Tema

1. Se citește un număr natural n și cele n cele n elemente ale unui vector. Să se înlocuiască elementele de pe pozițiile pare cu cea mai mică cifra din număr iar cele de pe pozițiile impare cu 0 dacă numărul e prim și cu 1 altfel.

#include <iostream>

using namespace std;

int Verify\_if\_the\_Number\_is\_Prime(int number)

{

if (number < 2 || (number % 2 == 0 && number != 2))

return 0;

else for (int index = 3; index \* index <= number; index = index + 2)

{

if (number % index == 0)

return 0;

}

return 1;

}

int Return\_the\_Biggest\_Digit(int number)

{

int maximum\_digit = 0;

while (number > 0)

{

if (maximum\_digit < number % 10)

{

maximum\_digit = number % 10;

}

number /= 10;

}

return maximum\_digit;

}

void Array(int number, int array[])

{

for (int index1 = 0; index1 < number; index1++)

{

cin >> array[index1];

}

}

void Modify\_Array(int number, int array[])

{

int new\_array[100];

Array(number, array);

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

{

if (index % 2 == 0)

new\_array[index] = Return\_the\_Biggest\_Digit(array[index]);

else

new\_array[index] = Verify\_if\_the\_Number\_is\_Prime(array[index]);

}

for (int index2=0;index2<number;index2++)

{

cout << new\_array[index2]<<" ";

}

}

int main()

{

int n, array[100];

cin >> n;

Modify\_Array(n, array);

return 0;

}

1. Se citește un număr natural n și cele n cele n elemente ale unui vector. Să se șteargă din vector acele elemente care sunt egale cu suma vecinilor săi.

#include <iostream>

using namespace std;

void Reading\_Array(int array[], int number)

{

for (int index1 = 0; index1 < number; index1++)

{

cin >> array[index1];

}

}

bool Verify\_Sum(int neighbor1, int number, int neighbor2)

{

if (neighbor1 + neighbor2 == number)

return true;

else

return false;

}

void Removing\_an\_Element\_from\_Array(int array[], int &number, int position)

{

for (int index2 = position; index2 < number; index2++)

{

array[index2] = array[index2 + 1];

}

number--;

}

void Modify\_Array(int array[], int number)

{

Reading\_Array(array, number);

for (int index3 = 1; index3 < number - 1; index3++)

{

if (Verify\_Sum(array[index3 - 1], array[index3], array[index3 + 1]) == true)

{

Removing\_an\_Element\_from\_Array(array, number, index3);

}

}

for (int index4 = 0; index4 < number; index4++)

{

cout << array[index4] << " ";

}

}

int main()

{

int n, array[100];

cin >> n;

Modify\_Array(array, n);

return 0;

}

1. Se citește un număr natural n și cele n cele n elemente ale unui vector. Să se insereze în vector o valoare între oricare două valori vecine suma și produsul acestora.

Obs: Determinarea sumei și a produsului a două numere se va realiza în cadrul unui singure funcții.

#include <iostream>

using namespace std;

void Citire\_Vector(int numar, int vector[])

{

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

{

cin >> vector[index];

}

}

void Returnare\_Vector(int vector[],int index,int &n)

{

if (n >= 100) cout << "Eroare";

else

{

for (int i = n - 1; i >= index; i--)

vector[i + 2] = vector[i];

n = n + 2;

vector[index] = vector[index - 1] + vector[index + 2];

vector[index + 1] = vector[index - 1] \* vector[index + 2];

for (int i = 0; i < n; i++)

{

cout << vector[i] << " ";

}

}

}

int main()

{

int n, vector[100], random\_value;

cin >> n >> random\_value;

Citire\_Vector(n, vector);

Returnare\_Vector(vector, random\_value, n);

}

1. Se citeșc două numere naturale n și k și cele n elemente ale unui vector. Să se deplaseze spre dreapta elementele vectorilor cu k-poziții. Primele k-elemente ale vectorului vor fi
2. zerorizate iar dimensiunea vectorului va crește k elemnete

*Ex: v = [1,2,3,4,5,6,7], k=3 afișare: [0,0,0,1,2,3,4,5,6,7]*

#include <iostream>

using namespace std;

void Citire\_Vector(int vector[], int numar\_de\_elemente)

{

for (int index = 0; index < numar\_de\_elemente; index++)

{

cin >> vector[index];

}

}

void Inserare\_in\_Vector\_de\_k\_ori(int vector[], int& numar\_de\_elemente, int numar\_de\_pozitii)

