Tema

1. Se da un vector de numere intregi pozitive de dimensiune “dim”. Sa se sorteze elementele palindroame din vectorul dat.

#include <iostream>

#include <cmath>

using namespace std;

void Citire\_Vector(int vector[], int marime\_vector)

{

for (int index = 0; index < marime\_vector; index++)

{

cin >> vector[index];

}

}

bool Verificare\_Palindrom(int numar, int dimensiune)

{

int numar1 = numar / pow(10, dimensiune / 2);

int numar2 = numar % ((int)pow(10, dimensiune / 2));

int oglindit\_numar1 = 0;

if (dimensiune % 2 == 1)

numar1 /= 10;

while (numar1 != 0)

{

oglindit\_numar1 = oglindit\_numar1 \* 10 + numar1 % 10;

numar1 /= 10;

}

if (oglindit\_numar1 == numar2)

return true;

else

return false;

}

void Afisare(int vector[], int marime\_vector)

{

for (int index = 0; index < marime\_vector; index++)

{

cout << vector[index] << " ";

}

}

void Sortare\_vector(int vector[], int marime\_vector, int dimensiune)

{

for (int index1 = 0; index1 < marime\_vector - 1; index1++)

{

for (int index2 = index1 + 1; index2 < marime\_vector; index2++)

{

if (vector[index1] > vector[index2] && Verificare\_Palindrom(vector[index1], dimensiune) && Verificare\_Palindrom(vector[index2], dimensiune))

{

int auxiliar = vector[index1];

vector[index1] = vector[index2];

vector[index2] = auxiliar;

}

}

}

Afisare(vector, marime\_vector);

}

int main()

{

int vector[100], marime\_vector, dim;

cin >> marime\_vector >> dim;

Citire\_Vector(vector, marime\_vector);

Sortare\_vector(vector, marime\_vector, dim);

}

1. Se da un vector de numere intregi pozitive de dimensiune “dim”. Sa se sorteze elementele crescator pana la jumatatea sirului, iar de la jumatate la final toate elementele sa se sorteze descrescator.

#include <iostream>

using namespace std;

void Citire\_Vector(int vector[], int marime\_vector)

{

for (int index = 0; index < marime\_vector; index++)

{

cin >> vector[index];

}

}

void Afisare(int vector[], int marime\_vector)

{

for (int index = 0; index < marime\_vector; index++)

{

cout << vector[index] << " ";

}

}

void Sortare\_vector(int vector[], int marime\_vector)

{

for (int index1 = 0; index1 < marime\_vector/2 -1; index1++)

{

for (int index2 = index1 + 1; index2 < marime\_vector/2; index2++)

{

if (vector[index1] > vector[index2])

{

int auxiliar = vector[index1];

vector[index1] = vector[index2];

vector[index2] = auxiliar;

}

}

}

for (int index1 = marime\_vector / 2; index1 < marime\_vector- 1; index1++)

{

for (int index2 = index1 + 1; index2 < marime\_vector; index2++)

{

if (vector[index1] < vector[index2])

{

int auxiliar = vector[index1];

vector[index1] = vector[index2];

vector[index2] = auxiliar;

}

}

}

Afisare(vector, marime\_vector);

}

int main()

{

int dim,vector[100];

cin >> dim;

Citire\_Vector(vector, dim);

Sortare\_vector(vector, dim);

}

1. Sa se testeze daca un vector de numere intregi da teste sortat crescator.

a. Daca da, atunci fiind dat un element “elem”, sa se insereze elementul la locul lui, adica astfel incat sa ramana un sir sortat crescator.

b. Daca nu este sortat crescator, atunci sa se identifice daca exista un eventual element care strica “ordinea”, iar daca exista unul singur, atunci sa se elimine acel element.

