

# T01. Algoritmai ir struktūros

2 ak. val.

# Temos klausimai

1. Rikiavimas burbuliuko metodu.
2. Dviejų tvarkingų rinkinių suliejimas.
3. Stuktūros (`struct`).
4. Struktūros: `DateTime`, `TimeSpan`.
5. Išvardijimo tipas (`enum`).



# *Porinių sukeitimų (burbuliuko) rikiavimo metodas*

# Minmax metodo trūkumai

Rudens semestre buvo nagrinėjamas masyvo rikiavimo metodas išrenkant reikiamus narius (minmax). Jo darbo imlumo įvertinimas yra  $n(n-1)/2$  narių palyginimų, kur  $n$  – rikiuojamo masyvo narių skaičius.

Kas atsitiks, jei šiuo metodu rikiuoti jau surikiuotą masyvą? Minmax metodas “nejaučia”, kad masyvas jau surikiuotas, ir atliks visus numatytus veiksmus.

Minmax metodas neefektyvus ir tuomet, kai masyvas dalinai surikiuotas arba kai jis išsirikiuoja anksčiau, nei po  $n-1$  peržiūros.

# Burbuliuko metodas (1/3)

Šio trūkumo leidžia išvengti **porinių sukeitimų** metodas, dar vadinamas **burbuliuko** metodu.

Jo esmė – lyginti tarpusavyje *gretimus* narius ir, jei jie neatitinka reikiamai tvarkai, sukeisti juos vietomis.

Nuosekliai peržiūrėjus poromis visus *gretimus* masyvo narius, vienas jų atsiranda savo vietoje, o pats masyvas tampa „tvarkingesniu“.

Peržiūrėjus masyvą  $n-1$  kartą, jo nariai bus išrikiuoti.

Šis nepagerintas metodas neduoda išlošimo, lyginant su **minmax**, net pablogina padėtį dėl didesnio sukeitimų skaičiaus vienos peržiūros metu.

# Burbuliuko metodas (2/3)

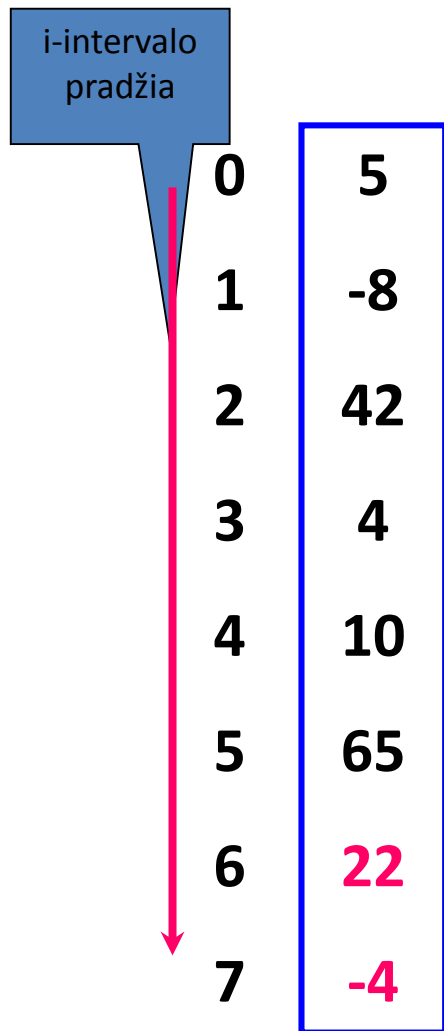
Burbuliuko metodas leidžia jį pagerinti – jei eilinės peržiūros metu masyvo nariai sukeisti nebuvo, reiškia masyvas surikiuotas ir darbą galima nutraukti.

Jei burbuliuko metodu rikiuoti jau surikiuotą masyvą, prireiks tik vienos peržiūros ( $n-1$  palyginimo).

Blogiausiu atveju reikės  $n(n-1)/2$  palyginimo. Bet dėl didesnio sukeitimų skaičiaus eilinės peržiūros metu burbuliuko metodas bus lėtesnis, nei minmax metodas.

Realus darbo imlumo įvertinimas yra intervale  $[n-1, n(n-1)/2]$ .

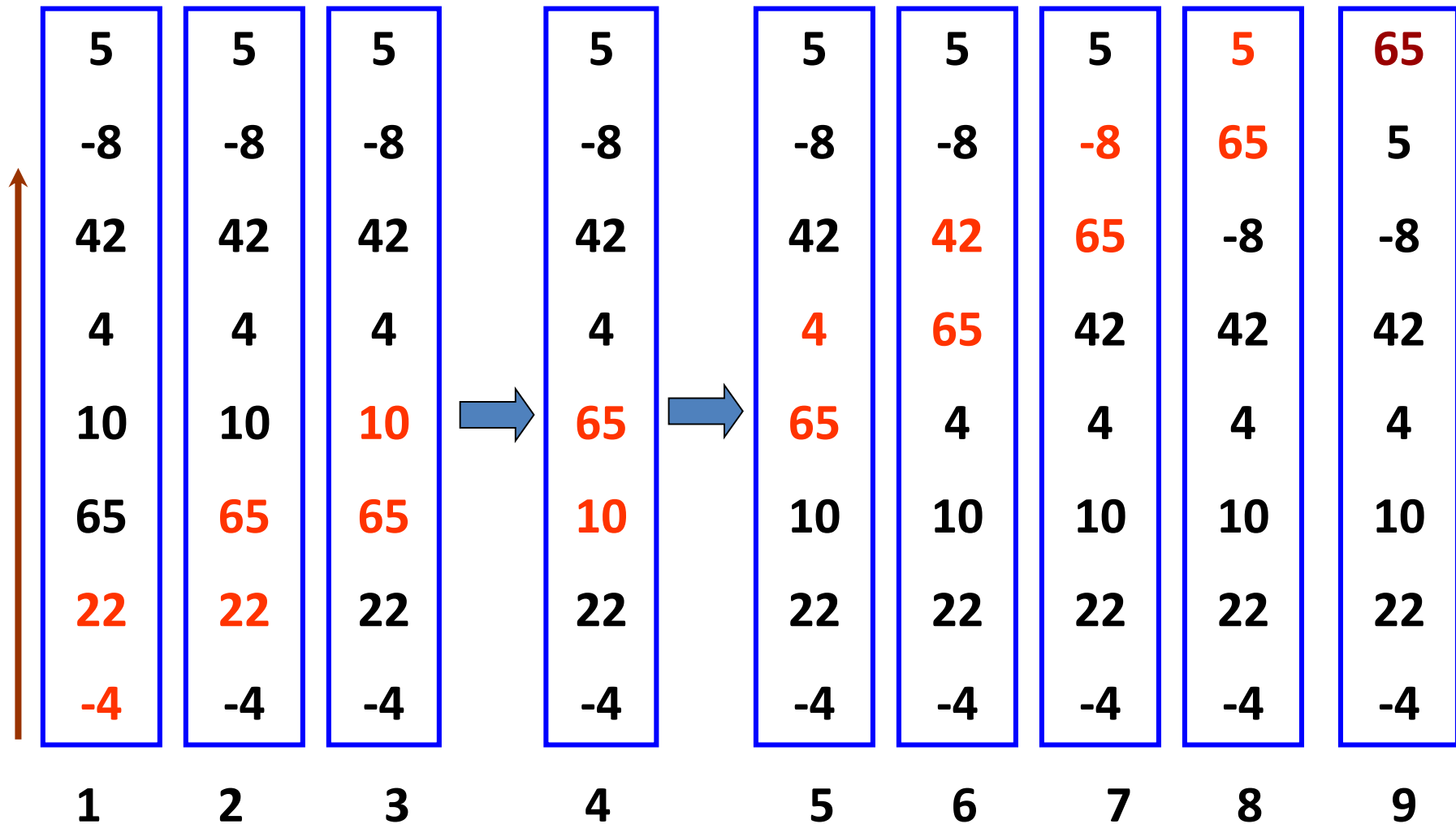
# Burbuliuko metodas (3/3)



