|  |  |
| --- | --- |
| Assignment No. : | 11 |
| Title: | Write a modular program using objectoriented programming features to implement different sorting methods. (Insertion Sort, Bubble Sort,  Selection Sort |
| Subject: | Data Structures Laboratory |
| Class: | S.Y. (C.s.E) |
| Roll No.: |  |
| Assessment (Marks): |  |
| Signature and Date of Assessment: |  |

Assignment No. 11

|  |  |
| --- | --- |
| Title :  Objectives :  Problem Statement :  Outcomes:  Theory: | Implementation of Selection sort and bubble sort and Insertion sort in CPP.   1. To understand the concept of sorting. 2. To understand the concept and use of selection sort. 3. To understand the concept and use of bubble sort.   Write a CPP program to store first year percentage of students in array. Write function for sorting array of floating point numbers in ascending order using   1. Selection Sort 2. Bubble sort and display top five scores.    * Understanding the concept of sorting.    * Understanding the concept and use of selection sort.  Understanding the concept and use of bubble sort.   Selection Sort:  Selection sort carries out a sequence of passes over the table. At the first pass an entry is selected on some criteria and placed in the correct position in the table. The possibk criteria for selecting an element are to pick the smallest or pick the largest. If the smallest is chosen then, for sorting in ascending order, the correct position to put it is at the beginning of |

the table.

Now that the correct entry is in the first place in the table the process is repeated on the remaining entries. Once this has been repeated n-l times the n-l smallest entries are in the first n-l places which leaves the largest element in the last place. Thus only nI passes are required.

The pseudocode can be described as follows: procedure selection sott list : array of items n : size of list fori= I to n- I

/\* set current element as minimum\*/ min = i

/\* check the element to be minimum \*/ forj= i+l ton if listljl < listlminl then min = j; end if end for  swap the minimum element with the current element\*/ if indexMin i then swap list[minl and list[il end if end for end procedure

For example consider the following example of selection sort being carried out on a sample set of data:

9 2 5 748 on pass I look for smallest in 1st to 6th swap 2nd with first giving

2 9 5 748 on pass 2 look for smallest in 2nd to 6th swap 5th with second giving

24 5 7 9 8 on pass 3 look for smallest in 3rd to 6th swap 3rd with third giving

24 5 798 on pass 4 look for smallest in 4th to 6th swap 4th with fourth giving 24 5 7 9 8 on pass 5 look for smallest in 5th to 6th swap 5th with 6th giving 24 5 789 sorted.

Bubble Sort:

Bubble Sort is the simplest sorting algorithm that works by repeatedly swapping the adjacent elements if they are in wrong order.

The pseudocode can be described as follows: procedure bubbleSort( list : array of items ) loop = list.count; for i = O to loop-I do: swapped = false for j = O to loop-I do:

/\* compare the adjacent elements if list[jl > list[j+l] then /\* swap them \*/ swap( list[j], list[j+ll ) swapped = true end if end for

/\*if no number was swapped that means array is sorted now, break the loop.\*/ if(not swapped) then break end if end for end procedure return list

Example :

First Pass:

( 5 1 4 2 8 ) ( 15 4 2 8 ), Here, algorithm compares the first two elements, and swaps since 5> 1.

( 1 54 2 ( 1 452 8 ), Swap since 5

# ( 1 4 52 ( 1 425 8 ), Swap since 5>2

( 1 4 2 5 8 ) 1 4 2 5 8 ), Now, since these elements are already in order (8 > 5), algorithm does not swap them.

Second Pass:

# 1 24 5 8 ), swap since

Now, the array is already sorted, but algorithm does not know if it is completed. The algorithm needs one whole pass without any swap to know it is sorted.

Third Pass:





After final pass will get the sorted list as : ( 1 24 58 )

Algorithm: Algorithm 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

Algorithm for Bubble Sort:

Step 1 — begin BubbleSort(list)

Step 2 — for all elements of list

Step 3 — if list[i] > list[i+l] then swap(list[i], list[i+l l) Step 4 — display list

Insertion Sort:

This is a in-place comparison based sorting algorithm. Here, a sub-list is maintained which is always sorted. For example, the lower part of an array is maintained to be sorted. A element which is to be 'inserted' in this sorted sub-list, has to find its appropriate place and insert it there. Hence the name insertion sort.

The array is searched sequentially and unsorted items are moved and inserted into sorted sub-list (in the same array). This algorithm is not suitable for large data sets as its average and worst case complexity are ofO(n2) where n are no. of items.

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Step i |  | | | | | | | | | | Checking second element of array with element before it and inserting it in proper position. In this case, 3 is inserted in position of |
|  |  | |  |  | |  | |  | |
|  | |  | |  | |  | |  | |
|  |  | | |  | |  | |  | |
| step 2 |  |  | |  |  |  |  | |  | | Checking third element Of array with elements before it and insertinq it in proper position. tn this inserted in position or 3. |
|  | | 12 | |  |  | 9 | | 8 | |
|  |  | | |  |  |  | |  | |
| step 3 |  |  | |  |  |  |  |  |  | | Checking fourth element Of array with elements before it and inserting it in proper position. tn this case, S is inserted position of 12.  Checking nfth of array with elements before it and inserting it in proper position. In this case, 8 is inserted in position of 12. |
|  | |  | |  | |  |  |  | |
|  | | | |  |  |  |  |  | |
| Step 4 |  |  | |  |  |  |  |  |  |  |
|  | |  | |  | | 12 | |  |  |
|  | | | |  | |  |  |  |  |
|  |  | | | |  | |  | |  | | Sorted Array Order |
|  | |  | | 3 | | 8 | |  | |

