**STRUCTURA RAPORTULUI**

Rezultatele practicii se vor descrie într-un raport cu următoarea structură:

* Foaie de titlu
* Cuprins
* Întroducere
* Conținutul activităților și sarcinilor de lucru
* Descrierea modului de elaborare a stagiului de practică
* Alte documente specifice a domeniului de formare
* Observații generale
* Bibliografie
* Anexe

Notă ! La elaborarea și redactarea raportului să se țină cont de următoarele specificații:

1. Concluzia:
   1. rezumă rezultatele cercetării și importanța lor în raport cu stadiul actual al temei cercetate;
   2. evidențiază complexitatea cercetării, fără să ignore dificultățile care fac cercetarea imperfectă;
   3. indică posibile cercetări viitoare, plecând chiar de la ceea ce nu s-a realizat în lucrarea actuală;
   4. oferă un comentariu personal despre rezultatul cercetării în raport cu obiectivele propuse, care au fost enunțate în introducere.
2. Realizați raportul practicii folosind un editor de text, cu următoarele setări:
   1. Parametri pagină: Mărimea - A4, margini: câmpul din stânga – 30 mm, de sus – 20 mm, de jos – 20 mm, din dreapta – 10 mm.
   2. Titlul: Font - Times New Roman, Mărime: 14, aliniere: Center,
   3. Corpul textului: Font - Times New Roman, Mărime: 12, Aliniere: justify, Spațiul dintre rânduri: 1.5 lines.
   4. Listing-ului programului: Font – CourierNew , Mărime: 10, Aliniere: left, Spațiul dintre rânduri: 1 line.
   5. Imagine, scheme: Numărul și denumirea se notează sub imagine, centrat.
   6. Numerotarea paginilor în partea de jos, centrat.
   7. Cuprinsul, bibliografia să fie creată cu ajutorul opțiunilor corespunzătoare.
   8. Foaia de titlu să fie elaborată conform modelului propus

Ministerul Educației, Culturii și Cercetării al Republicii Moldova

IP Colegiul „Iulia Hasdeu” din Cahul

**RAPORTUL**

**STAGIULUI DE PRACTICĂ**

Elevul: **Batir Daniel**

Grupa: **PAPP1921**

Specialitatea: **Programarea si analiza produselor program**

Baza de practică: **IP Colegiul “Iulia Hasdeu” din Cahul**

Conducătorul stagiului de practică

de la unitatea economică

**[Nume, prenume]**

Conducătorul stagiului de practică

de la IP CIHC

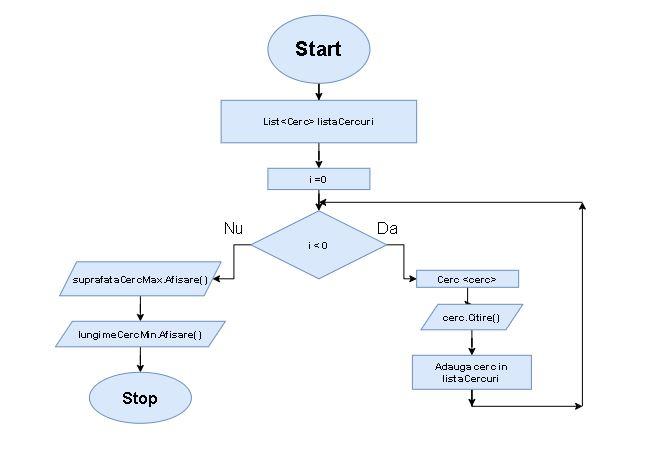
**[Nume, prenume]**

Cahul **2021**

**CUPRINS**

* Schema bloc generala a algoritmului realizat .
* Listingul programului .
* Rezultatele testarii aplicatiilor :
* Datele de intrare
* Datele de iesire (corespunzatoare datelor de intrare)
* Concluzii .
* Bibliografie .
* Anexe .

**Varianta 5**

**Tema 1 **

**Schema bloc generala a algoritmului realizat :**

using System;

using System.Collections.Generic;

using System.Linq;

namespace Tema\_1.Lucrul\_cu\_clasele

{

//Cream clasa cerc

class Cerc

{

//Proprietatea clasei lungimea razei

public double lungimeaRazei { get; set; }

//Metoda de initializare

public Cerc()

{

}

//Metoda Citire pentru citirea datelor despre cerc

public void Citire()

{

Console.Write("Lungimea razei : ");

lungimeaRazei = double.Parse(Console.ReadLine());

}

//Metoda Afisare pentru afisarea datelor despre cerc

public void Afisare()

{

Console.WriteLine($"Lungimea razei : {lungimeaRazei}");

Console.WriteLine($"Lungimea : {DeterminareLungimeCerc():F2}");

Console.WriteLine($"Suprafata : {DeterminareSuprafataCerc():F2}");

Console.WriteLine($"Diametrul : {DeterminareDiametruCerc()}");

}

//Metoda DeterminareLungimeCerc pentru determinarea lungimii cercului

public double DeterminareLungimeCerc() => 2 \* Math.PI \* lungimeaRazei;

//Metoda DeterminareSuprafataCerc pentru determinarea suprafetei cercului

public double DeterminareSuprafataCerc() => Math.PI \* lungimeaRazei \* lungimeaRazei;

//Metoda DeterminareDiametruCerc pentru determinarea diametrului cercului

public double DeterminareDiametruCerc() => 2 \* lungimeaRazei;

}

class Program

{

static void Main(string[] args)

{

//Cream o lista in care vor fi adaugate datele depre instantele clasei Cerc

List<Cerc> listaCercuri = new List<Cerc>();

for (int i = 0; i < 2; i++)

{

//Instanta clasei Cerc

Cerc cerc = new Cerc();

//Citim datele de la tastatura despre instante

Console.WriteLine("Citim datele despre cercul " + (i + 1) + " : ");

cerc.Citire();

//Spatiu

Console.WriteLine();

//Adaugam instanta in Lista cercuri

listaCercuri.Add(cerc);

}

//Afisam datele despre cercul cu suprafata cercului maxima

Console.WriteLine("----<Informatia despre cercul cu suprafata maxima>----");

double suprafataMax = listaCercuri.Max(cerc => cerc.DeterminareSuprafataCerc());

var suprafataCercMax = listaCercuri.First(cerc => cerc.DeterminareSuprafataCerc() == suprafataMax);

suprafataCercMax.Afisare();

//Spatiu

Console.WriteLine();

//Afisam datele despre cercul cu lungimea cercului minima

Console.WriteLine("----<Informatia despre cercul cu lungimea discului minima>----");

double lungimeMin = listaCercuri.Min(cerc => cerc.DeterminareLungimeCerc());

var lungimeCercMin = listaCercuri.First(cerc => cerc.DeterminareLungimeCerc() == lungimeMin);

lungimeCercMin.Afisare();

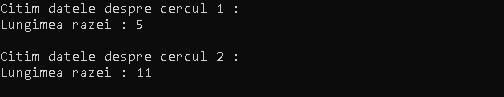
Console.ReadKey();

