AIM Insertion Sort, Selection Sort and Bubble Sort

Mayank Gupta 20BCE1538

```
#include <iostream>

using namespace std;
void swap(int *arr, int n,int &x,int &y){
   int temp=y;
   y=x;
   x=temp;
   cout<<x<<" is swapped with "<<y<<endl;
   for(int i=0;i<n;i++){
      cout<<arr[i]<<" ";
   }
   cout<<endl;
}


void insertionSort(int *arr, int n, int s){
   int key,j;
   for(int i=1;i<n;i++){
      key = arr[i];
      j=i;
      for(int i=0;i<n;i++){
      cout<<arr[i]<<" ";
      }
      cout<<endl;
      cout<<"-> the key taken is "<<key<<endl;
      while(j>0 && arr[j-1]>key){
         cout<<"Right shift "<<arr[j-1]<<" by 1 posn"<<endl;
         arr[j]=arr[j-1];
         j--;
      }
      cout<<"place the key at the posn of "<<arr[j]<<endl;
      arr[j]=key;
   }
}
void bubbleSort(int *arr, int n){
   for(int i=0;i<n;i++){
```

```
      for(int j=0;j<n-i-1;j++){
         if(arr[j]>arr[j+1]){
            swap(arr,n,arr[j],arr[j+1]);
         }
      }
   }
}

void display(int *arr, int n){
   cout<<"Final Correct Output"<<": ";
   for(int i=0;i<n;i++){
      cout<<arr[i]<<" ";
   }
   cout<<endl;
}
//11 10 7 9 '5' 6 4 8
void selectionSort(int arr[],int n){
   int int_min;
   for(int i=0;i<n-1;i++){
      int index=0;
      int_min=999;
      for(int j=i;j<n;j++){
         if(arr[j]<int_min){
            int_min=arr[j];
            index=j;
         }
      }
   swap(arr,n,arr[index],arr[i]);
   }
}
int main()
{
   int n;
   cin>>n;
   int arr[n];
   for(int i=0;i<n;i++){
      cin>>arr[i];
   }
   int arr1[n];
   for(int i=0;i<n;i++){
      arr1[i]=arr[i];
   }
```

```
   int s=0; //no. of swaps
   cout<<"Bubble sort"<<endl;
   bubbleSort(arr1,n);
   display(arr1,n);
   for(int i=0;i<n;i++){
      arr1[i]=arr[i];
   }
   cout<<"Insertion sort"<<endl;
   insertionSort(arr1,n,s);
   display(arr1,n);
   for(int i=0;i<n;i++){
      arr1[i]=arr[i];
   }
   cout<<"Selection sort"<<endl;
   selectionSort(arr1,n);
   display(arr1,n);
   //cout<<"no. of swaps" << s;
   return 0;
}
```

Output

- **Bubble Sort Output**

```
Bubble sort
10 is swapped with 11
10 11 7 9 5 6 4 8
7 is swapped with 11
10 7 11 9 5 6 4 8
9 is swapped with 11
10 7 9 11 5 6 4 8
5 is swapped with 11
10 7 9 5 11 6 4 8
6 is swapped with 11
10 7 9 5 6 11 4 8
4 is swapped with 11
10 7 9 5 6 4 11 8
8 is swapped with 11
10 7 9 5 6 4 8 11
7 is swapped with 10
7 10 9 5 6 4 8 11
9 is swapped with 10
7 9 10 5 6 4 8 11
5 is swapped with 10
7 9 5 10 6 4 8 11
6 is swapped with 10
7 9 5 6 10 4 8 11
4 is swapped with 10
7 9 5 6 4 10 8 11
8 is swapped with 10
7 9 5 6 4 8 10 11
5 is swapped with 9
7 5 9 6 4 8 10 11
6 is swapped with 9
7 5 6 9 4 8 10 11
4 is swapped with 9
7 5 6 4 9 8 10 11
8 is swapped with 9
7 5 6 4 8 9 10 11
5 is swapped with 7
5 7 6 4 8 9 10 11
6 is swapped with 7
5 6 7 4 8 9 10 11
4 is swapped with 7
5 6 4 7 8 9 10 11
4 is swapped with 6
5 4 6 7 8 9 10 11
4 is swapped with 5
4 5 6 7 8 9 10 11
Final Correct Output: 4 5 6 7 8 9 10 11
Insertion sort
```

Press F11 to exit full screen

- **Insertion Sort Output**

```
Insertion sort
11 10 7 9 5 6 4 8
-> the key taken is 10
Right shift 11 by 1 posn
place the key at the posn of 11
10 11 7 9 5 6 4 8
-> the key taken is 7
Right shift 11 by 1 posn
Right shift 10 by 1 posn
place the key at the posn of 10
7 10 11 9 5 6 4 8
-> the key taken is 9
Right shift 11 by 1 posn
Right shift 10 by 1 posn
place the key at the posn of 10
7 9 10 11 5 6 4 8
-> the key taken is 5
Right shift 11 by 1 posn
Right shift 10 by 1 posn
Right shift 9 by 1 posn
Right shift 7 by 1 posn
place the key at the posn of 7
5 7 9 10 11 6 4 8
-> the key taken is 6
Right shift 11 by 1 posn
Right shift 10 by 1 posn
Right shift 9 by 1 posn
Right shift 7 by 1 posn
place the key at the posn of 7
5 6 7 9 10 11 4 8
-> the key taken is 4
Right shift 11 by 1 posn
Right shift 10 by 1 posn
Right shift 9 by 1 posn
Right shift 7 by 1 posn
Right shift 6 by 1 posn
Right shift 5 by 1 posn
place the key at the posn of 5
4 5 6 7 9 10 11 8
-> the key taken is 8
Right shift 11 by 1 posn
Right shift 10 by 1 posn
Right shift 9 by 1 posn
place the key at the posn of 9
Final Correct Output: 4 5 6 7 8 9 10 11
```

- **Selection Sort**

```
Selection sort
11 is swapped with 4
4 10 7 9 5 6 11 8
10 is swapped with 5
4 5 7 9 10 6 11 8
7 is swapped with 6
4 5 6 9 10 7 11 8
9 is swapped with 7
4 5 6 7 10 9 11 8
10 is swapped with 8
4 5 6 7 8 9 11 10
9 is swapped with 9
4 5 6 7 8 9 11 10
11 is swapped with 10
4 5 6 7 8 9 10 11
Final Correct Output: 4 5 6 7 8 9 10 11
```

ALGO FOR INSERTION SORT

Algorithm

To sort an array of size n in ascending order:
1: Iterate from arr[1] to arr[n] over the array.
2: Compare the current element (key) to its predecessor.
3: If the key element is smaller than its predecessor, compare it to the elements before. Move the greater elements one position up to make space for the swapped element.

ALGO FOR BUBBLE SORT

We assume list is an array of n elements. We further assume that swap function swaps the values of the given array elements.

```
begin BubbleSort(list)

  for all elements of list
    if list[i] > list[i+1]
      swap(list[i], list[i+1])
    end if
  end for

  return list

end BubbleSort
```

ALGO FOR SELECTION SORT

Step 1 – Set MIN to location 0
Step 2 – Search the minimum element in the list
Step 3 – Swap with value at location MIN
Step 4 – Increment MIN to point to next element
Step 5 – Repeat until list is sorted