

# T05. Objektų rinkiniai

2 ak. val.

# Temos klausimai

1. Objektų rinkiniai.
2. Skaičių masyvas.
3. Objektų masyvai.
4. Ciklo operatoriai.
5. Nurodyto ilgio objektų masyvo įvestis.
6. Nežinomo ilgio objektų masyvo įvestis.
7. Nuorodos tipas.



# *Objektų rinkiniai*

# Pavyzdžio sąlyga

Faile "Studentai.txt" duota informacija apie vienos grupės studentų pažymius (pvz., kontrolinio darbo įvertinimai): studento pavardė ir vardas, pažymys.

Faile "Pazymiai.txt" duota informacija apie žinių vertinimo sistemą: pažymys, pažymio reikšmė.

**Reikia paskaičiuoti, kiek kokių pažymių gavo studentai ?**

## Studentai.txt

```
7
Jonaitis Jonas; 8;
Petraitis Petras; 7;
...
Ramunaitė Ramunė; 5;
```

## Pazymiai.txt

```
10;Puikiai;
9;Labai gerai;
...
2;Nepatenkinamai;
1;Nepatenkinamai;
```

# Problemos

Norint parašyti programą šiam pavyzdžiui, turime mokėti kaip:

- programose realizuoti objektų rinkinius;
- dirbti su objektų rinkiniais: įvesti ir išvesti jų reikšmes, panaudoti rinkiniuose esančių objektų savybes;
- paeiliui peržvelgti objektų rinkinio reikšmes;
- susieti skirtinguose objektų rinkiniuose esančius objektus (pvz., konkretų studento pažymį, su pažymiu, esančiu žinių vertinimo sistemoje);

Kaip realizuoti objektų rinkinius programose bei jų reikšmių įvestį ir išvestį, aptarsime šioje temoje, o kaip susieti objektus – aptarsime T06 temoje, joje ir užbaigsime šioje temoje pradėtą šio uždavinio realizaciją.

# Objektų rinkinių poreikis

Ankstesnėse temose nagrinėjome atskirus objektus. Jiems suteikdavome konkrečius vardus. Esant nedideliam objektų skaičiui, tai galima padaryti.

Gyvenime sutinkamos situacijos sudėtingesnės. Pavyzdžiui, dabartiniame mūsų pavyzdyje studentų skaičiai gali būti dideli ir jų atvaizduoti objektais su skirtingais vardais negalime. Reikia ieškoti kitų būdų.

# Objektų rinkinių sąvoka

Esant dideliame vienatipių objektų kiekiui, naudojami objektų rinkiniai.

Objektų rinkinys turi savo vardą. Atskiras objektas ar jo savybė pasiekiami, nurodant objektų rinkinio vardą, objekto eilės numerį rinkinyje ir, jei reikia, savybės vardą.

Susipažinsime su objektų rinkinio realizacija **masyvu**.

*Objektų rinkinius galima realizuoti ir kitais būdais, tai aptarsime vėliau.*

Pradžioje susipažinkime su skaičių masyvu.



# *Skaičių masyvas*



# Masyvas ir indeksuotas kintamasis

Masyvas – tai **vieno tipo** reikšmių, išdėstytų atmintyje **nuosekliai** iš eilės viena šalia kitos, rinkinys

Masyvo elementas – **indeksuotas kintamasis**

Masyvas sudarytas iš 6 elementų

<b>3</b>	<b>5</b>	<b>-1</b>	<b>0</b>	<b>4</b>	<b>8</b>
0	1	2	3	4	5

Masyvo elementas

Masyvo elemento indeksas

**Pirmojo** masyvo elemento indeksas yra 0, o reikšmė 3

**Paskutiniojo** masyvo elemento indekso reikšmė yra 5 ( $6 - 1$ ), o reikšmė 8

# Masyvo aprašymas C# (1/2)

Aprašymas:

**Tipas[] Vardas;**

**Tipas** – masyvo elementų bazinis tipas (**int**, **double**, **string**, ...) arba vartotojo sukurtas tipas (pvz., klasė)

**[]** – masyvo požymis

**Vardas** – masyvo vardas

Toks masyvo aprašymas **nesukuria** masyvo, o tik aprašo nuorodos (rodyklės) tipo kintamąjį, kurio reikšmė yra **null**, t.y. masyvo atmintyje dar **nėra**.

**Vardas** -----> **null**

# Masyvo sukūrimas C# (2/2)

Sukūrimas (atminties išskyrimas):

**Vardas = new Tipas[Konstanta];**

**Vardas** – masyvo vardas

**new** – raktinis žodis naudojamas atminties išskyrimui

**Tipas** – masyvo elementų bazinis tipas (int, double, string, ...) arba vartotojo sukurtas tipas (pvz., klasė)

**[]** – masyvo požymis

**Konstanta** – masyvo dydis (sveikasis skaičius)

Atmintyje sukuriamą vietą masyvo elementams saugoti. Tos vietos adresą saugoja masyvo vardas **Vardas**.

# Masyvo aprašymas ir sukūrimas C#

Galima vienu sakiniu aprašyti masyvą ir jam sukurti vietą atmintyje:

```
Tipas[] Vardas = new Tipas[Konstanta];
```

**Vardas** – masyvo vardas

**new** – raktinis žodis naudojamas atminties išskyrimui

**Tipas** – masyvo elementų bazinis tipas (int, double, string, ...)  
arba vartotojo sukurtas tipas (pvz., klasė)

**[]** – masyvo požymis

**Konstanta** – masyvo dydis (sveikasis skaičius)

Atmintyje sukuriamą vietą masyvo elementams saugoti. Tos vietos adresą saugoja masyvo vardas **Vardas**.

# Masyvo aprašų pavyzdžiai (1/6)

Masyvo vardas

```
int[] Mas = new int[7];
```

Elemento tipas

Masyvo dydis

*Elemento tipas: int; Masyvo dydis: 7;*

Mas

----->

0	0	0	0	0	0	0
0	1	2	3	4	5	6

Masyvas automatiškai (C# kalboje) užpildomas nuliais (0)

# Masyvo aprašų pavyzdžiai (2/6)

Masyvo vardas

```
int[] Mas = new int[7] {8, 7, 10, 5, 8, 4, 5};
```

Elemento tipas

Masyvo dydis

*Elemento tipas: int; Masyvo dydis: 7;*

Mas

----->

<b>8</b>	<b>7</b>	<b>10</b>	<b>5</b>	<b>8</b>	<b>4</b>	<b>5</b>
0	1	2	3	4	5	6

Masyvas užpildomas aprašyme nurodytomis reikšmėmis: 8, 7, 10, ...