Algoritmas:

- reikšmės lyginamos poromis nuo masyvo pabaigos
- jeigu reikia, sukeičiamos vietomis
- masyvas peržiūrimas  $n-1$  kartą
- jei eilinės peržiūros metu sukeitimų nebuvo, darbas nutraukiamas anksčiau

# Pavyzdys (1/7)




5	5	5	5	5	5	5	5	65
-8	-8	-8	-8	-8	-8	-8	65	5
42	42	42	42	42	42	65	-8	-8
4	4	4	4	4	65	42	42	42
10	10	10	65	65	4	4	4	4
65	65	65	10	10	10	10	10	10
22	22	22	22	22	22	22	22	22
-4	-4	-4	-4	-4	-4	-4	-4	-4
1	2	3	4	5	6	7	8	9

Pirmoji peržiūra



# Pavyzdys (2/7)



65	65	65	65	65	65	65	65
5	5	5	5	5	5	42	42
-8	-8	-8	-8	-8	42	5	5
42	42	42	42	42	-8	-8	-8
4	4	4	22	22	22	22	22
10	10	22	4	4	4	4	4
22	22	10	10	10	10	10	10
-4	-4	-4	-4	-4	-4	-4	-4
1	2	3	4	5	6	7	rezultatas

Antroji peržiūra

# Pavyzdys (3/7)



65	65	65	65	65	65		65
42	42	42	42	42	42		42
5	5	5	5	5	22		22
-8	-8	-8	-8	22	5		5
22	22	22	22	-8	-8		-8
4	4	10	10	10	10		10
10	10	4	4	4	4		4
-4	-4	-4	-4	-4	-4		-4
1	2	3	4	5	6	7	rezultatas