#include <iostream>

using namespace std;

void Citire\_Vector(int vector[], int marime\_vector)

{

for (int index = 0; index < marime\_vector; index++)

{

cin >> vector[index];

}

}

void Afisare(int vector[], int marime\_vector)

{

for (int index = 0; index < marime\_vector; index++)

{

cout << vector[index] << " ";

}

}

int Aflare\_Pozitie\_ELEM(int vector[], int marime\_vector, int elem)

{

for (int index = 0; index < marime\_vector-1; index++)

{

if (vector[index] <= elem && vector[index + 1] > elem)

return index + 1;

}

}

void Inserare\_ELEM(int vector[], int marime\_vector,int elem)

{

int pozitie = Aflare\_Pozitie\_ELEM(vector, marime\_vector, elem);

for (int index = marime\_vector; index > pozitie; index--)

{

vector[index] = vector[index - 1];

}

marime\_vector++;

vector[pozitie] = elem;

Afisare(vector, marime\_vector);

}

void Eliminare\_element(int vector[], int& dim, int pozitie)

{

for (int index = pozitie; index < dim; index++)

{

vector[index] = vector[index + 1];

}

dim--;

}

int Verificare\_Sortare(int vector[], int& marime\_vector,int elem)

{

for (int index1 = 0; index1 < marime\_vector-1; index1++)

{

for (int index2 = index1 + 1; index2 < marime\_vector; index2++)

{

if (vector[index1] > vector[index2])

{

Eliminare\_element(vector, marime\_vector, index1);

return Verificare\_Sortare(vector, marime\_vector, elem);

}

}

}

}

int main()

{

int dim, vector[100],elem;

cin >> dim>>elem;

Citire\_Vector(vector, dim);

Verificare\_Sortare(vector, dim,elem);

Inserare\_ELEM(vector, dim, elem);

}

1. Se citeste un vector de dim elemente dintr-un fisier text, “cautari.txt”. Sa se sorteze crescator vectorul si sa se returneze pozitiile fiecarui element sortat din vectorul initial folosind cautarea secventiala.

Exemplu: *35, 21, 8, 71, 93, 4, 20. Se afiseaza: 6, 3, 7, 2, 1, 4, 5*

**Metoda 1(vector de pozitie):**

#include <iostream>

#include <fstream>

using namespace std;

ifstream fin("cautari.txt");

void Afisare(int vector[], int marime\_vector)

{

for (int index = 1; index <= marime\_vector; index++)

{

cout << vector[index] << " ";

}

}

void Interschimbare(int& numar1, int& numar2)

{

int auxiliar = numar1;

numar1 = numar2;

numar2 = auxiliar;

}

void Ordonare\_Vector(int vector[], int dim)

{

int vector\_pozitie[100];

for (int index = 1; index <= dim; index++)

vector\_pozitie[index] = index;

for (int index1 = 1; index1 < dim; index1++)

{

for (int index2 = index1 + 1; index2 <= dim; index2++)

{

if (vector[index1] > vector[index2])

{

Interschimbare(vector[index1], vector[index2]);

Interschimbare(vector\_pozitie[index1], vector\_pozitie[index2]);

}

}

}

Afisare(vector\_pozitie, dim);

}

int main()

{

int vector[100], element\_vector, index = 1;

while (fin >> element\_vector)

{

vector[index++] = element\_vector;

}

int dim = --index;

fin.close();

Ordonare\_Vector(vector, dim);

}

**Metoda 2:**

#include <iostream>

#include <fstream>

using namespace std;

void Cautare\_Maxim(int& maxim,int& index\_maxim,int vector[],int dim)

{

maxim = vector[1];

for (int index = 2; index <= dim; index++)

{

if (maxim < vector[index])

{

maxim = vector[index];

index\_maxim = index;

}

}

}

void Cautare\_Minim(int vector[], int dim)

{

int minim, index\_minim;

Cautare\_Maxim(minim, index\_minim, vector, dim);

for (int index = 1; index <= dim; index++)

{

if (minim > vector[index] && vector[index] != -1)

{

minim = vector[index];

index\_minim = index;

}

}

vector[index\_minim] = -1;

cout << index\_minim << " ";

}

void Ordonare\_Vector(int vector[], int dim)

{

for (int index = 1; index <= dim; index++)

{

Cautare\_Minim(vector, dim);

}

}

int main()

{

ifstream fin("cautari.txt");

int vector[100], element\_vector, dim = 1;

while (fin >> element\_vector)

{

vector[dim++] = element\_vector;

}

fin.close();

Ordonare\_Vector(vector, --dim);

}

**Metoda 3(cautare secventiala):**

#include <iostream>

#include <fstream>

using namespace std;

void Copie\_Vector(int\* vector, int\* copie\_vector, int dimensiune)

{

for (int index = 1; index <= dimensiune; index++)

