LAB # 02 - ASSIGNMENT

30 OCT

2023

**GROUP 16**

MUHAMMAD YASIR | CT-22082

MUHAMMAD SHAHEER QURESHI | CT-22090

SYED SAAD WAQAR | CT-22097

AMMAR YASSER AHMED | CT-22103

DATA STRUCTURES ALGORITHMS AND APPLICATIONS (ct – 159)

TEACHER : SIR ABDUL KARIM KAZI

**EXERCISE**

/\*

DSAA LAB 2

29 OCT 2023

1. If the array is already sorted, we don’t want to continue with the comparisons. This can

be achieved with modified bubble sort. Update the code in example 02 to have a

modified bubble sort function.

\*/

#include <iostream>

void modifiedBubbleSort(int arr[], int size) {

    bool swapped;

    for (int i = 0; i < size - 1; i++) {

        swapped = false;

        for (int j = 0; j < size - i - 1; j++) {

            if (arr[j] > arr[j + 1]) {

                // Swap arr[j] and arr[j+1]

                int temp = arr[j];

                arr[j] = arr[j + 1];

                arr[j + 1] = temp;

                swapped = true;

            }

        }

               // If no two elements were swapped in the inner loop, the array is already sorted.

            if (!swapped) {

                std::cout << "No swap\n";

                  break;

        }

    }

}

  int main() {

     int arr[] = { 1, 2, 3, 4, 8};

     int size = sizeof(arr) / sizeof(arr[0]);

     modifiedBubbleSort(arr, size);

     std::cout << "Sorted array: ";

     for (int i = 0; i < size; i++) {

        std::cout << arr[i] << " ";

    }

     std::cout << std::endl;

     return 0;

}

OUTPUT:



/\*

DSAA LAB 2

29 OCT 2023

2. Create a Person class which has following attributes:

First Name

Last Name

Birth Year

Birth Month

Birth Date

Develop C++ solution such that day month and year are taken as input for N persons and

perform Sorting based on year, month and day using Selection Sort.

\*/

#include<iostream>

#include<vector>

using namespace std;

class Person{

    string firstName;

    string lastName;

    int birthYear;

    int birthMonth;

    int birthDate;

    public:

    Person(string f, string l, int y, int m, int d){

        setFirstName(f);

        setLastName(l);

        setBirthYear(y);

        setBirthMonth(m);

        setBirthDate(d);

    }

    Person(const Person& obj){

        setFirstName( obj.getFirstName() );

        setLastName( obj.getLastName() );

        setBirthYear( obj.getBirthYear() );

        setBirthMonth( obj.getBirthMonth() );

        setBirthDate( obj.getBirthDate() );

    }

    string getFirstName() const {

        return firstName;

    }

    string getLastName() const {

        return lastName;

    }

    int getBirthYear() const {

        return birthYear;

    }

    int getBirthMonth() const {

        return birthMonth;

    }

    int getBirthDate() const {

        return birthDate;

    }

    void setFirstName(string fn){

        firstName = fn;

    }

    void setLastName(string ln){

        lastName = ln;

    }

    void setBirthDate(int bd){

        birthDate = bd;

    }

    void setBirthMonth(int bm){

        birthMonth = bm;

    }

    void setBirthYear(int by){

        birthYear = by;

    }

    ~Person(){

    }

};

void swapIntElementsPerson(int i, int j, vector<Person> &obj){

    // \*a = \*a + \*b;

    // \*b = \*a - \*b;

    // \*a = \*a - \*b;

    obj.at(i).setBirthDate( obj.at(i).getBirthDate() + obj.at(j).getBirthDate() );

    obj.at(j).setBirthDate( obj.at(i).getBirthDate() - obj.at(j).getBirthDate() );

    obj.at(i).setBirthDate( obj.at(i).getBirthDate() - obj.at(j).getBirthDate() );

    obj.at(i).setBirthMonth( obj.at(i).getBirthMonth() + obj.at(j).getBirthMonth() );

    obj.at(j).setBirthMonth( obj.at(i).getBirthMonth() - obj.at(j).getBirthMonth() );

    obj.at(i).setBirthMonth( obj.at(i).getBirthMonth() - obj.at(j).getBirthMonth() );

    obj.at(i).setBirthYear( obj.at(i).getBirthYear() + obj.at(j).getBirthYear() );

    obj.at(j).setBirthYear( obj.at(i).getBirthYear() - obj.at(j).getBirthYear() );

    obj.at(i).setBirthYear( obj.at(i).getBirthYear() - obj.at(j).getBirthYear() );