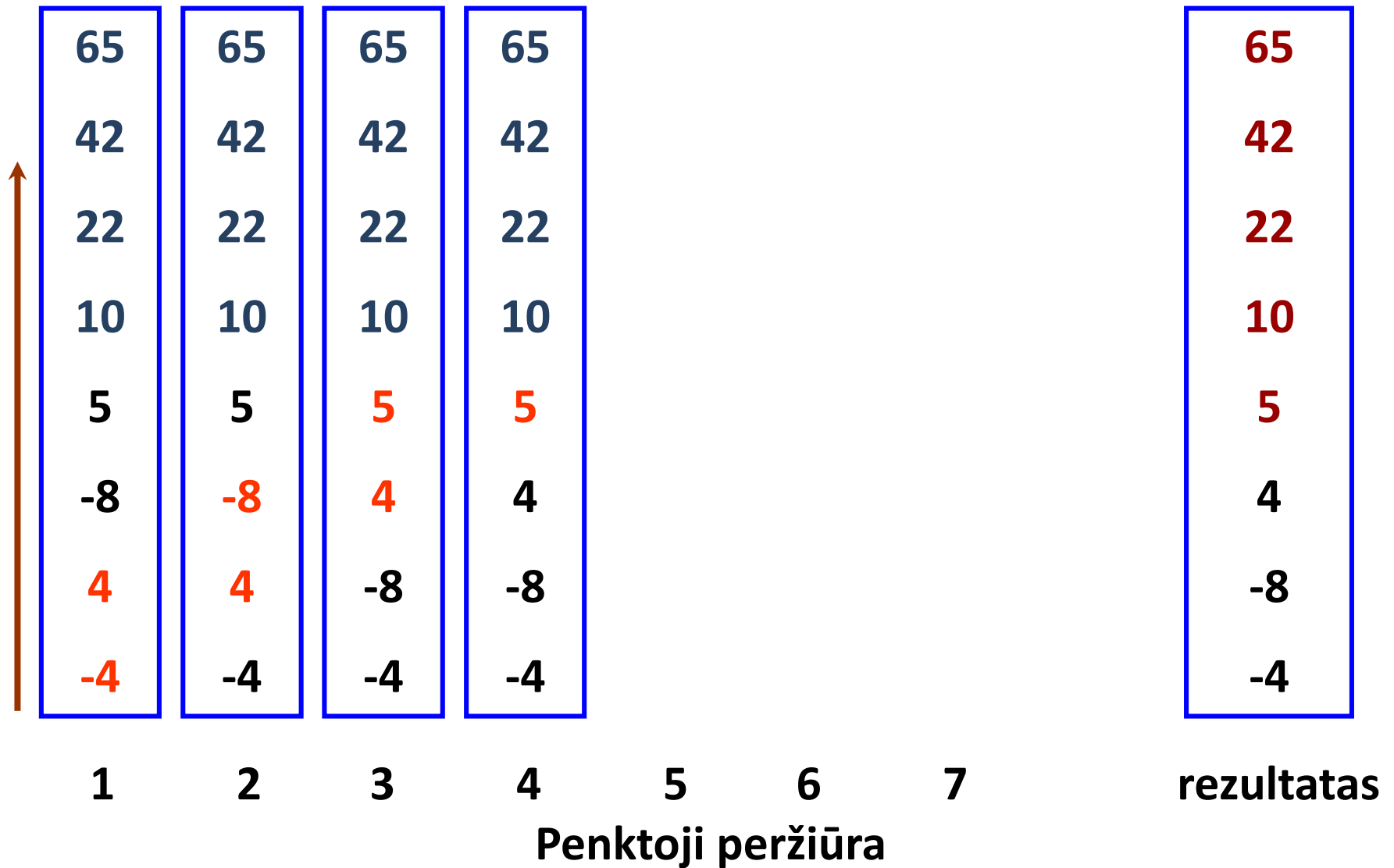
Trečioji peržiūra

# Pavyzdys (4/7)

65	65	65	65	65			65
42	42	42	42	42			42
22	22	22	22	22			22
5	5	5	5	10			10
-8	-8	-8	10	5			5
10	10	10	-8	-8			-8
4	4	4	4	4			4
-4	-4	-4	-4	-4			-4
1	2	3	4	5	6	7	rezultatas

Ketvirtoji peržiūra

# Pavyzdys (5/7)

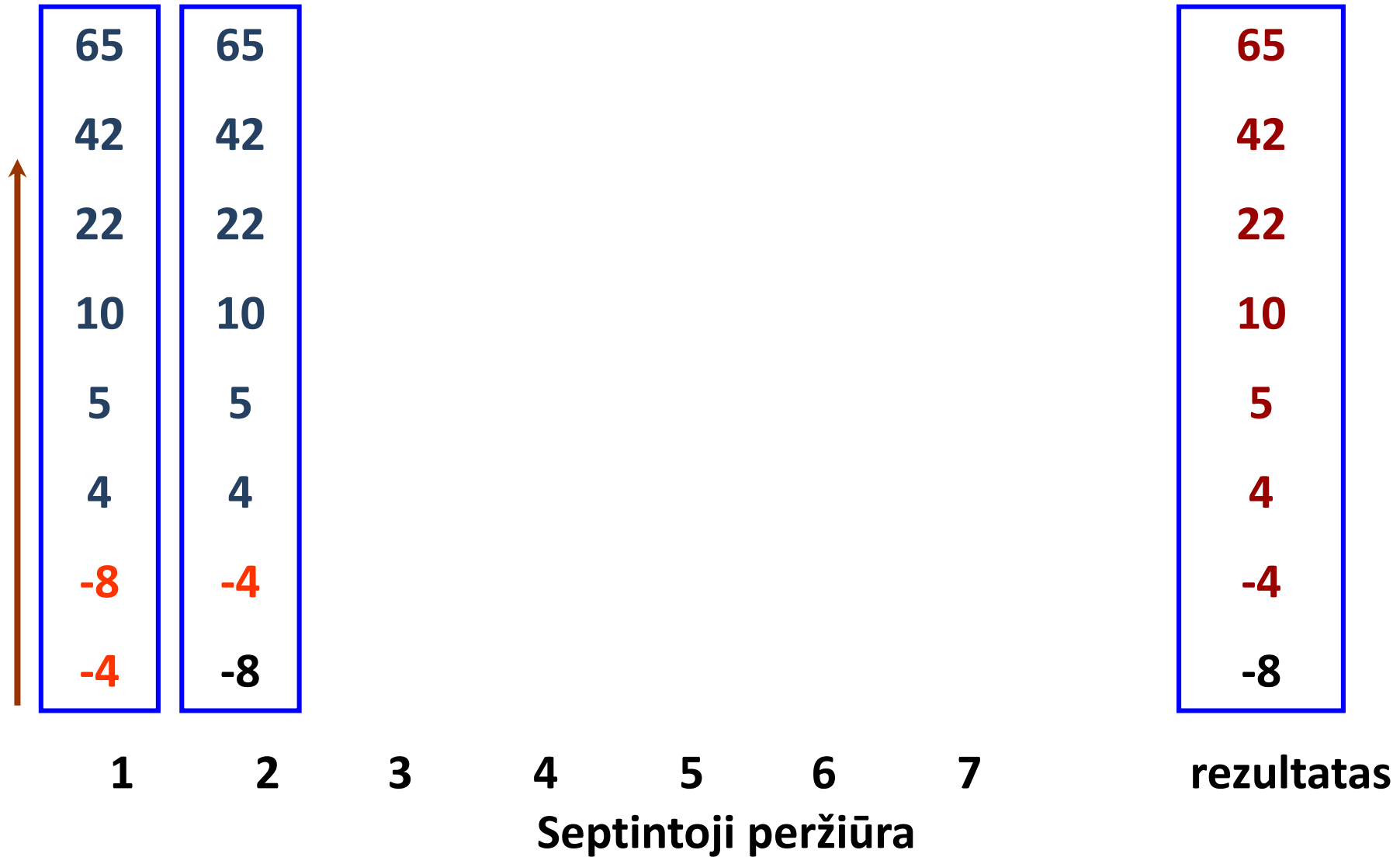


# Pavyzdys (6/7)

	65	65	65					65
	42	42	42					42
	22	22	22					22
	10	10	10					10
	5	5	5					5
	4	4	4					4
	-8	-4	-4					-4
	-4	-8	-8					-8
	1	2	3	4	5	6	7	rezultatas

Šeštoji peržiūra

# Pavyzdys (7/7)



# Burbuliuko metodas

// surikiuoja masyvą Mas(kiek) skaičių mažėjimo tvarka

```
static void Burbuliukas(int[] Mas, int kiek)
{
    int i = 0;           // intervalo pradžios žymeklis
    bool bk = true;      // požymis, ar buvo sukeitimų
    while (bk)
    {
        bk = false;
        for (int j = kiek - 1; j > i; j--)
            if (Mas[j] > Mas[j-1])
            {
                bk = true;
                int c = Mas[j];
                Mas[j] = Mas[j-1];
                Mas[j-1] = c;
            }
        i++;
    }
}
```

# Burbuliuko metodas objektų masyvui

// surikiuoja objektų masyvą Studentai(kiek) pagal pažymius  
// mažėjančia tvarka

```
static void Burbuliukas(Studentas[] studentai, int kiek)
{
    int i = 0;           // intervalo pradžios žymeklis
    bool bk = true;      // požymis, ar buvo sukeitimų
    while (bk)
    {
        bk = false;
        for (int j = kiek - 1; j > i; j--)
            if (Studentai[j] > Studentai[j-1])
            {
                bk = true;
                Studentas stud = Studentai[j];
                Studentai[j] = Studentai[j - 1];
                Studentai[j - 1] = stud;
            }
        i++;
    }
}
```

Naudojamas užklotas  
operatorius >



# Užklotas operatorius >

Burbuliuko rikiavimo metode naudojamas užklotas operatorius >:

```
// Užklotas operatorius >
public static bool operator >(Studentas stud1,
                             Studentas stud2)
{
    return stud1.pazym > stud2.pazym;
}

// Užklotas operatorius <
public static bool operator <(Studentas stud1,
                             Studentas stud2)
{
    return stud1.pazym < stud2.pazym;
}
```



## *Tvarkingų rinkinių suliejimas*

# Dviejų masyvų suliejimas

Pavyzdžiui, masyvų  $A(n)$  ir  $B(m)$  reikšmes reikia surašyti į masyvą  $C(k)$ .

Galimi atvejai:

- kai duomenys netvarkingi (**buvo nagrinėta anksčiau**).
- kai duomenys tvarkingi, t.y. surikiuoti.

Suliejant tvarkingus rinkinius naujajame rinkinyje turi būti *išlaikoma esama tvarka*.

Tokiu atveju *nereikia tvarkyti naujojo rinkinio, jo rikiuoti*.

# Suliejimas, kai duomenys tvarkingi

A	C	B
-1	-10	-10
0		-7
5		-3
7		3
9		10
		12
		13

Lyginame dvi reikšmes:

$$a_0 < b_0$$

Mažesnę įrašome į masyvą C :

$$b_0 \Rightarrow C$$

Toliau lyginimui imame to masyvo  
tolimesnę reikšmę:  $b_1$

# Suliejimas, kai duomenys tvarkingi

A	C	B
-1	-10	-10
0	-7	-7
5		-3
7		3
9		10
		12
		13

Lyginame dvi reikšmes:

$$a_0 < b_1$$

Mažesnę įrašome į masyvą C :

$$b_1 \Rightarrow C$$

Toliau lyginimui imame to masyvo  
tolimesnę reikšmę:  $b_2$

# Suliejimas, kai duomenys tvarkingi

A	C	B
-1	-10	-10
0	-7	-7
5	-3	-3
7		3
9		10
		12
		13

Lyginame dvi reikšmes:

$$a_0 < b_2$$

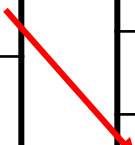
Mažesnę įrašome į masyvą C :

$$b_2 \Rightarrow C$$

Toliau lyginimui imame to masyvo  
tolimesnę reikšmę:  $b_3$

# Suliejimas, kai duomenys tvarkingi

A	C	B
-1	-10	-10
0	-7	-7
5	-3	-3
7	-1	-3
9		3
		10
		12
		13



Lyginame dvi reikšmes:

$$a_0 < b_3$$

Mažesnę įrašome į masyvą C :

$$a_0 \Rightarrow C$$

Toliau lyginimui imame to masyvo  
tolimesnę reikšmę:  $a_1$

# Suliejimas, kai duomenys tvarkingi

A	C	B
-1	-10	-10
0	-7	-7
5	-3	-3
	-1	-3
	0	3
		10
7		12
9		13

Lyginame dvi reikšmes:

$$a_1 < b_3$$

Mažesnę įrašome į masyvą C :

$$a_1 \Rightarrow C$$

Toliau lyginimui imame to masyvo  
tolimesnę reikšmę:  $a_2$



# Suliejimas, kai duomenys tvarkingi

A	C	B
-1	-10	-10
0	-7	-7
5	-3	-3
	-1	-3
	0	3
	3	10
7		12
9		13

Lyginame dvi reikšmes:

$$a_2 < b_3$$

Mažesnę įrašome į masyvą C :

$$b_3 \Rightarrow C$$

Toliau lyginimui imame to masyvo  
tolimesnę reikšmę:  $b_4$

# Suliejimas, kai duomenys tvarkingi

A	C	B
-1	-10	-10
0	-7	-7
5	-3	-3
7	-1	-3
9	0	3
	3	10
	5	12
		13

Lyginame dvi reikšmes:

$$a_2 < b_4$$

Mažesnę įrašome į masyvą C :

$$a_2 \Rightarrow C$$

Toliau lyginimui imame to masyvo tolimesnę reikšmę:  $a_3$

# Suliejimas, kai duomenys tvarkingi

A	C	B
-1	-10	-10
0	-7	-7
5	-3	-3
7	-1	-3
9	0	3
	3	10
	5	12
	7	13

Lyginame dvi reikšmes:

$$a_3 < b_4$$

Mažesnę įrašome į masyvą C :

$$a_3 \Rightarrow C$$

Toliau lyginimui imame to masyvo tolimesnę reikšmę:  $a_4$

# Suliejimas, kai duomenys tvarkingi

A	C	B
-1	-10	-10
0	-7	-7
5	-3	-3
7	-1	-3
9	0	3
	3	10
	5	12
	7	13
	9	

Lyginame dvi reikšmes:

$$a_4 < b_4$$

Mažesnę įrašome į masyvą C :

$$a_4 \Rightarrow C$$

Masyve A neliko reikšmių

# Suliejimas, kai duomenys tvarkingi

A	C	B
-1	-10	-10
0	-7	-7
5	-3	-3
7	-1	-1
9	0	3
	3	10
	5	12
	7	13
	9	
	10	
	12	
	13	

Masyvo B likusias reikšmes  
perrašome į masyvą C

# Dviejų tvarkingų masyvų suliejimas

```
static void SujungtiTvarkingus(int[] Mas1, int kiek1, int[] Mas2, int kiek2,  
                                int[] Mas3, out int kiek3)  
{  
    kiek3 = 0;  
    int i = 0, j = 0;  
    while ((i < kiek1) && (j < kiek2))  
        if (Mas1[i] < Mas2[j])  
        {  
            Mas3[kiek3] = Mas1[i]; kiek3 = kiek3 + 1; i = i + 1;  
        }  
        else  
        {  
            Mas3[kiek3] = Mas2[j]; kiek3 = kiek3 + 1; j = j + 1;  
        }  
    while (i < kiek1)  
    {  
        Mas3[kiek3] = Mas1[i]; kiek3 = kiek3 + 1; i = i + 1;  
    }  
    while (j < kiek2)  
    {  
        Mas3[kiek3] = Mas2[j]; kiek3 = kiek3 + 1; j = j + 1;  
    }  
}
```



# *Struktūros*

# Struktūros tipas

- Struktūros tipas yra reikšmės tipas (value type).
- Struktūros tipas atitinka įrašo tipui.
- Struktūros tipas tinka nesudėtingoms susietoms duomenų grupėms saugoti, pvz.: taško koordinatėms ar knygos aprašui saugoti ir pan.
- Struktūros tipui aprašyti naudojamas raktinis žodis **struct**.



# Struktūros pavyzdys

// Struktūra skirta taško koordinatėms ir taško spalvai aprašyti

public struct TaskasStruct

{

public int x; // taško koordinatė x

public int y; // taško koordinatė y

public char s; // taško spalva

}

# Struktūros pavyzdys

```
TaskasStruct TStr1; // 1-as taškas
TStr1.x = 4;
TStr1.y = 3;
TStr1.s = 'R';
Console.WriteLine("TStr1: x = {0} y = {1} spalva = {2}",
                  TStr1.x, TStr1.y, TStr1.s);
TaskasStruct TStr2; // 2-as taškas
TStr2.x = 0;
TStr2.y = 0;
TStr2.s = TStr1.s;
Console.WriteLine("TStr2: x = {0} y = {1} spalva = {2}",
                  TStr2.x, TStr2.y, TStr2.s);
```

Rezultatas ekrane:

```
TStr1: x = 4 y = 3 spalva = R
TStr2: x = 0 y = 0 spalva = R
```

# Struktūros tipas

Struktūra gali turėti:

- Kintamuosius (laukus)
- Savybes (properties)
- Konstantas
- Konstruktorius
- Metodus

# Struktūros pavyzdys

```
// Struktūra skirta taško koordinatėms ir spalvai aprašyti
public struct TaskasStruct
{
    public int x;          // taško koordinatė x
    public int y;          // taško koordinatė y
    public char s;         // taško spalva (pirmoji raidė)
    // Konstruktorius su parametrais
    public TaskasStruct(int x, int y, char s)
    {
        this.x = x;
        this.y = y;
        this.s = s;
    }
    // Užklotas metodas
    public override string ToString()
    {
        return "(" + x + ", " + y + "), " + s;
    }
}
```

# Struktūros pavyzdys

```
TaskasStruct TStr3 = new TaskasStruct(6, 8, 'G');  
TaskasStruct TStr4 = TStr3;  
Console.WriteLine("TStr: {0}", TStr3.ToString());  
Console.WriteLine("TStr: {0}", TStr4.ToString());
```

Rezultatas ekrane:

```
TStr3: (6, 8), G  
TStr4: (6, 8), G
```

# Struktūros pavyzdys

```
// Apskaičiuoja ir grąžina atstumą tarp dviejų taškų TStr1 ir TStr2
static double Atstumas(TaskasStruct TStr1,
                      TaskasStruct TStr2)
{
    return Math.Sqrt(Math.Pow(TStr1.x - TStr2.x, 2.0)
                    + Math.Pow(TStr1.y - TStr2.y, 2.0));
}
...
Console.WriteLine("Atstumas tarp taškų: {0, 7:f2}",
                  Atstumas(TStr1, TStr2));
```

Rezultatas ekrane:

Atstumas tarp taškų:	5,00
----------------------	------

# Struktūra ir klasė

Struktūros aprašas panašus į klasės aprašą, tačiau struktūra turi daugiau apribojimų:

- Struktūroje negalima aprašyti numatytojo konstruktoriaus (konstruktoriaus be parametrų).
- Struktūra yra reikšmės tipo, klasė nuorodos tipo.
- Struktūros kintamiesiems (laukams) reikšmės gali būti suteiktos nenaudojant **new** operatoriaus.
- Struktūra gali turėti konstruktorių su parametrais.
- Struktūra negali paveldėti kitos struktūros ar klasės.

# Struktūra **DateTime** (1/8)

Struktūra skirta datai ir laikui saugoti bei jais operuoti.

Struktūra turi:

- 11 užklotų konstruktorių.
- 16 savybių (properties).
- 59 metodus.
- 9 užklotus operatorius.



# Struktūra `DateTime` (2/8)

Pavyzdžiui:

```
DateTime DT1 = new DateTime(2016, 1, 7);  
Console.WriteLine("Data ir laikas: {0}",  
                  DT1.ToString("d"));  
Console.WriteLine("Data ir laikas: {0}", DT1);  
  
DateTime DT2 = new DateTime(2016, 1, 6, 10, 5, 55);  
Console.WriteLine("Data ir laikas: {0}", DT2);  
...
```

Rezultatas ekrane:

```
Data ir laikas: 2016-01-07  
Data ir laikas: 2016-01-07 00:00:00  
Data ir laikas: 2016-01-06 10:05:55
```

# Struktūra DateTime (3/8)

```
DateTime Dabar = DateTime.Now;    // data ir laikas programos darbo metu
Console.WriteLine("Šios dienos data ir laikas: {0}", Dabar.ToString());
```

```
int metai = Dabar.Year;
int menuo = Dabar.Month;
int diena = Dabar.Day;
int valandos = Dabar.Hour;
int minutes = Dabar.Minute;
int sekundes = Dabar.Second;
int milisek = Dabar.Millisecond;
```

```
Console.WriteLine("Metai:           {0}", metai);
Console.WriteLine("Mėnesis:        {0}", menuo);
Console.WriteLine("Diena:          {0}", diena);
Console.WriteLine("Valandos:       {0}", valandos);
Console.WriteLine("Minutės:        {0}", minutes);
Console.WriteLine("Sekundės:       {0}", sekundes);
Console.WriteLine("Milisekundės: {0}", milisek);
```

# Struktūra **DateTime** (4/8)

Rezultatas ekrane:

```
Šios dienos data ir laikas: 2016-01-06 15:17:23
Metai:                2016
Mėnesis:              1
Diena:                6
Valandos:            15
Minutės:             17
Sekundės:            23
Milisekundės:       131
```

# Struktūra `DateTime` (5/8)

```
string data = "2016-01-06";  
DateTime DT3 = DateTime.Parse(data);    // datos sudarymas iš eilutės  
Console.WriteLine("Data:      {0}", DT3.ToString("d"));  
Console.WriteLine("Metai:     {0}", DT3.Year);  
Console.WriteLine("Mėnesis:   {0}", DT3.Month);  
Console.WriteLine("Diena:     {0}", DT3.Day);
```

Rezultatas ekrane:

```
Data:      2016-01-06  
Metai:     2016  
Mėnesis:   1  
Diena:     6
```

# Struktūra DateTime (6/8)

```
int ppoz = DateTime.Compare(DT1, DT2);
string rezult;
if (ppoz < 0)
    rezult = "yra ankstesnė nei";
else if (ppoz == 0)
    rezult = "yra tokia pati, kaip ir";
else
    rezult = "yra vėlesnė nei";
Console.WriteLine("{0} {1} {2}", DT1.ToString("d"), rezult, DT2.ToString("d"));
```

Rezultatas ekrane:

```
2016-01-07 yra vėlesnė nei 2016-01-06
```

# Struktūra DateTime (7/8)

```
// sukurti datą laiką 2017-02-14 16:05:07.123
DateTime dt = new DateTime(2017, 2, 14, 16, 5, 7);
// skirtingų formatų naudojimas
Console.WriteLine(String.Format("{0:y yy yyy yyyy}", dt)); // "17 17 2017 2017"
Console.WriteLine(String.Format("{0:M MM MMM MMMM}", dt)); // "2 02 vas vasaris"
Console.WriteLine(String.Format("{0:d dd ddd dddd}", dt)); // "14 14 An antradienis"
Console.WriteLine(String.Format("{0:h hh H HH}", dt)); // "4 04 16 16"
Console.WriteLine(String.Format("{0:m mm}", dt)); // "5 05"
Console.WriteLine(String.Format("{0:s ss}", dt)); // "7 07"
```

Rezultatas ekrane:

```
17 17 2017 2017
2 02 vas vasaris
14 14 An antradienis
4 04 16 16
5 05
7 07
```

# Struktūra DateTime (8/8)

```
DateTime dt = new DateTime(2017, 2, 14, 16, 5, 7);
// skirtingų formatų naudojimas
Console.WriteLine(String.Format("{0:t}", dt)); // ShortTime
Console.WriteLine(String.Format("{0:d}", dt)); // ShortDate
Console.WriteLine(String.Format("{0:T}", dt)); // LongTime
Console.WriteLine(String.Format("{0:D}", dt)); // LongDate
Console.WriteLine(String.Format("{0:f}", dt)); // LongDate+ShortTime
Console.WriteLine(String.Format("{0:F}", dt)); // FullDateTime
Console.WriteLine(String.Format("{0:g}", dt)); // ShortDate+ShortTime
Console.WriteLine(String.Format("{0:G}", dt)); // ShortDate+LongTime
Console.WriteLine(String.Format("{0:m}", dt)); // MonthDay
Console.WriteLine(String.Format("{0:y}", dt)); // YearMonth
```

Rezultatas ekrane:

```
16:05
2017-02-14
16:05:07
2017 m. vasario 14 d.
2017 m. vasario 14 d. 16:05
2017 m. vasario 14 d. 16:05:07
2017-02-14 16:05
2017-02-14 16:05:07
vasario 14 d.
2017 m. vasaris
```

# Pavyzdys su DateTime (1/8)

Faile "StudIF.txt" duota studijų centro informacija apie **įstočiusius į fakultetą studentus:**

*studento pavardė ir vardas, gimimo data, kokią mokyklą (gimnaziją) baigė.*

Spausdinti studentų, gimusių nurodytais (klaviatūra) metais, sąrašą.



# Pavyzdys su DateTime (1/8)

Faile "StudIF.txt" duota studijų centro informacija apie **įstojusius į fakultetą studentus:**

*studento pavardė ir vardas, gimimo data, kokią mokyklą (gimnaziją) baigė.*

Spausdinti studentų, gimusių nurodytais (klaviatūra) metais, sąrašą.

# Pavyzdys su DateTime (2/8)

Anksčiau čia buvo  
studento asmens kodas.

