#) to understand the working of bubble sort technique.

Fill in the missing code so that it produces the desired result.

Sample Test Cases

+

BubbleSort...

Submit

```

1 #include<stdio.h>
2 void main() {
3     int a[20], i, n, j, temp;
4     printf("Enter value of n : ");
5     scanf("%d", &n);
6     for (i=0; i<n; i++) { // Complete
7         the code in for
8         printf("Enter element for
9         a[%d] : ", i);
10        scanf("%d", &a[i]); //
11        Complete the statement
12    }
13    printf("Before sorting the
14    elements in the array are\n");
15    for (i=0; i<n; i++) { // Complete
16        the code in for
17        for (j=0; j<(n-i-1); j++) {
18            //Complete the code in for
19            if ( a[j]>a[j+1]) { //
20                Complete the condition part
21                temp=a[j];
22                a[j]=a[j+1];
23                a[j+1]=temp;
24                // Complete the
25                statements to swap elements
26            }
27        }
28    }

```

Explorer

Plots

Debugger

Prev

Reset

Submit

Next >



GGU
GUJARATI GATEWAY
UNIVERSITY

5.2.2. Write a C program to Sort the elements... 05:02

Write a program to **sort** the given elements using **bubble sort technique**.

At the time of execution, the program should print the message on the console as:

Enter value of n :

For example, if the user gives the **Input** as:

Enter value of n : 3

Next, the program should print the messages one by one on the console as:

Enter element for a[0] :
Enter element for a[1] :
Enter element for a[2] :

if the user gives the **input** as:

Enter element for a[0] : 22
Enter element for a[1] : 33
Enter element for a[2] : 12

then the program should **print** the result as:

Before sorting the elements in the array are
Value of a[0] = 22
Value of a[1] = 33
Value of a[2] = 12
After sorting the elements in the array are
Value of a[0] = 12
Value of a[1] = 22
Value of a[2] = 33

Fill in the missing code so that it produces the desired result.

Sample Test Cases +

Question Hints +

BubbleSort...

Submit

```
1 #include<stdio.h>
2 void main() {
3     int a[20], i, n, j, temp;
4     printf("Enter value of n : ");
5     scanf("%d", &n);
6     // Write the for loop to read
    array elements
7     for(i=0;i<n;i++){
8         printf("Enter element for
a[%d] :- ", i);
9         scanf("%d", &a[i]);
10    }
11
12    printf("Before sorting the
elements in the array are\n");
13    // Write the for loop to display
array elements before sorting
14    for(i=0;i<n;i++){
15        printf("Value of a[%d] =
%d\n", i, a[i]);
16    }
17
18    //Write the code to sort elements
19    for(i=0;i<n;i++){
20        for(j=0;j<(n-i-1);j++){
21            if(a[j]>a[j+1]){
22                temp=a[j];
23                a[j]=a[j+1];
24                a[j+1]=temp;
25            }
26        }
27    }
28
29    printf("After sorting the
elements in the array are\n");
30    // Write the for loop to display
array elements after sorting
31    for(i=0;i<n;i++){
32        printf("Value of a[%d] =
%d\n", i, a[i]);
33    }
34
35 }
```

Debugger

Plots

Prev

Reset

Submit

Next



5.2.3. Understanding Insertion sort

05:02

Insertion sort is one that **sorts** a set of elements by inserting an element into the existing sorted elements.

The working procedure for **Insertion sort** is as follows:

1. Let us consider an array of **n** elements (i.e., **a[n]**) to be sorted.
2. The **first element a[0]** in the array is itself trivially sorted.
3. The **second element a[1]** is compared with first element **a[0]** and it will be inserted either before or after first element, so that **first** and **second** elements are sorted.
4. The **third element a[2]** is compared with **a[0]** and **a[1]** and it will be inserted into its proper place by checking conditions, so that first three elements are sorted.
5. Repeat the same process for **n - 1** passes.

Let us consider an example of array numbers "50 20 40 10 30", and sort the array from **lowest number to greatest number** using insertion sort.

In each step, elements written in **color** is compared with elements written in **bold**. Number of elements in the array are 5, so 4 passes will be required.

Pass - 1 :

(50 20 40 10 30) -> (20 50 40 10 30) // The second

Now, **Pass - 2** can compare **a[2]** with **a[0]** and **a[1]**.

Pass - 2 :

(20 50 40 10 30) -> (20 40 50 10 30) // Since 40 >

Now, **Pass - 3** can compare **a[3]** with **a[0]**, **a[1]** and **a[2]**.

Pass - 3 :

(20 40 50 10 30) -> (10 20 40 50 30) // Since 10 <

Now, **Pass - 4** can compare **a[4]** with **a[0]**, **a[1]**, **a[2]** and **a[3]**.

Pass - 4 :

(10 20 40 50 30) -> (10 20 30 40 50) // Since 30 >

After completion of **Pass - 4** all the elements of the array are **sorted**. So, the result is **10 20 30 40 50**.

Click on [Live Demo](#) to understand the working of [insertion sort technique](#).

Fill in the missing code so that it produces the desired output.

Sample Test Cases

+

Explorer

InsertionS...

Submit

```

1 #include<stdio.h>
2 void main() {
3     int a[20], i, n, j, temp;
4     printf("Enter value of n : ");
5     scanf("%d", &n);
6     for (i=0; i<n; i++) { // Complete
7         the code in for
8         printf("Enter element for
9         a[%d] : ", i);
10        scanf("%d", &a[i]); //
11        Complete the statement
12    }
13    printf("Before sorting the
14    elements in the array are\n");
15    for (i=0; i<n; i++) { // Complete
16        the code in for
17        printf("Value of a[%d] =
18        %d\n", i, a[i]); // Complete the
19        statement
20    }
21    for (i=1; i<n; i++) { // Complete
22        the code in for
23        temp = a[i]; // Complete the
24        statement
25        j = i - 1; // Complete the code
26        in for
27        while (j >= 0 && a[j] > temp) {
28            // Complete the condition part
29            a[j+1] = a[j];
30            j = j - 1;
31            // Complete the
32            statements to swap elements
33        }
34        a[j+1] = temp;
35    }
36    printf("After sorting the
37    elements in the array are\n");
38    for (i=0; i<n; i++) { // Complete
39        the code in for
40        printf("Value of a[%d] =
41        %d\n", i, a[i]); // Complete the
42        statement
43    }
44 }
```

Debugger

Plots

< Prev

Reset

Submit

Next >



5.2.4. C program to Sort the elements using I... 06:59

Write a program to sort the given elements using Insertion sort technique.

Fill in the missing code so that it produces the desired result.

Explorer

C InsertionS...

Submit

Debugger

```
1 #include<stdio.h>
2 void main() {
3     int a[20], i, n, j, temp, pos;
4     printf("Enter value of n : ");
5     scanf("%d", &n);
6     // write the for loop to read
    array elements
7     for(i=0;i<n;i++){
8         printf("Enter element for
a[%d] : ", i);
9         scanf("%d", &a[i]);
10    }
11    // write the for loop to display
    array elements before sorting
12    printf("Before sorting the
elements in the array are\n");
13    for(i=0;i<n;i++){
14        printf("Value of a[%d] =
%d\n", i, a[i]);
15    }
16    for(i=1;i<n;i++){
17        temp=a[i];
18        j=i-1;
19        while(j>=0&&a[j]>temp){
20            a[j+1]=a[j];
21            j=j-1;
22        }
23        a[j+1]=temp;
24    }
25    // write the code to sort
    elements
26
27    // write the for loop to display
    array elements after sorting
28    printf("After sorting the
elements in the array are\n");
29    for(i=0;i<n;i++){
30        printf("Value of a[%d] =
%d\n", i, a[i]);
31    }
32
33 }
```

Sample Test Cases





5.2.5. Selection sort - largest element method 07:27

Selection sort process can be done in **two** ways, one is the largest element method and the other is smallest element method.

The working procedure for selection sort using the **largest element method** is as follows:

1. Let us consider an array of **n** elements (i.e., **a[n]**) to be sorted.
2. In the first step, the **largest element** in the list is searched. Once the largest element is found, it is exchanged with the element which is placed at the **last position**. This completes the first pass.
3. In the next step, it searches for the **second largest element** in the list and it is interchanged with the element placed at **second last position**. This is done in second pass.
4. This process is repeated for **n - 1** passes to sort all the elements.

Let us consider an example of array numbers "80 10 50 20 40", and sort the array from **lowest number to greatest number** using selection sort by the largest element.

Pass - 1 :

(80 10 50 20 40) -> (40 10 50 20 80) // First finds

After completion of **Pass - 1**, the largest element is moved to the end of the array.

Now, **Pass - 2** can find the next largest element with out considering the last position element.

Pass - 2 :

(40 10 50 20 80) -> (40 10 20 50 80) // Largest in

After completion of **Pass - 2** the **second largest element** is moved to the **second last position** of the array.

Now, **Pass - 3** can find the next largest element with out considering the **last two position elements** because they are already sorted.

Pass - 3 :

(40 10 20 50 80) -> (20 10 40 50 80) // Largest in

After completion of **Pass - 3** the **third largest element** is moved to the **third last position** of the array.

Now, **Pass - 4** can find the next largest element with out considering the **last three position elements** because they are already sorted.

Pass - 4 :

(20 10 40 50 80) -> (10 20 40 50 80) // Largest in

After completion of **Pass - 4** all the elements of the array are **sorted**. So, the result is **10 20 40 50 80**.

Write a program to sort the elements using selection sort largest element method.

Input format:
The first line contains an integer n, size of the array.
The second line contains n space - separated array of integers.

Output format:
Print the elements of the sorted array.

Sample Test Cases

+

CTC35763.c

Submit

```

1  #include<stdio.h>
2  int main(){
3      int a[10],i,n,pos,j,temp;
4      scanf("%d",&n);
5      for(i=0;i<n;i++){
6          scanf("%d",&a[i]);
7      }
8      for(i=0;i<n-1;i++){
9          pos=i;
10         for(j=i+1;j<n;j++){
11             if(a[pos]>a[j]){
12                 pos=j;
13             }
14         }
15         if(pos!=i){
16             temp=a[i];
17             a[i]=a[pos];
18             a[pos]=temp;
19         }
20     }
21     for(i=0;i<n;i++){
22         printf("%d ",a[i]);
23     }
24     printf("\n");
25 }

```

Explorer

Plots

Debugger

Run

Reset

Submit

Next

5.3.4. Sort E-Books based on IDs using Quick...

You are developing a program for a digital library to manage its collection of e-books. Each e-book is identified by a unique Book ID. To optimize the process of listing e-books, you need to implement a sorting algorithm that arranges the e-books based on their Book ID's.

Your task is to design a C program to sort the e-books based on their Book ID's using the quick sort algorithm.

Constraints:

- The digital library manages a maximum of 1000 different e-books.

Input Format:

- The first line contains an integer, representing the total number of e-books.
- The next line contains space-separated integers, representing the Book ID's of the respective e-books.

Output Format:

- The first line should display the original array of Book ID's.
- The second line should display the sorted array of Book ID's.

Sample Test Cases

+

 BookCollec...

Submit

er

```

1 #include <stdio.h>
2 #include <stdlib.h>
3
4 // Quicksort partition function
5 int partition(int arr[], int low,
6 int high) {
7     int pivot=arr[high];
8     int i=low-1;
9     for(int j=low; j<=high-1; j++){
10         if(arr[j]<pivot){
11             i++;
12             int temp=arr[j];
13             arr[j]=arr[i];
14             arr[i]=temp;
15         }
16     }
17     int temp=arr[high];
18     arr[high]=arr[i+1];
19     arr[i+1]=temp;
20     return (i+1);
21 }
22
23 void swap(int* a,int* b){
24     int t=*a;
25     *a=*b;
26     *b=t;
27 }
28
29 // Quicksort function
30 void quickSort(int arr[], int low,
31 int high) {
32     // write your code here
33     if(low<high){
34         int pi=partition(arr, low, high);
35         quickSort(arr, low, pi-1);
36         quickSort(arr, pi+1, high);
37     }
38 }
39
40 // Function to print an array
41 void printArray(int arr[], int n) {
42     for (int i = 0; i < n; i++) {
43         printf("%d ", arr[i]);
44     }
45 }
46
47 int main() {
48     int n;
49     printf("no of e-books: ");
50     scanf("%d", &n);
51
52     int BID[n];
53     printf("Book ID's of e-books: ");
54     for (int i = 0; i < n; i++) {
55         scanf("%d", &BID[i]);
56     }
57
58     printf("Original Book ID's: ");
59     printArray(BID, n);
60     printf("\n");
61
62     // Sort the e-books based on
63     // their BIDs using quicksort
64     quickSort(BID, 0, n - 1);
65
66     printf("Sorted Book ID's: ");
67     printArray(BID, n);
68     printf("\n");
69
70     return 0;
71 }

```



GGU
GURU GURU
UNIVERSITY

5.4.2. Sort the Array using Selection Sort 04:37

Question description:

Sajid is a third-year student at a reputed institution. Although he scored well in many subjects, he is not an expert in computer programming languages. Sajid's computer examination is scheduled for the next week. As per the blueprint, many questions would come from the sorting topic.

He collected the previous year's questions and one of the repeated questions is to sort the given set of numbers using **Selection Sort**.

You are given a list A with N positive integers, Your task is to sort the given list using selection sort.

Can you write the code and help Sajid learn it?

Constraints:

- $1 \leq N \leq 10^5$
- $1 \leq A_i \leq 10^9$

Input format:

- The first line of the input contains the number of elements.
- The second line of the input contains the numbers to be sorted, separated by space.

Output format:

- Print the sorted list of integers separated with a space character

Sample Test Cases +

selectionS...

Submit

```
1 #include <stdio.h>
2
3 // Type Content here...
4 void selectionSort(int arr[], int n){
5     int i, j, minIndex, temp;
6     for(i=0; i<n-1; i++){
7         minIndex=i;
8         for(j=i+1; j<n; j++){
9             if(arr[j]<arr[minIndex]){
10                 minIndex=j;
11             }
12         }
13         if(minIndex!=i){
14             temp=arr[i];
15             arr[i]=arr[minIndex];
16             arr[minIndex]=temp;
17         }
18     }
19 }
20
21
22
23 int main() {
24     int n;
25     // Input: Number of elements in
26     // the array
27     scanf("%d", &n);
28
29     int arr[n], i;
30     // Input: Elements of the array
31     for (i = 0; i < n; i++) {
32         scanf("%d", &arr[i]);
33     }
34
35     // Call the selectionSort
36     // function to sort the array
37     selectionSort(arr, n);
38
39     // Output: Sorted array
40     for (i = 0; i < n; i++) {
41         printf("%d ", arr[i]);
42     }
43     printf("\n");
44     return 0;
45 }
```

Debugger

Prev

Reset

Submit

Next >