

**Ministerul Educației, Culturii și Cercetării al Republicii Moldova**  
**Colegiul Iulia Hasdeu din Cahul**

**Catedra TIC**

**RAPORT**

**practica de inițiere în specialitate**

**Specialitatea: 61210 Administrarea aplicațiilor Web**

**Calificarea: Tehnician de site-uri Web**

**Semestrul II**

**Grupa AAW 2012**

Efectuat      **Apareci Aurica AAW 2012**  
(Numele, prenumele studentului, grupa)

Verificat      **Baba Dorin**  
(Numele, prenumele coordonatorului)

Nota \_\_\_\_\_

**Cahul, 2021**

## Cuprins

Tema 1. Structura liniară. Expresii aritmetice. ....	1
<b>Listingul programului</b> .....	1
<b>Concluzii.</b> .....	2
Tema 2. Structura ramificată. ....	3
<b>Listingul programului</b> .....	3
<b>Rezultatele testării.</b> .....	4
<b>Concluzii.</b> .....	4
Tema 3. Metode template (Class Library).....	5
<b>Listingul programului</b> .....	5
<b>Rezultatele testării.</b> .....	6
<b>Concluzii.</b> .....	6
Tema 4. Fișiere .....	7
<b>Listingul programului</b> .....	7
<b>Rezultatele testării.</b> .....	8
<b>Concluzii.</b> .....	8
Tema 5. Șiruri de caractere.....	9
<b>Listingul programului</b> .....	9
<b>Rezultatele testării.</b> .....	9
<b>Concluzii.</b> .....	9
Tema 6. Structuri de date.....	10
<b>Listingul programului</b> .....	10
<b>Rezultatele testării</b> .....	13
<b>Concluzii.</b> .....	16

# Tema 1. Structura liniară. Expresii aritmetice.

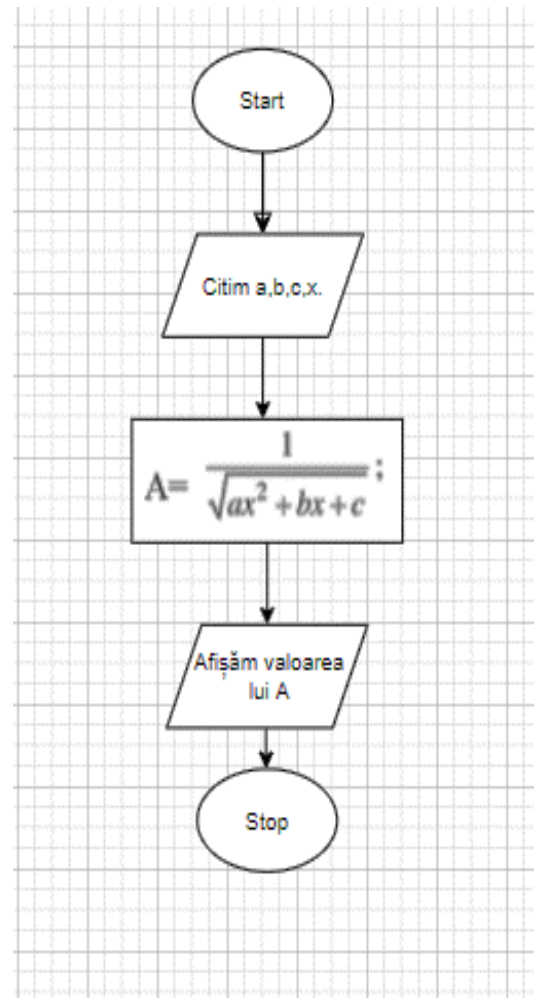
## Varianta I

De elaborat o aplicație de consolă ce va calcula valoarea expresiei  $A = \frac{1}{\sqrt{ax^2+bx+c}}$

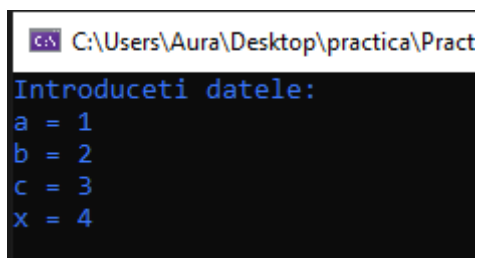
### Listingul programului

```
class Program
{
    static public int a;
    static public int b;
    static public int c;
    static public int x;
    static void Main(string[] args)
    {
        bool isValid = false;
        while (!isValid)
        {
            try
            {
                Citire();
                Console.WriteLine("A = " + Functie());
                isValid = true;
                Console.ReadKey();
            }
            catch (Exception)
            {
                Console.WriteLine("Eroare !");
                isValid = false;
            }
        }
    }
    private static double Functie()
    {
        double A = 1/(Math.Sqrt(a*Math.Pow(x,2) + b*x + c));
        return A;
    }
    private static void Citire()
    {
        Console.WriteLine("Introduceti datele: ");
        Console.Write("a = ");
        a = int.Parse(Console.ReadLine());
        Console.Write("b = ");
        b = int.Parse(Console.ReadLine());
        Console.Write("c = ");
        c = int.Parse(Console.ReadLine());
        Console.Write("x = ");
        x = int.Parse(Console.ReadLine());
        Console.Clear();
    }
}
```