StudIF.txt

```
Jonaitis Jonas;1996-01-01;Gimnazija 1;  
Petraitis Petras;1996-02-02;Mokykla 1;  
Antanaitis Antanas;1995-03-03;Gimnazija 2;  
Giedraitis Giedrius;1997-04-04;Gimnazija 1;  
Onaitytė Ona;1996-05-05;Mokykla 1;  
Juozaitis Juozas;1995-06-06;Mokykla 3;  
Ramunaitė Ramunė;1996-06-06;Gimnazija 2;
```

# Pavyzdys su DateTime (3/8)

Struktūros **DateTime**  
tipo savybė.

```
class StudentasFF
{
    public string PavVrd { get; set; } // savybė: studento pavardė ir vardas
    public DateTime GimData { get; set; } // savybė: studento gimimo data
    public string Mokykla { get; set; } // savybė: kurią mokyklą baigė
    public StudentasFF(string pavv, DateTime gd, string mok)
    {
        PavVrd = pavv;
        GimData = gd;
        Mokykla = mok;
    }
}
```

# Pavyzdys su DateTime (4/8)

```
static void SkaitytiStudF(string fv, StudentasFF[] Studentai, out int kiek)
{
    using (StreamReader srautas = new StreamReader(fv))
    {
        string eilute; // visa duomenų failo eilutė
        int i = 0;
        while ((eilute = srautas.ReadLine()) != null && (i < Cn))
        {
            string[] eilDalis = eilute.Split(';');
            string pavVrd = eilDalis[0];
            DateTime gd = DateTime.Parse(eilDalis[1]);
            string mokykl = eilDalis[2];
            Studentai[i++] = new StudentasFF(pavVrd, gd, mokykl);
        }
        kiek = i;
    }
}
```

# Pavyzdys su DateTime (5/8)

```
static void spausdintiStudF(string fv, StudentasFF[] studentai, int kiek,
                           string antraste)
{
    const string virusus =
        "-----\r\n"
        + " Nr.   Pavardė ir vardas      Gimimo data  Mokykla      \r\n"
        + "-----";
    using (var fr = File.AppendText(fv))
    {
        fr.WriteLine("\n" + antraste);
        fr.WriteLine(virusus);
        for (int i = 0; i < kiek; i++)
            fr.WriteLine("{0, 3}   {1, -20} {2, 8:Y}   {3, -14}",
                          i + 1, studentai[i].PavVrd,
                          studentai[i].GimData.ToString("d"), studentai[i].Mokykla);
        fr.WriteLine("-----\r\n");
    }
}
```

# Pavyzdys su DateTime (6/8)

```
static void Formuoti(StudentasFF[] StudentaiF, int kiek, int gmetai,  
    StudentasFF[] StudentaiFA, out int kiekA)  
{  
    kiekA = 0;  
    for (int i = 0; i < kiek; i++)  
    {  
        if (StudentaiF[i].GimData.Year == gmetai)  
            StudentaiFA[kiekA++] = StudentaiF[i];  
    }  
}
```

# Pavyzdys su DateTime (7/8)

```
const int Cn = 500; // maksimalus studentų skaičius
StudentasFF[] StudentaiFF = new StudentasFF[Cn];
StudentasFF[] StudentaiFFA = new StudentasFF[Cn];
int kiek, // studentų skaičius
    kiekA; // atrinktų studentų skaičius
SkaitytiStudF(CFdF, StudentaiFF, out kiek);
SpausdintiStudF(CFr, StudentaiFF, kiek, "Studentų sąrašas");
Console.Write("Užrašykite pageidaujamus metus: ");
int gMetai = int.Parse(Console.ReadLine());
Formuoti(StudentaiFF, kiek, gMetai, StudentaiFFA, out kiekA);
if (kiekA > 0)
    SpausdintiStudF(CFr, StudentaiFFA, kiekA,
        "Studentų sąrašas (gimę " + gMetai + ")");
else
    Console.WriteLine("Studentų gimusių nurodytais " + gMetai +
        " nėra!");
```

# Pavyzdys su DateTime (8/8)

## Rezultatai.txt

### Studentų sąrašas

Nr.	Pavardė ir vardas	Gimimo data	Mokykla
1	Jonaitis Jonas	1996-01-01	Gimnazija 1
2	Petraitis Petras	1996-02-02	Mokykla 1
3	Antanaitis Antanas	1995-03-03	Gimnazija 2
4	Giedraitis Giedrius	1997-04-04	Gimnazija 1
5	Onaitytė Ona	1996-05-05	Mokykla 1
6	Juozaitis Juozas	1995-06-06	Mokykla 3
7	Ramunaitė Ramunė	1996-06-06	Gimnazija 2

### Studentų sąrašas (gimę 1996)

Nr.	Pavardė ir vardas	Gimimo data	Mokykla
1	Jonaitis Jonas	1996-01-01	Gimnazija 1
2	Petraitis Petras	1996-02-02	Mokykla 1
3	Onaitytė Ona	1996-05-05	Mokykla 1
4	Ramunaitė Ramunė	1996-06-06	Gimnazija 2



# Struktūra TimeSpan (1/9)

Struktūra skirta laiko intervalui (trukmei) saugoti bei juo operuoti.

Struktūra turi:

- 4 užklotus konstruktorius.
- 11 savybių (properties).
- 33 metodus.
- 10 užklotų operatorių.

# Struktūra TimeSpan (2/9)

Pavyzdžiui:

```
TimeSpan TS1 = new TimeSpan(1, 10, 50);  
Console.WriteLine("Laiko intervalas TS1: {0}", TS1);  
TimeSpan TS2 = new TimeSpan(2, 50, 30);  
Console.WriteLine("Laiko intervalas TS2: {0}", TS2);  
TimeSpan TS3 = TS1 + TS2; // + operatorius  
Console.WriteLine("Laiko intervalų suma TS3: {0}", TS3);  
TS3 = TS2 - TS1;          // - operatorius  
Console.WriteLine("Laikų intervalų skirtumas TS3: {0}", TS3);  
...
```

Rezultatas ekrane:

```
Laiko intervalas TS1: 01:10:50  
Laiko intervalas TS2: 02:50:30  
Laiko intervalų suma TS3: 04:01:20  
Laikų intervalų skirtumas TS3: 01:39:40
```

# Struktūra `TimeSpan` (3/9)

Pavyzdžiui:

```
TimeSpan TS4 = new TimeSpan(2, 30, 00);  
Console.WriteLine("Laiko intervalas TS4: {0}", TS4);  
int valandosT = TS4.Hours;  
int minutesT = TS4.Minutes;  
int sekundesT = TS4.Seconds;  
Console.WriteLine("Valandos: {0}", valandosT);  
Console.WriteLine("Minutės: {0}", minutesT);  
Console.WriteLine("Sekundės: {0}", sekundesT);  
...
```

Rezultatas ekrane:

```
Laiko intervalas TS4: 02:30:00  
Valandos: 2  
Minutės: 30  
Sekundės: 0
```

# Struktūra `TimeSpan` (4/9)

Pavyzdžiui:

```
TimeSpan TS4 = new TimeSpan(2, 30, 00);  
...  
double valandosTT = TS4.TotalHours;  
double minutesTT = TS4.TotalMinutes;  
double sekundesTT = TS4.TotalSeconds;  
Console.WriteLine("Viso valandų: {0, 8:f}", valandosTT);  
Console.WriteLine("Viso minučių: {0, 8:f}", minutesTT);  
Console.WriteLine("Viso sekundžių: {0, 8:f}", sekundesTT);  
...
```

Rezultatas ekrane:

Viso valandų:	2,50
Viso minučių:	150,00
Viso sekundžių:	9000,00

# Struktūra `TimeSpan` (5/9)

Pavyzdžiui:

```
DateTime DT = new DateTime(2016, 01, 15, 12, 30, 00);
Console.WriteLine("Data ir laikas DT: {0}", DT);
(TimeSpan TS = new TimeSpan(1, 30, 40);
Console.WriteLine("Laiko intervalas TS: {0}", TS);
DateTime Suma = DT + TS; // + operatorius
Console.WriteLine("Datos ir laiko suma: {0}", Suma);
...
```

Rezultatas ekrane:

```
Data ir laikas DT: 2016-01-15 12:30:00
Laiko intervalas TS: 01:30:40
Datos ir laiko suma: 2016-01-15 14:00:40
```

# Struktūra TimeSpan (6/9)

Pavyzdžiui:

```
string laikas = "5:55:55";  
TS = TimeSpan.Parse(laikas);  
Console.WriteLine("Laiko intervalas TS: {0}", TS);  
...
```

Rezultatas ekrane:

```
Laiko intervalas TS: 05:55:55
```

# Struktūra TimeSpan (7/9)

Pavyzdžiui:

```
if (TS1 < TS2)
    Console.WriteLine("Laiko intervalas TS1 yra trumpesnis už TS2.");
else if (TS1 == TS2)
    Console.WriteLine("Laiko intervalai TS1 ir TS2 lygūs.");
else
    Console.WriteLine("Laiko intervalas TS2 yra trumpesnis už TS1.");
...
```

Rezultatas ekrane:

Laiko intervalas TS1 yra trumpesnis už TS2.

# Struktūra TimeSpan (8/9)

Pavyzdžiui:

```
int ppoz1 = TimeSpan.Compare(TS1, TS2);
string rezult1;
if (ppoz1 < 0)
    rezult1 = "yra trumpesnis nei";
else if (ppoz1 == 0)
    rezult1 = "yra toks pats, kaip ir";
else
    rezult1 = "yra ilgesnis nei";
Console.WriteLine("{0} {1} {2}", TS1, rezult1, TS2);
...
```

Rezultatas ekrane:

```
01:10:50 yra trumpesnis nei 02:50:30
```



Pavyzdžiui:

```
string laikas = "5:5:55";  
// skirtingų formatų naudojimas  
TimeSpan TS5 = TimeSpan.Parse(laikas);  
Console.WriteLine("Laiko intervalas TS5: {0:c}", TS5);  
Console.WriteLine("Laiko intervalas TS5: {0:g}", TS5);  
Console.WriteLine("Laiko intervalas TS5: {0:G}", TS5);  
...
```

Rezultatas ekrane:

```
Laiko intervalas TS5: 05:05:55  
Laiko intervalas TS5: 5:05:55  
Laiko intervalas TS5: 0:05:05:55,0000000
```

# Išvardijimo tipas **enum**

*Išvardijimas* yra rinkinys sveikojo tipo konstantų turinčių vardus.

Išvardijimo tipas yra užrašomas naudojant raktinį žodį **enum**.

Išvardijimas yra reikšmės (value) tipo.

Tipo **enum** aprašas:

```
enum Vardas {  
    // išvardijimo sąrašas  
};
```

# Išvardijimo tipo **enum** pavyzdys

```
enum SavDienos { pirm, antr, treč, ketv, penkt,
                šešt, sekm };
enum MetuLaikai { pavasaris = 1, vasara, ruduo, žiema };
...
int nr1 = (int)SavDienos.pirm;
int nr2 = (int)SavDienos.sekm;
Console.WriteLine("Pirmadienio nr. = {0}", nr1);
Console.WriteLine("Sekmadienio nr. = {0}", nr2);
int nr3 = (int)MetuLaikai.pavasaris;
int nr4 = (int)MetuLaikai.žiema;
Console.WriteLine("Pavasario nr. = {0}", nr3);
Console.WriteLine("Žiemos nr. = {0}", nr4);
```

```
Pirmadienio nr. = 0
Sekmadienio nr. = 6
Pavasario nr. = 1
Žiemos nr. = 4
```

# Išvardijimo tipo **enum** pavyzdys

```

Console.WriteLine("Užrašykite, koks metų laikas Jums labiausiai patinka:");
Console.WriteLine("                pavasaris, vasara, ruduo, žiema?");
string atsakymas = Console.ReadLine();
MetuLaikai metLaik = (MetuLaikai)Enum.Parse(typeof(MetuLaikai), atsakymas);
switch (metLaik)
{
    case MetuLaikai.pavasaris:
        Console.WriteLine("Pavasaris - nuostabus metų laikas.");
        break;
    case MetuLaikai.vasara:
        Console.WriteLine("Vasara - atostogų metas.");
        break;
    case MetuLaikai.ruduo:
        Console.WriteLine("Ruduo - gamtos subrendęs grožis.");
        break;
    case MetuLaikai.žiema:
        Console.WriteLine("Žiema - laikotarpis kai gamta ilsisi.");
        break;
}

```

Užrašykite, koks metų laikas Jums labiausiai patinka:  
                                  pavasaris, vasara, ruduo, žiema?    vasara  
 Vasara - atostogų metas.

# Šioje temoje susipažinote:

1. Rikiavimu burbuliuko metodu.
2. Dviejų tvarkingų rinkinių suliejimu.
3. Struktūromis (`struct`).
4. Struktūromis: `DateTime`, `TimeSpan`.
5. Išvardijimo tipu (`enum`).



*Klausimai?*