Short-term Hands-on Supplementary Course on C programming

Session 5:
Array Operations

Nandakishor. V
Nitheesh Kumar. N

Agenda

- Traverse
- Search
- Insertion
- Deletion
- Update
- Sort
- Live Code Demo - selection sort
- Tutorial

Traversing through an array

```
int balance[5] = {1000, 2, 3, 7, 50};
```

For loop-

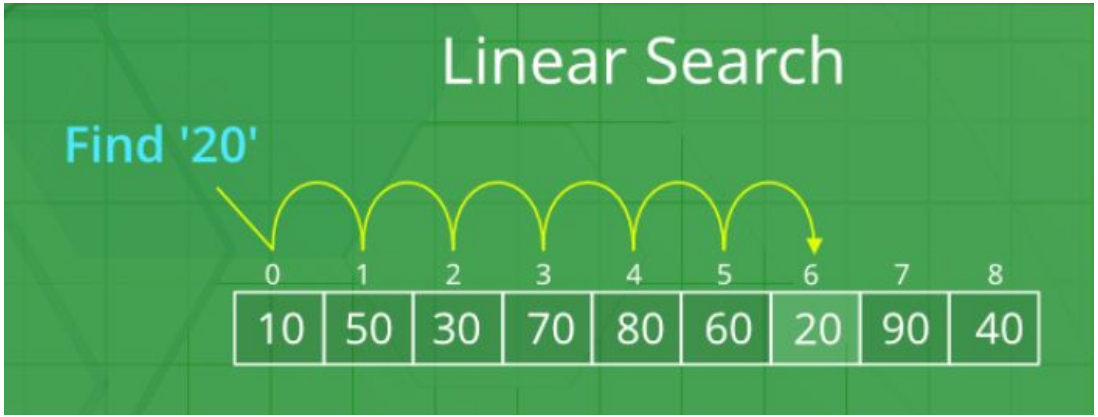
```
for(int i=0;i<5;i++){
  printf("Element %d-%.1lf\n",i,balance[i]);
}
```

While loop-

```
int j=0;
while(j<5){
  printf("Element %d-%.1lf\n",j,balance[j]);
  j++;
}
```

- Starts from 0th index
- i is incremented
- Prints ith element in array
- Till i exceeds the no. of elements in array.

searching - working



- Starts from 0th indexed element.
- Checks every element in array till element is found or there are no elements to be searched in array.

Searching-Linear Search

```
int balance[5] = {1000, 2, 3, 7, 50};
```

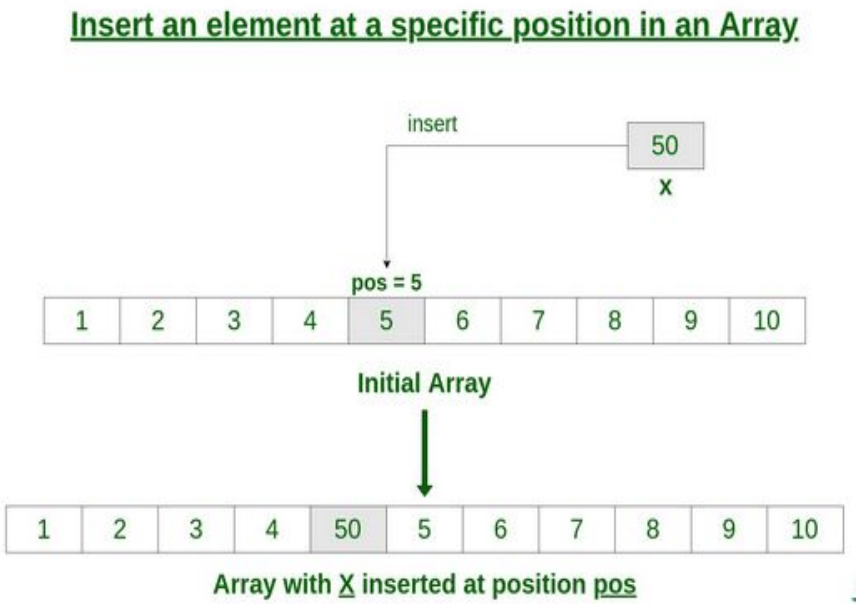
```
int i,ele=7;
for(i=0;i<5;i++){
  if(balance[i]==ele){
    printf("Found in %d\n",i);
    break;
  }
}
if(i==5){
  printf("Not found\n");
}
```

- Linear search goes through the entire array, to find the searched element.

Live Code Demo- Linear search

1) Searching an element in an array, if found return the index of the element, else output “Not found”.