{

while (numar\_de\_pozitii > 0)

{

for (int index = numar\_de\_elemente - 1; index >= 0; index--)

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

numar\_de\_elemente = numar\_de\_elemente + 1;

numar\_de\_pozitii--;

}

}

void Afisare\_Vector(int vector[], int numar\_de\_elemente, int numar\_de\_pozitii)

{

Citire\_Vector(vector, numar\_de\_elemente);

if (numar\_de\_elemente >= 100) cout << "Eroare";

else

{

Inserare\_in\_Vector\_de\_k\_ori(vector, numar\_de\_elemente, numar\_de\_pozitii);

for (int index = 0; index < numar\_de\_pozitii; index++)

{

vector[index] = 0;

}

for (int index2 = 0; index2 < numar\_de\_elemente; index2++)

{

cout << vector[index2] << " ";

}

}

}

int main()

{

int n, k, vector[100];

cin >> n >> k;

Afisare\_Vector(vector, n, k);

return 0;

}

1. înlocuite cu ultimele k elemente din vector astfel dimensiunea vectorului nu va suferi modificări

*Ex: v = [1,2,3,4,5,6,7], k=3 afișare: [5,6,7,1,2,3,4]*

#include <iostream>

using namespace std;

void Citire\_Vector(int vector[], int numar\_de\_elemente)

{

for (int index = 0; index < numar\_de\_elemente; index++)

{

cin >> vector[index];

}

}

void Inserare\_in\_Vector\_de\_k\_ori(int vector[], int numar\_de\_elemente, int numar\_de\_pozitii)

{

while (numar\_de\_pozitii > 0)

{

for (int index = numar\_de\_elemente - 1; index >= 0; index--)

{

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

}

vector[0] = vector[numar\_de\_elemente];

numar\_de\_pozitii--;

}

}

void Afisare\_Vector(int vector[], int numar\_de\_elemente, int numar\_de\_pozitii)

{

Citire\_Vector(vector, numar\_de\_elemente);

if (numar\_de\_elemente >= 100) cout << "Eroare";

else

{

Inserare\_in\_Vector\_de\_k\_ori(vector, numar\_de\_elemente, numar\_de\_pozitii);

for (int index2 = 0; index2 < numar\_de\_elemente; index2++)

{

cout << vector[index2] << " ";

}

}

}

int main()

{

int n, k, vector[100];

cin >> n >> k;

Afisare\_Vector(vector, n, k);

return 0;

}

1. Se dă un vector v cu n elemente. Să se elimine din vector elementul de cu indicele de poziție p (citit de la tastatură) iar p să se insereze la începutul vectorului. Să se afișeze noul vector. *Ex: 2 51 2 91 3 cu p = 3, afiseaza 3 2 51 2 3*

#include <iostream>

using namespace std;

void Citire\_Vector(int vector[], int numar\_de\_elemente )

{

for (int index = 0; index < numar\_de\_elemente; index++)

{

cin >> vector[index];

}

}

void Eliminare\_Element\_de\_pe\_Pozitia\_p(int vector[], int numar\_de\_elemente, int indicele\_de\_pozitie)

{

for (int index = indicele\_de\_pozitie; index < numar\_de\_elemente-1; index++)

{

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

}

numar\_de\_elemente--;

}

void Inserare\_p\_pe\_Prima\_Pozitie\_din\_Vector(int vector[], int numar\_de\_elemente, int indicele\_de\_pozitie)

{

for (int index = numar\_de\_elemente + 1; index >= 0; index--)

{

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

}

numar\_de\_elemente++;

vector[0] = indicele\_de\_pozitie;

}

void Afisare\_Vector(int vector[], int numar\_de\_elemente, int indicele\_de\_pozitie)

{

if (numar\_de\_elemente >= 100) cout << "Eroare";

else

{

Eliminare\_Element\_de\_pe\_Pozitia\_p(vector, numar\_de\_elemente, indicele\_de\_pozitie);

Inserare\_p\_pe\_Prima\_Pozitie\_din\_Vector(vector, numar\_de\_elemente, indicele\_de\_pozitie);

for (int index = 0; index < numar\_de\_elemente; index++)

{

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

}

}

}

int main()

{

int v[100], n, p;

cin >> n >> p;

Citire\_Vector(v, n);

Afisare\_Vector(v, n, p);