# Masyvo aprašų pavyzdžiai (3/6)

Masyvo vardas

```
int[] Mas = {8, 7, 10, 5, 8, 4, 5};
```

Elemento tipas

Masyvo elementų reikšmės

*Elemento tipas: int;*

*Masyvo dydis: 7;*

<b>Mas</b>	----->	<b>8</b>	<b>7</b>	<b>10</b>	<b>5</b>	<b>8</b>	<b>4</b>	<b>5</b>
		0	1	2	3	4	5	6

Kompiliatorius automatiškai suskaičiuoja kiek riestiniuose skliaustuose {} yra reikšmų, išskiria masyvui atmintį ir užrašo į ją tas reikšmes

# Masyvo aprašų pavyzdžiai (4/6)

Masyvo vardas

`const int Cn = 10;`

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

Elementų tipas

Masyvo dydis

		<i>Elemento tipas: int;</i> <i>Masyvo dydis: const int Cn = 10;</i>									
Mas	----->	0	0	0	0	0	0	0	0	0	0
		0	1	2	3	4	5	6	7	8	9

Masyvo dydis aprašytas vardine sveikojo tipo konstanta Cn.



# Masyvo aprašų pavyzdžiai (5/6)

```
const int Cn = 10;           // masyvo dydis
int kiek = 7;               // masyvo elementų skaičius
int[] Mas = new int[Cn];    // masyvas (nuoroda į masyvą)
```

```
Mas[0] = 8;
Mas[1] = 7;
Mas[2] = 10;
Mas[3] = 5;
Mas[4] = 8;
Mas[5] = 4;
Mas[6] = 5;
```

Reikšmių  
suteikimas masyvo  
7 elementams

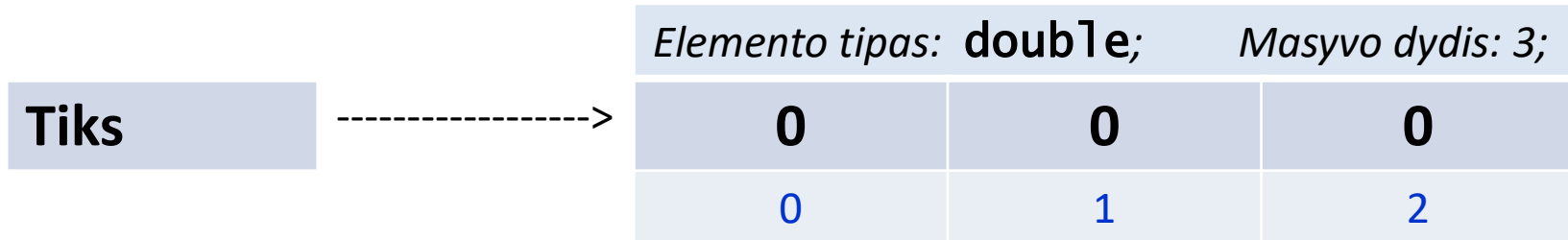
		Elemento tipas: <b>int</b> ;					Masyvo dydis: <b>Cn = 10</b> ;				
Mas	----->	<b>8</b>	<b>7</b>	<b>10</b>	<b>5</b>	<b>8</b>	<b>4</b>	<b>5</b>	<b>0</b>	<b>0</b>	<b>0</b>
		0	1	2	3	4	5	6	7	8	9

**Masyvo dydis** aprašytas vardine sveikojo tipo (int) konstanta **Cn**

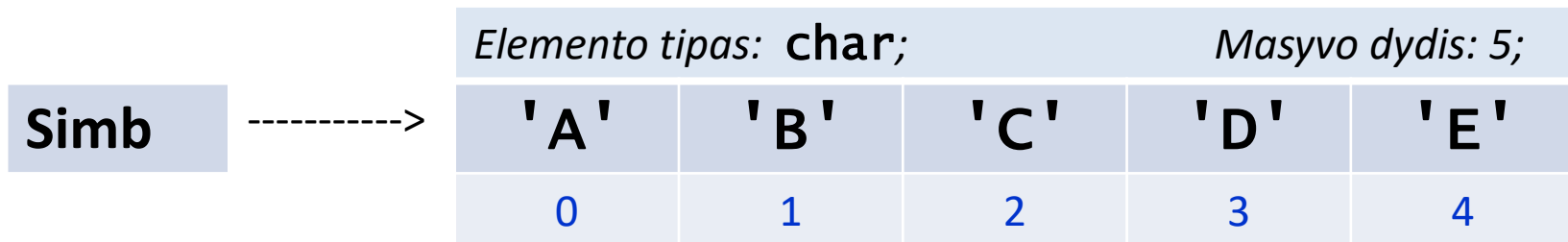
**Masyvo elementų skaičius** yra sveikojo tipo (int) kintamasis **kiek**.

# Masyvo aprašų pavyzdžiai (6/6)

```
double[] Tiks = new double[3];
```



```
char[] Simb = new char[5] { 'A', 'B', 'C', 'D', 'E' };
```



# Masyvas: svarbu atsiminti!

```

const int Cn = 10;           // masyvo dydis
int[] Mas = new int[Cn];    // masyvas (nuoroda į masyvą)
int kiek = 7;               // masyvo elementų skaičius
  
```

		Elemento tipas: <b>int;</b>					Masyvo dydis: <b>Cn = 10;</b>				
Mas	----->	<b>8</b>	<b>7</b>	<b>10</b>	<b>5</b>	<b>8</b>	<b>4</b>	<b>5</b>	<b>0</b>	<b>0</b>	<b>0</b>
		0	1	2	3	4	5	6	7	8	9

- Nepamiršti aprašyti ir išskirti atmintį (**new**) masyvui. Atminties apimtis – masyvo dydis (**Cn**) – didžiausias galimas masyvo elementų skaičius.
- Užpildyti masyvą reikšmėmis. Reikšmių skaičius, tai masyvo elementų skaičius (**kiek**)
- Pirmojo masyvo elemento (**Mas[0]**) reikšmės vieta (indeksas): **0**
- Paskutiniojo masyvo elemento (**Mas[kiek-1]**) reikšmės vieta (indeksas): **kiek-1**
- Masyvo elementų skaičius **kiek** niekuomet negali viršyti masyvo dydžio **Cn**:  
**kiek <= Cn**

# Masyvo užpildymas klaviatūra

```
// Masyvo įvedimas klaviatūra
Console.WriteLine("Kiek elementų?");
kiek = int.Parse(Console.ReadLine());
Console.WriteLine("Užrašykite {0} skaičius ", kiek);
for (int i = 0; i < kiek; i++)
    Mas[i] = int.Parse(Console.ReadLine());
```

Ekrane matysite:

Kiek elementų?

7

Užrašykite 7 skaičius

8

7

10

5

8

4

5

Po kiekvieno užrašyto  
skaičiaus paspauskite  
ENTER klavišą.

Apie masyvo įvedimą iš tekstinio duomenų failo – vėliau.

# Masyvo spausdinimas ekrane

```
// Masyvo spausdinimas ekrane
Console.WriteLine("\nĮvertinimai:");
for (int i = 0; i < kiek; i++)
    Console.WriteLine("Mas[ {0} ] = {1}", i, Mas[i]);
```

**Rezultatai ekrane:**

```
Įvertinimai:
Mas[ 0 ] = 8
Mas[ 1 ] = 7
Mas[ 2 ] = 10
Mas[ 3 ] = 5
Mas[ 4 ] = 8
Mas[ 5 ] = 4
Mas[ 6 ] = 5
Press any key to continue . . .
```

Apie masyvo spausdinimą tekstiniame faile – vėliau.

# Demo

Demonstracinė programa MVS: masyvo aprašymas,  
sukūrimas, reikšmių suteikimas



# *Objektų masyvai*

# Objektų rinkiniai – masyvai

Objektų rinkiniai – masyvai aprašomi panašiai, kaip ir skaičių masyvai.

Pradžioje aprašoma visų objektų bendras savybes ir elgseną nustatanti klasė, kurios pagrindu bus kuriamas šios klasės objektų rinkinys.

Po to aprašomas objektų rinkinys – masyvas, nurodant maksimalų rinkinio narių – objektų kiekį (maksimalų masyvo ilgį).

Objektų rinkinio – masyvo nuorodos ir masyvo sukūrimo tipinis aprašas:

```
KlasėsVardas[] MasyvoVardas =  
    new KlasėsVardas[MaksimalusIlgis]
```



# Masyvo aprašo dalys

**KlasėsVardas[] MasyvoVardas =**  
**new** **klasėsVardas[MaksimalusIlgis]**

**KlasėsVardas** – objektą aprašančios klasės vardas

**MasyvoVardas** – objektų rinkinio – masyvo vardas

**MaksimalusIlgis** – maksimalus masyvo ilgis (maksimalus objektų rinkinio objektų skaičius). Jį nurodysime konstanta (tarkime, **Cn**)

Naudojama dar viena svarbi charakteristika – esamas masyvo narių skaičius. Ji saugosime **int** tipo kintamajame (tarkime, **kiek**)

Visuomet turi būti išlaikoma sąlyga:

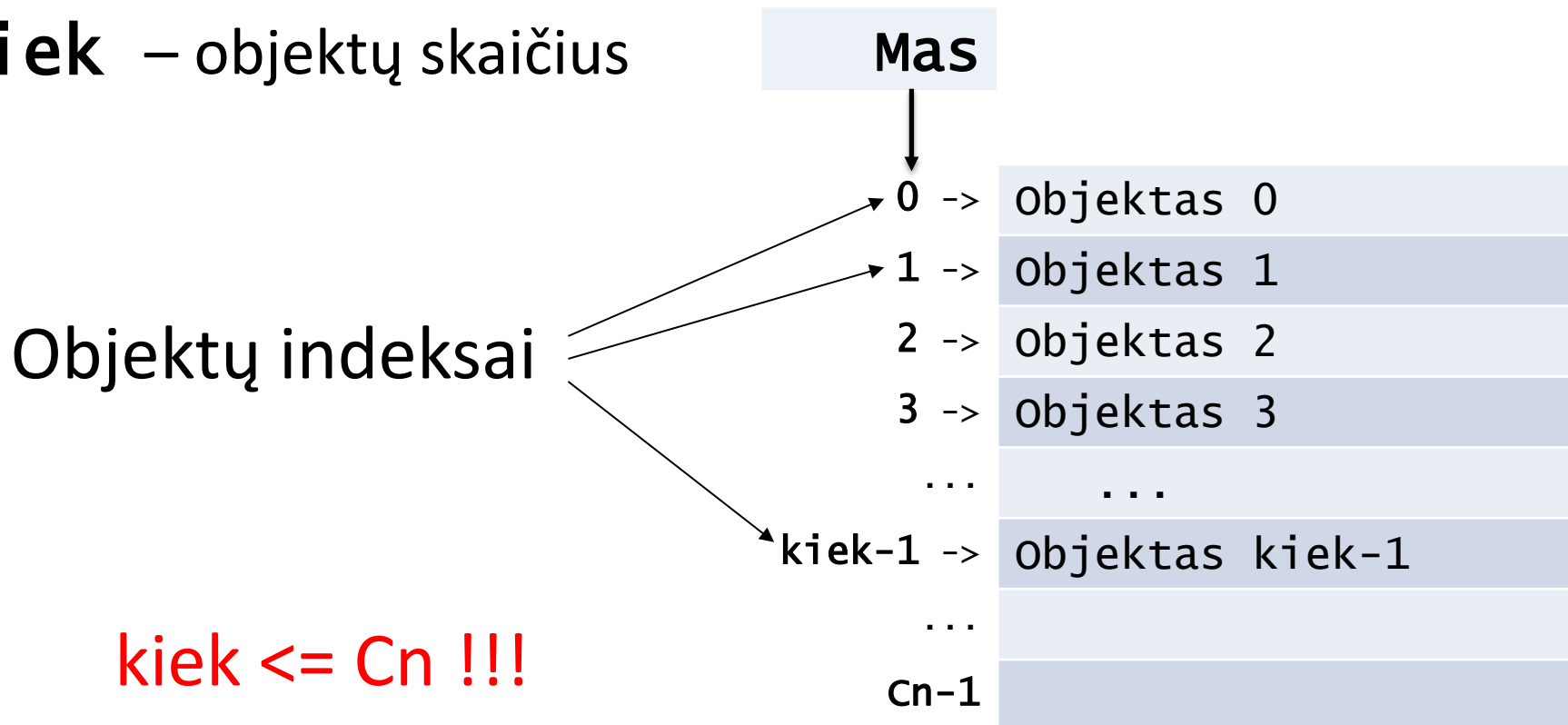
**kiek <= MaksimalusIlgis**

# Objektų masyvo išdėstymas atmintyje

`KlasėsVardas[] Mas = new KlasėsVardas[Cn];`

**Cn** – masyvo dydis

**kiek** – objektų skaičius



**kiek <= Cn !!!**

# Klasės pavyzdys (1/4)

```
class Studentas
```

```
{
```

```
    private string pavVrd;    // studento pavardė ir vardas
```

```
    private int pazym;       // pažymys (įvertinimas)
```

```
    public Studentas(string pavv, int pazym)
```

```
    {
```

```
        pavVrd = pavv;
```

```
        this.pazym = pazym;
```

```
    }
```

```
    public string ImtiPavv() { return pavVrd; }
```

```
    public int ImtiPazym()   { return pazym; }
```

```
}
```

# Objektų masyvo pavyzdys (2/4)

```
const int Cn = 7;    // maksimalus studentų skaičius
int kiek = 7;       // studentų skaičius
Studentas[] Studentai = new Studentas[Cn] // masyvas
{
    new Studentas("Jonaitis Jonas", 8),
    new Studentas("Petraitis Petras", 7),
    new Studentas("Antanaitis Antanas", 10),
    new Studentas("Giedraitis Giedrius", 5),
    new Studentas("Onaitytė Ona", 8),
    new Studentas("Juozaitis Juozas", 4),
    new Studentas("Ramunaitė Ramunė", 5)
};
```

**Pastaba:** pradinį reikšmių suteikimui {} skliaustuose privalo būti `kiek = Cn`.

# Klasės pavyzdys (3/4)

```
class Pazymys
```

```
{
```

```
    private int pazym; // pažymys (skaičius: 1..10)  
    private string pazzodR; // pažymio žodinė reikšmė  
    private int kiekis; // pažymio pasikartojimų skaičius  
    public Pazymys(int paz, string pazR)
```

```
{
```

```
        pazym = paz;  
        pazzodR = pazR;
```

```
}
```

```
    public int ImtiPazym() { return pazym; }
```

```
    public string ImtiPazReiksme() { return pazzodR; }
```

```
    public int ImtiPazKieki() { return kiekis; }
```

```
    public void DetiPazKieki(int kiekis)  
        { this.kiekis = kiekis; }
```

```
}
```

# Objektų masyvo pavyzdys (4/4)

```
const int CnP = 10;           // maksimalus pažymių skaičius
Pazymys[] Pazymiai = new Pazymys[CnP] // pažymių masyvas
{
    new Pazymys(10, "Puikiai"),
    new Pazymys(9, "Labai gerai"),
    new Pazymys(8, "Gerai"),
    new Pazymys(7, "Vidutiniškai"),
    new Pazymys(6, "Patenkinamai"),
    new Pazymys(5, "Silpnai"),
    new Pazymys(4, "Nepatenkinamai"),
    new Pazymys(3, "Nepatenkinamai"),
    new Pazymys(2, "Nepatenkinamai"),
    new Pazymys(1, "Nepatenkinamai")
};
```

# Masyvai ir jų dalys

Masyvo vardas atitinka kreipinį į visą masyvą.  
Pavyzdžiui, **Studentai** arba **Pazymiai**.

Kreiptis į masyve saugoma objektą galima nurodžius masyvo vardą ir, laužtiniuose skliaustuose, objekto indeksą. Pavyzdžiui, **Studentai[5]** arba, jei **i** yra sveikųjų skaičių tipo kintamasis, tai: **Studentai[i]**. Pirmas masyvo indeksas visuomet yra 0.

Kreiptis į masyve saugomo objekto savybę galima nurodžius masyvo vardą, indeksą ir, už taško, sąsajos metodo vardą.

# Pavyzdžiai

Reikšmių panaudojimo pavyzdžiai:

```
int ind = 6;  
string eilute;  
eilute = Studentai[ind].ImtiPavv();  
Console.WriteLine(eilute);  
Console.WriteLine(Pazymiai[0].ImtiPaz());
```

Ekrane matysite:

Ramunaitė Ramunė  
10





## *Ciklo operatorius*

# Pasikartojantys veiksmai

Dirbant su objektų rinkiniu, tenka tuos pačius veiksmus kartoti daug kartų su kiekvienu jo objektu.

Šiems veiksmams atlikti naudojami ciklo operatoriai **for** ir **while**. Cikle kinta ciklo parametras (dažnai tai masyvo indeksas) ir tikrinama ciklo pabaigos sąlyga.

Ciklo operatorius **while** paprastai naudojamas, kai ciklų skaičius iš anksto nežinomas, o ciklas vykdomas, kol tenkinama sąlyga.

Ciklo operatorius **for** naudojamas, kai iš anksto yra žinomas ciklų pasikartojimų skaičius. C# kalboje ciklas **for**, lyginant su kitomis programavimo kalbomis, turi išplėstas galimybes ir dengia ciklo **while** galimybes.

# Ciklo operatorius **for**

Ciklo operatoriaus **for** bendrinė forma:

**for** (R1; R2; R3) **KartojamasSakinys**;

**R1** – išraiška, suteikianti ciklo parametrai pradinę reikšmę;

**R2** – ciklo vykdymo sąlyga;

**R3** – ciklo parametro keitimo išraiška;

**KartojamasSakinys** – ciklo kamienas (vienas sakiny, gali būti ir kitas ciklo sakiny). Jei reikia kartoti kelis sakinius, jie skliaudžiami skliaustais { }.

# Operatoriaus **for** algoritmas

**for** (R1; R2; R3) **KartojamasSakinys**;

1. Vieną kartą prieš ciklą skaičiuojama išraiška **R1**.
2. Skaičiuojama išraiška **R2**. Jei jos reikšmė tiesa (**true**), vykdomas **KartojamasSakinys**, jei netiesa (**false**), einama į 5 žingsnį.
3. Skaičiuojama išraiška **R3**.
4. Grįžtama į 2 žingsnį.
5. Vykdoma programa už ciklo kamieno.

# Ciklo pradžia

Ciklo parametras yra kintamasis, kuriam pradinė reikšmė suteikiama išraiškoje **R1**.

Pavyzdžiai:

```
i = 0;      // kintamasis yra aprašytas prieš ciklą  
int j = 1;  // kintamasis yra lokalus
```

**Pirmuoju atveju**, ciklui pasibaigus, t.y. už ciklo, ciklo kintamojo **i** reikšmė išlieka apibrėžta ir ją galima toliau naudoti programoje.

