Name: Chaudhari Dhanshri Santosh

Roll no.: 8

//Write a program for sorting from given array in ascending / descending order

// n = 1000, 2000, 300 find the exact time of execution.

#include<iostream>

#include<conio.h>

#include<chrono>

using namespace std;

using namespace std::chrono;

class HeapSort

{

public:

    int done, maxchild, temp;

    int A[1000];

    int i, n;

    void shiftdown(int[], int, int);

    void heapsort(int[], int);

    void getdata();

    void display();

};

void HeapSort::getdata()

{

    cout << "Enter size of array:";

    cin >> n;

    cout << "Enter the array elements=";

    for (int i = 0; i < n; i++)

    {

        cin >> A[i];

    }

}

void HeapSort::shiftdown(int A[], int root, int bottom)

{

    done = 0;

    while ((root \* 2 + 1 <= bottom) && (!done))

    {

        if (root \* 2 + 1 == bottom || A[root \* 2 + 1] > A[root \* 2 + 2])

        {

            maxchild = root \* 2 + 1;

        }

        else

        {

            maxchild = root \* 2 + 2;

        }

        if (A[root] < A[maxchild])

        {

            temp = A[root];

            A[root] = A[maxchild];

            A[maxchild] = temp;

            root = maxchild;

        }

        else

        {

            done = 1;

        }

    }

}

void HeapSort::heapsort(int A[], int ub)

{

    for (int i = (ub / 2.0) - 1; i >= 0; i--)

    {

        shiftdown(A, i, ub);

    }

    for (int i = ub; i >= 1; i--)

    {

        temp = A[0];

        A[0] = A[i];

        A[i] = temp;

        shiftdown(A, 0, i - 1);

    }

}

void HeapSort::display()

{

    cout << "Elements you entered:";

    for (int i = 0; i < n; i++)

    {

        cout << A[i] << " ";

    }

    heapsort(A, n - 1);

    cout << "\nSorted element  in ascending order:";

    for (int i = 0; i < n; i++)

    {

        cout << A[i] << " ";

    }

    cout << endl;

    cout << "\nSorted element  in descending order:";

    for (int i = n; i >= 0; i--)

    {

        cout << A[i] << " ";

    }

    cout << endl;

}

int main()

{

    HeapSort h;

    h.getdata();

    auto start = high\_resolution\_clock::now();

    h.display();

    auto stop = high\_resolution\_clock::now();

    auto duration = duration\_cast<seconds>(stop - start);

    cout << "\n Exact time of execution:" << duration.count() << "seconds\n" << endl;

}

Output:

Enter size of array:5

Enter the array elements=34

54

23

78

99

Elements you entered:34 54 23 78 99

Sorted element in ascending order:23 34 54 78 99

Sorted element in descending order: 99 78 54 34 23

Exact time of execution:0seconds

//Write a program for sorting given array in ascending/descending order using merge sort.

#include<iostream>

using namespace std;

class MergeSortDemo

{

    int A[16];

    int n;

public:

    void getData()

    {

        cout << "Enter the number of elements:";

        cin >> n;

        cout << "Enter the element:";

        for (int i = 0; i < n; i++)

        {

            cin >> A[i];

        }

    }

    void display()

    {

        cout << "Sorted elements in ascending order :";

        for (int i = 0; i < n; i++)

        {

            cout << A[i] << "\t";

        }

        cout << endl;

        cout << "Sorted elements in descending order :";

        for (int i = n; i >= 0; i--)

        {

            cout << A[i] << "\t";

        }

        cout << endl;

    }

    void merge(int A[], int temp[] ,int left, int mid, int right);

    void m\_sort(int A[], int temp[], int left, int right);

    void mergeSort();

};

void MergeSortDemo::merge(int A[], int temp[], int left, int mid, int right)

{

    int t\_pos, left\_end, n, i;

    t\_pos = left;

    left\_end = mid - 1;

    n = right - left + 1;

    while (left <= left\_end && mid <= right)

    {

        if (A[left] < A[mid])

        {

            temp[t\_pos] = A[left];

            t\_pos = t\_pos + 1;

            left = left + 1;

        }

        else

        {

            temp[t\_pos] = A[mid];

            t\_pos = t\_pos + 1;

            mid = mid + 1;

        }

    }

    while (left <= left\_end)

    {

        temp[t\_pos] = A[left];

        t\_pos = t\_pos + 1;

        left = left + 1;

    }

    while (mid <= right)

    {

        temp[t\_pos] = A[mid];

        t\_pos = t\_pos + 1;

        mid = mid + 1;

    }

    for (i = 0; i < n; i++)

    {

        A[right] = temp[right];

        right--;

    }

}

void MergeSortDemo::m\_sort(int A[], int temp[], int left, int right)

{

    int mid;

    if (right > left)

    {

        mid = (left + right) / 2;

        m\_sort(A, temp, left, mid);

        m\_sort(A, temp, mid + 1, right);

        merge(A, temp, left, mid + 1, right);

    }

}

void MergeSortDemo::mergeSort()

{

    int temp[10];

    m\_sort(A, temp, 0, n - 1);

}

int main(int argc, char\* argv[])

{

    MergeSortDemo o;

    o.getData();

    o.mergeSort();

    o.display();

}

Outuput:

Enter the number of elements:5

Enter the element:23 45 76 99 45

Sorted elements in ascending order :23 45 45 76 99

Sorted elements in descending order : 99 76 45 45 23

//Write a program for sorting given array in ascending/descending order using Quick sort.

#include<iostream>

using namespace std;

class QuickSortDemo

{

    int A[16];

    int n;

public:

    void getData()

    {

        cout << "Enter the number of elements:";

        cin >> n;

        cout << "Enter the element:";

        for (int i = 0; i < n; i++)

        {

            cin >> A[i];

        }

    }

    void QuickSort()

    {

        QuickSort(A, 0, n - 1);

    }

    void display()

    {

        cout << "Sorted elements in ascending order :";

        for (int i = 0; i < n; i++)

        {

            cout << A[i] << "\t";

        }

        cout << endl;

        cout << "Sorted elements in descending order :";

        for (int i = n; i >= 0; i--)

        {

            cout << A[i] << "\t";

        }

        cout << endl;

    }

    int partition(int A[], int lb, int ub);

    void QuickSort(int A[], int lb, int ub);

};

int QuickSortDemo::partition(int A[], int lb, int ub)

{

    int temp;

    int start = lb, end = ub;

    int pivot = A[lb];

    while (start < end)

    {

        while (A[start] <= pivot)start++;

        while (A[end] > pivot)end--;

        if (start < end)

        {

            temp = A[start];

            A[start] = A[end];

            A[end] = temp;

        }

    }

    temp = A[lb];

    A[lb] = A[end];

    A[end] = temp;

    return end;

}

void QuickSortDemo::QuickSort(int A[], int lb, int ub)

{

    int loc;

    if (lb < ub)

    {

        loc = partition(A, lb, ub);

        QuickSort(A, lb, loc - 1);

        QuickSort(A, loc + 1, ub);

    }

}

int main(int argc, char\* argv[])

{

    QuickSortDemo o;

    o.getData();

    o.QuickSort();

    o.display();

}

Output:

Enter the number of elements:5

Enter the element:23

45

65

77

34

Sorted elements in ascending order :23 34 45 65 77

Sorted elements in descending order : 77 65 45 34 23