{

copie\_vector[index] = vector[index];

}

}

void Citire\_Vector(int\* vector, int& dimensiune,int\* vector\_ordonat)

{

ifstream fin("cautari.txt");

int element\_vector;

while (fin >> element\_vector)

{

vector[dimensiune++] = element\_vector;

}

dimensiune--;

fin.close();

Copie\_Vector(vector, vector\_ordonat, dimensiune);

}

void Interschimbare(int& numar1, int& numar2)

{

int auxiliar = numar1;

numar1 = numar2;

numar2 = auxiliar;

}

void Ordonare\_Vector(int\* vector, int dimensiune)

{

for (int index = 1; index < dimensiune; index++)

{

for (int index2 = index + 1; index2 <= dimensiune; index2++)

{

if (vector[index] > vector[index2])

{

Interschimbare(vector[index], vector[index2]);

}

}

}

}

void Afisare\_Pozitii(int\* vector, int\* vector\_ordonat, int dimensiune)

{

Ordonare\_Vector(vector\_ordonat, dimensiune);

for (int index = 1; index <= dimensiune; index++)

{

bool cautare\_element = false;

for (int index2 = 1; index2 <= dimensiune && cautare\_element==false; index2++)

{

if (vector\_ordonat[index] == vector[index2])

{

cautare\_element = true;

cout << index2 << " ";

}

}

}

}

int main()

{

int vector[100],vector\_ordonat[100], dim = 1;

Citire\_Vector(vector, dim, vector\_ordonat);

Afisare\_Pozitii(vector, vector\_ordonat, dim);

}

1. Se citeste un vector de n numere intregi ordonat descrescator din fisier. Se citeste un al doilea vector de m numere intregi din consola. Folosind cautarea binara sa se determine care elemente din al doilea vector apar in primul si pe ce poziții. Rezultatul se va scrie in fisier.

#include <iostream>

#include <fstream>

using namespace std;

void Citire\_Vector\_din\_FISIER(int\* vector, int& dimensiune)

{

ifstream fin("cautari.in");

int element\_vector;

while (fin >> element\_vector)

{

vector[dimensiune++] = element\_vector;

}

dimensiune--;

fin.close();

}

void Citire\_Vector(int vector[], int& dimensiune)

{

cin >> dimensiune;

for (int index = 0; index < dimensiune; index++)

{

cin >> vector[index];

}

}

void Interschimbare(int& numar1, int& numar2)

{

int auxiliar = numar1;

numar1 = numar2;

numar2 = auxiliar;

}

int Cautare\_Binara(int\* vector, int dimensiune, int element\_cautat)

{

int stanga = 0, dreapta = dimensiune - 1, pozitie = -1;

while (stanga < dreapta && pozitie == -1)

{

int mijloc = (stanga + dreapta) / 2;

if (vector[mijloc] == element\_cautat)

{

pozitie = mijloc;

}

else

{

if (vector[mijloc] > element\_cautat)

{

stanga = ++mijloc;

}

else

{

dreapta = --mijloc;

}

}

}

if (vector[stanga] == element\_cautat)

{

pozitie = stanga;

}

return pozitie;

}

void Afisare\_in\_Fisier(int\* vector1\_din\_fisier, int\* vector2\_din\_consola, int dimensiune1, int dimensiune2)

{

ofstream fout("cautari.out");

for (int index = 0; index < dimensiune2; index++)

{

int pozitie = Cautare\_Binara(vector1\_din\_fisier, dimensiune2, vector2\_din\_consola[index]);

if (pozitie != -1)

{

fout << "Elementul " << vector2\_din\_consola[index] << " din al doilea vector apare si in primul pe pozitia " << pozitie << endl;;

}

else

{

fout << "Elementul " << vector2\_din\_consola[index] << " din al doilea vector NU apare si in primul vector" << endl;;

}

}

fout.close();

}

int main()

{

int vector1[100], vector2[100];

int dim1 = 0, dim2;

Citire\_Vector\_din\_FISIER(vector1, dim1);

Citire\_Vector(vector2, dim2);

Afisare\_in\_Fisier(vector1, vector2, dim1, dim2);

}

1. Sa se genereze un sir de numere cu ajutorul funcției rand din stdlib.h. Sa se sorteze crescator toate elementele de pe poziții pare din acest sir de numere.