}

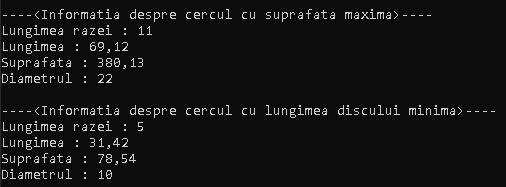
}

}

**Listingul programului :**



**Datele de intrare :**

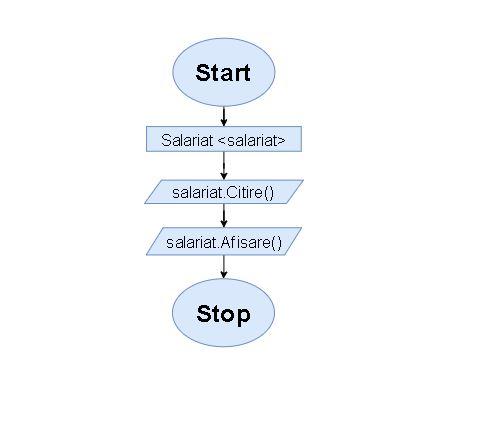


**Datele de iesire :**

Realizand aplicatia respectiva am apelat la toate cunostintele si informatiile invatate in semestrul precedent la modulul „Asistenta pentru programarea orientata pe obiecte” , am utilizat clase,constructori,metode,liste generice si instantele claselor . Aplicatia Cerc contine o clasa cerc in care sunt create proprietati si metodele de citire , de afisare si calcul a suprafatei,lungimii si diametrului care vor fi apelate in metoda Main() cu ajutorul instantei clasei astfel se va afisa la ecran informatiile despre cercurile adaugate in lista .

**Concluzie :**

**Tema 2**

****

**Schema bloc generala a algoritmului realizat :**

using System;

namespace Tema\_2.Mostenire

{

//Cream clasa Persoana

class Persoana

{

//Proprietatile clasei

public string Nume { get; set; }

public string Prenume { get; set; }

public int AnulNasterii { get; set; }

public string CNP { get; set; }

//Metoda Citire pentru citirea datelor despre persoane

public void Citire()

{

Console.Write("Numele : ");

Nume = Console.ReadLine();

Console.Write("Prenumele : ");

Prenume = Console.ReadLine();

Console.Write("Anul nasterii : ");

AnulNasterii = int.Parse(Console.ReadLine());

Console.Write("CNP : ");

CNP = Console.ReadLine();

}

//Metoda Afisare pentru afisarea datelor despre persoane

public void Afisare()

{

Console.WriteLine($"Numele : {Nume}");

Console.WriteLine($"Prenumele : {Prenume}");

Console.WriteLine($"Anul nasterii : {AnulNasterii}");

Console.WriteLine($"CNP : {CNP}");

Console.WriteLine($"Varsta : {Varsta()}");

}

//Metoda Varsta care calculeaza varsta persoanei

public double Varsta() => DateTime.Now.Year - AnulNasterii;

}

//Cream clasa Salariat care este derivata clasei persoana

class Salariat : Persoana

{

//Proprietatile clasei

public double nrOreLucrate { get; set; }

public double plataPerOra { get; set; }

//Metoda Salariul care calculeaza salariul persoanei

public double Salariul() => nrOreLucrate \* plataPerOra;

//Metoda Citire pentru citirea datelor despre salariat

public void Citire()

{

Console.Write("Numarul de ore lucrate : ");

nrOreLucrate = double.Parse(Console.ReadLine());

Console.Write("Plata pentru o ora : ");

plataPerOra = double.Parse(Console.ReadLine());

base.Citire();

}

//Metoda Afisare pentru afisarea datelr despre persoana

public void Afisare()

{

Console.WriteLine($"Numarul de ore lucrate : {nrOreLucrate}");

Console.WriteLine($"Plata pentru o ora : {plataPerOra}");

base.Afisare();

}

}

class Program

{

static void Main(string[] args)

{

//Instanta clasei Salariat

Salariat salariat = new Salariat();

//Citim datele despre salariat

Console.WriteLine("Citim datele despre salariat : ");

salariat.Citire();

//Spatiu

Console.WriteLine();

Console.WriteLine("----------------------------------");

Console.WriteLine();

//Afisam datele despre salariat

Console.WriteLine("Informatiile despre salariat: ");

salariat.Afisare();

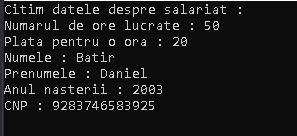
Console.ReadKey();

}

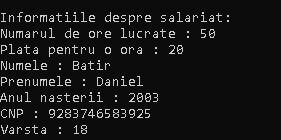
}

}

**Listingul programului :**

****

**Datele de intrare :**

****

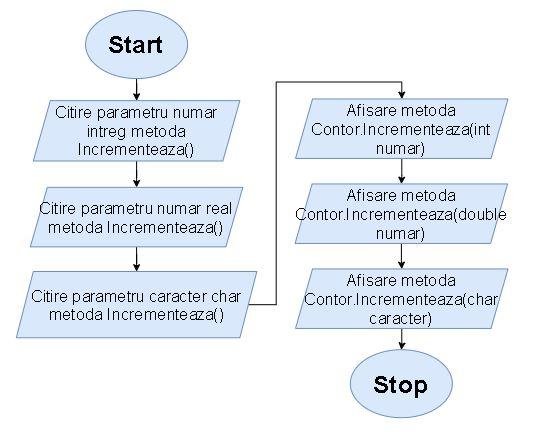
**Datele de iesire :**

La realizarea aplicatiei Persoana am utilizat clase,metode,liste generice si instante ale claselor , clasa Persoana este clasa de baza si are 4 campuri numele,prenumele,anul nasterii,cnp clasa derivata clasei Persoana este clasa Salariat care are 2 campuri numarul de ore lucrate si plata pentru o ora .

In metoda Main am creat o instanta a clasei derivate si am apelat toate metodele care sunt prezente in clasa , si anume metodele de citire si de afisare astfel aplicatia nostra poate sa citeasca datele despre persoane sa le calculeze Varsta si sa afiseze datele despre Persoane .

**Concluzie :**

**Tema 3**

****

**Schema bloc generala a algoritmului realizat :**

using System;

namespace Tema\_3.Polimorfismul\_metodelor\_claselor

{

//Cream clasa Contor

class Contor

{

//Metoda Incrementeaza, ce primește ca parametru un număr întreg și returnează numărul respectiv incrementat cu o unitate

public static int Incrementeaza(int numar)

{

return ++numar;

}

// Metoda Incrementează, ce primește ca parametru un număr real și returnează numărul respectiv incrementat cu 0.1

public static double Incrementeaza(double numar)

{

return numar + 0.1;

}

//Metoda Incremenează, ce primește ca parametru un caracter și returnează caracterul specific codului ascii al caracterului inițial, incrementat cu o valoare.

public static int Incrementeaza(char caracter)

{

int numar = (int)caracter;

return numar += 2;

}

}

class Program

{

static void Main(string[] args)

{

//Afisarea primei metode

Console.WriteLine($"Prima metoda incrementeaza un numar intreg 3 cu o unitate : " + Contor.Incrementeaza(3));

Console.WriteLine();

//Afisare a doua metoda

Console.WriteLine($"A doua metoda incrementeaza un numar real 20.5 cu 0.1 : {Contor.Incrementeaza(20.5):F1} ");

Console.WriteLine();

//Afisare a treia metoda

Console.WriteLine("A treia metoda primeste un caracter <<!>> si returnează caracterul specific codului ascii incrementat " +

"cu o valoare care este <<2>> : " + Contor.Incrementeaza('!'));

Console.ReadKey();

}

}

}

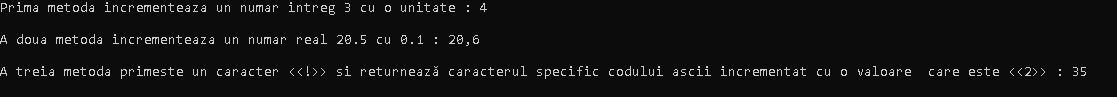
**Listingul programului :**

****

****

****

**Datele de intrare :**

****

**Datele de iesire :**

La realizarea aplicatiei Contor am utilizat clase,metode si proprietati , aplicatia respectiva se bazeaza mai mult pe incrementarea parametrilor metodelor dupa care se afiseaza numarul final incrementat cu unitatea respectiva , in cadrul aplicatiei prima metoda apelata in Main() incrementeaza un numar intreg cu o unitate , a doua metoda incrementeaza un numar real cu 0.1 si a treia metoda incrementeza un caracter returnat din codurile ascii,realizand aplicatia am aflat noi informatii ceea ce a ajutat la dezvoltarea mea ca programator .

**Concluzie :**

**Tema 4**

****

**Schema bloc generala a algoritmului realizat :**

using System;

using System.Collections.Generic;

using System.IO;

namespace Tema\_4.Utilizare\_fisiere\_text

{

//Cream clasa Melodie

class Melodie

{

//Proprietatile clasei

public string Denumire { get; set; }

public string Interpret { get; set; }

public DateTime dataLansarii { get; set; }

public double Durata { get; set; }

}

class Program

{

static void Main(string[] args)

{

//Cream o lista in care o sa adaug cel putin datele despre 5 melodii

List<Melodie> melodii = new List<Melodie>();

melodii.Add(new Melodie() { Denumire = "I see you again - for Paul Walker", Interpret = "Wiz Khalifa",

dataLansarii = DateTime.Parse("01/01/2013"), Durata = 150 });

melodii.Add(new Melodie() { Denumire = "Unity", Interpret = "Alan Walker",

dataLansarii = DateTime.Parse("08/08/2019"), Durata = 220 });

melodii.Add(new Melodie() { Denumire = "Take away", Interpret = "Marshmello",

dataLansarii = DateTime.Parse("4/6/2019"), Durata = 200 });

melodii.Add(new Melodie() { Denumire = "Something Just Like This", Interpret = "Coldplay & The Chainsmokers",

dataLansarii = DateTime.Parse("10/12/2017"), Durata = 110 });

melodii.Add(new Melodie() { Denumire = "Wonderland", Interpret = "Axel Johansson",

dataLansarii = DateTime.Parse("12/04/2020"), Durata = 92 });

//Adaugam datele din lista melodii in fisierul date.out

using (StreamWriter fisier = new StreamWriter("date.out"))

{

foreach (var item in melodii)

// <<//>> este spatiul dintre date

fisier.WriteLine(item.Denumire + "//" + item.Interpret + "//" + item.dataLansarii.ToString("dd/MM/yyyy") +

"//" + item.Durata);

}

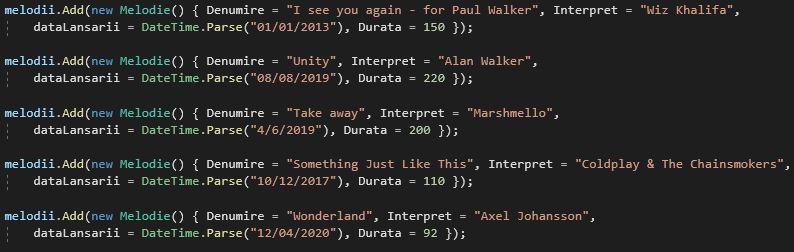
Console.ReadKey();

}

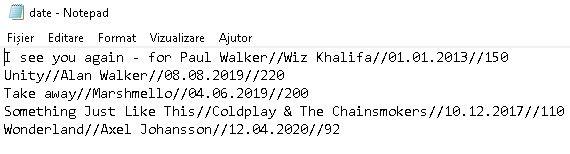
}

}

**Listingul programului :**

****

**Datele de intrare :**

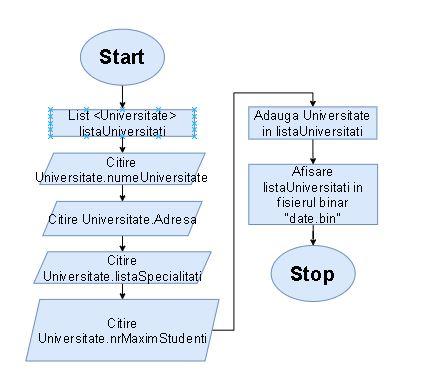
****

**Datele de iesire :**

La realizarea aplicatiei Melodie am utilizat clase,proprietati,lista generica si fisier StreamWriter , am creat clasa Melodie in care au fost date 4 proprietati Denumire,Interpret,DataLansarii,durata . In metoda Main() am creat o lista generica in care am inscris datele despre 5 melodii,respectiv dupa ce am inscris datele despre melodii am creat un fisier text in care datele despre melodii vor fi adaugate cand vom da la executie aplicatia . Aplicatia Melodie mi-a fost de foarte mare ajutor pentru ca am recapitulat utilizarea fisierelor text despre care am invatat in semestrul 2,dezvoltandumi abilitatile de programator .