Algorithm:

Algorithm for Insertion Sort:

Step I — If it is the first element, it is already sorted. retum l; Step 2 — Pick next element

Step 3 — Compare with all elements in the sorted sub-list

Step 4 — Shift all the elements in the sorted sub-list that is greater than the value to be sorted

Step 5 — Insert the value

Step 6 — Repeat until list is sorted

Algorithm for Shell Sort:

Step I — Initialize the value of h

Step 2 — Divide the list into smaller sub-list of equal interval h

Step 3 — Sort these sub-lists using insertion sort

Step 3 — Repeat until complete list is sorted

|  |  |
| --- | --- |
| Test Cases: | Input :Unsorted Array: [65 35 45 15 85 5 95 251  Output :SortedArray: [5 15 25 35 45 65 85 951  Insertion Sort:  Test Cases: Input: Unsorted Array: [65 35 45 15 85 5 95 251 Output :Sorted Array: [5 15 25 35 45 65 85 951 |
| Conclusion : | Thus, we implemented selection sort and bubble sort to sort and Insertion Sort the given army in ascending order. |
| Questions : | 1. What is sorting? 2. Why we need sorting? 3. Explain best, average, worst case analysis for selection and bubble sort. 4)What is stability in sorting? Which sorting algorithms are not stable? |

Code:-Insertion Sort.

#include <iostream>

using namespace std;

void insert(int a[], int n) /\* function to sort an aay with insertion sort \*/

{

    int i, j, temp;

    for (i = 1; i < n; i++) {

        temp = a[i];

        j = i - 1;

        while(j>=0 && temp <= a[j])  /\* Move the elements greater than temp to one position ahead from their current position\*/

        {

            a[j+1] = a[j];

            j = j-1;

        }

        a[j+1] = temp;

    }

}

void printArr(int a[], int n) /\* function to print the array \*/

{

    int i;

    for (i = 0; i < n; i++)

        cout << a[i] <<" ";

}

int main()

{

    int a[10],n,i;

    cout<<"Enter no of elements in the array";

    cin>>n;

    cout<<" Enter array elements"<<endl;

    for(i=0;i<n;i++)

    cin>>a[i];

    cout<<"Before sorting array elements are - "<<endl;

    printArr(a, n);

    insert(a, n);

    cout<<"\nAfter sorting array elements are - "<<endl;

    printArr(a, n);

    return 0;

}

Output:-

Enter no of elements in the array: 3

Enter array elements

12

23

34

Before sorting array elements are -

12 23 34

After sorting array elements are -12 23 34

Code:-Bubble Sort.

#include <iostream>

using namespace std;

class BubbleSort

{

    int a[20], i, n, j, temp;

public:

    void get()

    {

        cout << "ENTER THE NUMBER OF ELEMENTS IN THE ARRAY: " << endl;

        cin >> n;

        cout << "ENTER THE ARRAY ELEMENTS: " << endl;

        for (i = 0; i < n; i++)

        {

            cin >> a[i];

        }

    }

public:

    void display()

    {

        cout << "UNSORTED ARRAY ELEMENTS ARE: " << endl;

        for (i = 0; i < n; i++)

        {

            cout << a[i] << " ";

        }

    }

public:

    void sort()

    {

        cout << "\nSORTED ELEMENTS ARE: " << endl;

        for (i = 0; i < n; i++)

        {

            for (j = i + 1; j < n; j++)

            {

                if (a[i] > a[j])

                {

                    temp = a[i];

                    a[i] = a[j];

                    a[j] = temp;

                }

            }

            cout << a[i]<<" ";

        }

    }

};

int main()

{

    BubbleSort b;

    b.get();

    b.display();

    b.sort();

}

Output:-

Enter no of elements in the array: 3

Enter array elements

12

23

34

Before sorting array elements are -

12 23 34

After sorting array elements are -

12 23 34

Code:- Selection Sort.

#include <iostream>

using namespace std;

void selection(int arr[], int n)

{

    int i, j, small;

    for (i = 0; i < n-1; i++)    // One by one move boundary of unsorted subarray

    {

        small = i; //minimum element in unsorted array

        for (j = i+1; j < n; j++)

        if (arr[j] < arr[small])

            small = j;

// Swap the minimum element with the first element

    int temp = arr[small];

    arr[small] = arr[i];

    arr[i] = temp;

    }

}

void printArr(int a[], int n) /\* function to print the array \*/

{

    int i;

    for (i = 0; i < n; i++)

        cout<< a[i] <<" ";

}

int main()

{

    int a[10],n,i;

    cout<<"Enter no of elements in the array";

    cin>>n;

    cout<<" Enter array elements";

    for(i=0;i<n;i++)

    cin>>a[i];

    cout<< "Before sorting array elements are - "<<endl;

    printArr(a, n);

    selection(a, n);

    cout<< "\nAfter sorting array elements are - "<<endl;

    printArr(a, n);

    return 0;

}

Output:-

Enter no of elements in the array: 3

Enter array elements

12

23

34

Before sorting array elements are -

12 23 34

After sorting array elements are –

12 23 34