#include <iostream>

#include <stdlib.h>

using namespace std;

void Generare\_Sir(int\* vector, int& dimensiune)

{

cin >> dimensiune;

for (int index = 0; index < dimensiune; index++)

{

vector[index] = rand();

}

}

void Interschimbare(int& numar1, int& numar2)

{

int auxiliar = numar1;

numar1 = numar2;

numar2 = auxiliar;

}

void Sortare(int\* vector,int dimensiune)

{

for (int index1 = 0; index1 < dimensiune-1; index1 += 2)

{

for (int index2 = index1 + 2; index2 < dimensiune; index2 += 2)

{

if (vector[index1] > vector[index2])

{

Interschimbare(vector[index1], vector[index2]);

}

}

}

}

void Afisare\_Vector\_Sortat(int\* vector, int dimensiune)

{

Sortare(vector, dimensiune);

for (int index = 0; index < dimensiune; index++)

{

cout << vector[index] << " ";

}

}

int main()

{

int vector[100],dimensiune;

Generare\_Sir(vector, dimensiune);

Afisare\_Vector\_Sortat(vector, dimensiune);

}

1. Se citeste o matrice cu n linii si m coloane. Sa se genereze un sir de 100 numere random si sa se afiseze câte dintre aceste elemente sunt pe prima linie a matricei.

Obs: Se va aplica o operație de sortare asupra liniilor matricei astfel incât prima coloana sa fie sortata crescator, ulterior se va apela cautarea binara.

#include <iostream>

#include <stdlib.h>

using namespace std;

void Citire\_Matrice(int matrice[][100], int& dimensiune\_linii, int& dimensiune\_coloane)

{

cin >> dimensiune\_linii >> dimensiune\_coloane;

for (int index = 0; index < dimensiune\_linii; index++)

{

for (int index2 = 0; index2 < dimensiune\_coloane; index2++)

{

cin >> matrice[index][index2];

}

}

}

void Interschimbare\_Linii(int matrice[][100], int dimensiune\_coloane, int numar\_linie1, int numar\_linie2)

{

for (int index = 0; index < dimensiune\_coloane; index++)

{

int auxiliar = matrice[numar\_linie1][index];

matrice[numar\_linie1][index] = matrice[numar\_linie2][index];

matrice[numar\_linie2][index] = auxiliar;

}

}

void Sortare\_dupa\_Linii(int matrice[][100], int dimensiune\_linii, int dimensiune\_coloane)

{

for (int index1 = 0; index1 < dimensiune\_linii - 1; index1++)

{

for (int index2 = index1 + 2; index2 < dimensiune\_linii; index2++)

{

if (matrice[index1][0] > matrice[index2][0])

{

Interschimbare\_Linii(matrice, dimensiune\_coloane, index1, index2);

}

}

}

}

bool Cautare\_Binara(int matrice[][100], int dimensiune, int element\_cautat)

{

int stanga = 1, dreapta = dimensiune, pozitie = -1;

while (stanga < dreapta && pozitie == -1)

{

int mijloc = (stanga + dreapta) / 2;

if (matrice[0][mijloc] == element\_cautat)

{

pozitie = mijloc;

}

else

{

if (matrice[0][mijloc] < element\_cautat)

{

stanga = ++mijloc;

}

else

{

dreapta = --mijloc;

}

}

}

if (matrice[0][stanga] == element\_cautat)

{

pozitie = stanga;

}

if (pozitie == -1) return false;

return true;

}

void Generare\_Sir\_de\_Numere(int matrice[][100], int dimensiune\_linii, int dimensiune\_coloane)

{

int Numar\_de\_Elemente = 0;

Sortare\_dupa\_Linii(matrice, dimensiune\_linii, dimensiune\_coloane);

for (int index = 0; index < 100; index++)

{

int element\_random = rand(), verificare = 0;

verificare = Cautare\_Binara(matrice, dimensiune\_linii, element\_random);

if (verificare == true)

{

cout << element\_random << " ";

}

}

}

int main()

{

int matrice[100][100], dimensiune\_linii, dimensiune\_coloane;

Citire\_Matrice(matrice, dimensiune\_linii, dimensiune\_coloane);

Generare\_Sir\_de\_Numere(matrice, dimensiune\_linii, dimensiune\_coloane);

}