Insertion



- First get the element to be inserted, say x
- Then get the position at which this element is to be inserted, say pos
- Then shift the array elements from this position to one position forward(towards right), and do this for all the other elements next to pos.
- Insert the element x now at the position pos, as this is now empty.

Insertion

```
int LA[] = {1,3,5,7,8};
int item = 10, k = 3, n = 5,i = 0, j = n;
```

```
while( j >= k) {
    LA[j+1] = LA[j];
    j = j - 1;
}

LA[k] = item;
```

Moving all element from kth position to the right to make space of the new element

Inserting new element

Deletion

```
int LA[] = {1,3,5,7,8};
int item = 10, k = 3, n = 5,i = 0, j = n;
```

1. Start
2. Set J = K
3. Repeat steps 4 and 5 while J < N
4. Set LA[J] = LA[J + 1]
5. Set J = J+1
6. Set N = N-1
7. Stop

```
The original array elements are :
LA[0] = 1
LA[1] = 3
LA[2] = 5
LA[3] = 7
LA[4] = 8
The array elements after deletion :
LA[0] = 1
LA[1] = 3
LA[2] = 7
LA[3] = 8
```

Update

```
int LA[] = {1,3,5,7,8};
int item = 10, k = 3, n = 5,i = 0, j = n;
```

1. Start
2. Set LA[K-1] = ITEM
3. Stop

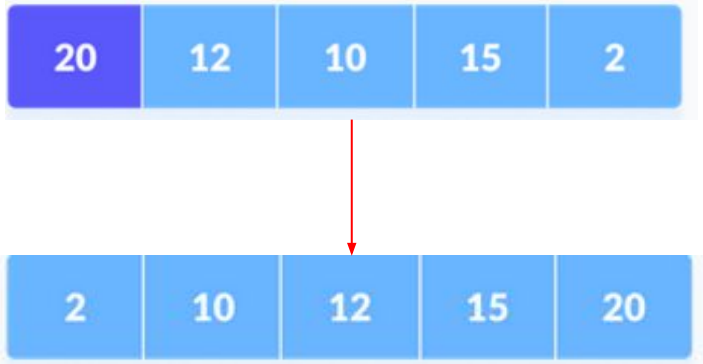
```
LA[k-1] = item;
```

Updation can be implemented directly by changing the array[i-th position]

Sorting

An unordered array can be sorted in ascending or descending order, by using many algorithms-

- Selection sort
- Bubble sort
- Insertion sort
- Quick sort
- Merge sort



Sorting - selection sort

Step 1 – Set min to the first location

Step 2 – Search the minimum element in the array

Step 3 – swap the first location with the minimum value in the array

Step 4 – assign the second element as min.

Step 5 – Repeat the process until we get a sorted array.

Sorting - selection sort

Step 1

Set the first element as minimum

Sorting - selection sort

Step 2

Compare minimum with the second element. If the second element is smaller than minimum, assign the second element as minimum.

Compare minimum with the third element. Again, if the third element is smaller, then assign minimum to the third element otherwise do nothing. The process goes on until the last element.

Sorting - selection sort

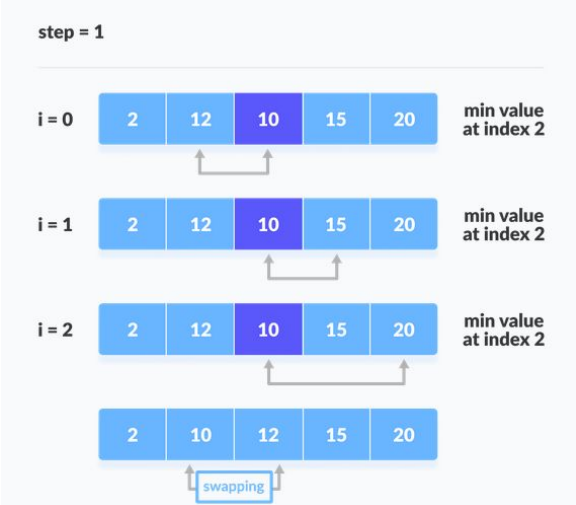
Step 3

After each iteration, minimum is placed in the front of the unsorted list.