### Schema logică

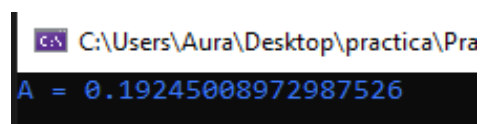


## Rezultatele testării.



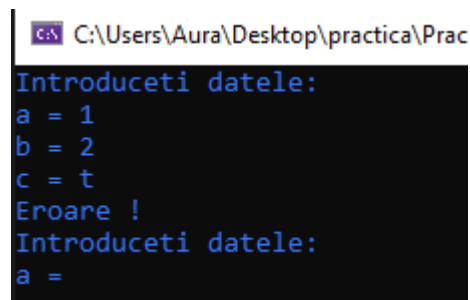
```
C:\Users\Aura\Desktop\practica\Pract
Introduceti datele:
a = 1
b = 2
c = 3
x = 4
```

Fig nr. 1 Introducerea datelor de la tastatura.



```
C:\Users\Aura\Desktop\practica\Pra
A = 0.19245008972987526
```

Fig nr. 2 Afișarea rezultatului



```
C:\Users\Aura\Desktop\practica\Prac
Introduceti datele:
a = 1
b = 2
c = t
Eroare !
Introduceti datele:
a =
```

Fig nr. 3 Tratarea exceptiilor

## Concluzii.

Pentru crearea acestei aplicații cu structură liniară, ce calculează valoarea unei expresii matematice complexe, am folosit funcțiile încorporate ale clasei Math, cum ar fi Math.Pow() și Math.Sqrt().

Aplicația include :

1. Citirea datelor ();
2. Folosirea instrucțiunii *Try/Catch* pentru tratarea excepțiilor;
3. Calcularea expresiei();
4. Afișarea rezultatului.

De-asemena am utilizat blocurile try-catch pentru a trata excepțiile ce pot apărea în decursul rulării programului (astfel tratez cazul în care utilizatorul introduce un sir de caractere care nu poate fi convertit într-o valoare de tip întreg). Pentru tratarea excepțiilor au fost înaintate 2 metode:

Am selectat prima metoda deoarece este mai practică și poate fi utilizată pentru mai multe proceduri simultan. Aceasta sarcină m-a ajutat să-mi formeze noi abilități datorită faptului că am aplicat pentru prima dată tratarea excepțiilor.

## Tema 2. Structura ramificată.

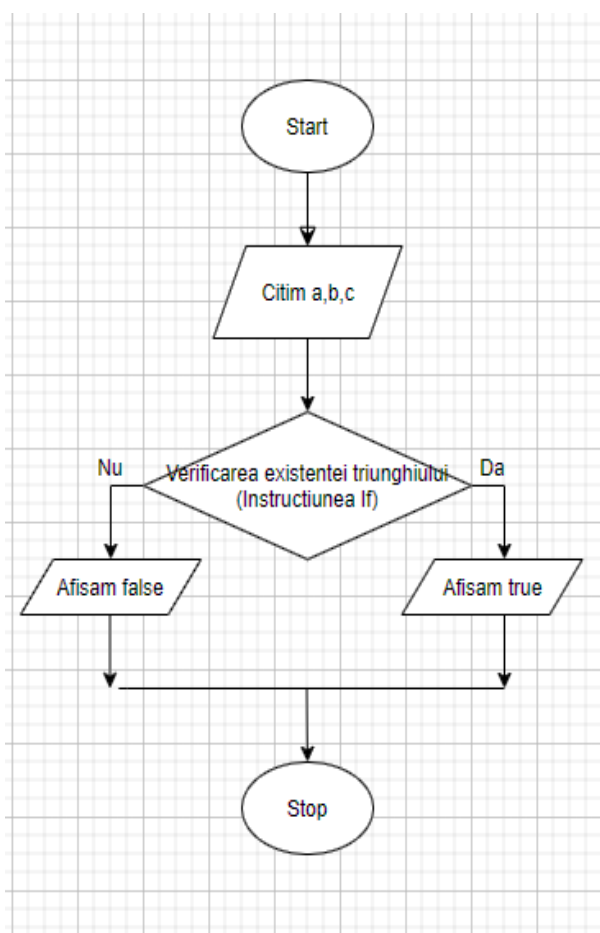
### Varianta I

De la tastatură se citesc 3 numere reale. Creați o funcție ce va primi aceste numere ca parametri de intrare și va returna valoarea true dacă numerele pot reprezenta lungimile laturilor unui triunghi. În caz contrar - va returna false. Trei numere pot fi laturile unui triunghi doar dacă suma oricăror două este mai mare decât al 3-lea număr.

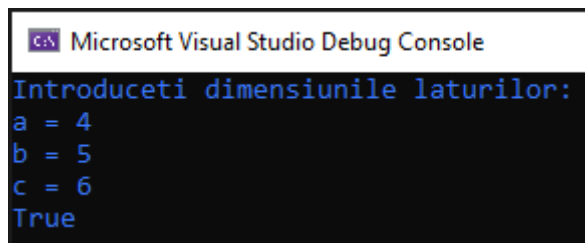
### Listingul programului

```
class Program
{
    static public int a;
    static public int b;
    static public int c;
    static void Main(string[] args)
    {
        bool isValid = false;
        while (!isValid)
        {
            try
            {
                Citire();
                Console.WriteLine(Verificare(a, b, c));
                isValid = true;
            }
            catch (Exception)
            {
                Console.WriteLine("Eroare !");
                isValid = false;
            }
        }
    }
    private static bool Verificare(int a, int b, int c)
    {
        if ((a + b > c) & (a + c > b) & (c + b > a))
        {
            return true;
        }
        else return false;
    }
    private static void Citire()
    {
        Console.WriteLine("Introduceti dimensiunile
laturilor: ");
        Console.Write("a = ");
        a = int.Parse(Console.ReadLine());
        Console.Write("b = ");
        b = int.Parse(Console.ReadLine());
        Console.Write("c = ");
        c = int.Parse(Console.ReadLine());
    }
}
```

