Laborator 6a

Tablouri multidimensionale

În cazul tablourilor cu mai multe dimensiuni facem distincţie între **tablouri regulate** şi **tablouri neregulate (tablouri de tablouri)**

Declararea în cazul **tablourilor regulate bidimensionale** se face astfel:

```
Tip[,] nume;
```

iar **instanţierea**:

```
nume = new Tip[Linii,Coloane];
```

Accesul:

```
nume[indice1,indice2]
```

Exemplu: Declararea instanţierea şi iniţializarea

```
int[,] mat = new int[,] {{1,2,3},{4,5,6},{7,8,9}};
```

sau

```
int[,] mat = {{1,2,3},{4,5,6},{7,8,9}};
```

În cazul **tablourilor neregulate (jagged array) declararea** se face:

```
    Tip [][] nume; //tablou neregulat cu doua
                   //dimensiuni
```

iar **instanţierea şi iniţializarea**:

```
Tip [][] nume = new Tip[][]
              {
                   new Tip[] {sir_0},
                   new Tip[] {sir_1},
                   ...
                   new Tip[] {sir_n}
              };
```

sau

```
Tip [][] nume = {
                   new Tip[] {sir_0},
                   new Tip[] {sir_1},
                   ...
                   new Tip[] {sir_n}
                };
```

Acces

```
nume[indice1][indice2]
```

Exemple:

```
int[][] mat = new int[][]
{
     new int[3] {1,2,3},
     new int[2] {4,5},
     new int[4] {7,8,9,1}
};
```

sau

```
int[][] mat = {
                    new int[3] { 1, 2, 3 },
                    new int[2] { 4, 5 },
                    new int[4] { 7, 8, 9, 1 }
                };
```

Observaţie: Este posibilă declararea vectorilor de dimensiuni mai mari.

Exemple:

```
int[, ,] vect = new int[2, 3, 5];
```

```
int[, , ,] vect = new int[6, 2, 4, 8];
```

Vectorii 3-D sunt utilizaţi frecvent în aplicaţiile grafice.

Exemplul 48: Descompunerea unui număr în sumă de numere naturale consecutive. Se citeşte un număr natural n. Să se memoreze toate posibilităţile de descompunere a numărului n în sumă de numere consecutive.

Dacă numărul n se scrie ca sumă de numere naturale consecutive, atunci rezultă că există

i,k∈N* astfel încât

i+(i+1)+(i+2)+(i+3)+……+(k)=n

↔ (1+2+...+k)-(1+2+...+i-1)=n↔k*(k+1)/2-i*(i-1)/2=n

↔k^2+k-i^2+i-2n=0

↔k=(-1+$\sqrt{1+8n-4i+4i^2}$)/2

Vom memora descompunerile în matricea neregulată a (descompunerile au dimensiuni variabile).

```
using System;
namespace Exemplul_48
{
    class Program
    {
        static void Main()
        {
            Console.Write("Introduceti un numar natural ");
            int n = Convert.ToInt32(Console.ReadLine());
            int[][] a = new int[n / 2][];
            int l = 0, i, j;
            for (i = 1; i <= n / 2; i++)
            {
                double k = (Math.Sqrt(1 + 8 * n - 4 * i + 4 * i * i) - 1)/ 2;
                if (k == (int)k)
                {
                    a[l] = new int[(int)k - i + 1];
                    for (j = i; j <= k; j++) a[l][j - i] = j;
                    l++;
                }
            }
            Console.WriteLine("Descompunerea lui {0} in suma de numere
naturale consecutive", n);
            for (i = 0; i < l; i++)
            {
                for (j = 0; j < a[i].Length; j++)
                    Console.Write(a[i][j] + " ");
                Console.WriteLine();
            }
        }
    }
}
```

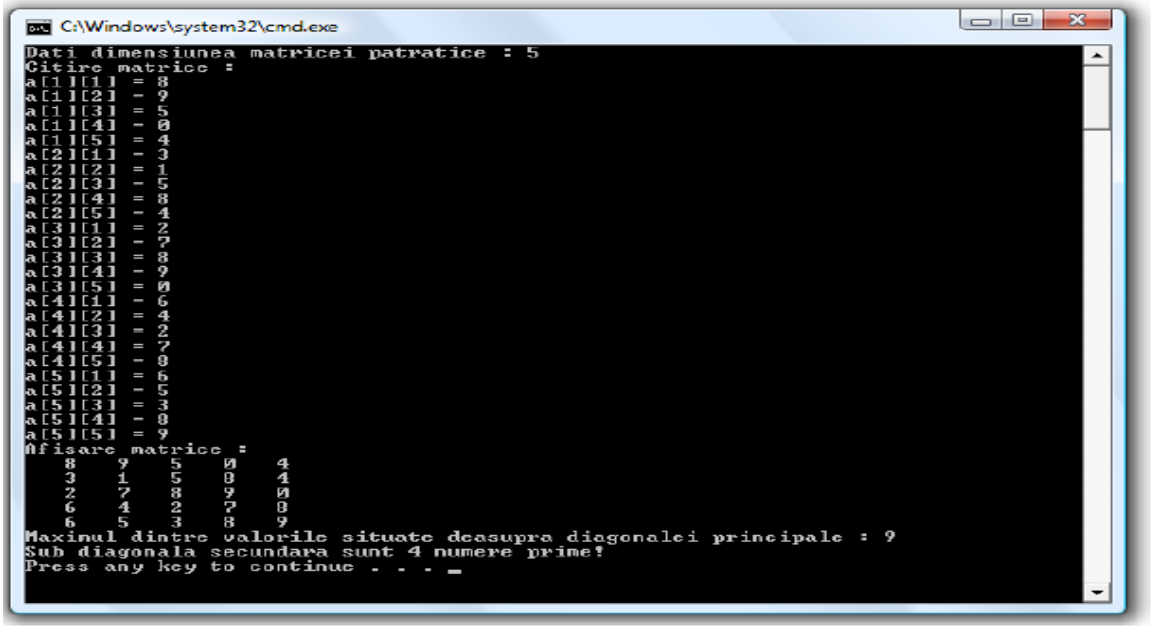
Exemplul_49: Pentru o matrice pătratică, ale cărei elemente întregi se citesc de la tastatură, să se determine:

- maximul dintre valorile situate deasupra diagonalei principale
- numărul de numere prime (dacă acestea există) situate sub diagonala secundară

```
using System;
using System.Collections.Generic;
using System.Text;

namespace Exemplul_49
{
    class Program
    {
        static void Main(string[] args)
        {
            int i, j, n;
            Console.Write("Dati dimensiunea matricei patratice : ");
            n = Convert.ToInt32(Console.ReadLine());
            int[,] a;
            a = new int[n + 1, n + 1];
            Console.WriteLine("Citire matrice : ");
            for (i = 0; i < n; i++)
                for (j = 0; j < n; j++)
                {
                    Console.Write("a[{0}][{1}] = ", i + 1, j + 1);
                    a[i, j] = Convert.ToInt32(Console.ReadLine());
                }
            Console.WriteLine("Afisare matrice : ");
            for (i = 0; i < n; i++)
            {
                for (j = 0; j < n; j++)
                    Console.Write("{0, 4}", a[i, j]);
                Console.WriteLine();
            }
            int max = a[0, 1];
            for (i = 0; i < n - 1; i++)
                for (j = i + 1; j < n; j++)
                    if (a[i, j] > max) max = a[i, j];
            Console.WriteLine("Maximul dintre valorile situate deasupra
diagonalei principale : {0}", max);
            int k = 0;
            for (i = 1; i < n; i++)
                for (j = n - i; j < n; j++)
                    if (prim(a[i, j]) == 1) k++;
            if (k == 0)
                Console.WriteLine("Sub diagonala secundara nu sunt numere
prime!");
            else
                Console.WriteLine("Sub diagonala secundara sunt {0} numere
prime!", k);


        }
        static int prim(int x)
        {
            if (x == 1) return 0;
            if (x % 2 == 0 && x != 2) return 0;
            for (int d = 3; d * d <= x; d += 2)
                if (x % d == 0) return 0;
            return 1;
        }
    }
}
```



Laborator 6b

Stocarea informaţiilor în fişiere

Administrarea fişierelor

Tehnica de citire şi scriere a datelor în şi din fişiere, utilizată pentru a păstra aceste informaţii, reprezintă administrarea fişierelor.

Pentru accesarea unui fişier de pe disc se folosesc funcţii din spaţiul de nume **System.IO**.

În acest spaţiu există mai multe clase: **File, StreamWriter, BinaryReader şi BinaryWriter**.

Aceste clase sunt folosite pentru operaţiile de intrare-ieşire cu fişiere.

Obiectul **File** este o reprezentare a unui fişier de pe disc, iar pentru a-l utiliza trebuie să îl conectăm la un flux (**stream**).

Pentru a scrie datele pe disc, se ataşează unui flux un obiect **File**. Astfel se face administrarea datelor.

Limbajul C# oferă două tipuri de fişiere: fişiere text şi fişiere binare.

Scrierea şi citirea datelor din fişiere text

Fişierele de ieşire necesită utilizarea unui obiect **StreamWriter**.

Funcţia **CreateText()**, ce face parte din clasa **File**, deschide un fişier şi creează obiectul **StreamWriter**.

Exemplul 56:

```
using System;
using System.Collections.Generic;
using System.Text;
using System.IO;

namespace Exemplul_56
{
    class Program
    {
        static void Main(string[] args)
        {
            string[] a = { "primul", "fisier", "creat", "de mine", };
            //deschiderea unui fisier si atasarea lui la un flux
            StreamWriter outputFile = File.CreateText("C:\\C#\\fisier1.txt");
```

```
            foreach (string b in a)
            {
                outputFile.WriteLine(b);//scrierea textului in fisier
            }
            //inchiderea fisierului
            outputFile.Close();
            //deschidem din nou fisierul de data aceasta pentru a citi din el
            StreamReader inputFile = File.OpenText("C:\\C#\\fisier1.txt");
            //definim o variabila string care va parcurge fisierul pana la
final
            string x;
            while ((x = inputFile.ReadLine()) != null)
            {
                System.Console.WriteLine(x);
            }
            //inchidem fisierul
            inputFile.Close();
        }

    }
}
```

Scrierea şi citirea datelor din fişiere binare

Dacă la fişierele text tipul de flux folosit era **StreamWriter**, la cele binare, pentru scrierea datelor programul creează un obiect **FileStream**, la care trebuie ataşat şi un obiect **BinaryWriter**.

Exemplul 57:

```
using System;
using System.Collections.Generic;
using System.Text;
using System.IO;
namespace Exemplul_57
{
    class Program
    {
        static void Main(string[] args)
        {
            int i, j, x;
            int[,] a = new int[10, 10];
            //se creeaza un fisier si un flux
            FileStream f = new FileStream("C:\\C#\\fisier2.dat",
FileMode.CreateNew);
            // se creeaza un scriitor binar si il ataseaza la flux
            //acesta traduce datele fluxului in format binar
            BinaryWriter outputFile = new BinaryWriter(f);
            for (i = 1; i <= 4; i++)
                for (j = 1; j <= 4; j++)
                    if (i == j) a[i, j] = 1;
                    else if (j == 5 - i) a[i, j] = 2;
                    else a[i, j] = 0;
            for (i = 1; i <= 4; i++)
                for (j = 1; j <= 4; j++)
                    outputFile.Write(a[i, j]);
            //se inchide fisierul creat
            outputFile.Close();
            f.Close();
            //incepe citirea datelor din fisierul creat mai sus
            //se creeaza un obiect FileStream
            FileStream g = new FileStream("C:\\C#\\fisier2.dat",
FileMode.Open);
            //se creeaza un obiect BinaryReader
            BinaryReader inputFile = new BinaryReader(g);
            bool final;
            for (final = false, i = 1; !final; i++)
            {
                for (final = false, j = 1; !final; j++)
                {//se apeleaza functia PeekChar care face parte din clasa
BinaryReader
                    //si examineaza urmatorul caracter din flux, daca acesta
este diferit de -1
                    // atunci se executa citirea urmatorului caracter din
flux prin functia ReadInt32()
                    if (inputFile.PeekChar() != -1)
                    {
                        x = inputFile.ReadInt32();
                        System.Console.Write("{0} ", x);
                    }
                }
                System.Console.Write("\n");
            }
            inputFile.Close();
            g.Close();
        }
    }
}
```