Sorting - selection sort

Step 4

The first iteration

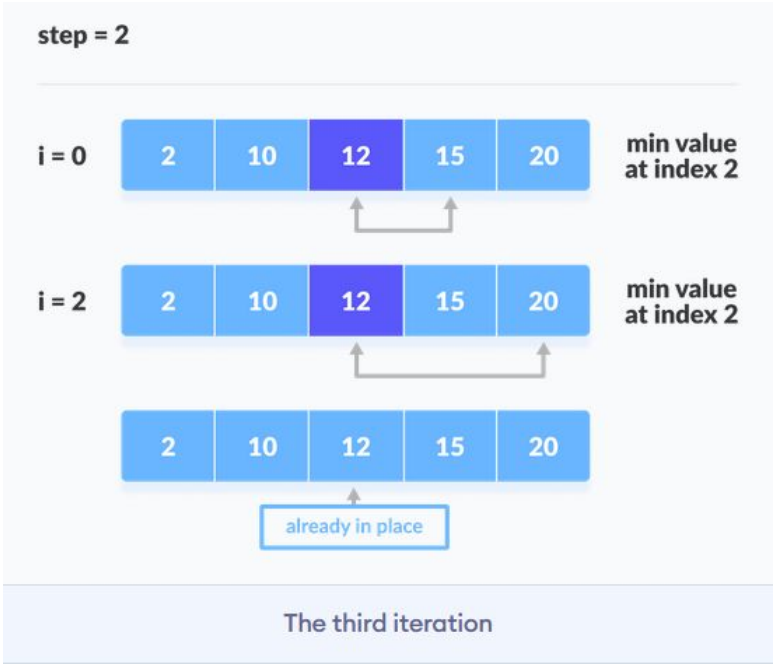


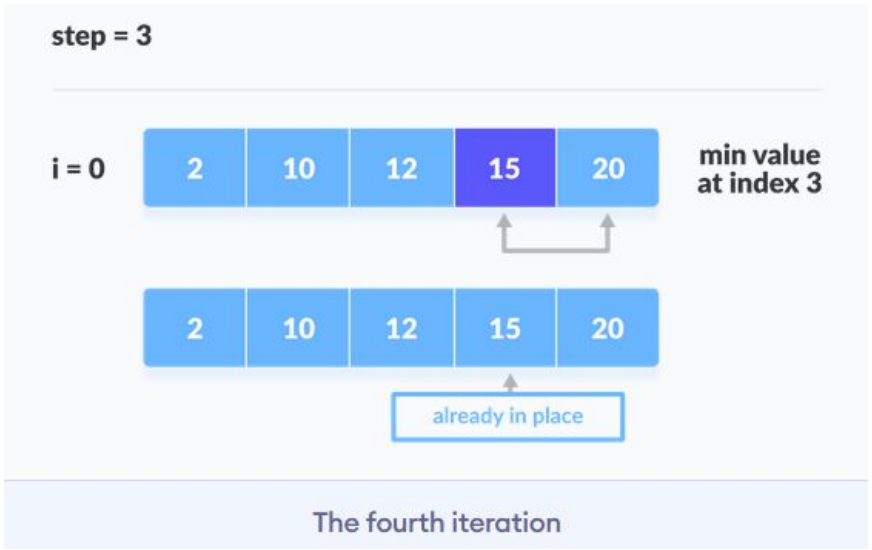
The second iteration

For each iteration, indexing starts from the first unsorted element. Step 1 to 3 are repeated until all the elements are placed at their correct positions.

Sorting - selection sort

Step 4



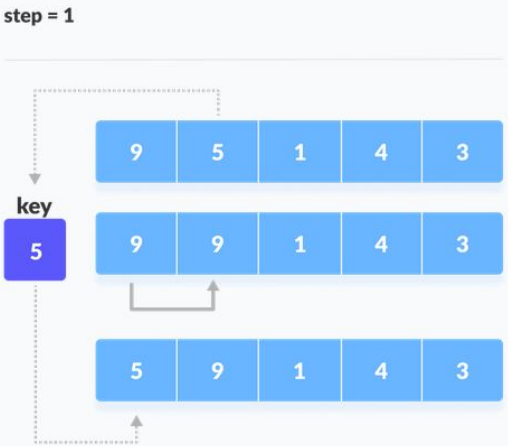


Live Code Demo - Selection sort

1) Write a C program to implement selection sort

Sorting - Insertion sort

Step 1



If the first element is greater than key, then key is placed in front of the first element.

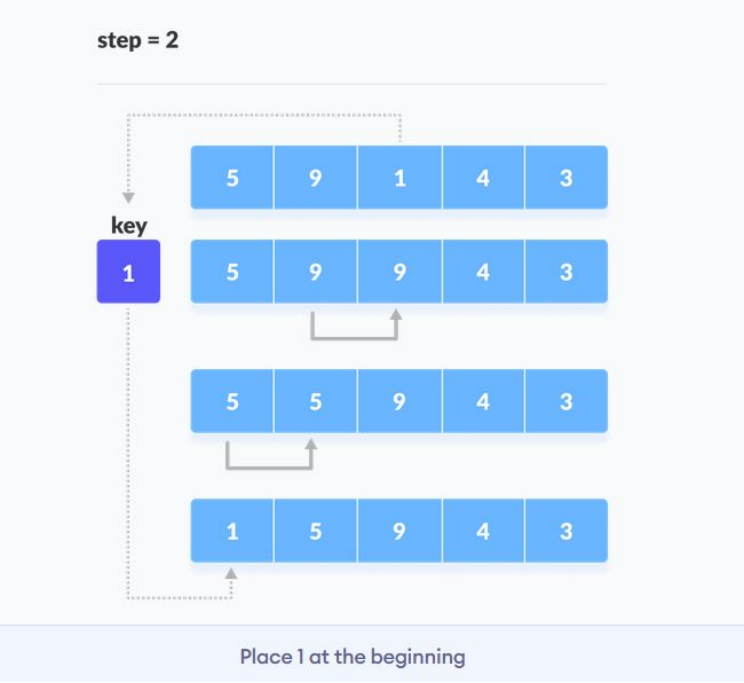
Initial array

The first element in the array is assumed to be sorted. Take the second element and store it separately in key.

Compare key with the first element. If the first element is greater than key, then key is placed in front of the first element.

Sorting - Insertion sort

Step 2

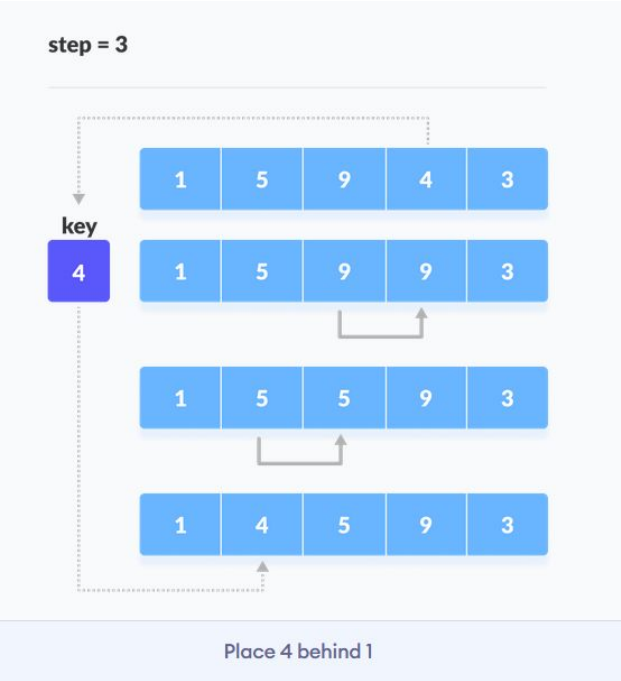


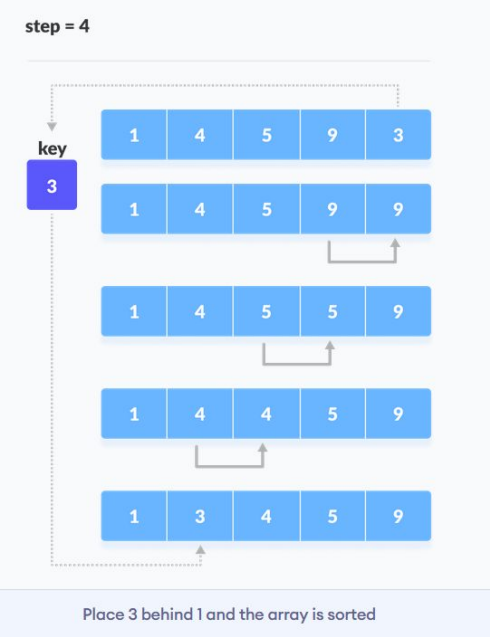
Now, the first two elements are sorted.

Take the third element and compare it with the elements on the left of it. Placed it just behind the element smaller than it. If there is no element smaller than it, then place it at the beginning of the array.

Sorting - Insertion sort

Step 3





Similarly, place every unsorted element at its correct position.

Tutorial

1) Write a C program to implement insertion sort

Thank You for attending!

Contact us regarding any questions through email

nandakishor2010608@ssn.edu.in
nitheesh2010343@ssn.edu.in