**Concluzie :**

**Tema 5**

****

**Schema bloc generala a algoritmului realizat :**

using System;

using System.Collections.Generic;

using System.IO;

namespace Tema\_5.Utilizare\_fisiere\_binare

{

//Cream clasa Universitate

class Universitate

{

//Proprietatile clasei

public string numeUniversitate { get; set; }

public string Adresa { get; set; }

public List<string> listaSpecialitati = new List<string>();

public double nrMaximStudenti { get; set; }

}

class Program

{

static void Main(string[] args)

{

//Cream lista listaUniversitati in care vom adauga datele despre fiecare universitate

List<Universitate> listaUniversitati = new List<Universitate>();

listaUniversitati.Add(new Universitate() { numeUniversitate = "Universitatea de stat ''Bogdan Petri" +

"ceicu Hasdeu'' ", Adresa = "str.Stefan cel Mare", listaSpecialitati = { "Matematica", "Informatica", "Fizica", "Limba Romana" }, nrMaximStudenti = 30 });

listaUniversitati.Add(new Universitate() { numeUniversitate = "Universitatea de medicina" +

" de stat ''Nicolae Testemiteanu'' ", Adresa = "str.Bihai Viteazu", listaSpecialitati = { "Chimia", "Fizica", "Anatomia" }, nrMaximStudenti = 29 });

listaUniversitati.Add(new Universitate() { numeUniversitate = "Universitatea pedagogica" +

" de stat ''Ion Creanga'' ", Adresa = "str.Stefan cel Mare", listaSpecialitati = { "Matematica", "Informatica", "Fizica", "Limba Romana" }, nrMaximStudenti = 31 });

//Inscriem lista listaUniversitati intrun fisier binar

using (BinaryWriter fisier = new BinaryWriter(File.Open("date.bin", FileMode.OpenOrCreate)))

{

//Cream un foreach prin care vom citi datele din lista listaUniversitati in fisierul "date.bin"

foreach (var item in listaUniversitati)

{

fisier.Write(item.numeUniversitate);

fisier.Write("\t");

fisier.Write(item.Adresa);

fisier.Write("\t");

//Mai cream inca un foreach prin care vom citi datele din lista listaSpecialitati in fisierul "date.bin"

foreach (var elem in item.listaSpecialitati)

{

fisier.Write(elem + " ");

}

fisier.Write("\n\t");

fisier.Write(item.nrMaximStudenti.ToString());

fisier.Write("\n\n");

}

}

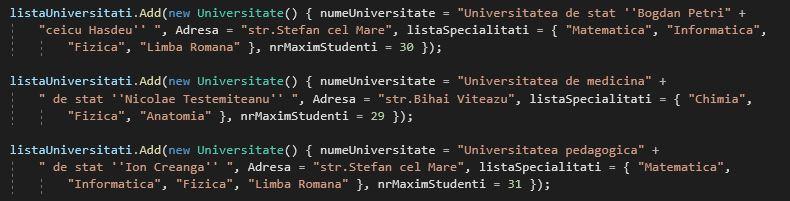
Console.ReadKey();

}

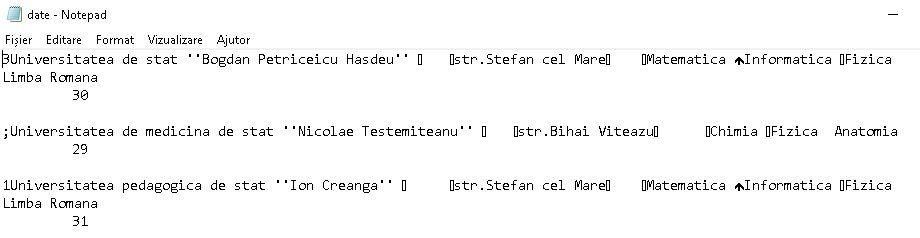
}

}

**Listingul programului :**

****

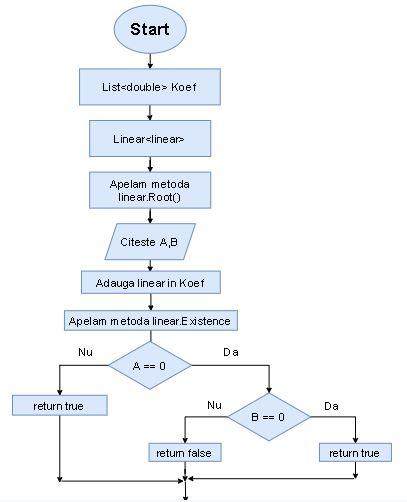
**Datele de intrare :**

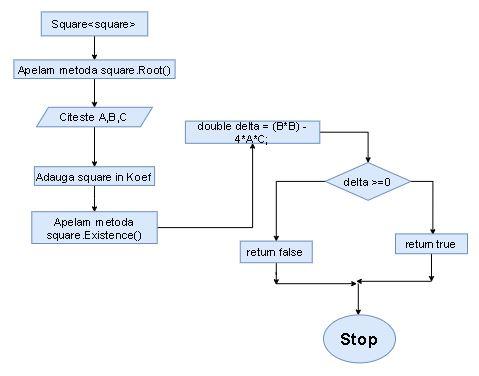
****

**Datele de iesire :**

Pentru realizarea acestei aplicatii am utilizat clase,proprietati ale clasei,lista generica si fisier BinaryWriter .Clasa Universitate contine 4 proprietati numeUniversitate ,Adresa ,listaSpecialitati si nrMaximStudenti . In metoda Main() am creat o lista generica in care am inscris datele despre 3 universitati ,respectiv dupa ce citim datele cream un fisier binar si scriem lista in fisierul „date.bin” .Aplicatia Universitate consider ca a fost de mare ajutor majoritatii grupei pentru ca nu am utilizat fisierele binare din semestrul 2 si ar fi bine sa le mai recapitulam ,pentru ca vom avea nevoie de ele pe viitor in calitate de programatori .

**Concluzie :**

**Tema 6**



**Schema bloc generala a algoritmului realizat :**

using System;

using System.Collections.Generic;

using System.Linq;

namespace Tema\_6.Interfete

{

class Program

{

//Cream interfata ISolution (ecuatii)

interface Isolution

{

//Proprietatile interfetei

public List<double> Koef { get; set; }

public bool Existence();

public void Root();

}

class Linear : Isolution

{

public double A { get; set;}

public double B { get; set; }

public List<double> Koef { get; set; }

public bool Existence()

{

if (A == 0 )

{

if (B == 0)

{

Console.WriteLine($"<<<<<<<< Ecuatia {A}x + {B} = 0 admite o mulțime infinită de soluții ! >>>>>>>>>");

return true;

}

else

{

Console.WriteLine($"<<<<<<<< Ecuatia {A}x + {B} = 0 nu are solutii ! >>>>>>>>>");

return false;

}

}

else

{

Console.WriteLine($"<<<<<< Ecuatia {A}x + {B} = 0 are solutii ! >>>>>>>>");

return true;

}

}

public void Root()

{

Console.Write("a = ");

A = double.Parse(Console.ReadLine());

Console.Write("b = ");

B = double.Parse(Console.ReadLine());

Koef = new List<double>();

//Adaugam in lista coeficientii ecuatiei

Koef.Add(A);

Koef.Add(B);

Console.WriteLine("------------------------------------------");

Console.WriteLine("Lista coeficientilor ecuatiei de gradul 1: ");

foreach (var item in Koef)

{

Console.WriteLine(item);

}

Console.WriteLine();

}

public Linear(double a, double b)

{

A = a;

B = b;

}

public Linear()

{

}

}

class Square : Isolution

{

public double A { get; set; }

public double B { get; set; }

public double C { get; set; }

public List<double> Koef { get; set; }

public bool Existence()

{

double delta = B \* B - 4 \* A \* C;

if (delta >= 0)

{

Console.WriteLine($"<<<<<<<< Ecuatia {A}x^2 + {B}x + {C} = 0 are solutii ! >>>>>>>>>");

return true;

}

else

{

Console.WriteLine($"<<<<<<<< Ecuatia {A}x^2 + {B}x + {C} = 0 nu are solutii ! >>>>>>>>>");

return false;

}

}

public void Root()

{

Console.Write("a = ");

A = double.Parse(Console.ReadLine());

Console.Write("b = ");

B = double.Parse(Console.ReadLine());

Console.Write("c = ");

C = double.Parse(Console.ReadLine());

Koef = new List<double>();

//Adaugam coeficientii ecuatiei in lista

Koef.Add(A);

Koef.Add(B);

Koef.Add(C);

Console.WriteLine("------------------------------------------");

Console.WriteLine();

Console.WriteLine("Lista coeficientilor ecuatiei de gradul 2: ");

foreach(var item in Koef)

{

Console.WriteLine(item);

}

Console.WriteLine();

}

public Square(double a, double b, double c)

{

A = a;

B = b;

C = c;

}

public Square()

{

}

}

static void Main(string[] args)

{

Linear linear = new Linear();

Console.WriteLine("Citim datele despre ecuatia de gradul I :");

linear.Root();

Console.WriteLine("------------------------------------------");

linear.Existence();

Console.WriteLine();

Console.WriteLine("==============================================");

Console.WriteLine();

Square square = new Square();

Console.WriteLine("Citim datele despre ecuatia de gradul II :");

square.Root();

Console.WriteLine("------------------------------------------");

square.Existence();

Console.ReadKey();

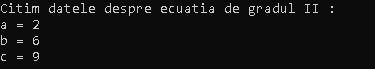
}

}

}

**Listingul programului :**

****

****

**Datele de intrare :**

****

**Datele de iesire :**

La aplicatia respectiva am utilizat clase,interfata,instante ale claselor ,polimorfismul ,mostenirea si lista generica , interfata Isolution contine doua metode una de tip bool si una void care va citi informatia si o lista generica Koef in care vom adauga coeficienti ecuatiilor ambelor clase , interfata va trebui realizata in ambele clase Linear si Square.

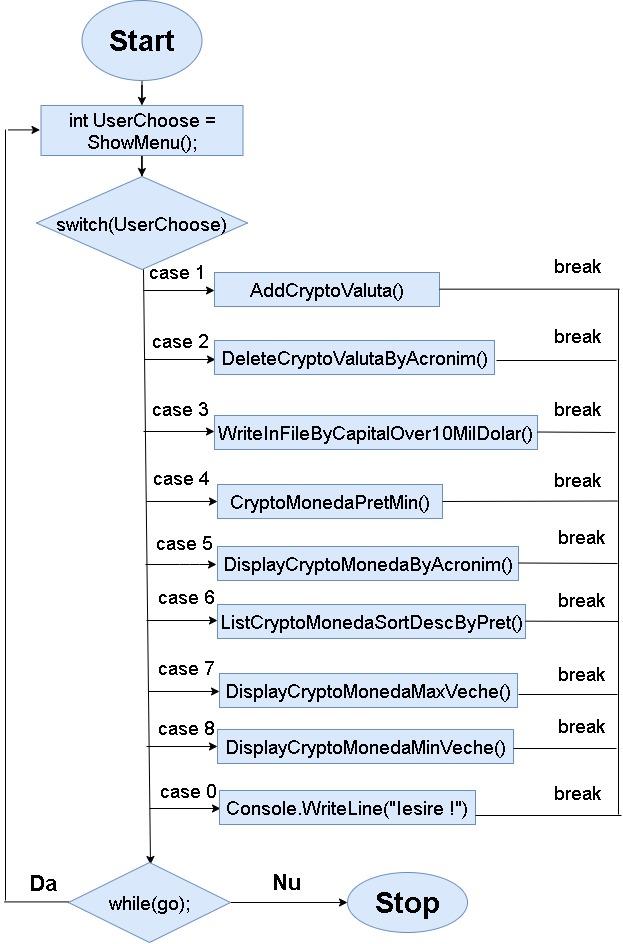
In clasa Linear vom adauga doua variabile A si B care vor fi coeficientii ecuatiei de gr 1 ,vom realiza metoda bool care va returna daca ecuatia are sau nu solutie si metoda Root care va citi datele ecuatiei .

In clasa Square vom adauga trei variabile A, B si C care vor fi coeficientii ecuatiei de gr 2 ,vom realiza metoda bool care va returna daca ecuatia are sau nu solutie si metoda Root() care va citi datele ecuatiei .In metoda Main() facem pentru fiecare clasa cate o instanta cu care vom apela cele doua metode Existence si Root.

Aplicatia 6 este una printre cele mai importante aplicatii pe care leam realizat pe parcursul practicii pentru ca am recapitulat utilizarea interfetei,metodelor supradefinite si altele care pe parcursul anilor de studiu vom avea foarte mare nevoie de ele in realizarea diverselor aplicatii .

**Concluzie :**

**Tema 7**

****

**Schema bloc generala a algoritmului realizat :**

using System;

using System.IO;

using System.Collections.Generic;

using System.Linq;

using System.Threading;

namespace Tema\_7.Aplicatie\_complexa

{

class Program

{

class CriptoValuta

{

public string Nume { get; set; }

public string Acronim { get; set; }

public DateTime dataCrearii { get; set; }

public double capitalPePiata { get; set; }

public double nrTotalMonedeCePotFiMinate { get; set; }

public double pretMedCriptoMonedaInZiuaInregistrarii { get; set; }

}

static List<CriptoValuta> listaCriptoValute = new List<CriptoValuta>();

public static int ShowMenu()

{

int UserChoose\_int = 0;

bool ok;

do

{

Console.WriteLine("1. Adauga o CriptoValuta :");

Console.WriteLine("2. Exclude o CriptoValuta dupa acronim :");

Console.WriteLine("3. Înscrie într-un fisier datele despre criptomonele cu capitalul de market de peste 10 miliarde de dolari :");

Console.WriteLine("4. Afiseaza datele despre CriptoMonedele cu cel mai mic pret :");

Console.WriteLine("5. Afiseaza datele despre o CriptoMoneda dupa acronim :");

Console.WriteLine("6. Afisează lista tuturor CriptoMonedelor, în ordine descrescătoare a preturilor :");

Console.WriteLine("7. Afisează criptomoneda cu cea mai mare vechime pe piată :");

Console.WriteLine("8. Afisează criptomoneda cu cea mai mică vechime pe piată :");

Console.WriteLine("0. Iesire:");

Console.WriteLine("\nAlegeti o optiune ! ");

string UserChoose = Console.ReadLine();

ok = int.TryParse(UserChoose, out UserChoose\_int);

if (!ok)

