

T09. Operacijos su objektais

2 ak. val.

Temos klausimai

1. Veiksmai su objektais. Objektų palyginimas (<, >, ==). Operacijų užklojimas.
2. Nuosekli paieška nerikiuotame masyve panaudojant užklotą == operaciją.
3. Nario įterpimas į masyvą. Atvejai, kai masyvas nerikiuotas ir rikiuotas.
4. Nario išmetimas iš masyvo. Atvejai, kai masyvas nerikiuotas ir rikiuotas.



Operatorių užklojimas

Veiksmai su objektais

Priskyrimo veiksmas (=).

Yra įgyvendintas.

Aritmetiniai veiksmai (+, -, +=, -=, ++, --, ...)

Veiksmą apibrėžia vartotojas.

Objektų palyginimas (<, >, ==).

Veiksmą apibrėžia vartotojas. Yra papildomų reikalavimų.

Veiksmų su objektais realizavimas (1/2)

Kuriant ir naudojant **išorinius metodus**, pvz.:

// Objekto **stud** lauką **pazym** padidina dydžiu **pokytis**

```
public void PadidintiPazymi(Studentas stud, int pokytis);
```

// Gražina **true**, jei studento **stud1** pažymys didesnis už studento **stud2** pažymį, kitaip - **false**

```
public bool ArDidesnis(Studentas stud1, Studentas stud2);
```

Veiksmų su objektais realizavimas (2/2)

Užklojant ir taikant baziniams tipams įprastus veiksmus, pvz.:

```
// Objekto stud1 nuorodą priskiria objekto stud2 nuorodai  
Studentas stud1, stud2;
```

```
...
```

```
stud2 = stud1;
```

```
// Objekto stud lauką pazym padidina dydžiu pokytis
```

```
Studentas stud;
```

```
int pokytis;
```

```
...
```

```
stud = stud + pokytis;
```

```
// Priskiria true, jei studentas stud1 geresnis už stud2, kitaip - false
```

```
Studentas stud1, stud2;
```

```
...
```

```
bool p = stud1 > stud2;
```

Operatorių užklojimo privalumas

Galima naudoti **bazinių operatorių sintaksę** veiksmuose su mūsų sukurtų klasių **objektais**, pvz.:

a = b + c;

kur **a**, **b**, **c** – sukurtos klasės **X** objektai.

Pastaba: užklotam operatoriui tikslinga suteikti tokį turinį, kuris bent kiek primintų bazinį operatorių.

Operatorių užklojimo esmė

Tas pats užklotas operatorius (pvz., **+**, **-**) kiekvienoje klasėje turi skirtingą prasmę.

Užklojimo operatoriaus esmė glūdi klasės metode, kuris iškviečiamas kiekvieną kartą, kai programos tekste užklotas operatorius taikomas tos klasės objektams.

Užklojamų operatorių sąrašas

+(ženklas)	-(ženklas)	!	~	++
--	+	-	*	/
%	&		^	<<
>>	==	!=	>	>
>=	<=			

+=, -= ir t.t. netiesiogiai per atitinkamus **+**, **-**

&& ir **||** netiesiogiai per **&** ir **|**, atitinkamai

Palyginimo operatoriai **užklojami poromis** (**<**, **>**, **==**, **!=**)

Operatoriams (**==**, **!=**) rekomenduojama užkloti **Equals()**,
HashCode()

Operatorių užklojimo sintaksė

```
public static <tipas> operator<operatoriaus simbolis>  
([<parametrai>])  
{  
    ...  
}
```

<tipas> – nustato užkloto operatoriaus grąžinamo rezultato tipą (**void**, **bool**, **Studentas**, ...),

operator – bazinis žodis,

<operatoriaus simbolis> – parodo, kuris operatorius užklotas,

<parametrai> – operatoriaus parametų sąrašas.

Operatoriaus antraštės taisyklės

- Antraštė pradedama **public static**.
- Antraštėje svarbu tinkamai apibrėžti *grąžinamos reikšmės tipą*.
2 rūšys:
 - *klasės objektas* – operatoriams, kurie nekeičia pirmojo objekto reikšmės. Tai **+**, **-**, **/** ir t.t.
 - **bool** – palyginimo operatoriams **==**, **!=** ir t.t.
- Toliau - **operator** operatoriaus simbolis
- Jei operatorius *vienvietis*, vienas parametras.
- Jei operatorius *dvivietis*, visada pirmasis parametras turi formą **X pirmas**, kur **X** – vardas klasės, kuriai užklojamas operatorius, o antrasis parametras – tai su kuo dalyvauja operacijoje pirmasis parametras.

Antraštė ir kamienas

- Nuo antraštės priklauso kamienas.
- Jei reikia grąžinti klasės objektą, tai metodo viduje skelbiamas laikinas klasės tipo objektas, kuris sukaupia operatoriaus rezultatą, ir grąžinama šio objekto reikšmė. Esamos klasės objektas išlaiko savo reikšmę nepakitusią.