### Schema logică

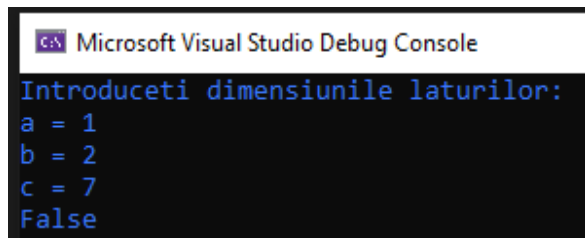


## Rezultatele testării.



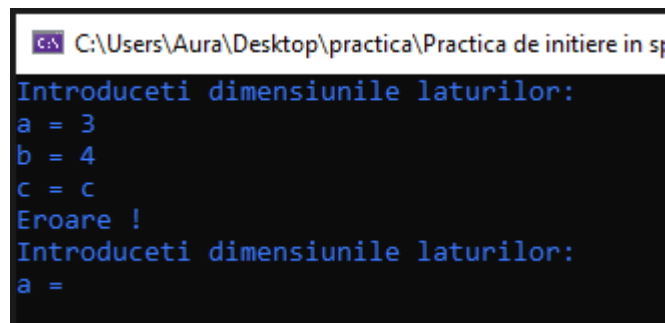
```
Microsoft Visual Studio Debug Console
Introduceti dimensiunile laturilor:
a = 4
b = 5
c = 6
True
```

Fig nr. 1 Programul se rulează corect



```
Microsoft Visual Studio Debug Console
Introduceti dimensiunile laturilor:
a = 1
b = 2
c = 7
False
```

Fig nr. 2 Programul se rulează corect



```
C:\Users\Aura\Desktop\practica\Practica de initiere in s
Introduceti dimensiunile laturilor:
a = 3
b = 4
c = c
Eroare !
Introduceti dimensiunile laturilor:
a =
```

Fig nr. 3 Tratarea exceptiilor

## Concluzii.

Pentru crearea acestei aplicații am folosit procedura *Citire()* și funcția *Verificare()*, cu ajutorul cărora se introduc 3 variabile de tip integer, și se verifică dacă acestea pot fi dimensiunile laturilor triunghiului. Pentru a determina dacă aceste dimensiuni formează un triunghi sau nu am folosit instrucțiunea *If* (suma oricăror 2 laturi este mai mare decât a 3), în caz afirmativ se returnează și afișează *true*, iar în caz negativ *false*.

Aplicația include :

1. Citirea datelor;
2. Folosirea instrucțiunii *Try{} Catch(){}*  pentru tratarea excepțiilor;
3. Funcția *Verificare()* care determină dacă dimensiunile formează un triunghi.
4. Afișarea rezultatului.

## Tema 3. Metode template (Class Library)

### Varianta I

**Sarcina:** Să se completeze în mod manual un masiv format din n rânduri și m coloane. De creat o bibliotecă de metode în care se va conține metoda, care va efectua calculele conform variantei din tabelul de mai jos:

n*m	Tipul elementelor masivului	Problema
4*4	int	Să se determine suma elementelor pare din acest masiv. Citirea masivului, determinarea sumei și determinarea parității unui număr vor fi efectuate de către metode separate.

### Listingul programului

```
static void Main(string[] args)
{
    Class1.Citire();
    Class1.Afisare();
    Class1.Pare();
    Console.WriteLine($"Suma elementelor pare: {Class1.Suma()}");
    Console.ReadKey();
}
```

private static void Citire()	private static void Afisare()
<pre>static int[,] masiv = new int[4, 4]; static public void Citire() {     Random el = new Random();     for (int i = 0; i &lt; 4; i++)     {         for (int j = 0; j &lt; 4; j++)         {             masiv[i,j] = el.Next(-100, 100);         }     } }</pre>	<pre>static public void Afisare() {     for (int i = 0; i &lt; 4; i++)     {         for (int j = 0; j &lt; 4; j++)         {             Console.Write(masiv[i,j]+"");         }         Console.WriteLine();     } }</pre>
private static void Pare()	
<pre>static public int[] elemente; static public void Pare() {     string s = "";     for (int i = 0; i &lt; 4; i++)     {         for (int j = 0; j &lt; 4; j++)         {             if (masiv[i,j]%2==0)             {                 s += \$"{masiv[i,j]} ";             }         }     }     Console.WriteLine("\nNumere pare:\n"+s);     elemente = s.Trim().Split(" ").Select(x =&gt; int.Parse(x.Trim())).ToArray(); }</pre>	
private static int Suma()	
<pre>static public int Suma() {     return elemente.Sum(); }</pre>	

## Rezultatele testării.

C:\Users\Aura\Desktop\practica\Practica de ini	C:\Users\Aura\Desktop\practica\Practica de initiere in special
<pre>-22    97    -94    60 -29    25    -57    -5 -77    46    55    -35 -42    -89   -91    43  Numere pare: -22 -94 60 46 -42  Suma elementelor pare: -52</pre>	<pre>13     80     96    -78 -30    38     45     87 -28    86     84     46 0       72    -78     56  Numere pare: 80 96 -78 -30 38 -28 86 84 46 0 72 -78 56  Suma elementelor pare: 344</pre>

Fig nr. 1/2 Programul ruleaza corect

## Concluzii.

Bibliotecile dinamice sunt biblioteci ce conțin resurse (în general funcții) ce pot fi utilizate (apelate) de alte programe. Utilizarea DLL-urilor este foarte utila pentru aplicații complexe datorita posibilității de a actualiza programul la necesitate.