}

void swapVectors(int i, int j, vector<Person> &obj){

    string fn, ln;

    fn = obj.at(i).getFirstName();

    obj.at(i).setFirstName( obj.at(j).getFirstName() );

    obj.at(j).setFirstName( fn );

    ln = obj.at(i).getLastName();

    obj.at(i).setLastName( obj.at(j).getLastName() );

    obj.at(j).setLastName( ln );

    swapIntElementsPerson(i, j, obj);

}

void bubbleSortPersonClass(vector<Person> &obj){

    for(int i = 0; i < obj.size() - 1; i++){

        for(int j = 0; j < obj.size() - 1; j++){

            if(obj.at(j).getBirthYear() > obj.at(j+1).getBirthYear()){

                swapVectors(j, j+1, obj);

            }

            if((obj.at(j).getBirthYear() == obj.at(j+1).getBirthYear()) &&

               (obj.at(j).getBirthMonth() > obj.at(j+1).getBirthMonth()) ){

                    swapVectors(j, j+1, obj);

            }

            if((obj.at(j).getBirthYear() == obj.at(j+1).getBirthYear()) &&

               (obj.at(j).getBirthMonth() == obj.at(j+1).getBirthMonth()) &&

               (obj.at(j).getBirthDate() > obj.at(j+1).getBirthDate())){

                    swapVectors(j, j+1, obj);

            }

        }

    }

}

void printVectorPersonElements(vector<Person> &obj) {

    cout << "FirstName: \tLastName: \tDOB\n";

    for(int i = 0; i < obj.size(); i++){

        cout << obj.at(i).getFirstName() << "\t" << obj.at(i).getLastName() << "\t"

             << obj.at(i).getBirthDate() << "/" << obj.at(i).getBirthMonth() << "/"

             << obj.at(i).getBirthYear() << endl;

    }

}

int main(){

    Person  P1("Muhammad", "Qureshi", 1985, 2, 14);

    Person  P2("Ahmad", "Zafar", 1990, 5, 23);

    Person  P3("Ali", "Abu Bakar", 1978, 10, 9);

    Person  P4("Umar", "Farooq", 1995, 7, 30);

    Person  P5("Farooq", "Ahmad", 1972, 9, 22);

    Person  P6("Uzair", "Qureshi", 1972, 3, 17);

    Person  P7("Zafar", "Qureshi", 1972, 9, 21);

    Person  P8("Ali", "Murtaza", 1998, 6, 11);

    Person  P9("Abu Bakar", "Siddique", 1997, 4, 2);

    Person  P10("Hasan", "Hussain", 1980, 11, 13);

    vector<Person> VectorPerson;

    VectorPerson.push\_back(P1);

    VectorPerson.push\_back(P2);

    VectorPerson.push\_back(P3);

    VectorPerson.push\_back(P4);

    VectorPerson.push\_back(P5);

    VectorPerson.push\_back(P6);

    VectorPerson.push\_back(P7);

    VectorPerson.push\_back(P8);

    VectorPerson.push\_back(P9);

    VectorPerson.push\_back(P10);

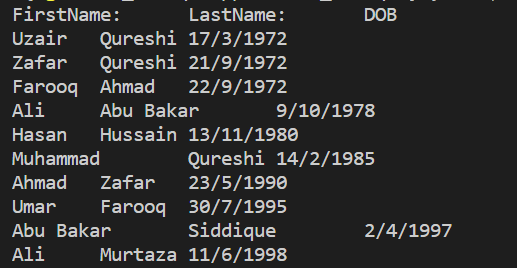
    bubbleSortPersonClass(VectorPerson);

    printVectorPersonElements(VectorPerson);

    return 0;

}

OUTPUT:



/\*

DSAA LAB 2

29 OCT 2023

3. Given an array arr[ ] of length N consisting cost of N toys and an integer K the amount

with you. The task is to find maximum number of toys you can buy with K amount.