}

1. Se citeșc două numere naturale n și m și doi vectori crescători de lungime n, respectiv m. Să se creeze și să se afișeze vectorul ordonat crescător obținut prin reuniunea celor doi vector. *Ex: v1 = [1,2,5,10,13,20] v2 = [1,6,7,9,15,25], afisare: [1,1,2,5,6,7,9,10,13,15,20,25]*

#include <iostream>

using namespace std;

void Citire\_Vector(int vector[], int numar\_de\_elemente)

{

for (int index = 0; index < numar\_de\_elemente; index++)

{

cin >> vector[index];

}

}

void Interclasare\_v1\_si\_v2(int vector\_final[], int vector1[], int vector2[],int numar\_de\_elemente1,int numar\_de\_elemente2, int& numar\_de\_elemente\_final)

{

int index1 = 0, index2 = 0, index\_final = 0;

while (index1 < numar\_de\_elemente1 && index2 < numar\_de\_elemente2)

{

if (vector1[index1] < vector2[index2])

{

vector\_final[index\_final++] = vector1[index1++];

}

else

{

vector\_final[index\_final++] = vector2[index2++];

}

}

while (index1 < numar\_de\_elemente1)

{

vector\_final[index\_final++] = vector1[index1++];

}

while (index2 < numar\_de\_elemente2)

{

vector\_final[index\_final++] = vector2[index2++];

}

numar\_de\_elemente\_final = index\_final;

}

void Afisare\_Vector(int vector1[], int numar\_de\_elemente1, int vector2[], int numar\_de\_elemente2)

{

int vector\_final[201],numar\_de\_elemente\_total;

if (numar\_de\_elemente1 >= 100 || numar\_de\_elemente2>=100) cout << "Eroare";

else

{

Interclasare\_v1\_si\_v2(vector\_final, vector1, vector2, numar\_de\_elemente1, numar\_de\_elemente2, numar\_de\_elemente\_total);

for (int index = 0; index < numar\_de\_elemente\_total; index++)

{

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

}

}

}

int main()

{

int v1[100],v2[100], n, m;

cin >> n >> m;

Citire\_Vector(v1, n);

Citire\_Vector(v2, m);

Afisare\_Vector(v1, n,v2,m);

}

1. Se citește un număr natural n și n medii (numere reale cu doua zecimale cu valori cuprinse intre 1 şi 10). Să se afişeze o statistică a mediilor astfel: numărul de medii cuprinse în intervalul [1,2], numărul de medii cuprinse în intervalul (2,3], …, numărul de medii cuprinse în intervalul (9,10]. Cat este complexitatea gasita? Comentati.

#include <iostream>

using namespace std;

void Citire\_Medii(int interval[], int numar\_de\_elemente)

{

float medie;

for (int index1 = 0; index1 < numar\_de\_elemente; index1++)

{

cin >> medie;

if ((int)medie == medie)

interval[(int)medie]++;

else

interval[(int)medie + 1]++;

}

}

void Statistica\_Medii(int numar\_de\_elemente)

{

int interval\_medii[15] = { 0 };

Citire\_Medii(interval\_medii, numar\_de\_elemente);

cout << "Numarul de medii cuprinse in intervalul [ " << 1 << " , " << 2 << " ] este " << interval\_medii[1] + interval\_medii[2] << endl;

for (int index2 = 2; index2 < 10; index2++)

{

cout << "Numarul de medii cuprinse in intervalul ( " << index2 << " , " << index2 + 1 << " ] este " << interval\_medii[index2 + 1] << endl;

}

}

int main()

{

int n;

cin >> n;

Statistica\_Medii(n);

return 0;

}

**BONUS:**

#include <iostream>

using namespace std;

void Citire\_Medii(int interval[], int numar\_de\_elemente)

{

float medie;

for (int index1 = 0; index1 < numar\_de\_elemente; index1++)

{

cin >> medie;

if ((int)medie == medie)

interval[(int)medie]++;

else

interval[(int)medie + 1]++;

}

}

void Statistica\_Medii(int numar\_de\_elemente)

{

int interval\_medii[15] = { 0 };

Citire\_Medii(interval\_medii, numar\_de\_elemente);

cout << "Numarul de medii cuprinse in intervalul [ " << 1 << " , " << 2 << " ] este " << interval\_medii[1] + interval\_medii[2] << endl;

for (int index2 = 2; index2 < 10; index2++)

{

cout << "Numarul de medii cuprinse in intervalul ( " << index2 << " , " << index2 + 1 << " ] este " << interval\_medii[index2 + 1] << endl;