Problema realizată în cadrul acestei teme este una mai complexă datorită utilizării unei biblioteci care la rândul ei conține subprograme. Au fost create procedurile *Citire()*, *Afișare()* și *Pare()* cu algoritmul cărora au fost îndeplinite sarcinile.

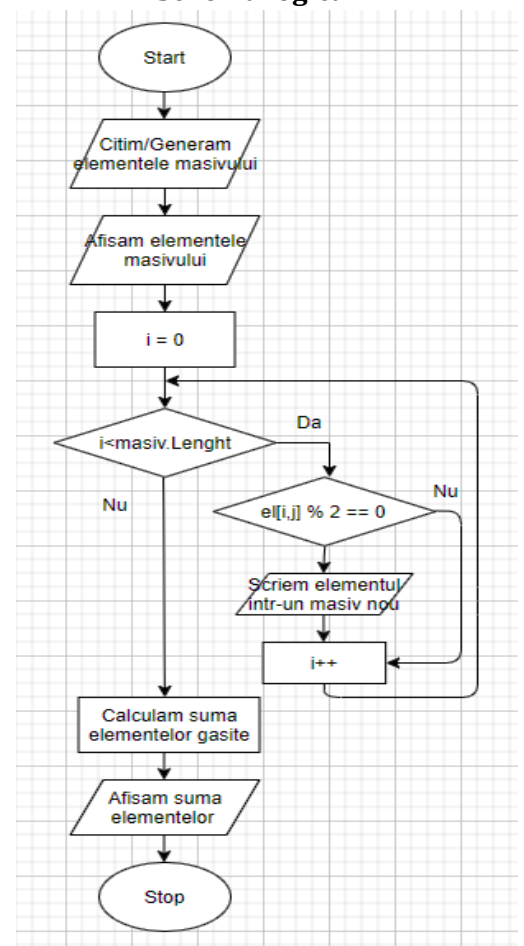
Am generat cu ajutorul metodei *Next()* din cadrul obiectului de tip *Random* masivul bidimensional după care l-am afișat la ecran.

Am parcurs masivul generat și cu ajutorul instrucțiunii *If* am determinat paritatea elementelor. În caz afirmativ am înscris elementul într-un șir de caractere pe care l-am transformat într-un masiv unidimensional.

Ulterior am calculat suma elementelor cu ajutorul funcției predefinite de *System.Linq* și am afișat rezultatul la ecran.

Acest exercițiu m-a ajutat să îmi formez capacități în lucrul cu bibliotecile dinamice dar și în prelucrarea masivelor.

## Schema logică





## Tema 4. Fișiere

### Varianta I

Pentru stocarea parolelor elevilor în baza de date, Colegiul Iulia Hasdeu dispune de un algoritm propriu de criptare a parolelor - CIHCrypt. Acest algoritm transformă o parolă într-un șir de 8 caractere ce se regăsesc în șirul "Colegiul Iulia Hasdeu". Algoritmul nu este case sensitive (Se acceptă și caractere 'c', 'h', 'A' etc.). Algoritmul de criptare uneori generează șiruri invalide. Până când această problema va fi soluționată, se dorește elaborarea unui program ce va verifica validitatea fiecărui șir generat de către CIHCrypt.

Șirurile generate sunt înscrise în fișierul cihcrypt.in. Elaborați un algoritm ce va verifica validitatea fiecărui șir înscris. În fișierul cihcrypt.out vor fi înscrise doar acele șiruri ce satisfac următoarele condiții:

- Nu conțin alte caractere decât cele ce se regăsesc în șirul „Colegiul Iulia Hasdeu”,
- Este format din exact 8 caractere
- Conține cel puțin 3 caractere distincte (Unde ,c' este diferit de ,C').

cihcrypt.in	cihcrypt.out	Explicatie
ccOleHdd cociechu kelegGcC iauliASDu eeeeeeie	ccOleHdd cociechu	Ultimele 3 șiruri generate sunt invalide: al 3-ea conține caracterul k, al 4-lea – este format din 9 caractere, iar ultimul – este format doar din 2 caractere distincte.

### Listingul programului

```
static void Main(string[] args)
{
    using (StreamReader rd = new StreamReader("cihcrypt.in"))
    {
        using (StreamWriter wr = new StreamWriter("cihcrypt.out"))
        {
            string line;
            string valid = "Colegiul Iulia Hasdeu";
            while ((line = rd.ReadLine()) != null)
            {
                if (line.Length == 8)
                {
                    if (line.ToCharArray().Distinct().Count() >= 3)
                    {
                        if
                        (line.ToLower().ToCharArray().Distinct().Intersect(valid.ToLower().ToCharArray().Distinct()).
                        Count()== line.ToCharArray().Distinct().Count())
                        {
                            wr.WriteLine(line);
                        }
                    }
                }
            }
            Console.WriteLine("Programul a fost executat cu succes !");
        }
    }
}
```

## Rezultatele testării.

```
Microsoft Visual Studio Debug Console
Programul a fost executat cu succes !
```

Fig nr. 1 Afisarea mesajului de executie al programului

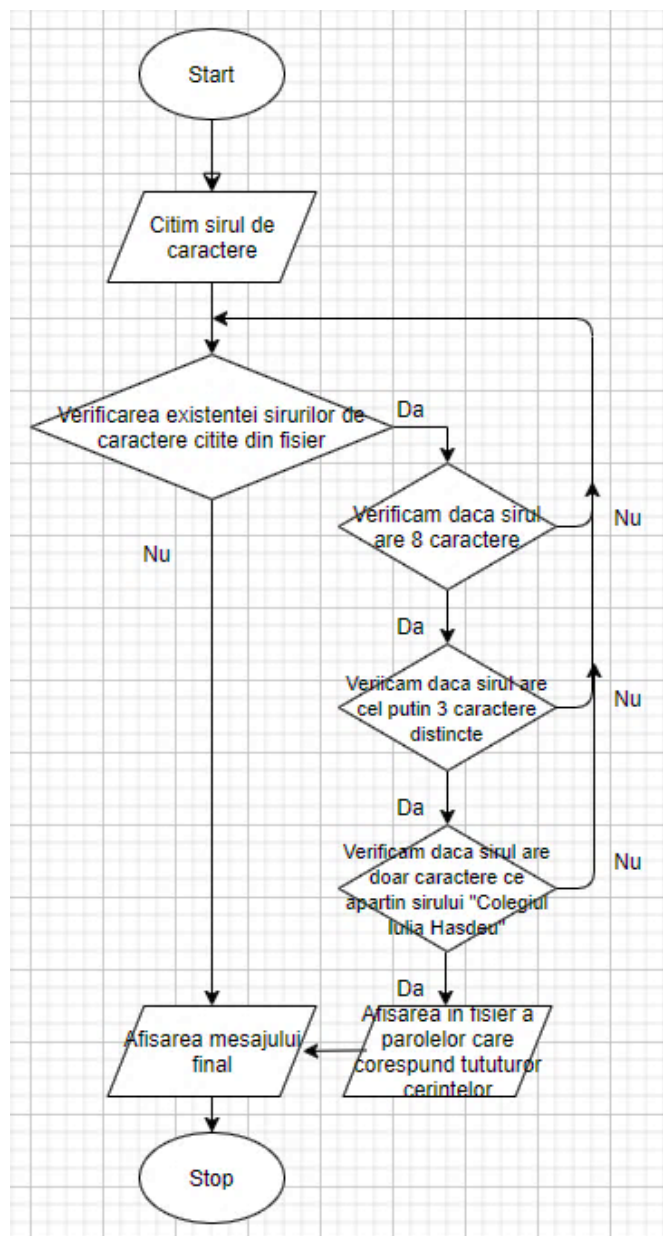
```
cihcrypt.in X
C: > Users > Aura > Desktop > practica > Practica de initiere
1 |cc0leHdd
2 |cociechu
3 |kelegGcC
4 |iauliASDu
5 |eeeeeeie
```

Fig nr. 2 Fisierul cihcrypt.in

```
cihcrypt.out X
C: > Users > Aura > Desktop > practica > Practica de initiere in
1 |cc0leHdd
2 |cociechu
3 |
```

Fig nr. 3 Fisierul cihcrypt.out

## Schema logică



## Concluzii.

Un lucru important in elaborarea acestei sarcini l-au jucat cunoștințele de lucru cu fișierele, și anume crearea, citirea, scrierea și prelucrarea datelor din fișiere. În realizarea problemei am folosit instrucțiunea while care se va executa atât timp cât șirul citit de pe fiecare linie din fișier este diferit de null. Pentru fiecare sir de caractere preluat, cu ajutorul instrucțiunii If am verificat dacă dimensiunea șirului este de 8 caractere, dacă are cel puțin 3 caractere distincte și dacă toate caracterele utilizate aparțin șirului “Colegiul Iulia Hasdeu”.

În urma prelucrării șirurilor cu diferite funcții predefinite funcții, cele care corespund tuturor condițiilor au fost înscrise în fișierul cihcrypt.out. Sarcina propusă la acestă temă nu a fost una dificilă însă necesită atenție și precauție pentru a lua în calcul orice detaliu.

## Tema 5. Șiruri de caractere.

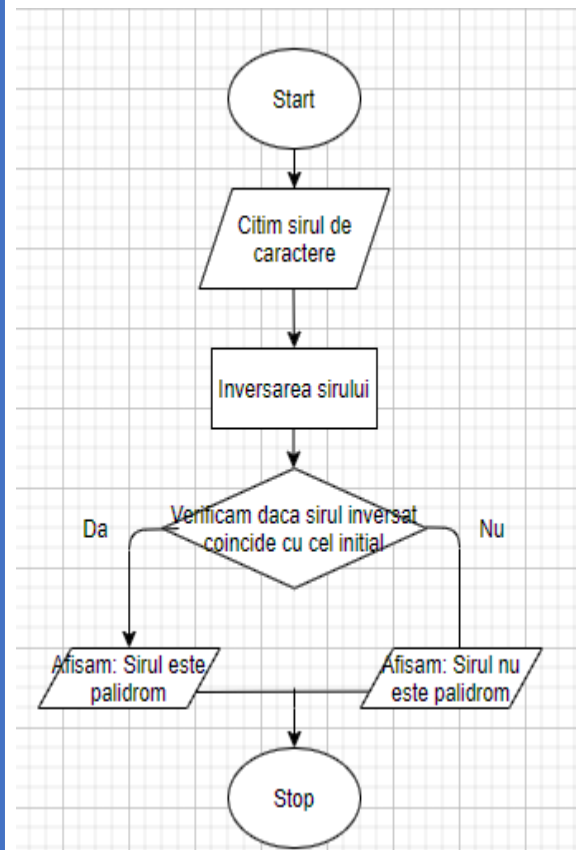
### Varianta I

Se considera un șir de caractere. Elaborați un program cu funcții/proceduri prin intermediul căruia se va determina dacă șirul este palindrom.

### Listingul programului

```
static public string text;
static public string invers;
static void Main(string[] args)
{
    Console.WriteLine("Introduceti un sir de caractere: ");
    text = Console.ReadLine();
    invers = Inversare(text, text.Length - 1);
    Verificare(text);
}
private static void Verificare(string text)
{
    if (text!=invers)
    {
        Console.WriteLine($"Sirul '{text}' nu este palidrom.");
    }
    else
    {
        Console.WriteLine($"Sirul '{text}' este palidrom.");
    }
}
static public string Inversare(string sir, int length)
{
    if (length == -1) return "";
    else
    {
        return sir[length] + Inversare(sir, length - 1);
    }
}
```

### Schema logică



### Rezultatele testării.

Microsoft Visual Studio Debug Console

Introduceti un sir de caractere:  
aerisirea  
Sirul 'aerisirea' este palidrom.

Fig nr. 1 Programul ruleaza corect (caz afirmativ)

Microsoft Visual Studio Debug Console

Introduceti un sir de caractere:  
AAW 2012  
Sirul 'AAW 2012' nu este palidrom.

Fig nr. 2 Programul ruleaza corect (caz negativ)

### Concluzii.

Programul din cadrul temei 5 ne testează abilitățile de a lucra cu șirurile de caractere. La crearea aplicației care întoarce rezultatul necesar, am folosit funcției recursive *Inversare()* care va întoarce șirul citit de la tastatura inversat și procedura *Verificare()* care cu ajutorul instrucțiunii *If* determina respectiv dacă șirul introdus de la tastatura este palindrom sau nu, prin compararea șirului inițial cu cel inversat. Sarcina acestei teme a fost una accesibilă pentru mine și nu am întâmpinat oarecare dificultăți.

## Tema 6. Structuri de date.

### Varianta I

De elaborat o aplicație ce conține structura conform variantei. În aplicație trebuie să se conțină un meniu pentru introducerea/afișarea datelor în fișiere binare.

Structura	Problema
Denumire Autor Furnizor Preț Anul_editării	În meniul aplicației se va adăuga opțiunea ce va afișa informația despre cărțile ce corespunde furnizorului „PRUT”, denumirea căreia se începe cu consoană.

### Listingul programului

public struct Carte	
<pre>public string Denumire; public string Autor; public string Furnizor; public double Pret; public uint Anul_editarii;</pre>	<pre>public void Citire() {     try     {         Console.WriteLine("=====         =====\n\t\tIntroduceti datele cartii: ");         Console.Write("Denumire: ");         Denumire = Console.ReadLine();         if (Denumire.Length == 0)         {             throw new Exception();         }         Console.Write("Autor: ");         Autor = Console.ReadLine();         if (Autor.Length == 0)         {             throw new Exception();         }         Console.Write("Furnizor: ");         Furnizor = Console.ReadLine();         if (Furnizor.Length == 0)         {             throw new Exception();         }         Console.Write("Pret: ");         Pret = int.Parse(Console.ReadLine());         if (Pret == double.NaN)         {             throw new Exception();         }         Console.Write("Anul editarii: ");         Anul_editarii = uint.Parse(Console.ReadLine());         if (Anul_editarii == 0)         {             throw new Exception();         }     }     catch (Exception)     {         Console.WriteLine("Eroare! Introduceti date veridice. ");         Console.ReadKey();         Console.Clear();         Citire();     } }</pre>

```

public void Afisare()
{
    Console.WriteLine("=====Datele cartii =====");
    Console.WriteLine($"\\t\\tDenumire: {Denumire}");
    Console.WriteLine($"\\t\\tAutor: {Autor}");
    Console.WriteLine($"\\t\\tFurnizor: {Furnizor}");
    Console.WriteLine($"\\t\\tPret: {Pret}");
    Console.WriteLine($"\\t\\tAnul editarii: {Anul_editarii}");
    Console.WriteLine();
}

static void Main(string[] args)

```

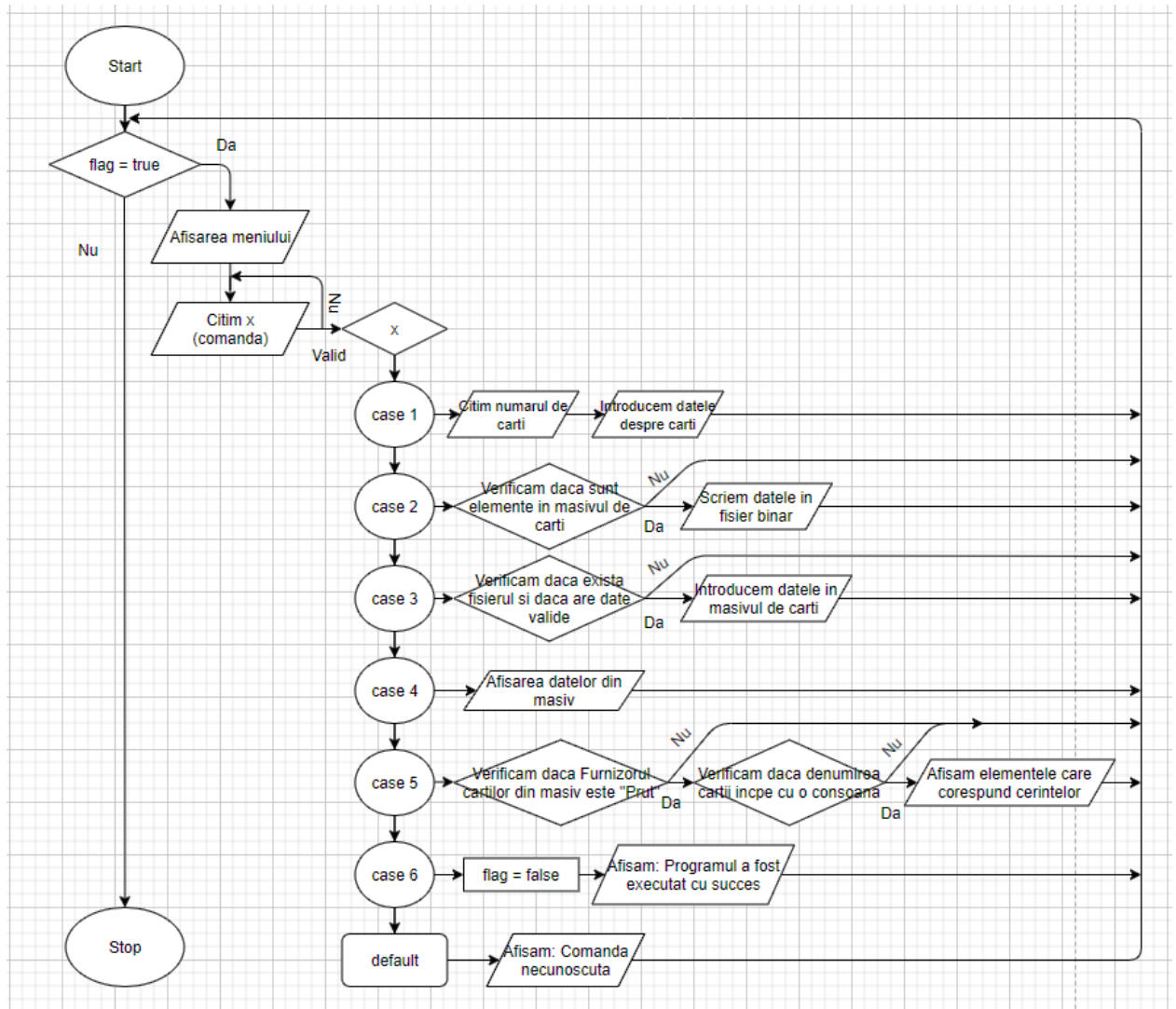
```

public static Carte[] masiv;
bool flag = true;
while (flag)
{
    Console.WriteLine(" \\t\\t\\t\\t Meniul:");
    Console.WriteLine("\\t\\t\\t1. Adaugati carti.");
    Console.WriteLine("\\t\\t\\t2. Adaugati cartile in fisier binar.");
    Console.WriteLine("\\t\\t\\t3. Introducerea datelor din fisier binar. ");
    Console.WriteLine("\\t\\t\\t4. Afisarea datelor citite din fisier binar. ");
    Console.WriteLine("\\t\\t\\t5. Afisati cartile, denumirea carora se incepe cu o consoana.");
    Console.WriteLine("\\t\\t\\t6. Iesire.");
    Console.WriteLine();
    bool isValid = false;
    int c = 0;
    while (!isValid)
    {
        try
        {
            Console.WriteLine("=====");
            Console.WriteLine("\\tIntroduceti comanda: ");
            c = int.Parse(Console.ReadLine());
            isValid = true;
        }
        catch (Exception)
        {
            Console.WriteLine("Comanda invalida!\\nIntroduceti alta comanda.");
            isValid = false;
        }
    }
    switch (c)
    {
        case 1: Citire(); break;
        case 2: ScriereFisier(); break;
        case 3: CitFisier(); break;
        case 4: Afisare(); break;
        case 5: Consoana(); break;
        case 6: flag = false; Console.WriteLine("Aplicatia a fost inchisa!"); break;
        default: Console.WriteLine("Comanda necunoscuta!"); break;
    }
    Console.ReadKey();
    Console.Clear();
}

```

private static void Citire()	private static void ScriereFisier()
<pre> try {     Console.Write("Numarul de carti: ");     int nr = int.Parse(Console.ReadLine());     masiv = new Carte[nr];     Console.WriteLine("Introduceti datele despre carti: ");     Console.WriteLine();     for (int i = 0; i &lt; nr; i++)     {         masiv[i].Citire();     } } catch (Exception) {     Console.WriteLine("Eroare! Introduceti date veridice. ");     Console.ReadKey();     Console.Clear();     Citire(); } Console.Clear(); </pre>	<pre> try {     if (masiv.Length!=0)     {         using (BinaryWriter bw = new BinaryWriter(new FileStream("date.bin", FileMode.Create)))         {             bw.Write(masiv.Length);             foreach (var item in masiv)             {                 bw.Write(item.Denumire);                 bw.Write(item.Autor);                 bw.Write(item.Furnizor);                 bw.Write(item.Pret);                 bw.Write(item.Anul_editarii);             }         }         Console.WriteLine("Fisierul a fost creat cu succes !");     }     else     {         Console.WriteLine("Asigurati-va ca a-ti inregistrat cartile mai intai.");     } } catch (Exception) { Console.WriteLine("Ceva nu a mers cum trebuie, m-ai incercati o data"); } </pre>
private static void Afisare()	
<pre> foreach (var item in masiv) {     item.Afisare(); } </pre>	
private static void CitFisier()	private static void Consoana()
<pre> if (File.Exists("date.bin")) {     if (File.ReadAllText("date.bin")!=null)     {         try         {             using (BinaryReader br = new BinaryReader(new FileStream("date.bin", FileMode.Open)))             {                 int nr = br.ReadInt32();                 masiv = new Carte[nr];                 for (int i = 0; i &lt; nr; i++)                 {                     masiv[i].Denumire = br.ReadString();                     masiv[i].Autor = br.ReadString();                     masiv[i].Furnizor = br.ReadString();                     masiv[i].Pret = br.ReadDouble();                     masiv[i].Anul_editarii = br.ReadInt32();                 }                 Console.WriteLine("Datele au fost citite !");             }         }         catch (Exception)         { Console.WriteLine("Fisierul a fost gasit, insa nu este formatat corect."); }         else {Console.WriteLine("Fisierul a fost gasit, insa nu este formatat corect."); }         else {Console.WriteLine("Fisierul nu a fost gasit.");}     } } </pre>	<pre> string consoane = "bcdfhjklmnpqrstvwxyz"; var sh = from c in masiv.ToList() where c.Furnizor.Contains("Prut") &amp;&amp; consoane.Contains(c.Denumire[0].ToString().ToLower()) select c; if (sh.Count() != 0) {     Console.WriteLine("Lista cartilor denumirea carora incepe cu o consoana: \n=====");     foreach (var item in sh)     {         item.Afisare();     } } else {     Console.WriteLine("Nu au fost gasite carti care sa indeplineasca conditia"); } </pre>

## Schema logică



## Rezultatele testării

```

Meniul:
1. Adaugati carti.
2. Adaugati cartile in fisier binar.
3. Introducerea datelor din fisier binar.
4. Afisarea datelor citite din fisier binar.
5. Afisati cartile, denumirea carora se incepe cu o consoana.
6. Iesire.

=====
Introduceti comanda:
  
```

Fig nr 1. Meniul aplicatiei.

```

=====
      Introduceti comanda: 1
Numarul de carti: 3
Introduceti datele despre carti:

=====
      Introduceti datele cartii:
Denumire: Mara
Autor: Ioan Slavici
Furnizor: Prut
Pret: 230
Anul editarii: 1995
=====
      Introduceti datele cartii:
Denumire: Baltagul
Autor: Mihail Sadoveanu
Furnizor: Arc
Pret: 150
Anul editarii: 1960
=====
      Introduceti datele cartii:
Denumire: Arcada
Autor: George Calinescu
Furnizor: Prut
Pret: 90
Anul editarii: 2005

```

Fig nr. 2 Testarea comenzii 1 (Adaugati carti)

```

=====
      Introduceti comanda: 2
Fisierul a fost creat cu succes !

```

Fig nr. 3 Testarea comenzii 2 (Adaugati cartile in fisier)

```

=====
      Introduceti comanda: 3
Datele au fost citite cu succes !

```

Fig nr. 4 Testarea comenzii 3 (Introducerea datelor din fisiere binare)

```

=====
      Introduceti comanda: 4
===== Datele cartii =====
      Denumire: Mara
      Autor: Ioan Slavici
      Furnizor: Prut
      Pret: 230
      Anul editarii: 1995

===== Datele cartii =====
      Denumire: Baltagul
      Autor: Mihail Sadoveanu
      Furnizor: Arc
      Pret: 150
      Anul editarii: 1960

===== Datele cartii =====
      Denumire: Arcada
      Autor: George Calinescu
      Furnizor: Prut
      Pret: 90
      Anul editarii: 2005

```

Fig nr 5 Testarea comenzii 4 (Afisarea datelor citite din fisier)





## Concluzii.

Aplicația din cadrul temei 6 a fost o provocare pentru mine deoarece aceasta necesita în prelucrare atenție și un bagaj vast de cunoștințe în lucrul cu structurile de date. În elaborarea meniului aplicației am luat în considerare ca acesta să fie unul practic și accesibil, astfel încât să fie utilizat cu ușurință.

Am utilizat instrucțiunea *while* pentru a afișa meniul și a citi comanda la necesitate de câte ori e nevoie până la ieșirea din program. Pentru a simplifica aplicația, fiecărei funcționalități din meniu îi corespunde o funcție sau o procedură, atribuită în urma instrucțiunii *switch*, care prelucrează comanda. Respectiv subprogramele *Citire()*, *ScriereFisier()*, *Afisare()*, *CitFisier()*, *Consoane()* efectuează la cerința utilizatorilor sarcinile. Consider ca complexitatea acestui program a constat în tratarea excepțiilor, astfel încât aplicația să prelucreze datele indiferent de erorile întâlnite.

Pentru îndeplinirea sarcinilor lor am lucrat de asemenea cu fișierele binare, astfel încât datele pot fi citite și din fișier, în cazul în care există un fișier și dacă acesta conține date. Citirea și afișarea masivului, citit de la tastatură sau generat din fișier se îndeplinesc cu ajutorul procedurilor care se afla în *structura Carte*.

În testarea rezultatelor am avut nevoie de atenție și precauție, pentru a observa dacă aceasta lucrează în totalitate. Într-un final sunt mandra că am reușit să îndeplinesc toate sarcinile, fapt ce mi-a demonstrat că pot aplica toate cunoștințele acumulate în decursul unui an de studiu.

### **Bibliografie și resurse:**

- 1. Literatura din suport de curs.**
- 2. [metanit.com](http://metanit.com)**
- 3. [tproger.ru](http://tproger.ru)**
- 4. [visualstudio.com](http://visualstudio.com)**
- 5. [msdn.com](http://msdn.com)**
- 6. [ulearn.me](http://ulearn.me)**
- 7. [stackoverflow.com](http://stackoverflow.com)**