{

Console.WriteLine("Nu exista astfel de optiune !");

}

} while (!ok);

return UserChoose\_int;

}

static void Main(string[] args)

{

bool go = true;

do

{

Console.ForegroundColor = ConsoleColor.Red;

Console.WriteLine("--------------< Meniul Aplicatiei >--------------" );

Console.WriteLine();

int UserChoose = ShowMenu();

switch (UserChoose)

{

case 1:

Console.WriteLine("1. Adauga o CriptoValuta :");

Console.WriteLine();

AddCriptoValuta();

break;

case 2:

Console.WriteLine("2. Exclude o CriptoValuta dupa acronim :");

Console.WriteLine();

DeleteCriptoValutaByAcronim();

break;

case 3:

Console.WriteLine("3. Înscrie într-un fisier datele despre criptomonele cu capitalul de market de peste 10 miliarde de dolari :");

Console.WriteLine();

WriteInFileByCapitalOver10MilDolar();

break;

case 4:

Console.WriteLine("4. Afiseaza datele despre CriptoMonedele cu cel mai mic pret :");

Console.WriteLine();

CriptoMonedaPretMin();

break;

case 5:

Console.WriteLine("5. Afiseaza datele despre o CriptoMoneda dupa acronim :");

Console.WriteLine();

DisplayCriptoMonedaByAcronim();

break;

case 6:

Console.WriteLine("6. Afisează lista tuturor CriptoMonedelor, în ordine descrescătoare a preturilor :");

Console.WriteLine();

ListCriptoMonedaSortDescByPret();

break;

case 7:

Console.WriteLine("7. Afisează criptomoneda cu cea mai mare vechime pe piată :");

Console.WriteLine();

DisplayCriptoMonedaMaxVeche();

break;

case 8:

Console.WriteLine("8. Afisează criptomoneda cu cea mai mică vechime pe piată :");

Console.WriteLine();

DisplayCriptoMonedaMinVeche();

break;

case 0:

go = false;

Console.WriteLine("Iesire");

Console.WriteLine("The application has closed !");

break;

default:

Console.WriteLine("Nu exista astfel de optiune !");

break;

}

} while (go);

Console.ReadKey();

}

//Metoda pentru prima optiune din meniu

private static void AddCriptoValuta()

{

Console.Clear();

CriptoValuta criptoValuta = new CriptoValuta();

Console.WriteLine("Citim datele despre CriptoValuta :");

try

{

Console.Write("Numele : ");

criptoValuta.Nume = Console.ReadLine();

Console.Write("Acronim : ");

criptoValuta.Acronim = Console.ReadLine();

Console.Write("Data crearii : ");

criptoValuta.dataCrearii = DateTime.Parse(Console.ReadLine());

Console.Write("Capital pe piata : ");

criptoValuta.capitalPePiata = double.Parse(Console.ReadLine());

Console.Write("Numarul total de monede ce pot fi minate : ");

criptoValuta.nrTotalMonedeCePotFiMinate = double.Parse(Console.ReadLine());

Console.Write("Pretul mediu al criptomonedei în ziua înregistrării : ");

criptoValuta.pretMedCriptoMonedaInZiuaInregistrarii = double.Parse(Console.ReadLine());

listaCriptoValute.Add(criptoValuta);

}

catch (Exception e)

{

Console.WriteLine("MESAJUL DE EROARE !");

Console.WriteLine(e.Message);

}

Console.WriteLine();

}

//Metoda pentru a doua optiune din meniu

private static void DeleteCriptoValutaByAcronim()

{

Console.Clear();

Console.WriteLine("Scrieti acronimul criptovalutei care doriti sa o excludeti din lista de mai jos : ");

foreach (var item in listaCriptoValute)

{

Console.WriteLine(item.Acronim);

}

Console.WriteLine();

Console.WriteLine("--------------------");

Console.WriteLine();

try

{

string acronim = Console.ReadLine();

Console.Write("Find data by Acronim..");

for (int i = 0; i < 10; i++)

{

Console.Write(".");

Thread.Sleep(400);

}

var itemToRemove = listaCriptoValute.Single(r => r.Acronim == acronim);

listaCriptoValute.Remove(itemToRemove);

Console.WriteLine("\nCriptoValute has been deleted !");

}

catch (Exception e)

{

Console.WriteLine("\nMesajul de EROARE : " + e.Message);

}

Console.WriteLine();

}

//Metoda pentru a treia optiune din meniu

private static void WriteInFileByCapitalOver10MilDolar()

{

Console.Clear();

using (StreamWriter fisier = new StreamWriter("fisier\_capital\_max\_10\_mil\_dolar.out"))

{

foreach (var item in listaCriptoValute)

{

if (item.capitalPePiata > 10000000)

{

fisier.WriteLine("Datele despre CriptoValuta : ");

fisier.WriteLine();

fisier.WriteLine($"Numele : {item.Nume}");

fisier.WriteLine($"Acronim : {item.Acronim}");

fisier.WriteLine($"Data crearii : {item.dataCrearii.ToString("dd/MM/yyyy")}");

fisier.WriteLine($"Capital pe piata : {item.capitalPePiata}");

fisier.WriteLine($"Numarul total de monede ce pot fi minate : {item.nrTotalMonedeCePotFiMinate}");

fisier.WriteLine($"Pretul mediu al criptomonedei in ziua inregistrarii : {item.pretMedCriptoMonedaInZiuaInregistrarii}");

Console.WriteLine("Datele au fost inscrise in fisier !");

}

else

{

fisier.WriteLine("Nu exista CriptoValuta cu un capital mai mare de 10 miliarde dolar ! ");

Console.WriteLine("Nu exista CriptoValuta cu un capital mai mare de 10 miliarde dolar ! ");

}

}

}

Console.WriteLine();

}

//Metoda pentru a patra optiune din meniu

private static void CriptoMonedaPretMin()

{

Console.Clear();

double pretMin = listaCriptoValute.Min(criptoValuta => criptoValuta.pretMedCriptoMonedaInZiuaInregistrarii);

var criptoMonedaPretMin = listaCriptoValute.First(criptoValuta => criptoValuta.pretMedCriptoMonedaInZiuaInregistrarii == pretMin);

//Afisarea datelor despre CriptoMoneda cu pretul minim

Console.WriteLine($"Numele : {criptoMonedaPretMin.Nume}");

Console.WriteLine($"Acronim : {criptoMonedaPretMin.Acronim}");

Console.WriteLine($"Data crearii : {criptoMonedaPretMin.dataCrearii.ToString("dd/MM/yyyy")}");

Console.WriteLine($"Capital pe piata : {criptoMonedaPretMin.capitalPePiata}");

Console.WriteLine($"Numarul total de monede ce pot fi minate : {criptoMonedaPretMin.nrTotalMonedeCePotFiMinate}");

Console.WriteLine($"Pretul mediu al criptomonedei in ziua inregistrarii : {criptoMonedaPretMin.pretMedCriptoMonedaInZiuaInregistrarii}");