}

}

void Input\_Averages(int interval\_of\_averages[], int number\_of\_elements)

{

float average;

for (int counter = 0; counter < number\_of\_elements; counter++)

{

cin >> average;

if ((int)average == average)

interval\_of\_averages[(int)average]++;

else

interval\_of\_averages[(int)average + 1]++;

}

}

void Statistic\_of\_Averages(int number\_of\_elements)

{

int interval\_of\_averages[15] = { 0 };

Input\_Averages(interval\_of\_averages, number\_of\_elements);

cout << "Number of averages contained in the interval [ " << 1 << " , " << 2 << " ] is " << interval\_of\_averages[1] + interval\_of\_averages[2] << endl;

for (int counter = 2; counter < 10; counter++)

{

cout << "Number of averages contained in the interval ( " << counter << " , " << counter + 1 << " ] is " << interval\_of\_averages[counter + 1] << endl;

}

}

void Moyennes\_d\_Entrée(int intervall\_de\_moyennes[], int numbre\_d\_éléments)

{

float moyenne;

for (int compteur = 0; compteur < numbre\_d\_éléments; compteur++)

{

cin >> moyenne;

if ((int)moyenne == moyenne)

intervall\_de\_moyennes[(int)moyenne]++;

else

intervall\_de\_moyennes[(int)moyenne + 1]++;

}

}

void Statistiques\_de\_Moyennes(int numbre\_d\_éléments)

{

int intervall\_de\_moyennes[15] = { 0 };

Moyennes\_d\_Entrée(intervall\_de\_moyennes, numbre\_d\_éléments);

cout << "Nombre de moyennes contenues dans l'intervalle [ " << 1 << " , " << 2 << " ] est " << intervall\_de\_moyennes[1] + intervall\_de\_moyennes[2] << endl;

for (int compteur = 2; compteur < 10; compteur++)

{

cout << "Nombre de moyennes contenues dans l'intervalle ( " << compteur << " , " << compteur + 1 << " ] est " << intervall\_de\_moyennes[compteur + 1] << endl;

}

}

void Input\_Durchschnitte(int Intervall\_von\_Durchschnitten[], int Anzahl\_der\_Elemente)

{

float Durchschnitt;

for (int Zähler = 0; Zähler < Anzahl\_der\_Elemente; Zähler++)

{

cin >> Durchschnitt;

if ((int)Durchschnitt == Durchschnitt)

Intervall\_von\_Durchschnitten[(int)Durchschnitt]++;

else

Intervall\_von\_Durchschnitten[(int)Durchschnitt + 1]++;

}

}

void Statistik\_der\_Durchschnitte(int Anzahl\_der\_Elemente)

{

int Intervall\_von\_Durchschnitten[15] = { 0 };

Input\_Durchschnitte(Intervall\_von\_Durchschnitten, Anzahl\_der\_Elemente);

cout << "Anzahl der im Intervall enthaltenen Durchschnittswerte [ " << 1 << " , " << 2 << " ] ist " << Intervall\_von\_Durchschnitten[1] + Intervall\_von\_Durchschnitten[2] << endl;

for (int Zähler = 2; Zähler < 10; Zähler++)

{

cout << "Anzahl der im Intervall enthaltenen Durchschnittswerte ( " << Zähler << " , " << Zähler + 1 << " ] ist " << Intervall\_von\_Durchschnitten[Zähler + 1] << endl;

}

}

void Promedios\_de\_Entrada(int intervalo\_de\_medios[], int número\_de\_elementos)

{

float media;

for (int contador = 0; contador < número\_de\_elementos; contador++)

{

cin >> media;

if ((int)media == media)

intervalo\_de\_medios[(int)media]++;

else

intervalo\_de\_medios[(int)media + 1]++;

}

}

void Estadística\_de\_Medios(int número\_de\_elementos)

{

int intervalo\_de\_medios[15] = { 0 };

Promedios\_de\_Entrada(intervalo\_de\_medios, número\_de\_elementos);

cout << "Número de promedios contenidos en el intervalo [ " << 1 << " , " << 2 << " ] es " << intervalo\_de\_medios[1] + intervalo\_de\_medios[2] << endl;

for (int contador = 2; contador < 10; contador++)

{

cout << "Número de promedios contenidos en el intervalo ( " << contador << " , " << contador + 1 << " ] es " << intervalo\_de\_medios[contador + 1] << endl;

}

}

void Médias\_de\_Entrada(int intervalo\_de\_medidas[], int número\_de\_elementos)

{

float média;

for (int balcão = 0; balcão < número\_de\_elementos; balcão++)

{

cin >> média;

if ((int)média == média)

intervalo\_de\_medidas[(int)média]++;

else

intervalo\_de\_medidas[(int)média + 1]++;

}

}

void Estatísticas\_de\_Médias(int número\_de\_elementos)

{

int intervalo\_de\_medidas[15] = { 0 };

Médias\_de\_Entrada(intervalo\_de\_medidas, número\_de\_elementos);

cout << "Número de médias contidas no intervalo [ " << 1 << " , " << 2 << " ] é " << intervalo\_de\_medidas[1] + intervalo\_de\_medidas[2] << endl;

for (int balcão = 2; balcão < 10; balcão++)

{

cout << "Número de médias contidas no intervalo ( " << balcão << " , " << balcão + 1 << " ] é " << intervalo\_de\_medidas[balcão + 1] << endl;

}

}

void Alegere\_Limba\_Program(int& n)

{

int limba;

cout << "Pentru limba ROMANA tastati 1." << endl;

cout << "Pentru limba ENGLEZA tastati 2. / For English type 2." << endl;

cout << "Pentru limba FRANCEZA tastati 3. / Pour langue française saisissez 3." << endl;

cout << "Pentru limba GERMANA tastati 4. / Für Deutsch Typ 4." << endl;

cout << "Pentru limba SPANIOLA tastati 5. / Para el español tipo 5." << endl;

cout << "Pentru limba PORTUGHEZA tastati 6. / Para português tipo 6." << endl;

cin >> limba;

switch (limba)

{

case 1:

cout << "Numarul de medii = ";

cin >> n;

Statistica\_Medii(n);

break;

case 2:

cout << "The number of averages = ";

cin >> n;

Statistic\_of\_Averages(n);

break;

case 3:

cout << "Nombre d'environnements = ";

cin >> n;

Statistiques\_de\_Moyennes(n);

break;

case 4:

cout << "Die Anzahl der Durchschnittswerte = ";

cin >> n;

Statistik\_der\_Durchschnitte(n);

break;

case 5:

cout << "El número de promedios =";

cin >> n;

Estadística\_de\_Medios(n);

break;

case 6:

cout << "O número de médias =";

cin >> n;

Estatísticas\_de\_Médias(n);

break;

}

}

int main()

{

int n;

Alegere\_Limba\_Program(n);

return 0;

}

1. Un număr natural se reţine într-un vector (număr mare), astfel încât fiecare componentă a vectorului conţine câte o cifră a numărului. Să se înmulţească numărul cu un număr între 1 şi 9 (random generat).

#include <iostream>

using namespace std;

void Citire\_Numar(int numar[], int numar\_de\_cifre)

{

for (int index = 0; index < numar\_de\_cifre; index++)

{

cin >> numar[index];

}

}

void Inserare\_o\_Noua\_Unitate\_in\_Numar(int numar[], int& numar\_de\_cifre, int unitate\_noua)

{

for (int index = numar\_de\_cifre - 1; index >= 0; index--)

{

numar[index + 1] = numar[index];

}

numar[0] = unitate\_noua;

numar\_de\_cifre++;

}

void Inmultire\_Numar\_cu\_Numarul\_Generat(int numar[], int& numar\_cifre, int numar\_generat)

{

int transport = 0;

for (int index = numar\_cifre - 1; index >= 0; index--)

{

numar[index] = numar[index] \* numar\_generat;

if (transport != 0)

{

numar[index] += transport;

transport = 0;

}

if (numar[index] > 9)

transport = numar[index] / 10;

else transport = 0;

numar[index] = numar[index] % 10;

}

if (transport != 0)

Inserare\_o\_Noua\_Unitate\_in\_Numar(numar, numar\_cifre, transport);

}

void Afisare\_Produs(int numar[], int numar\_de\_cifre, int numar\_generat)

{

Citire\_Numar(numar, numar\_de\_cifre);

Inmultire\_Numar\_cu\_Numarul\_Generat(numar, numar\_de\_cifre, numar\_generat);

for (int index = 0; index < numar\_de\_cifre; index++)

{

cout << numar[index];

}

}

int main()

{

int numar\_cifre, numar[100], numar\_generat;

cin >> numar\_cifre;

//numar\_generat = rand() % 9 + 1;

numar\_generat = 9;

Afisare\_Produs(numar, numar\_cifre, numar\_generat);

return 0;

}