**Antruoju atveju**, kintamojo **j** reikšmė išlieka apibrėžta tik cikle, t.y. lokaliai ir už ciklo kintamasis apamai neegzistuoja ir toliau negali būti naudojamas.

Ciklo parametras naudojamas ciklo kamienne **{}**. Tinkamai kisdamas cikle, jis garantuoja ciklo užbaigimą.

# Ciklo vykdymo sąlyga

**Ciklas turi būti baigtinis.**

Tai kontroliuoja ciklo vykdymo sąlyga **R2**, kuriai nustojus galioti (tampa lygi **false**) ciklas užbaigiamas.

**R2** pavyzdžiai:

```
i < kiek;  
j >= 10;  
x < 3.5
```

Išraiškoje **R2** dažnai naudojama santykio operacija, tačiau tai nebūtina. Išraiška gali būti bet kokia, tačiau jos rezultatas turi būti **true** arba **false**.

# Santykio operacijos

Ženklas	Operacijos pavadinimas
==	ar lygu
!=	ar nelygu
>	ar daugiau
<	ar mažiau
>=	ar daugiau arba lygu
<=	ar mažiau arba lygu

# Ciklo parametro keitimas

Ciklo parametras turi kisti, užtikrindamas ciklo baigtinumą. Tai atlieka ciklo pabaigoje skaičiuojama išraiška **R3**, keičianti ciklo parametro reikšmę.

**R3** pavyzdžiai:

```
i = i + 1;    // arba i++ arba i += 1  
j = j - 1;    // arba j-- arba j -= 1  
x = x + 0.5;  // arba x += 0.5
```



# Operatoriaus **for** pavydžiai (1/2)

```
int[] MasSv = new int[] { 1, 2, 3, 4, 5 };
int kiekSv = MasSv.Length;
int i;
for (i = 0; i < kiekSv; i = i + 2)
    MasSv[i] = i * MasSv[i];
```

//-----

```
double[] MasRe = new double[100];
int kiekRe = 7;
int j;
for (j = kiekRe; j >= 0; j--)
{
    double x = 1.0 / j;
    MasRe[j] = 3 * (x + 0.001);
}
```

Cikle esant keliems  
sakiniams, jie  
apgaubiami  
riestiniais skliaustais

# Operatoriaus **for** pavydžiai (2/2)

```
int n = 9;
```

```
double x = 3.33;
```

```
double y;
```

P – lokalus ciklo  
parametras

```
for (int p = 1; p < n - 1; p = p + 1)  
    y = x * p;
```

# Operatorius **while** (1/2)

Ciklo operatoriaus **while** bendrinė forma:

**while** (Sąlyga) KartojamasSakinys;

**Sąlyga** – loginė išraiška; ciklas karojamas, kol ji **true**.

**KartojamasSakinys** – ciklo kamienas (C# kalbos sakiny, taip pat gali būti ir kitas ciklo sakiny). Jei cikle turi būti keli sakiniai, jie skliaudžiami skliaustais { }.

Kartojamajame sakinyje turi būti išraiška, keičianti ciklo sąlygą taip, kad ciklas kada nors baigtųsi. Todėl, dažniausiai, ciklo operatoriaus **while** kamienne būna daugiau nei vienas sakiny.

# Operatorius **while** (2/2)

## **while** (Sąlyga) Kartojamasis Sakinys;

1. Tikrinama reiškinių **Sąlyga** reikšmė. Jei ji tenkinama (**true**), vykdomas **Kartojamasis Sakinys**, jei netenkinama (**false**), einama į 3 punktą.
2. Eiti į 1 punktą.
3. Vykdomas pirmas sakiny, esantis žemiau ciklo kamieno.

# Operatoriaus **while** pavydžiai

```
int i = -13;
while (i < 0)
{
    // Kartojami sakiniai
    i = i + 1;
}
//-----
double x = 3.14;
while (x > 0)
{
    // Kartojami sakiniai
    x = x - 0.01;
}
```

# Ciklas **foreach**

Kaip atskirą ciklo tipą galima išskirti ciklą **foreach**. Šio ciklo pagalba galima nuosekliai „pereiti“ per visą masyvą nuo pradžios iki pabaigos.

**Būtina sąlyga** šio ciklo naudojimui: masyvo elementų skaičius privalo būti lygus maksimaliam masyvo elementų skaičiui.

**foreach** (**tipas** kint **in** Masyvas)

```
{  
    // veiksmai su ciklo kintamuoju kint  
    ...  
}
```

**tipas** – masyvo elementų tipas

**kint** – ciklo kintamojo vardas

**in** – raktinis žodis

**Masyvas** – masyvo vardas

# Ciklas foreach pavyzdys

```
const int Cn = 7;
int[] Mas = new int [Cn] { 3, -1, 0, 7, 2, 11, 9 };
int kiek = Mas.Length; // arba kiek = Cn;

int i = 0;
foreach (int elem in Mas)
    Console.WriteLine("Mas[{0}, 2]} = {1, 3}", i++, elem);
```

		Elemento tipas: <b>int</b> ; Masyvo dydis: Cn = 7;						
Mas	----->	3	-1	0	7	2	11	9
		0	1	2	3	4	5	6

Mas[ 0]	=	3
Mas[ 1]	=	-1
Mas[ 2]	=	0
Mas[ 3]	=	7
Mas[ 4]	=	2
Mas[ 5]	=	11
Mas[ 6]	=	9



# *Nurodyto ilgio objektų rinkinio įvestis*



# Įvesties ir išvesties metodai

Objektų rinkinių informacijos įvedimui ir išvedimui rašomos atskiros programos dalys – **įvesties** ir **išvesties metodai**.

Objektų rinkinių informacijos įvesties metodų vardus rekomenduojame pradėti žodžiu **Įvesti (Skaityti)**, pavadinime nurodant ir objektų rinkinio vardą.

Objektų rinkinių informacijos išvesties metodų vardus rekomenduojame pradėti žodžiu **Išvesti (Spausdinti)**, pavadinime nurodant ir objektų rinkinio vardą.

# Rinkinio įvestis ir išvestis

Įvedant objektų rinkinio informaciją reikia žinoti, kiek bus objektų.

*Priimsime*, kad rinkinio objektų skaičius nurodytas failo pirmoje eilutėje. Informacija apie objektus prasideda antrąja eilute. Tačiau gali būti ir kitokių duomenų pateikimo faile variantų.

Įvedant informaciją svarbu sekti, ar neperpildomi masyvai (esamas narių skaičius neviršija maksimalaus masyvo ilgio).

# Pavyzdžio tąsa

Pratęskime temos pradžioje (4 skaidrėje) pateiktą pavyzdį apie studentų įvertinimus ir žinių vertinimo sistemą.

Įvedus informaciją, galima atlikti uždavinio sąlygoje numatytus paskaičiavimus (**Reikia paskaičiuoti, kiek kokių pažymių gavo studentai ?**). Parašysime tam skirtą metodą **PapildytiPazMasyva()**.

Studento pažymys yra nurodytas studento aprašyme, pažymys, jo reikšmė ir kiekis – pažymio aprašyme. Kad galėtumėm paskaičiuoti kiek kokių pažymių gavo studentai kiekvieno studento pažymį reikia **susieti** su pažymiu žinių vertinimo sistemoje – rasti atitinkamo pažymio vietą. Apie objektų rinkinių susiejimą kalbėsime kitoje temoje, todėl šio pavyzdžio kol kas pilnai neužbaigsime.

# Pavyzdys (1/12)

```
namespace MasyvasObjektu
{
    class Studentas
    {
        ...
    }
    class Pazymys
    {
        ...
    }
    class Program
    {
        ...
    }
}
```

# Pavyzdys (2/12)

```
class Studentas
{
    private string pavVrd;    // studento pavardė ir vardas
    private int pazym;       // pažymys (įvertinimas)
    public Studentas(string pavv, int pazym)
    {
        pavVrd = pavv;
        this.pazym = pazym;
    }
    public string ImtiPavv() { return pavVrd; }
    public int ImtiPazym()  { return pazym; }
}
```

# Pavyzdys (3/12)

```
class Pazymys
{
    private int pazym;      // pažymys (skaičius: 1..10)
    private string pazZodR;  // pažymio žodinė reikšmė
    private int kiekis;     // pažymio pasikartojimų skaičius
    public Pazymys(int paz, string pazR)
    {
        pazym = paz;
        pazZodR = pazR;
    }
    public int ImtiPazym()      { return pazym; }
    public string ImtiPazReiksme() { return pazZodR; }
    public int ImtiPazKieki()   { return kiekis; }
    public void DetiPazKieki(int kiekis)
                                { this.kiekis = kiekis; }
}
```

# Pavyzdys (4/12)

```
class Program
```

```
{
```

```
    const int Cn = 100;    // maksimalus studentų skaičius  
    const int CnP = 10;   // pažymių skaičius vertinimo sistemoje  
    const string CFd = "..\\..\\Studentai.txt"; // duomenų failas  
    const string CFr = "..\\..\\Rezultatai.txt"; // rezultatų failas
```

```
    static void Main(string[] args)
```

```
{
```

```
    int kiek;                // studentų skaičius  
    Studentas[] Studentai = new Studentas[Cn]; // studentų masyvas  
    Pazymys[] Pazymiai;      // pažymių masyvas  
    Pazymiai = ...           // žiūr. kitoje skaidrėje
```

```
    ...
```

```
}
```

```
}
```

# Pavyzdys (5/12)

```
Pazymiai = new Pazymys[CnP]
{
    new Pazymys(10, "Puikiai"),
    new Pazymys(9, "Labai gerai"),
    new Pazymys(8, "Gerai"),
    new Pazymys(7, "Vidutiniškai"),
    new Pazymys(6, "Patenkinamai"),
    new Pazymys(5, "Silpnai"),
    new Pazymys(4, "Nepatenkinamai"),
    new Pazymys(3, "Nepatenkinamai"),
    new Pazymys(2, "Nepatenkinamai"),
    new Pazymys(1, "Nepatenkinamai")
};
```

**Pastaba:** masyvą galima įvesti ir iš failo (savarankiška užduotis).



# Pavyzdys (6/12)

```
static void skaitytiStud(string fv, Studentas[] Studentai, out int kiek)
{
    using (StreamReader srautas = new StreamReader(fv))
    {
        string eilute; // duomenų failo eilutė
        kiek = int.Parse(srautas.ReadLine());
        for (int i = 0; i < kiek; i++)
        {
            eilute = srautas.ReadLine();
            string[] eilDalis = eilute.Split(';'); // failo eilutės dalys
            string pavVrd = eilDalis[0];
            int pazym = int.Parse(eilDalis[1]);
            Studentai[i] = new Studentas(pavVrd, pazym);
        }
    }
}
```

```
7
Jonaitis Jonas;      8;
...
Ramunaitė Ramunė;    5;
```

# Pavyzdys (7/12)

```
static void spausdintiStud(string fv, studentas[] Studentai,  
                           int kiek, string antraste)  
{  
    const string virsus =  
        "-----\n"  
        + " Nr.   Pavardė ir vardas      Pažymys\n"  
        + "-----";  
    using (var fr = File.AppendText(fv))  
    {  
        fr.WriteLine("\n      " + antraste);  
        fr.WriteLine(virsus);  
        for (int i = 0; i < kiek; i++)  
            fr.WriteLine("{0, 3}   {1, -20}   {2, 2}",  
                          i + 1, Studentai[i].ImtiPavv(), Studentai[i].ImtiPazym());  
        fr.WriteLine("-----\n");  
    }  
}
```

# Pavyzdys (8/12)

```
static void SpausdintiPazym(string fv, Pazymys[] Pazymiai,
                             int kiek, string antraste)
{
    const string virsus =
        "-----\n"
        + " Pažymys Pažymio reikšmė kiekis\n"
        + "-----";
    using (var fr = File.AppendText(fv))
    {
        fr.WriteLine("\n" + antraste);
        fr.WriteLine(virsus);
        for (int i = 0; i < kiek; i++)
            fr.WriteLine("{0, 5}    {1, -14}    {2, 2}",
                          Pazymiai[i].ImtiPazym(), Pazymiai[i].ImtiPazReiksme(),
                          Pazymiai[i].ImtiPazKieki());
        fr.WriteLine("-----\n");
    }
}
```

# Pavyzdys (9/12)

```
static void SpausdintiPazym1(string fv, Pazymys[] Pazymiai,  
                             int kiek, string antraste)  
{  
    const string virusus =  
        "-----\n"  
        + " Pažymys Pažymio reikšmė kiekis\n"  
        + "-----";  
    using (var fr = File.AppendText(fv))  
    {  
        fr.WriteLine("\n" + antraste);  
        fr.WriteLine(virusus);  
        foreach (Pazymys elem in Pazymiai)  
            fr.WriteLine("{0, 5} {1, -14} {2, 2}",  
                          elem.ImtiPazym(), elem.ImtiPazReiksme(),  
                          elem.ImtiPazKieki());  
        fr.WriteLine("-----\n");  
    }  
}
```