Test Case:

Input: N = 7, K = 50, arr[] = {1, 12, 5, 111, 200, 1000, 10}

Output: 4

Explanation: The costs of the toys. You can buy are 1, 12, 5 and 10.

\*/

#include <iostream>

#include <algorithm>

using namespace std;

int maximumToys(int arr[], int N, int K) {

    sort(arr, arr + N); // Sort the array in ascending order

    int count = 0;

    int totalCost = 0;

    for (int i = 0; i < N; i++) {

        if (totalCost + arr[i] <= K) {

            totalCost += arr[i];

            count++;

        } else {

            break; // If you can't afford the next toy, stop.

        }

    }

    return count;

}

int main() {

    int N = 7;

    int K = 50;

    int arr[] = {1, 12, 5, 111, 200, 1000, 10};

    int result = maximumToys(arr, N, K);

    cout << "Maximum number of toys you can buy: " << result << endl;

    return 0;

}

OUTPUT:



/\*

DSAA LAB 2

29 OCT 2023

4. Create a single class Sort, which will provide the user the option to choose between all 3

sorting techniques. The class should have following capabilities:

⎫ Take an array and a string (indicating the user choice for sorting technique) as

input and perform the desired sorting.

⎫ Should allow the user to perform analysis on a randomly generated array. The

analysis provides number of comparisons and number of swaps performed for

each technique.

⎫ After printing all the results, the class should highlight the best and worst

techniques.

\*/

#include<iostream>

using namespace std;

void swap(int \*a, int \*b){

    \*a = \*a + \*b;

    \*b = \*a - \*b;

    \*a = \*a - \*b;

}

class Sort{

    int comparisons;

    public:

    void sortingTechnique(string sorting\_technique, int array[], int size){

        comparisons = 0;

        if(sorting\_technique == "Bubble Sort"){

            bubbleSort(array, size);

            cout << "Bubble Sort: ";

        }

        else if(sorting\_technique == "Selection Sort"){

            selectionSort(array, size);

            cout << "Selection Sort: ";

        }

        else if(sorting\_technique == "Insertion Sort"){

            insertionSort(array, size);

            cout << "Insertion Sort: ";

        }

        printArray(array, size);

        printComparisons();

    }

    void bubbleSort(int array[], int size){

        for(int i = 0; i < size-1; i++){

            comparisons++;

            for(int j = 0; j < size-i-1; j++){

                comparisons++;

                if(array[j] > array[j+1]){

                    swap(array[j], array[j+1]);

                }

                comparisons++;

            }

        }

    }

    void selectionSort(int array[], int size){

        int i, j, min\_index;

        for(i = 0; i < size-1; i++){

            comparisons++;

            min\_index = i;

            for(j = i+1; j < size; j++){

                comparisons++;

                if(array[j] < array[min\_index]){

                    min\_index = j;

                }

                comparisons++;

            }

            if(min\_index != i){

                swap(&array[min\_index], &array[i]);

            }

            comparisons++;

        }

    }

    void insertionSort(int array[], int size){

        int i, key, j;

        for(i = 1; i < size; i++){

            comparisons++;

            key = array[i];

            j = i - 1;

            while(j >= 0 && array[j] > key){

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

                j = j - 1;

            }

            comparisons += 3;

            array[j+1] = key;

        }

    }

    void printArray(int array[], int size){

        cout << "Array Elements:" << endl;

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

            cout << array[i] << " ";

        }

        cout << endl;

    }

    void printComparisons(){

        cout << "Number of Comparisons: " << comparisons << endl;

    }

};

int main(){

    int array1[10] = {5, 6, 4, 3, 2, 8, 9, 10, 1, 7}, array2[10], array3[10];

    for(int i = 0; i < 10; i++){

        array3[i] = array2[i] = array1[i];

    }

    Sort S1;

    S1.sortingTechnique("Bubble Sort", array1, 10);

    S1.sortingTechnique("Selection Sort", array2, 10);

    S1.sortingTechnique("Insertion Sort", array3, 10);

    return 0;

}

OUTPUT:

