Vasile Denisa-Georgiana
Grupa 204
Informatica ZI
Anul I
Facultatea de Matematica si Informatica

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;
}
```

2. 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;
}
```

3. 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);
}
```

4. Se citesc 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

    a. 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;
}
```

b. 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;
			}
```

5. 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);
}
```

6. Se citesc 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);
```

```
}
```

7. 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;
}
```

8. 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;
	}
```