Ciklo **foreach**  
panaudojimas darbe su  
objektų masyvo  
elementais

# Pavyzdys (10/12)

7

Jonaitis Jonas;	8;
Petraitis Petras;	7;
Antanaitis Antanas;	10;
Giedraitis Giedrius;	5;
Onaitytė Ona;	8;
Juozaitis Juozas;	4;
Ramunaitė Ramunė;	5;

## Studentų sąrašas

Nr.	Pavardė ir vardas	Pažymys
1	Jonaitis Jonas	8
2	Petraitis Petras	7
3	Antanaitis Antanas	10
4	Giedraitis Giedrius	5
5	Onaitytė Ona	8
6	Juozaitis Juozas	4
7	Ramunaitė Ramunė	5

## Žinių vertinimo sistema

### Pažymys Pažymio reikšmė Kiekis

10	Puikiai	0
9	Labai gerai	0
8	Gerai	0
7	Vidutiniškai	0
6	Patenkinamai	0
5	Silpnai	0
4	Nepatenkinamai	0
3	Nepatenkinamai	0
2	Nepatenkinamai	0
1	Nepatenkinamai	0

# Pavyzdys (11/12)

```
static void PapildytiPazMasyva(Pazymys[] Pazymiai,  
                                int kiekP,  
                                Studentas[] Studentai,  
                                int kiek)  
{  
    // Metodo realizacija bus pateikta kitoje temoje  
    ...  
}
```

# Pavyzdys (12/12)

```
class Program
{
    // Konstantos
    ...
    static void Main(string[] args)
    {
        // Objektų masyvų aprašai
        int kiek; // studentų skaičius
        Studentas[] Studentai = new Studentas[Cn]; // studentų masyvas
        Pazymys[] Pazymiai; // pažymių masyvas
        Pazymiai = ...
        // Rezultatų failo sukūrimas (išvalymas) pagal poreikį
        ...
        // Duomenų įvedimas ir išvedimas
        SkaitytiStud(CFd, Studentai, out kiek);
        SpausdintiStud(CFr, Studentai, kiek, "Studentų sąrašas");
        SpausdintiPazym(CFr, Pazymiai, CnP, "Žinių vertinimo sistema");
        // Skaičiavimai bus pateikti kitoje temoje
        ...
        // Atsakymų išvedimas
    }
}
```

# Demo

Demonstracinė programa MVS: klasių aprašymas, objektų masyvų sukūrimas, masyvų įvedimas ir išvedimas.





# *Nežinomo ilgio objektų rinkinio įvestis*

# Duomenų įvedimo problema

Įvedant duomenis, iš anksto nežinoma, kiek jų yra. Skaičiuoti ir nurodyti failo pradžioje duomenų kiekį, kaip darėme iki šiol, nepatogu.

Dažniausiai duomenys iš failo įvedami visi, tačiau ypač svarbu stebėti, ar dar įvedamiems duomenims atmintyje (masyve) yra vietos, ir neleisti jos perpildyti (būtina apsauga).

# Pavyzdys

Pakeiskime mūsų pavyzdį (apie studentų pažymius ir žinių vertinimo sistemą) taip, kad nereikėtų failo pradžioje nurodinėti objektų (studentų) skaičiaus.

Tam įvedimo metu tikrinsime, ar iš failo nuskaityti visi duomenys ir ar, įvedamų objektų skaičius neviršija maksimalios masyvo ribos.

Koreguosime tik duomenų įvedimą, kitos programos dalys nesikeičia.

# Pavyzdys (6/11)

```
static void SkaitytiStud(string fv, Studentas[] Studentai, out int kiek)
{
    using (StreamReader srautas = new StreamReader(fv))
    {
        string eilute; // duomenų failo eilutė
        int i = 0;
        while ((eilute = srautas.ReadLine()) != null && (i < Cn))
        {
            string[] eilDalis = eilute.Split(';'); // failo eilutės dalys
            string pavVrd = eilDalis[0];
            int pazym = int.Parse(eilDalis[1]);
            Studentai[i++] = new Studentas(pavVrd, pazym);
        }
        kiek = i;
    }
}
```

Apsauga nuo  
masyvo  
perpildymo

Jonaitis Jonas;	8;
...	
Ramunaitė Ramunė;	5;



## *Nuorodos tipas (angl. reference)*

# Kintamasis ir jo adresas

**Aprašius kintamąjį**, nepanaudotoje atminties vietoje, jam pagal nurodytą tipą išskiriama atmintis.

Ši vieta įsimenama ir tai yra kintamojo **adresas** (šešioliktainis skaičius).

Pavyzdžiui:

```
int    nr = 9;  
double suma = 0.3;  
char   sim = 'x';
```

001ECA10

9

**nr**

001ECA12

0.3

**suma**

001ECAA0

x

**sim**

Taip atmintis skiriama **baziniams** duomenų **tipams** (vadinamiems **reikšmės tipams**): **int**, **double**, ...

# Reikšmės ir nuorodos tipai

```
double kaina = 0.8; Prekė pr = new Prekė();
```

kaina

0.8

pr

FAB10048

FAB10048

Pieštukas

0.8

100

Atminties ląstelė (kintamasis) **kaina** saugo duomenis (**bazinis tipas**), o atminties ląstelė (kintamasis) **pr** – adresą į duomenis (**nuorodos tipas**).

# Nuorodos tipas

Nuorodos tipas yra skirtas išskirtos atminties vietai (**adresui**) (šešioliktainis skaičius) saugoti.

**Nuorodos tipo** kintamieji gali būti:

- **string** tipo;
- klasės (**class**) tipo;
- masyvo (pvz., **Prekė[]**) tipo.

Kiekvienas tokio tipo kintamasis pačių duomenų nesaugo. Paskelbus tokio tipo kintamuosius, jų reikšmės yra **null**, o tai reiškia, kad jie nerodo į jokią atminties vietą.

Tik vėliau, panaudojus operatorių **new** (išskyrus **string**), jie įgauna išskirtos atminties vietos adresą, kur bus saugomi duomenys.



# Nuorodų privalumas

- Vartojant nuorodas, programos darbo metu atmintis valdoma dinamiškai, t.y. reikalui esant, programa jos paprašo, o po to, kai ji tampa nereikalinga *automatiškai* sugrąžina. Tam naudojamas taip vadinamas „šiukšlių surinkėjas“ (*angl.* garbage collector).
- Atmintis bereikalingai nešvaistoma.

# Nuorodos metodų parametruose (**ref** arba **out**)

Rašomos prieš parametrus, kurių pakitusios reikšmės turi grįžti iš metodo į jį kvietusį metodą (dažniausiai **Main()**).

Tam naudojami du būdai:

- **ref** naudojamas kintamiesiems, kurių **pradinės reikšmės jau yra priskirtos iki kreipinio**. Tokiam kintamajam nėra būtina priskirti reikšmę metodo viduje;
- **out** naudojamas kintamiesiems, kurių **pradinės reikšmės priskiriamos metodo viduje**;
- šie raktiniai žodžiai **ref** ir **out** turi būti naudojami tiek **metodo antraštėje**, tiek ir **kreipinyje į metodą**.

# Metodo su **out** parametru pavyzdys

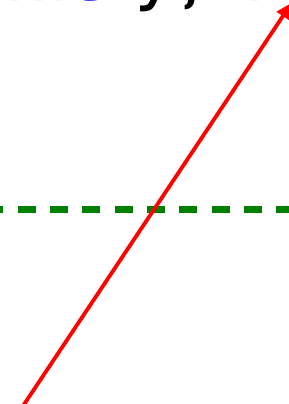
```
// Metoda padalina x iš y ir gauta rezultata priskiria dalmeniui dalmuo
static void Dalmuo(int x, int y, out int dalmuo)
{
    dalmuo = x / y; // dėl 'out' būtina priskirti reikšmę !
}
//-----
int a1 = 7,
    b1 = 2,
    dlm; // dalmuo
Dalmuo(a1, b1, out dlm);
Console.WriteLine("Po iškvietimo: ");
Console.WriteLine("{0} / {1} = {2}", a1, b1, dlm);
```

Ekrane:

Po iškvietimo: 7 / 2 = 3

# Metodo su **ref** parametru pavyzdys

```
// Metodas padalina x iš y ir gautą liekaną priskiria liekanai liekana
static void Liekana(int x, int y, ref int liekana)
{
    liekana = x % y;
}
//-----
int a2 = 7,
    b2 = 2,
    lkn = 0; // liekana (dėl 'ref' turi būti suteikta reikšmė!)
Liekana(a2, b2, ref lkn);
Console.WriteLine("Po iškvietimo: ");
Console.WriteLine("{0} % {1} = {2}", a2, b2, lkn);
```



Ekrane:

Po iškvietimo: 7 % 2 = 1

# Nuorodų perdavimo į metodą mechanizmas

Kreipiantis į metodą, kuris savo parametrų apraše turi nuorodą (**out** arba **ref**), metodui yra perduodami ne patys argumentai, o jų saugojimo atmintyje vieta, t.y. **adresas**. Tokiu būdu metodo viduje atliekami veiksmai su argumento reikšme atsispindi jo (argumento) iškvietimo vietoje.

`int a = 5;`

~~5~~ 6  
FF100002

Perduodamas argumento a  
adresas

Kreipinys į metodą, kurio viduje yra: `a = a + 1;`

# Masyvo perdavimas metodui (1/2)

Kreipiantis į metodą, jam yra perduodama tik **nuoroda** (adresas) į masyvo saugojimo vietą atmintyje (nulinį elementą). Metode atlikus masyvo reikšmių pakeitimus, masyvo nuoroda nesikeičia, pasikeičia tik masyvo elementų reikšmės. Panagrinėkime pavyzdį, kuris demonstruoja masyvo perdavimo metodui mechanizmą, t.y. kaip keičiasi masyvo elementų reikšmės (o masyvo nuoroda nesikeičia).

// Metoda grąžina suskaičiuotą masyvo Mas(kiek) reikšmių sumą

```
static int Suma(int[] Mas, int kiek)
```

```
{
```

```
    int s = 0;
```

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

```
        s += Mas[i];
```

```
    return s;
```

```
}
```

```
//-----
```

// Masyvo Mas(kiek) pabaigoje įrašoma suskaičiuota suma

```
static void IrasytiSuma(int[] Mas, ref int kiek)
```

```
{
```

```
    Mas[kiek] = Suma(Mas, kiek);
```

```
    kiek++;
```

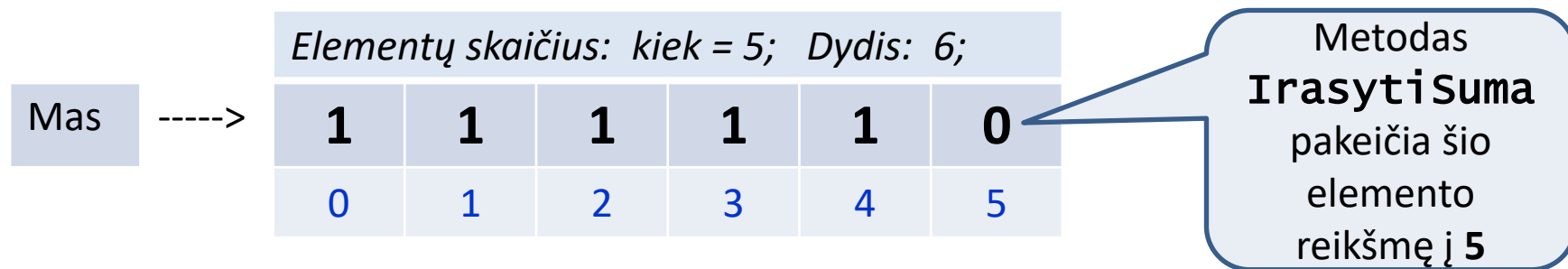
```
}
```

Reikia **ref**, kadangi  
metode didinamas  
masyvo elementų skaičius

# Masyvo perdavimas metodui (2/2)

```
int[] Mas = new int [6] { 1, 1, 1, 1, 1, 0 };
int kiek = 5;
int suma = Suma(Mas, kiek);
Console.WriteLine("Suma prieš: {0}", suma);
IrasytiSuma(Mas, ref kiek);
suma = Mas[kiek-1];
Console.WriteLine("Suma po: {0}", suma);
```

Suma prieš: 5  
Suma po: 5



# Šioje temoje susipažinome su:

1. Objektų rinkinių realizavimu masyvais.
2. Skaičių masyvu, indeksuotais kintamaisiais.
3. Ciklo operatoriais **for**, **while** ir **foreach**.
4. Nurodyto ilgio objektų masyvo įvestimi.
5. Nežinomo ilgio objektų masyvo įvestimi.
6. Nuorodos tipu.





*Klausimai?*