Console.WriteLine();

}

//Metoda pentru a cincea optiune din meniu

private static void DisplayCriptoMonedaByAcronim()

{

Console.Clear();

Console.Write("Scrieti acronimul criptovalutei care doriti sa o afisati din lista de mai jos : ");

Console.WriteLine();

foreach(var item in listaCriptoValute)

{

Console.WriteLine(item.Acronim);

}

Console.WriteLine();

Console.WriteLine("--------------------");

Console.WriteLine();

try

{

string acronim = Console.ReadLine();

Console.WriteLine();

Console.Write("Find data by Acronim...");

for (int i = 0; i < 10; i++)

{

Console.Write(".");

Thread.Sleep(400);

}

Console.WriteLine();

var itemToDisplay = listaCriptoValute.Single(r => r.Acronim == acronim);

//Afisare date despre CriptoValuta cu acronimul ales din lista

Console.WriteLine("\nDatele despre CriptoValuta : ");

Console.WriteLine($"Numele : {itemToDisplay.Nume}");

Console.WriteLine($"Acronim : {itemToDisplay.Acronim}");

Console.WriteLine($"Data crearii : {itemToDisplay.dataCrearii.ToString("dd/MM/yyyy")}");

Console.WriteLine($"Capital pe piata : {itemToDisplay.capitalPePiata}");

Console.WriteLine($"Numarul total de monede ce pot fi minate : {itemToDisplay.nrTotalMonedeCePotFiMinate}");

Console.WriteLine($"Pretul mediu al criptomonedei in ziua inregistrarii : {itemToDisplay.pretMedCriptoMonedaInZiuaInregistrarii}");

Console.WriteLine();

}

catch (Exception e)

{

Console.WriteLine("\nMesajul de EROARE : " + e.Message);

}

}

//Metoda pentru a sasea optiune din meniu

private static void ListCriptoMonedaSortDescByPret()

{

Console.Clear();

List<CriptoValuta> SortedList = listaCriptoValute.OrderBy(o => o.pretMedCriptoMonedaInZiuaInregistrarii).ToList();

SortedList.Reverse();

Console.WriteLine("Lista de criptovalute sortate descrescator dupa pret : ");

Console.WriteLine();

foreach (var item in SortedList)

{

Console.WriteLine($"Numele : {item.Nume}");

Console.WriteLine($"Acronim : {item.Acronim}");

Console.WriteLine($"Data crearii : {item.dataCrearii.ToString("dd/MM/yyyy")}");

Console.WriteLine($"Capital pe piata : {item.capitalPePiata}");

Console.WriteLine($"Numarul total de monede ce pot fi minate : {item.nrTotalMonedeCePotFiMinate}");

Console.WriteLine($"Pretul mediu al criptomonedei in ziua inregistrarii : {item.pretMedCriptoMonedaInZiuaInregistrarii}");

Console.WriteLine();

}

Console.WriteLine();

}

//Metoda pentru a saptea optiune din meniu

private static void DisplayCriptoMonedaMaxVeche()

{

Console.Clear();

DateTime dateVecheMax = listaCriptoValute.Min(criptoValuta => criptoValuta.dataCrearii);

var criptoMonedaDateVecheMax = listaCriptoValute.First(criptoValuta => criptoValuta.dataCrearii == dateVecheMax);

Console.WriteLine($"Numele : {criptoMonedaDateVecheMax.Nume}");

Console.WriteLine($"Acronim : {criptoMonedaDateVecheMax.Acronim}");

Console.WriteLine($"Data crearii : {criptoMonedaDateVecheMax.dataCrearii.ToString("dd/MM/yyyy")}");

Console.WriteLine($"Capital pe piata : {criptoMonedaDateVecheMax.capitalPePiata}");

Console.WriteLine($"Numarul total de monede ce pot fi minate : {criptoMonedaDateVecheMax.nrTotalMonedeCePotFiMinate}");

Console.WriteLine($"Pretul mediu al criptomonedei in ziua inregistrarii : {criptoMonedaDateVecheMax.pretMedCriptoMonedaInZiuaInregistrarii}");

Console.WriteLine();

}

//Metoda pentru a opta optiune din meniu

private static void DisplayCriptoMonedaMinVeche()

{

Console.Clear();

DateTime dateVecheMin = listaCriptoValute.Max(criptoValuta => criptoValuta.dataCrearii);

var criptoMonedaDateVecheMin = listaCriptoValute.First(criptoValuta => criptoValuta.dataCrearii == dateVecheMin);

Console.WriteLine($"Numele : {criptoMonedaDateVecheMin.Nume}");

Console.WriteLine($"Acronim : {criptoMonedaDateVecheMin.Acronim}");

Console.WriteLine($"Data crearii : {criptoMonedaDateVecheMin.dataCrearii.ToString("dd/MM/yyyy")}");

Console.WriteLine($"Capital pe piata : {criptoMonedaDateVecheMin.capitalPePiata}");

Console.WriteLine($"Numarul total de monede ce pot fi minate : {criptoMonedaDateVecheMin.nrTotalMonedeCePotFiMinate}");

Console.WriteLine($"Pretul mediu al criptomonedei in ziua inregistrarii : {criptoMonedaDateVecheMin.pretMedCriptoMonedaInZiuaInregistrarii}");

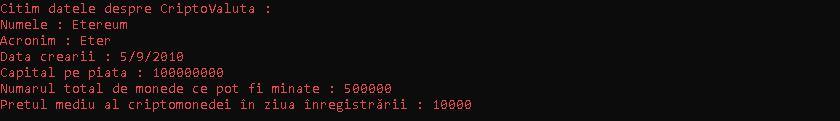
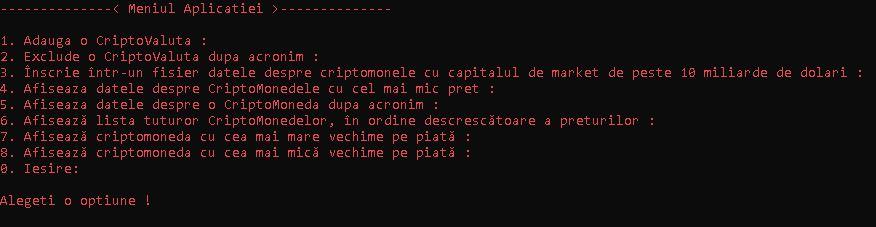
Console.WriteLine();

