Assignment 1 – Sorting and Searching

Name: Kartik Banshi Katkar

Rollno : 29

SY-IT-A

Q**. Write a Menu driven Program to implement**

**1. Insertion sort**

**2. Merge sort**

**3. Binary search**

**4. Exit**

Code:

#include <iostream>

using namespace std;

void displayarray(int array[],int end){

for (int i=0;i<=end;i++){

    cout<<array[i]<<"\t";

    }

   cout<<"\n";

}

void merge(int array[], int const left, int const mid,

           int const right)

{

    auto const subArrayOne = mid - left + 1;

    auto const subArrayTwo = right - mid;

    // Create temp arrays

    auto \*leftArray = new int[subArrayOne],

         \*rightArray = new int[subArrayTwo];

    // Copy data to temp arrays leftArray[] and rightArray[]

    for (auto i = 0; i < subArrayOne; i++)

        leftArray[i] = array[left + i];

    for (auto j = 0; j < subArrayTwo; j++)

        rightArray[j] = array[mid + 1 + j];

    auto indexOfSubArrayOne

        = 0, // Initial index of first sub-array

        indexOfSubArrayTwo

        = 0; // Initial index of second sub-array

    int indexOfMergedArray

        = left; // Initial index of merged array

    // Merge the temp arrays back into array[left..right]

    while (indexOfSubArrayOne < subArrayOne

           && indexOfSubArrayTwo < subArrayTwo) {

        if (leftArray[indexOfSubArrayOne]

            <= rightArray[indexOfSubArrayTwo]) {

            array[indexOfMergedArray]

                = leftArray[indexOfSubArrayOne];

            indexOfSubArrayOne++;

        }

        else {

            array[indexOfMergedArray]

                = rightArray[indexOfSubArrayTwo];

            indexOfSubArrayTwo++;

        }

        indexOfMergedArray++;

    }

    // Copy the remaining elements of

    // left[], if there are any

    while (indexOfSubArrayOne < subArrayOne) {

        array[indexOfMergedArray]

            = leftArray[indexOfSubArrayOne];

        indexOfSubArrayOne++;

        indexOfMergedArray++;

    }

    // Copy the remaining elements of

    // right[], if there are any

    while (indexOfSubArrayTwo < subArrayTwo) {

        array[indexOfMergedArray]

            = rightArray[indexOfSubArrayTwo];

        indexOfSubArrayTwo++;

        indexOfMergedArray++;

    }

    delete[] leftArray;

    delete[] rightArray;

}

void mergeSort(int array[], int const begin, int const end)

{

    if (begin >= end)

        return; // Returns recursively

    auto mid = begin + (end - begin) / 2;

    mergeSort(array, begin, mid);

    mergeSort(array, mid + 1, end);

    merge(array, begin, mid, end);

    displayarray(array,end);

}

void binarysearch(int arr1[],int num,int n)

{

    int low,mid,high;

    low = 0;

    high = n-1;

    do{

    mid = (low+high)/2;

    if (arr1[mid] == num)

    {

      cout<< "number found at pos  "<<mid+1<<"\n";

    }

    if (arr1[mid < num])

    {

      low = mid+1;

    }

    else{

        high = mid -1;

    }

    }while(low<=high);

}

int insertionsort(int n,int\* arr1)

{

    int i, key, j;

    for (i = 1; i < n; i++)

    {

        key = arr1[i];

        j = i - 1;

        // Move elements of arr[0..i-1],

        // that are greater than key, to one

        // position ahead of their

        // current position

        while (j >= 0 && arr1[j] > key)

        {

            arr1[j + 1] = arr1[j];

            j = j - 1;

        }

        arr1[j + 1] = key;

    }

    displayarray(arr1,n - 1);

    return 0;

}

int selectionsort(int n,int\* arr1)

{

    int i, j, min\_idx;

    // One by one move boundary of

    // unsorted subarray

    for (i = 0; i < n-1; i++)

    {

        // Find the minimum element in

        // unsorted array

        min\_idx = i;

        for (j = i+1; j < n; j++)

        if (arr1[j] < arr1[min\_idx])

            min\_idx = j;

        // Swap the found minimum element

        // with the first element

        if(min\_idx!=i)

            {

                int temp = arr1[min\_idx];

                arr1[min\_idx] = arr1[i];

                arr1[i] = temp;

            }

    }

    displayarray(arr1, n-1);

    return 0;

}

int bubblesort(int n,int\* arr1)

{

    int i, j;

   bool swapped;

   for (i = 0; i < n-1; i++)

   {

     swapped = false;

     for (j = 0; j < n-i-1; j++)

     {

        if (arr1[j] > arr1[j+1])

        {

           swap(arr1[j], arr1[j+1]);

           swapped = true;

        }

     }

     // IF no two elements were swapped

     // by inner loop, then break

     if (swapped == false)

        break;

   }

   displayarray(arr1,n-1);

   return 0;

}

int bubblesortandbinary(int n,int\* arr1,int num)

{

    int i, j;

   bool swapped;

   for (i = 0; i < n-1; i++)

   {

     swapped = false;

     for (j = 0; j < n-i-1; j++)

     {

        if (arr1[j] > arr1[j+1])

        {

           swap(arr1[j], arr1[j+1]);

           swapped = true;

        }

     }

     // IF no two elements were swapped

     // by inner loop, then break

     if (swapped == false)

        break;

   }

   displayarray(arr1,n-1);

   binarysearch(arr1,num,n);

   return 0;

}

int main()

{

    int choice, n;

    do{

    cout << "Enter the number of elements you want in array ";

    cin >> n;

    int arr[n];

    for (int i = 0; i < n; i++)

    {

        cout << "Enter the element at array position " << i << " ";

        cin >> arr[i];

    }

    cout << "Your Array before sorting is \n";

    for (int i = 0; i < n; i++)

    {

        cout << arr[i] << "\t";

    }

    cout << "\n Menu \n 1.Merge sort \n 2.Selection sort \n 3.Bubble sort\n 4.Insertion Sort \n 5.Binary Search\n 6.Exit \n";

    cin >> choice;

    switch (choice)

    {

    case 1:

        mergeSort(arr,0,n-1);

        break;

    case 2:

        selectionsort(n,arr);

        break;

    case 3:

        bubblesort(n,arr);

        break;

    case 4:

        insertionsort(n,arr);

        break;

    case 5:

    int search\_number;

        cout<<"Enter the number to be searched : ";

        cin>>search\_number;

        bubblesortandbinary(n,arr,search\_number);

        break;

    case 6:

        cout << "\n Thank you for Sorting  \n";

        break;

    default:

        cout << "\n Invalid Choice \n";

        break;

    }

    }while(choice != 6);

}

Input/Output:





