

T12. Sudėtingesnis konteineris

2 ak. val.

Temos klausimai

1. Savybės (properties).
2. Dvimatis masyvas (matrica).
3. Aprašymas ir atminties išskyrimas.
4. Veiksmai eilutėse ir stulpeliuose.
5. Dvimatis objektų konteineris.
6. Kvadratinė matrica.
7. Masyvų masyvas.



Savybės (properties)

Savybės sąvoka

Savybės - tai *specialus* **public** klasės narių tipas.

Savybės leidžia *saugiai* ir paprastai keisti privačių (**private**) klasės narių reikšmes.

Jos realizuoja klasės sąsajos metodus **Get()** ir **Set()**.

Trijų savybių pavyzdžiai:

```
public int Sv { get; set; }
```

```
public double Re { get; set; }
```

```
public string Eil { get; set; }
```

Klasės aprašymo pavyzdys (1/8)

Palyginimui, sukursime tris vienodas klases taško koordinatėms plokštumoje aprašyti. Kiekvieną klasę skirtingai realizuosime:

- *Pirmosios klasės nariai*: du kintamieji, konstruktorius ir 4 sąsajos metodai;
- *Antrosios klasės nariai*: du kintamieji, konstruktorius ir dvi savybės;
- *Trečiosios klasės nariai*: dvi automatinės savybės ir konstruktorius.

Visos klasės vienodai gerai leis sukurti šių klasių objektus ir atlikti veiksmus su jais.

Klasės aprašymo pavyzdys (2/8)

```
// Pirmoji taško klasė
```

```
class Taskas
```

```
{
```

```
    private int x;        // taško koordinatė x
```

```
    private int y;        // taško koordinatė y
```

```
    // Konstruktorius su numatytosiomis reikšmėmis
```

```
    public Taskas(int x = 0, int y = 0)
```

```
    {
```

```
        this.x = x;
```

```
        this.y = y;
```

```
    }
```

```
    // Sąsajos metodai
```

```
    public int GetX() { return x; }
```

```
    public void SetX(int x) { this.x = x; }
```

```
    public int GetY() { return y; }
```

```
    public void SetY(int y) { this.y = y; }
```

```
}
```

Klasės aprašymo pavyzdys (3/8)

```
// Antroji taško klasė su dviem savybėmis x ir y
class TaskasP
{
    private int x;        // taško koordinatė x
    private int y;        // taško koordinatė y

    public int X // savybė x: taško koordinatė x
    {
        get { return x; }
        set { x = value; }
    }
    public int Y // savybė y: taško koordinatė y
    {
        get { return y; }
        set { y = value; }
    }

    // konstruktorius su numatytosiomis reikšmėmis
    public TaskasP(int x = 0, int y = 0)
    {
        X = x;
        Y = y;
    }
}
```

Greitam savybės šablono sukūrimui reikia užrašyti žodį `prop` ir du kartus greitai paspausti `Tab` klavišą.

Klasės aprašymo pavyzdys (4/8)

// Trečioji taško klasė su automatinėmis savybėmis x ir y

class TaskasPA

{

public int x { get; set; } // savybė x: taško koordinatė x

public int y { get; set; } // savybė y: taško koordinatė y

// konstruktorius su numatytosiomis reikšmėmis

public TaskasPA(int x = 0, int y = 0)

{

 x = x;

 y = y;

}

}


```
static void Main(string[] args)
{
    Taskas T1 = new Taskas();
    Console.WriteLine("T1: x = {0} y = {1}",
                      T1.ImtiX(), T1.ImtiY());
    T1.DetiX(9);
    Console.WriteLine("T1 x = {0} y = {1}",
                      T1.ImtiX(), T1.ImtiY());
    Taskas T2 = new Taskas(4, 6);
    Console.WriteLine("T2: x = {0} y = {1}",
                      T2.ImtiX(), T2.ImtiY());
    ...
}
```

```
static void Main(string[] args)
{
    ...
    TaskasP TP1 = new TaskasP();
    Console.WriteLine("TP1: x = {0} y = {1}",
                      TP1.X, TP1.Y);
    TP1.X = 9;
    Console.WriteLine("TP1: x = {0} y = {1}",
                      TP1.X, TP1.Y);
    TaskasP TP2 = new TaskasP(4, 6);
    Console.WriteLine("TP2: x = {0} y = {1}",
                      TP2.X, TP2.Y);
    ...
}
```

```
static void Main(string[] args)
{
    ...
    TaskasPA TPA1 = new TaskasPA();
    Console.WriteLine("TPA1: x = {0} y = {1}",
                      TPA1.X, TPA1.Y);
    TPA1.X = 9;
    Console.WriteLine("TPA1: x = {0} y = {1}",
                      TPA1.X, TPA1.Y);
    TaskasPA TPA2 = new TaskasPA(4, 6);
    Console.WriteLine("TPA2: x = {0} y = {1}",
                      TPA2.X, TPA2.Y);
}
```

Rezultatai (8/8)

T1: $x = 0$ $y = 0$

T1: $x = 9$ $y = 0$

T2: $x = 4$ $y = 6$

TP1: $x = 0$ $y = 0$

TP1: $x = 9$ $y = 0$

TP2: $x = 4$ $y = 6$

TPA1: $x = 0$ $y = 0$

TPA1: $x = 9$ $y = 0$

TPA2: $x = 4$ $y = 6$

Pastaba: Visais atvejais (3) gaunami tie patys rezultatai, kadangi atliekami identiškai veiksmai su objektais.



Dvimatis masyvas

Dvimatis masyvas grafiškai (1/2)

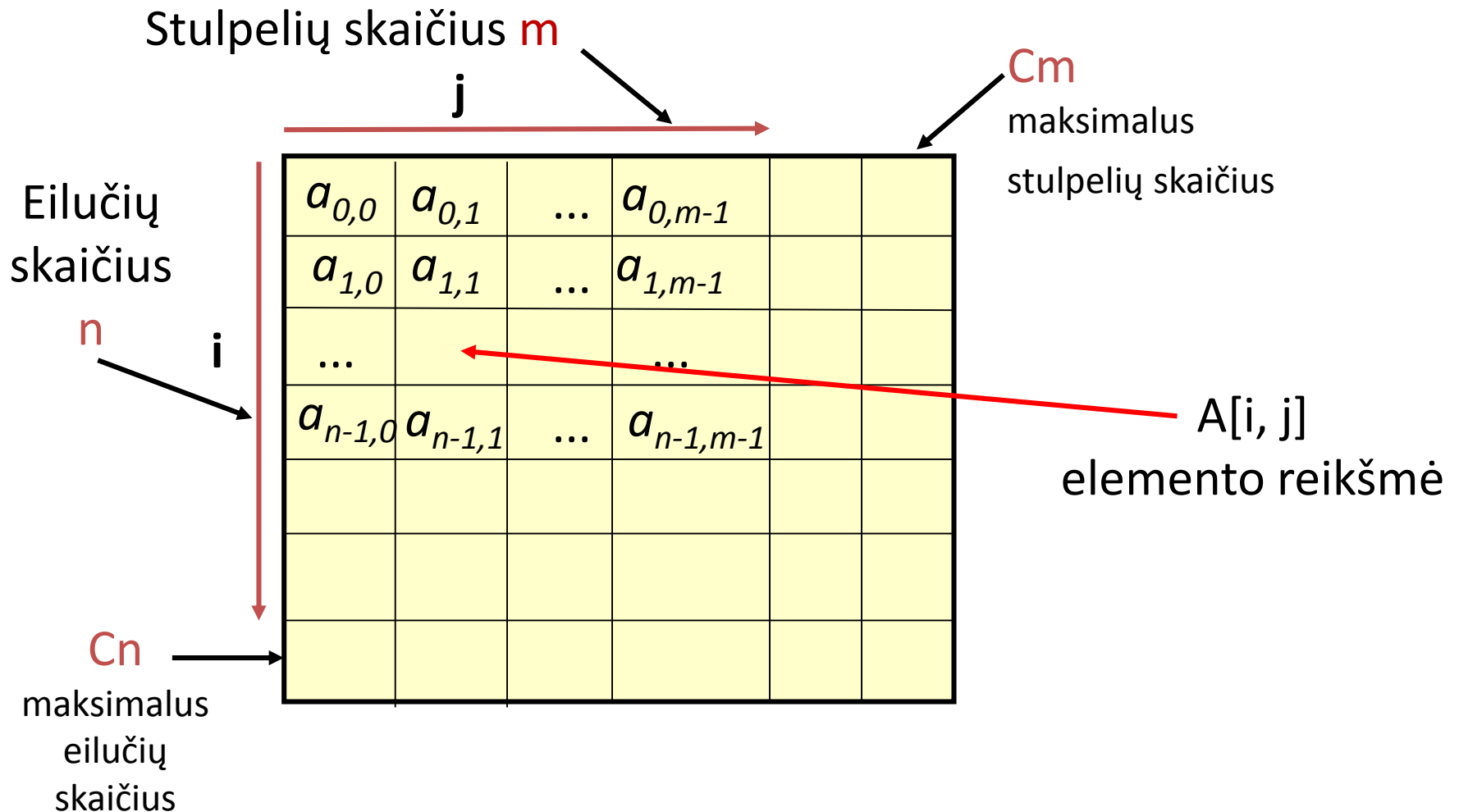
3 stulpeliai

4 eilutės

Elementas:
2 eilutė
0 stulpelis

	0	1	2
0	5	8	6
1	9	2	0
2	-5	1	-2
3	4	-8	7

Dvimatis masyvas grafiškai (2/2)



Dvimačio masyvo nuorodos sukūrimas:

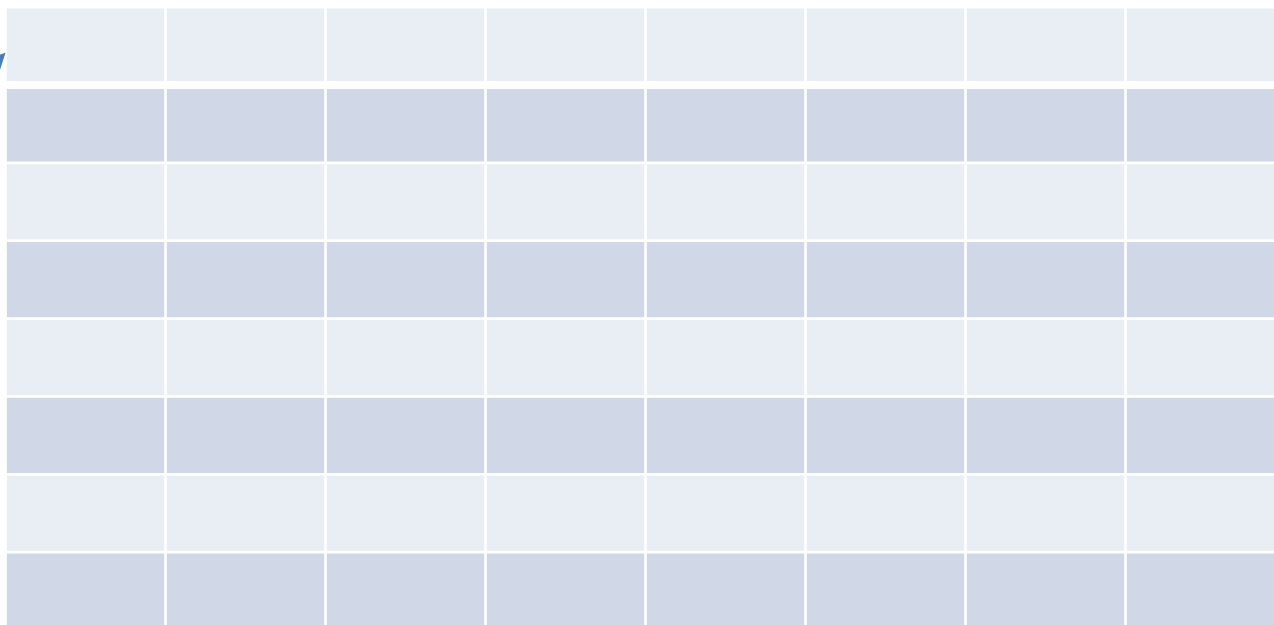
```
int[,] DMas;
```

Dvimačio masyvo nuorodos sukūrimas ir atminties išskyrimas:

```
int[,] DMas = new int[Cn,Cm];
```


Dvimačio masyvo aprašymas ir atminties išskyrimas (2/2)

DMas



Maksimalus eilučių skaičius C_n

Maksimalus stulpelių skaičius C_m

Dvimačio masyvo aprašymas, indeksacija, kreipiniai (1/3)

```
const int Cn = 5;    // maksimalus eilučių skaičius  
const int Cm = 10;   // maksimalus stulpelių skaičius  
int[,] DMas;         // dvimačio sveikų skaičių masyvo nuoroda  
int n, m;            // eilučių ir stulpelių skaičius
```

Rekomenduojama:

indeksas eilutei: $i = 0, n - 1$

indeksas stulpeliui: $j = 0, m - 1$

Kreipinys į dvimačio masyvo elementą i, j : **DMas[i, j]**

Dvimačio masyvo aprašymas, indeksacija, kreipiniai (2/3)

`DMas = new int[Cn, Cm];`

`DMas.GetLength(0)` – grąžina maksimalų **eilučių** skaičių

`DMas.GetLength(1)` – grąžina maksimalų **stulpelių** skaičių

Pastaba: maksimalus eilučių skaičius bus lygus C_n , o maksimalus stulpelių skaičius bus lygus C_m .

Dvimačio masyvo aprašymas, indeksacija, kreipiniai (3/3)

```
int[,] DMas = {
    { 1, 2, 3, 4 }, // eilutės 0 reikšmės
    { 5, 6, 7, 8 }, // eilutės 1 reikšmės
};

for (int i = 0; i < DMas.GetLength(0); i++)
{
    for (int j = 0; j < DMas.GetLength(1); j++)
    {
        Console.Write("{0, 2:d} ", DMas[i, j]);
    }
    Console.WriteLine();
}
```

1	2	3	4
5	6	7	8

Konteinerinė klasė su dvimačiu masyvu (1/2)

```
class Matrica
```

```
{
```

```
    public const int Cn = 50;  
    public const int Cm = 100;
```

```
// maksimalus eilučių skaičius  
// maksimalus stulpelių skaičius
```

```
    private int[,] DMas;  
    public int N { get; set; }  
    public int M { get; set; }
```

```
// dvimatis sveikų skaičių masyvas  
// savybė N: eilučių skaičius  
// savybė M: stulpelių skaičius
```

```
    public Matrica()
```

```
{
```

```
        DMas = new int[Cn, Cm];  
        N = 0;  
        M = 0;
```

Atminties išskyrimas
dvimačiam masyvui

```
}
```

```
    ... // sąsajos metodai
```

```
}
```

Konteinerinė klasė su dvimačiu masyvu (2/2)

```
class Matrica
{
    ...
    public void Deti(int i, int j, int sk)
    {
        DMas[i, j] = sk;
    }

    public int Imti(int i, int j)
    {
        return DMas[i, j];
    }
}
```

Duomenų failo pavyzdys

4	5			
5	2	1	6	3
7	0	5	2	-4
-8	6	4	-7	0
3	-5	-8	9	7

Rekomenduojama:

- pirmoje eilutėje rašyti tik eilučių ir stulpelių skaičių;
- reikšmes eilutėse surašyti tvarkingai, lygiuojant stulpelius.

Dvimačio masyvo reikšmių skaitymas

```
// Konteinerio A užpildymas duomenimis iš failo fv
static void skaityti(string fv, Matrica A)
{
    using (StreamReader reader = new StreamReader(fv)) {
        int skaicius;
        string line = reader.ReadLine();
        char[] skyr = { ' ' };
        string[] skaiciai = line.Split(skyr,
                                      StringSplitOptions.RemoveEmptyEntries);
        A.N = int.Parse(skaiciai[0]);
        A.M = int.Parse(skaiciai[1]);
        for (int i = 0; i < A.N; i++) {
            line = reader.ReadLine();
            skaiciai = line.Split(skyr,
                                StringSplitOptions.RemoveEmptyEntries);
            for (int j = 0; j < A.M; j++) {
                skaicius = int.Parse(skaiciai[j]);
                A.Deti(i, j, skaicius);
            }
        }
    }
}
```

i-osios eilutės
užpildymas


```
// konteinerio A duomenų spausdinimas faile fv
```

```
static void Spausdinti(string fv, Matrica A)
```

```
{
```

```
    using (var fr = File.AppendText(fv))
```

```
    {
```

```
        for (int i = 0; i < A.N; i++)
```

```
        {
```

```
            for (int j = 0; j < A.M; j++)
```

```
            {
```

```
                fr.Write("{0, 4:d}", A.Imti(i, j));
```

```
            }
```

```
            fr.WriteLine();
```

```
        }
```

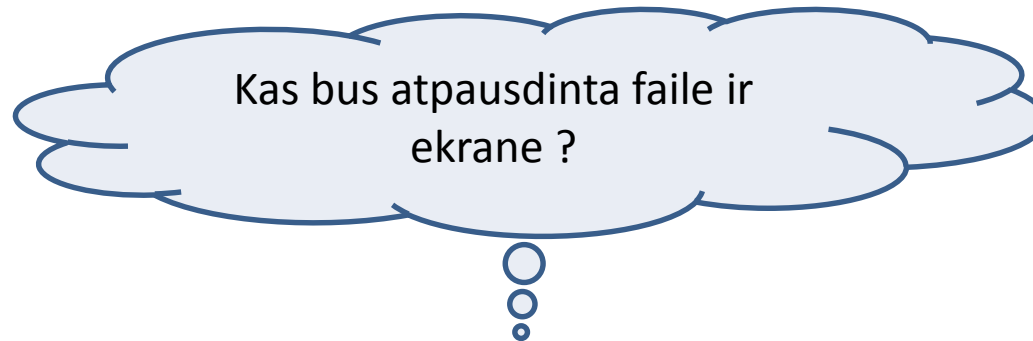
```
    }
```

```
}
```

i-osios eilutės
spausdinimas

Dvimačio masyvo reikšmių įvedimas ir spausdinimas

```
static void Main(string[] args)
{
    Matrica Mtr = new Matrica(); // konteineris su dvimačiu masyvu
    Skaityti(CFd, Mtr);
    Spausdinti(CFr, Mtr);
    Console.WriteLine("Eilučių skaičius:{0,4:d}", Mtr.N);
    Console.WriteLine("Stulpelių skaičius:{0,4:d}", Mtr.M);
}
```



Rezultatai faile ([pageidaujami](#)) ir ekrane

	1	2	3	4	5

1 :	5	2	1	6	3
2 :	7	0	5	2	-4
3 :	-8	6	4	-7	0
4 :	3	-5	-8	9	7

Savarankiškai
papildykite
spausdinimo
metodą taip, kad
rezultatas būtų
toks.

Spausdinant dvimatį masyvą rekomenduojama nurodyti eilučių ir stulpelių numerius.

Eilučių skaičius: 4
Stulpelių skaičius: 5

- su viso masyvo elementais
- su eilutės elementais
- su stulpelio elementais
- su masyvo dalies elementais (kvadratinė matrica)




Veiksmiai su viso masyvo reikšmėmis

Visos dvimačio masyvo reikšmės

$a_{0,0}$	$a_{0,1}$	$a_{0,2}$	$a_{0,3}$	$a_{0,4}$	$a_{0,5}$
$a_{1,0}$	$a_{1,1}$	$a_{1,2}$	$a_{1,3}$	$a_{1,4}$	$a_{1,5}$
$a_{2,0}$	$a_{2,1}$	$a_{2,2}$	$a_{2,3}$	$a_{2,4}$	$a_{2,5}$
$a_{3,0}$	$a_{3,1}$	$a_{3,2}$	$a_{3,3}$	$a_{3,4}$	$a_{3,5}$
$a_{4,0}$	$a_{4,1}$	$a_{4,2}$	$a_{4,3}$	$a_{4,4}$	$a_{4,5}$


Eilučių skaičius $n = 5$; stulpelių skaičius $m = 6$

```
// Paruošiamieji veiksmai
for (int i = 0; i < n; i++)
{
    for (int j = 0; j < m; j++)
    {
        // veiksmai su Dmas[i,j]
    }
}
// skaičiavimų rezultatai
```



Elementai nagrinėjami **eilutėmis**

```
// Paruošiamieji veiksmai
for (int j = 0; j < m; j++)
{
    for (int i = 0; i < n; i++)
    {
        // veiksmai su Dmas[i,j]
    }
}
// skaičiavimų rezultatai
```



Elementai nagrinėjami **stulpeliais**

Pavyzdys: dvimačio masyvo reikšmių aritmetinis vidurkis

```
static double vidurkis(Matrica A)
```

```
{
```

```
    int suma = 0;
```

```
    for (int i = 0; i < A.N; i++)
```

```
    {
```

```
        for (int j = 0; j < A.M; j++)
```

```
        {
```

```
            suma += A.Imti(i, j);
```

```
        }
```

```
    }
```

```
    if (A.N * A.M != 0)
```

```
        return (double)suma / (A.N * A.M);
```

```
    else
```

```
        return 0.0;
```

```
}
```

```
...
```

```
Console.WriteLine("Vidurkis:{0, 7:f2}", vidurkis(Mtr));
```

5	2	1	6	3
7	0	5	2	-4
-8	6	4	-7	0
3	-5	-8	9	7

Pavyzdys: didžiausios reikšmės vieta

```
static void RastiDidVieta(Matrica A, out int di, out int dj)
{
    int did = A.Imti(0, 0);
    di = 0;
    dj = 0;
    for (int i = 0; i < A.N; i++)
        for (int j = 0; j < A.M; j++)
            if (A.Imti(i, j) > did)
            {
                did = A.Imti(i, j);
                di = i; dj = j;
            }
}

...
int iMax, jMax;
RastiDidVieta(Mtr, out iMax, out jMax);
Console.WriteLine("Didžiausias skaičius:{0, 4:d}",
    Mtr.Imti(iMax, jMax));
```

5	2	1	6	3
7	0	5	2	-4
-8	6	4	-7	0
3	-5	-8	9	7



Veiksmai su eilutės reikšmėmis

Viena dvimačio masyvo eilutė

$a_{0,0}$	$a_{0,1}$	$a_{0,2}$	$a_{0,3}$	$a_{0,4}$	$a_{0,5}$
$a_{1,0}$	$a_{1,1}$	$a_{1,2}$	$a_{1,3}$	$a_{1,4}$	$a_{1,5}$
$a_{2,0}$	$a_{2,1}$	$a_{2,2}$	$a_{2,3}$	$a_{2,4}$	$a_{2,5}$
$a_{3,0}$	$a_{3,1}$	$a_{3,2}$	$a_{3,3}$	$a_{3,4}$	$a_{3,5}$
$a_{4,0}$	$a_{4,1}$	$a_{4,2}$	$a_{4,3}$	$a_{4,4}$	$a_{4,5}$

Eilučių skaičius $n = 5$; stulpelių skaičius $m = 6$

Veiksmai su dvimačio masyvo eilutėmis

```
for (int i = 0; i < n; i++)  
{  
    // Paruošiamieji veiksmai  
    for (int j = 0; j < m; j++)  
    {  
        // veiksmai su Dmas[i,j] reikšme  
    }  
    // skaičiavimų rezultatai  
}
```

Pavyzdys: veiksmas su dvimačio masyvo eilutėmis

// Eilučių didžiausias reikšmės pakeičia x

```
static void PakeistiEilDidziausia(Matrica A, int x)
{
    for (int i = 0; i < A.N; i++)
    {
        int did = A.Imti(i, 0);
        int dj = 0;
        for (int j = 1; j < A.M; j++)
            if (A.Imti(i, j) > did)
            {
                did = A.Imti(i, j);
                dj = j;
            }
        A.Deti(i, dj, x);
    }
}
```

5	2	1	6	3
7	0	5	2	-4
-8	6	4	-7	0
3	-5	-8	9	7

Koks bus
rezultatas?

...
PakeistiEilDidziausia(Mtr, 99);



Veiksmai su stulpelio reikšmėmis

Viena dvimačio masyvo stulpelis

$a_{0,0}$	$a_{0,1}$	$a_{0,2}$	$a_{0,3}$	$a_{0,4}$	$a_{0,5}$
$a_{1,0}$	$a_{1,1}$	$a_{1,2}$	$a_{1,3}$	$a_{1,4}$	$a_{1,5}$
$a_{2,0}$	$a_{2,1}$	$a_{2,2}$	$a_{2,3}$	$a_{2,4}$	$a_{2,5}$
$a_{3,0}$	$a_{3,1}$	$a_{3,2}$	$a_{3,3}$	$a_{3,4}$	$a_{3,5}$
$a_{4,0}$	$a_{4,1}$	$a_{4,2}$	$a_{4,3}$	$a_{4,4}$	$a_{4,5}$

Eilučių skaičius $n = 5$; stulpelių skaičius $m = 6$

```
for (int j = 0; j < m; j++)  
{  
    // Paruošiamieji veiksmai  
    for (int i = 0; i < n; i++)  
    {  
        // veiksmai su Dmas[i,j] reikšme  
    }  
    // skaičiavimų rezultatai  
}
```


Pavyzdys: veiksmas su dvimačio masyvo stulpeliais

// Stulpelių didžiausias reikšmės pakeičia x

```
static void PakeistiStulpDidziausia(Matrica A, int x)
{
    for (int j = 0; j < A.M; j++)
    {
        int did = A.Imti(0, j);
        int di = 0;
        for (int i = 1; i < A.N; i++)
            if (A.Imti(i, j) > did)
            {
                did = A.Imti(i, j);
                di = i;
            }
        A.Deti(di, j, x);
    }
}

...
PakeistiStulpDidziausia(Mtr, 99);
```

5	2	1	6	3
7	0	5	2	-4
-8	6	4	-7	0
3	-5	-8	9	7

Koks bus
rezultatas?



Dvimatis objektų konteineris. Programos pavyzdys

Užduotis

Stačiakampiame sodo sklype augalai pasodinti eilėmis (iš viso n eilių) vakarų – rytų kryptimi. Kiekvienoje eilėje yra m augalų. Faile **Sodas.txt** užrašyta kiekvieno augalo pavadinimas ir amžius.

Savininkas ruošiasi atnaujinti sodą. Kiekvieno augalo "kritinis" amžius yra žinomas. Šie duomenys yra faile **Augalai.txt**.

Kiek kokių augalų reikia atnaujinti?

Duomenų failų pavyzdžiai

Kodėl taip galima?

Sodas.txt

4 3

Kriaušė;	10;	obelis;	3;	vyšnia;	8;
obelis	3;	slyva;	5;	obelis;	14;
obelis	6;	slyva;	9;	slyva;	10;
Agrastas	3;	Agrastas;	2;	Agrastas;	3;

Augalai.txt

6

Agrastas;	5;
Kriaušė;	8;
obelis;	10;
Serbentas;	5;
slyva;	8;
vyšnia;	6;

Sodo augalų amžius

"Kritinis" amžius

Laukiami rezultatai

Agrastas	0
Kriaušė	1
Obelis	1
Serbentas	0
Slyva	2
Vyšnia	1

Augalų, kuriuos reikia
pakeisti naujais, kiekis

Reikalingos klasės (1/3)

```
class Augalas
{
    public string Pavadinimas { get; set; } // savybė: augalo pavadinimas
    public int Skaicius { get; set; }       // savybė: augalo amžius arba kiekis
    public Augalas(string pav = "", int sk = 0)
    {
        Pavadinimas = pav;
        Skaicius = sk;
    }
    public void DidintiSkaiciu(int k)
    {
        Skaicius += k;
    }
    public static bool operator >(Augalas a1, Augalas a2)
    {
        return a1.Skaicius > a2.Skaicius;
    }
    public static bool operator <(Augalas a1, Augalas a2)
    {
        return a1.Skaicius < a2.Skaicius;
    }
}
```

Reikalingos klasės (2/3)

```
class Konteineris
{
    public const int CMax = 100;
    private Augalas[] Augalai;
    public int N { get; set; }    // savybė N: augalų skaičius
    public Konteineris()
    {
        Augalai = new Augalas[CMax];
        N = 0;
    }
    public void DetiAugala(int i, Augalas a)
    {
        Augalai[i] = a;
    }
    public Augalas ImtiAugala(int i)
    {
        return Augalai[i];
    }
}
```

Reikalingos klasės (3/3)

```
class DvimKonteineris
{
    public const int CMaxEil = 10;
    public const int CMaxSt = 100;
    private Augalas[,] Augalai;
    public int N { get; set; } // savybė N: eilučių skaičius
    public int M { get; set; } // savybė M: stulpelių skaičius
    public DvimKonteineris()
    {
        Augalai = new Augalas[CMaxEil, CMaxSt];
        N = 0; M = 0;
    }
    public void DetiAugala(int i, int j, Augalas a)
    {
        Augalai[i, j] = a;
    }
    public Augalas ImtiAugala(int i, int j)
    {
        return Augalai[i, j];
    }
}
```


Duomenų įvedimas (1/2)

```
static void skaityti(string fv, Konteineris A)
{
    using (StreamReader reader = new StreamReader(fv))
    {
        string line = reader.ReadLine();
        A.N = int.Parse(line);
        string[] Eilutes;
        char[] skyr = { ' ', ';' };
        for (int i = 0; i < A.N; i++)
        {
            line = reader.ReadLine();
            Eilutes = line.Split(skyr,
                                StringSplitOptions.RemoveEmptyEntries);
            string pav = Eilutes[0];
            int skaic = int.Parse(Eilutes[1]);
            Augalas a = new Augalas(pav, skaic);
            A.DetiAugala(i, a);
        }
    }
}
```

Duomenų įvedimas (2/2)

```
static void Skaityti(string fv, DvimKonteineris S)
{
    using (StreamReader reader = new StreamReader(fv)) {
        string line = reader.ReadLine();
        char[] skyr = { ' ', ';' };
        string[] Eilutes = line.Split(skyr, StringSplitOptions.RemoveEmptyEntries);
        S.N = int.Parse(Eilutes[0]);
        S.M = int.Parse(Eilutes[1]);
        for (int i = 0; i < S.N; i++)
        {
            line = reader.ReadLine();
            Eilutes = line.Split(skyr, StringSplitOptions.RemoveEmptyEntries);
            int k = 0;
            for (int j = 0; j < S.M; j++)
            {
                string pav = Eilutes[k];
                int skaic = int.Parse(Eilutes[k+1]);
                Augalas a = new Augalas(pav, skaic);
                S.DetiAugala(i, j, a);
                k = k + 2;
            }
        }
    }
}
```

Duomenų spausdinimas (1/2)

```
static void Spausdinti(string fv, Konteineris A, string antraste)
{
    const string virsus =
        "-----\n"
        + " Nr.   Augalas           Amžius \n"
        + "-----";
    using (var fr = File.AppendText(fv))
    {
        fr.WriteLine("\n " + antraste);
        fr.WriteLine(virsus);
        for (int i = 0; i < A.N; i++)
        {
            Augalas a = A.ImtiAugala(i);
            fr.WriteLine("{0, 3}   {1, -14}   {2, 2:d}",
                          i + 1, a.Pavadinimas, a.Skaicius);
        }
        fr.WriteLine("-----\n");
    }
}
```

Duomenų spausdinimas (2/2)

```
static void Spausdinti(string fv, DvimKonteineris S, string antraste)
{
    using (var fr = File.AppendText(fv))
    {
        fr.WriteLine("\n          " + antraste);
        for (int i = 0; i < S.N; i++)
        {
            for (int j = 0; j < S.M; j++)
            {
                Augalas a = S.ImtiAugala(i, j);
                fr.Write("{0, -14} {1, 2:d} ", a.Pavadinimas, a.Skaicius);
            }
            fr.WriteLine();
        }
        fr.WriteLine();
    }
}
```

Pagalbiniai metodai (1/2)

```
// Konteinerio A augalų pavadinimus perrašo į konteinerį B,  
// o skaičiams suteikia reikšmes 0  
static void SuteiktiPradinesReiksmes(Konteineris A, Konteineris B)  
{  
    B.N = A.N;  
    for (int i = 0; i < A.N; i++)  
    {  
        string pav = A.ImtiAugala(i).Pavadinimas;  
        Augalas a = new Augalas(pav, 0);  
        B.DetiAugala(i, a);  
    }  
}
```

Pagalbiniai metodai (2/2)

```
// Konteineryje A ieško nurodyto pavadinimo pav objekto  
// Gražina indeksą, jei rado, priešingu atveju -1  
static int AugaloInd(Konteineris A, string pav)  
{  
    for (int i = 0; i < A.N; i++)  
    {  
        Augalas a = A.ImtiAugala(i);  
        if (a.Pavadinimas == pav)  
            return i;  
    }  
    return -1;  
}
```

Pagrindiniai skaičiavimai

```
// skaičiuoja kiek augalų reikės sode atnaujinti
// S - sodo augalų konteineris
// A - augalų kritinio amžiaus konteineris
// B - augalų atnaujinimo konteineris
static void Skaičiuoti(DvimKonteineris S, Konteineris A, Konteineris B)
{
    for (int i = 0; i < S.N; i++)
    {
        for (int j = 0; j < S.M; j++)
        {
            string pav = S.ImtiAugala(i, j).Pavadinimas;
            int ind = AugaloInd(A, pav);
            if (ind > -1 && S.ImtiAugala(i, j) > A.ImtiAugala(ind))
            {
                B.ImtiAugala(ind).DidintiSkaiciu(1);
            }
        }
    }
}
```

Naudojamas
užklotas
operatorius

Konstantos ir Main metodo fragmentas

```
const string CFd1 = @"..\..\Sodas.txt";
const string CFd2 = @"..\..\Augalai.txt";
const string CFr = @"..\..\Rezultatai.txt";

...

Konteineris Augalai = new Konteineris();           // augalų konteineris
Skaityti(CFd2, Augalai);
Spausdinti(CFr, Augalai, "Augalų \"kritinis\" amžius");

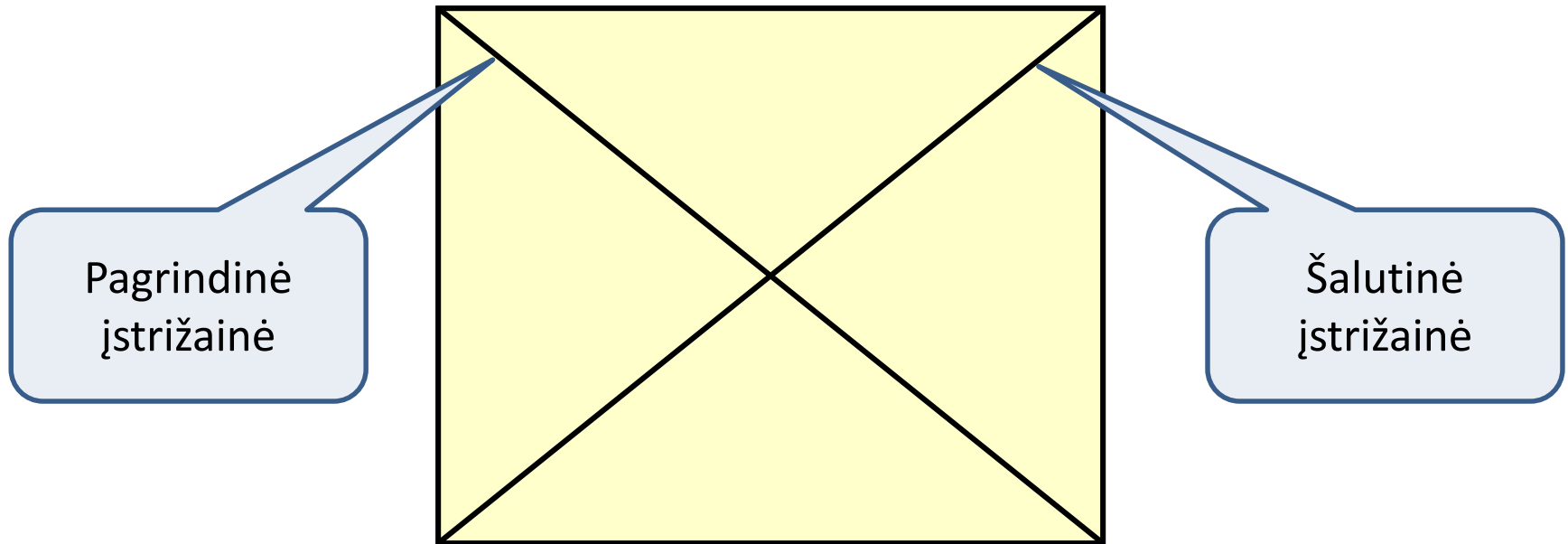
DvimKonteineris Sodas = new DvimKonteineris(); // sodo konteineris
Skaityti(CFd1, Sodas);
Console.WriteLine("Eilių skaičius sode:{0, 4:d}", Sodas.N);
Console.WriteLine("Augalų skaičius eilėje{0, 4:d}", Sodas.M);
Spausdinti(CFr, Sodas, "Sodo augalai (pasodinti eilėmis)");

// Augalų konteineris, kuriuos reikia atnaujinti:
Konteineris AugalaiAtn = new Konteineris();
SuteiktiPradinesReiksmes(Augalai, AugalaiAtn);
Skaičiuoti(Sodas, Augalai, AugalaiAtn);
Spausdinti(CFr, AugalaiAtn, "Augalų atnaujinimas");
```




Kvadratinė matrica

Matricos $A(n,n)$ įstrižainės (1/2)



Matricos $A(n,n)$ įstrižainės (2/2)

$a_{0,0}$	$a_{0,1}$	$a_{0,2}$	$a_{0,3}$	$a_{0,4}$
$a_{1,0}$	$a_{1,1}$	$a_{1,2}$	$a_{1,3}$	$a_{1,4}$
$a_{2,0}$	$a_{2,1}$	$a_{2,2}$	$a_{2,3}$	$a_{2,4}$
$a_{3,0}$	$a_{3,1}$	$a_{3,2}$	$a_{3,3}$	$a_{3,4}$
$a_{4,0}$	$a_{4,1}$	$a_{4,2}$	$a_{4,3}$	$a_{4,4}$

Eilučių (stulpelių) skaičius $n = 5$

$a_{0,0}$	$a_{0,1}$	$a_{0,2}$	$a_{0,3}$
$a_{1,0}$	$a_{1,1}$	$a_{1,2}$	$a_{1,3}$
$a_{2,0}$	$a_{2,1}$	$a_{2,2}$	$a_{2,3}$
$a_{3,0}$	$a_{3,1}$	$a_{3,2}$	$a_{3,3}$

Eilučių (stulpelių) skaičius $n = 4$

Veiksmai su įstrižainių elementų reikšmėmis (1/2)

$a_{0,0}$	$a_{0,1}$	$a_{0,2}$	$a_{0,3}$	$a_{0,4}$
$a_{1,0}$	$a_{1,1}$	$a_{1,2}$	$a_{1,3}$	$a_{1,4}$
$a_{2,0}$	$a_{2,1}$	$a_{2,2}$	$a_{2,3}$	$a_{2,4}$
$a_{3,0}$	$a_{3,1}$	$a_{3,2}$	$a_{3,3}$	$a_{3,4}$
$a_{4,0}$	$a_{4,1}$	$a_{4,2}$	$a_{4,3}$	$a_{4,4}$

Pagrindinės įstrižainės elementai:
 $A[i, i]$

// Pagrindinės įstrižainės reikšmių suma

```
int sumaP = 0;
```

```
for (int i = 0; i < n; i++)  
    sumaP = sumaP + A[i, i];
```

Veiksmai su įstrižainių elementų reikšmėmis (2/2)

$a_{0,0}$	$a_{0,1}$	$a_{0,2}$	$a_{0,3}$	$a_{0,4}$
$a_{1,0}$	$a_{1,1}$	$a_{1,2}$	$a_{1,3}$	$a_{1,4}$
$a_{2,0}$	$a_{2,1}$	$a_{2,2}$	$a_{2,3}$	$a_{2,4}$
$a_{3,0}$	$a_{3,1}$	$a_{3,2}$	$a_{3,3}$	$a_{3,4}$
$a_{4,0}$	$a_{4,1}$	$a_{4,2}$	$a_{4,3}$	$a_{4,4}$

Šalutinės įstrižainės elementai:
 $A[i, n-i-1]$

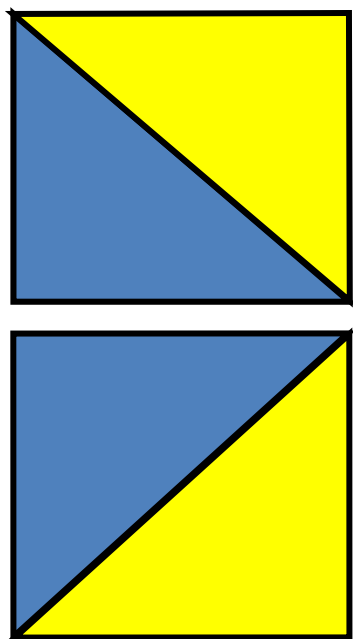
// Šalutinės įstrižainės reikšmių suma

```
int sumas = 0;
```

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

```
    sumas = sumas + A[i, n-i-1];
```

Matricos $A(n,n)$ dalys ribojamos įstrižainėmis (1/2)



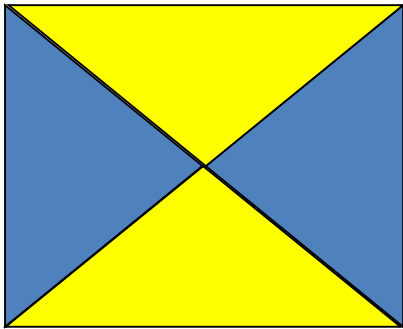
```
int suma = 0;  
for (int i = 0; i < n; i++)  
    for (int j = 0; j <= i; j++)  
        suma = suma + A[i,j];
```

Kurios matricos, kurios dalies
reikšmių suma skaičiuojama?

Savarankiškai: parašykite kitų (trijų) dalių reikšmių sumų skaičiavimų fragmentus.

Matricos $A(n,n)$ dalys ribojamos įstrižainėmis (2/2)

```
int m, suma = 0;
if (n % 2 == 0) m = n / 2;
else m = n / 2 + 1;
for (int i = 0; i < m; i++)
    for (int j = i; j <= n - 1 - i; j++)
        suma = suma + A[i, j];
```



Kurios matricos dalies
reikšmių suma skaičiuojama?

Savarankiškai: parašykite kitų (trijų) matricos dalių reikšmių sumų skaičiavimų fragmentus.

Atstumų matrica – tai kvadratinė matrica, kuri yra simetrinė pagrindinės įstrižainės atžvilgiu

	Kaunas	Vilnius	Klaipėda	Panevėžys	Jurbarkas	Utena
Kaunas	0	101	213	122	86	130
Vilnius	101	0	311	136	188	95
Klaipėda	213	311	0	242	146	325
Panevėžys	122	136	242	0	164	94
Jurbarkas	86	188	146	164	0	217
Utena	130	95	325	94	217	0

- Rasti du tarpusavyje tolimiausius miestus.
- Rasti tris miestus, tarp kurių atstumų suma minimali.



Masyvų masyvas (jagged array)

Masyvų masyvo aprašymas ir atminties išskyrimas (1/3)

Masyvų masyvo nuorodos sukūrimas:

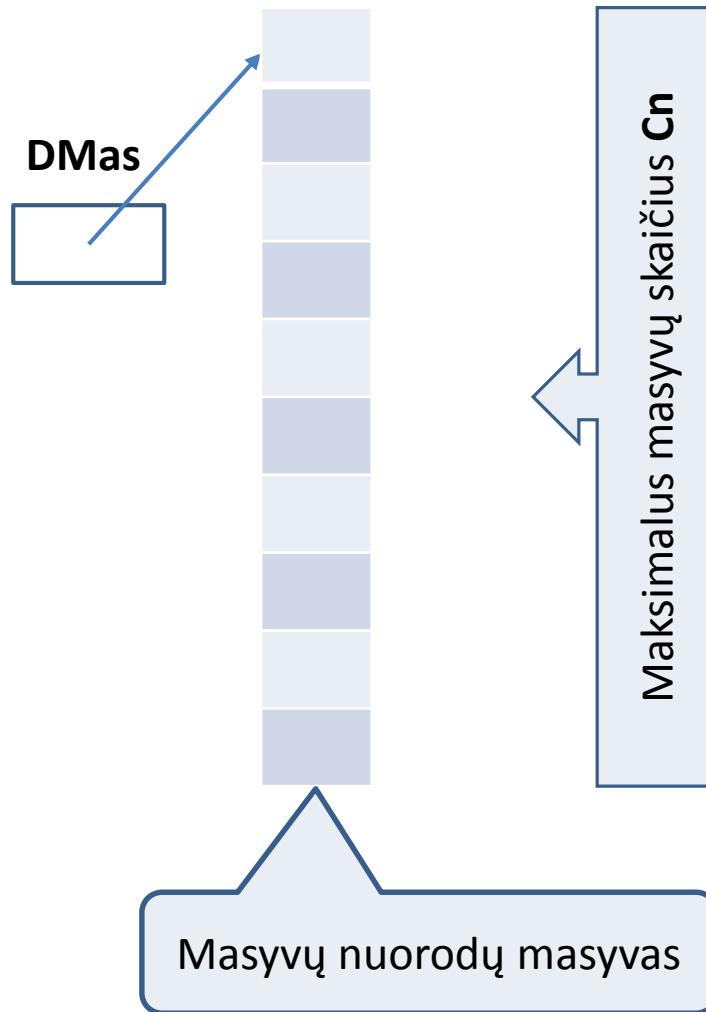
```
int[][] DMas;
```

Masyvų masyvo nuorodos sukūrimas ir atminties išskyrimas:

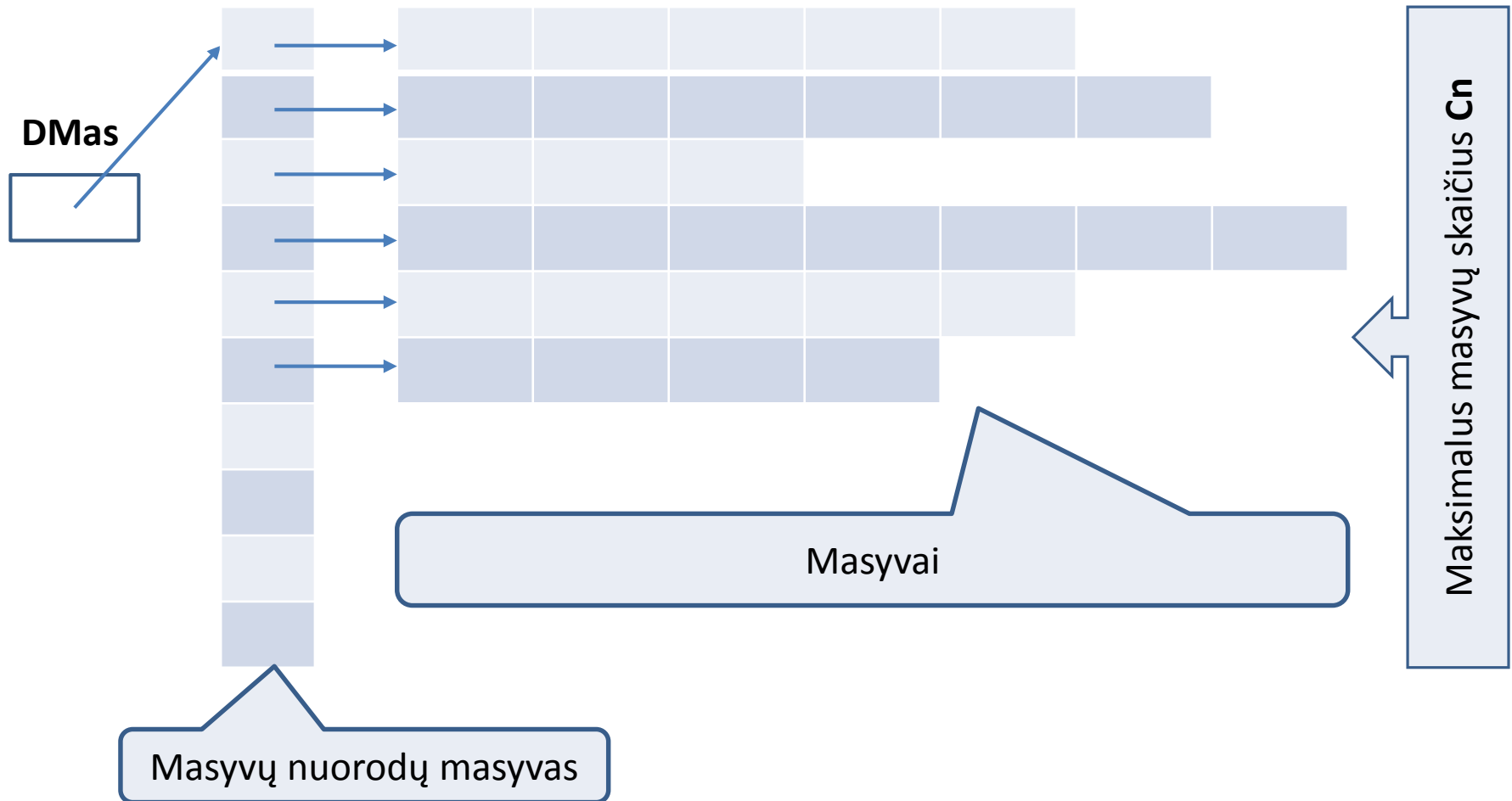
```
int[][] DMas = new int[Cn][];
```

Atmintis bus išskirta
programos darbo metu

Masyvų masyvo aprašymas ir atminties išskyrimas (2/3)



Masyvų masyvo aprašymas ir atminties išskyrimas (3/3)



Konteinerinė klasė su masyvų masyvu (1/2)

```
class MasMasyvas
```

```
{
    public const int Cn = 50;           // maksimalus masyvų skaičius
    private int[][] DMas;              // sveikų skaičių masyvų masyvas
    public int N { get; set; }         // savybė N: masyvų skaičius
    public MasMasyvas()
    {
        DMas = new int[Cn][];
        N = 0;
    }
    public void DetiMasyva(int i, int[] Mas)
    {
        DMas[i] = Mas;
    }
    public int[] ImtiMasyva(int i)
    {
        return DMas[i];
    }
    ... // tęsinys kitoje skaidrėje
}
```

Atminties išskyrimas
masyvų masyvui

Konteinerinė klasė su masyvų masyvu (2/2)

```
class MasMasyvas
```

```
{  
    ...  
    public void Deti(int i, int j, int sk)  
    {  
        DMas[i][j] = sk;  
    }  
    public int Imti(int i, int j)  
    {  
        return DMas[i][j];  
    }  
    public int ImtiKiekMasyve(int i)  
    {  
        return DMas[i].Length;  
        // arba return DMas[i].GetLength(0);  
    }  
}
```

Duomenų failo pavyzdys

4						
5	2	1	6	3	2	-5
7	0	5	2			
-8	6	4	-7	0	1	
3	-5	-8	9	7		

Rekomenduojama:

- pirmoje eilutėje rašyti tik eilučių skaičių;
- reikšmes eilutėse surašyti tvarkingai, lygiuojant stulpelius.

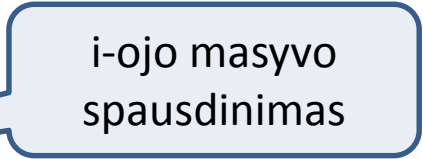
Masyvų masyvo reikšmių skaitymas

```
static void skaitytiD(string fv, MasMasyvas A)
{
    using (StreamReader reader = new StreamReader(fv)) {
        char[] skyr = { ' ', ' ' };
        string[] skaiciai; // vienos eilutės skaičiai
        int[] Mas; // vienos eilutės skaičių masyvas
        string line = reader.ReadLine();
        A.N = int.Parse(line);
        for (int i = 0; i < A.N; i++)
        {
            line = reader.ReadLine();
            skaiciai = line.Split(skyr,
                                StringSplitOptions.RemoveEmptyEntries);
            Mas = new int[skaiciai.Length];
            for (int j = 0; j < skaiciai.Length; j++)
            {
                Mas[j] = int.Parse(skaiciai[j]);
            }
            A.DetiMasyva(i, Mas);
        }
    }
}
```

i-ojo masyvo
formavimas

Masyvų masyvo reikšmių spausdinimas

```
static void spausdintiD(string fv, MasMasyvas A)
{
    using (var fr = File.AppendText(fv))
    {
        for (int i = 0; i < A.N; i++)
        {
            for (int j = 0; j < A.ImtiKiekMasyve(i); j++)
            {
                fr.Write("{0, 4:d}", A.Imti(i, j));
            }
            fr.WriteLine();
        }
    }
}
```



i-ojo masyvo
spausdinimas

Masyvų vidurkių skaičiavimas

```
static void vidurkiaid(MasMasyvas A, double[] vid)
{
    int suma;
    for (int i = 0; i < A.N; i++)
    {
        suma = 0;
        for (int j = 0; j < A.ImtiKiekMasyve(i); j++)
        {
            suma += A.Imti(i, j);
        }
        if (A.ImtiKiekMasyve(i) != 0)
            vid[i] = (double)suma / A.ImtiKiekMasyve(i);
        else
            vid[i] = 0.0;
    }
}
```

i-ojo masyvo
sumos
skaičiavimas

Main metodas

```
static void Main(string[] args)
{
    using (var fr = File.CreateText(CFr))
    { }
    // Konteineris su masyvų masyvu
    MasMasyvas MasM = new MasMasyvas();
    SkaitytiD(CFd, MasM);
    SpausdintiD(CFr, MasM);
    double[] vid = new double[MasM.N]; // masyvų vidurkiai
    Vidurkiaid(MasM, vid); //
    using (var fr = File.AppendText(CFr))
    {
        fr.WriteLine("{0}", "Masyvų vidurkiai");
        foreach (double v in vid)
            fr.WriteLine("{0, 7:f2} ", v);
        fr.WriteLine();
    }
}
```

Masyvų vidurkiai

2,00

3,50

-0,67

1,20

Šioje temoje susipažinsite su:

1. Dvimačiu masyvu (matrica) ir jo (jos) aprašymu.
2. Veiksmais eilutėse ir stulpeliuose.
3. Dvimačiu objektų konteineriu.
4. Kvadratine matrica.
5. Masyvų masyvu.



Klausimai?