}

}

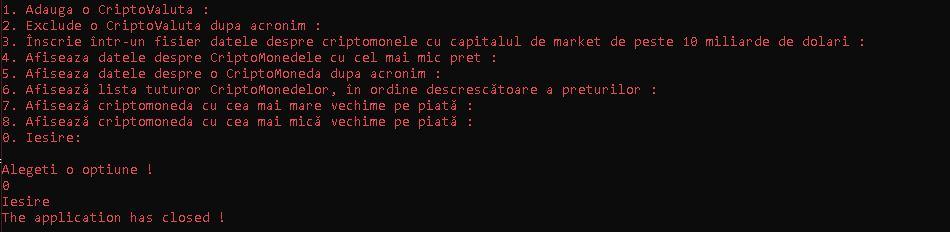
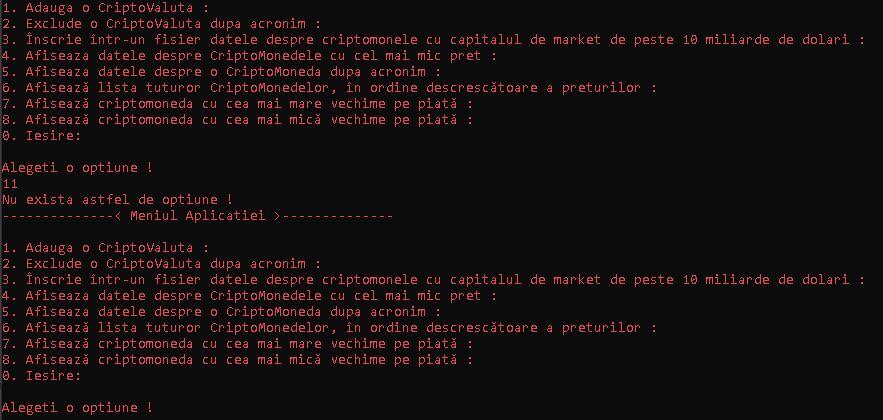
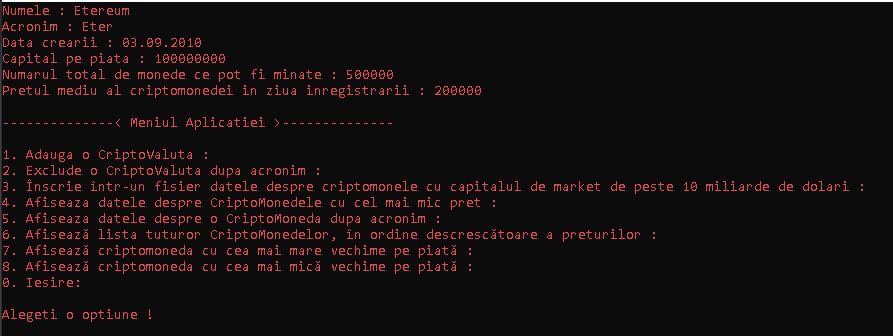
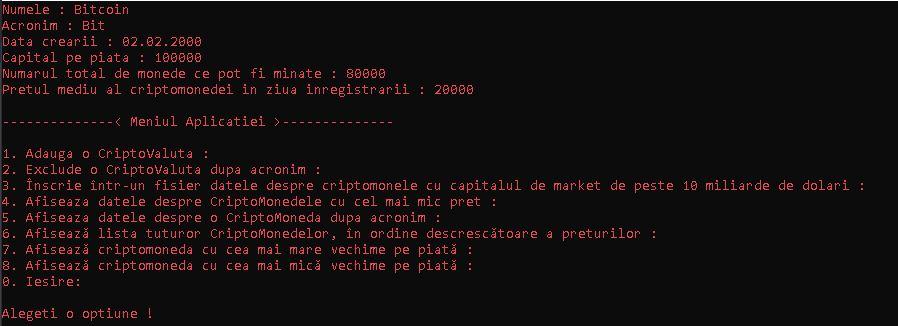
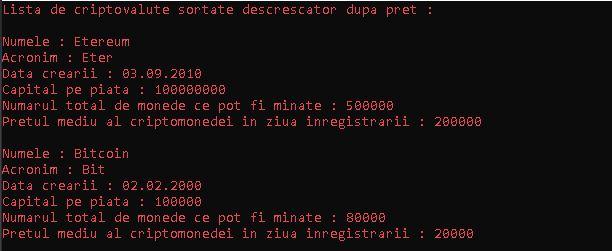
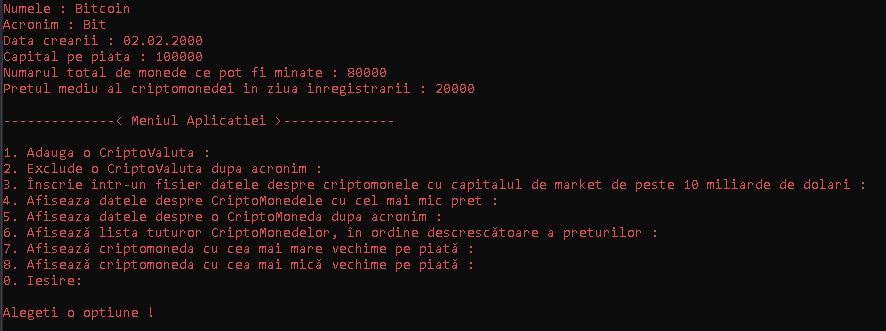
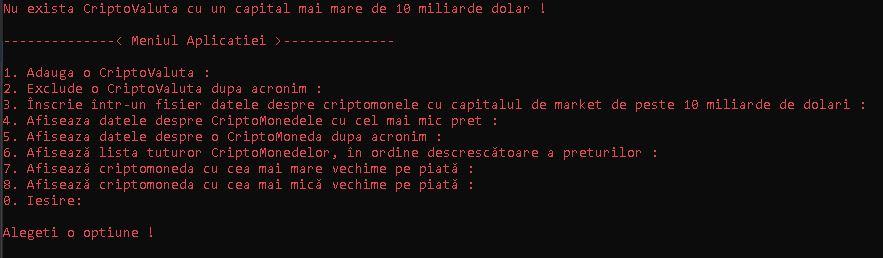
}

**Listingul programului :**



**Datele de intrare :**





**Datele de iesire :**

Aplicatia 7 este una dintre cele mai importante aplicatii pe care le-am creat pana acum in timpul practicii , pentru a o crea am folosit diverse metode si optiuni pe care le-am dezvoltat dupa propria logica si cunostintele extrase pe parcursul a 4 semestre .

In acesta aplicatie am folosit clase , metode , lista generica , instantele clasei , adaugarea si stergerea a elementelor din lista si cel mai important lucru utilizarea Meniului , meniul respective contine 8 optiuni , pentru fiecare optiune am creat cate o metoda care va fi apelata de meniu , am creat aplicatia respectiva in cateva ore si am mai recapitulat din temele anterioare din celelalte semestre .

Aplicatiile care leam creat pana acum respectiv si aplicatia 7 m-au ajutat foarte mult la dezvoltarea mea profesionala pe plan de programator si consider ca aceste aplicatii au fost de ajutor si colectivului cu care studiez , practica de initiere este un mod foarte bun de recapitulare a temelor studiate si de examinare a studentilor .

**Concluzie :**

**Bibliografie și resurse:**

1. **Literatura din suport de curs.**
2. **metanit.com**
3. **tproger.ru**
4. **visualstudio.com**