Operatorių variantai

Operatoriai gali būti:

- dviviečiai, pvz.: **+**, **-**, **&&**
- vienviečiai, pvz.: **++**, **--**, **!**.

Dviviečiai operatoriai turi operandus iš abiejų pusių, pvz., **a + b**, **c - d**, **a = d** ir panašiai.

Vienviečiai operatoriai operandą turi tik iš vienos pusės, pavyzdžiui, **!a**, **b++** ir panašiai.

Užklojamo operatoriaus ir metodo panašumai ir skirtumai

Operatorių užklojimo sakinyss taip pat kaip ir bet kuris metodas talpinamas tos klasės aprašymo viduje, kurios objektams ruošiamės naudoti operatorių.

Šis sakinyss analogiškas klasės metodo antraštei, skiriasi tik tuo, kad yra papildomas bazinis žodis **operator**, o metodo vardas – tai užklojamo operatoriaus simbolis (pvz.: +, -, ++, += ir pan.).

Dviviečiam operatoriui *pirmasis parametras* yra klasės objektas, nepriklausomai nuo to, koks operatorius apibrėžiamas, o *antrasis parametras* – tos pačios klasės objektas ar bazinio tipo kintamasis.

Vienviečiam operatoriui *parametras* yra klasės objektas.

Objektai užklotoje operacijoje

- Kai dviviečiame operatoriuje dalyvauja 2 to paties tipo objektai, jie abu turi šį užklotą operatorių.
- Atliekant užklotą operatorių, *visada kviečiamas kairėje operatoriaus pusėje esančio objekto metodas*, kuris užkloja minėtą operatorių.
- Kairysis objektas perduodamas per pirmąjį parametą, o dešinysis – per antrąjį parametą.



Operatorių užklojimo pavyzdys

Pirminė (bazinė) klasė **Object**

Kiekviena naujai kuriama klasė paveldi bazinę klasę **Object**.

Ši klasė turi **metodus**, kuriuos tam tikrais atvejais reikia užkloti naujai kuriamose klasėse:

```
// Gražina eilutę (string), kurioje yra sukaupta  
// informacija apie objektą (kintamųjų reikšmes)
```

```
virtual ToString()
```

```
// Naudojamas hash lentelėse
```

```
virtual GetHashCode()
```

```
// Palygina du objektus:
```

```
// gražina true, jei objektai lygus;
```

```
// false – priešingu atveju
```

```
virtual Equals()
```

Klasė Studentas

```
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; }
    ... // sąsajos metodai
}
```

Klasė **Studentas** (papildyta užklotais metodais)

```
class Studentas
```

```
{
    ...
    // užklotas metodas ToString()
    public override string ToString()
    {
        string eilute;
        eilute = string.Format("{0, -20}    {1, 2}", pavVrd, pazym);
        return eilute;
    }
    // užklotas metodas Equals()
    public override bool Equals(object objektas)
    {
        Studentas stud = objektas as Studentas;
        return stud.pavVrd == pavVrd;
    }
    // užklotas metodas GetHashCode()
    public override int GetHashCode()
    {
        return base.GetHashCode();
    }
}
```

```
Studentas stud = new Studentas("Pavardenis Vardenis", 9);
```

```
Console.WriteLine(stud.ToString());
```

```
// arba
```

```
Console.WriteLine(stud);
```

Palyginkite, kaip rašėte anksčiau:

```
Console.WriteLine("{0, 3}    {1, -20}    {2, 2}",  
    i + 1, Studentai[i].ImtiPavv(), Studentai[i].ImtiPazym());
```

Pirmųjų dviejų spausdinimo sakinių `writeLine()` rezultatas ekrane:

Pavardenis Vardenis 9

Klasė **Studentas** (papildyta užklotais operatoriais == ir !=)

```
class Studentas
{
    ...
    // užklotas operatorius ==
    public static bool operator ==(Studentas stud1, Studentas stud2)
    {
        return stud1.pavVrd == stud2.pavVrd;
        // arba
        //int poz = String.Compare(stud1.pavVrd, stud2.pavVrd,
        //                           StringComparison.CurrentCulture);
        //return poz == 0;
    }
    // užklotas operatorius !=
    public static bool operator !=(Studentas stud1, Studentas stud2)
    {
        return !(stud1.pavVrd == stud2.pavVrd);
        // arba
        //int poz = String.Compare(stud1.pavVrd, stud2.pavVrd,
        //                           StringComparison.CurrentCulture);
        //return poz != 0;
    }
}
```

Klasė **Studentas** (papildyta užklotais operatoriais > ir <)

```
class Studentas
{
    ...
    // užklotas operatorius >
    public static bool operator >(Studentas stud1, Studentas stud2)
    {
        int poz = String.Compare(stud1.pavVrd, stud2.pavVrd,
                                   StringComparison.CurrentCulture);

        return poz > 0;
    }
    // užklotas operatorius <
    public static bool operator <(Studentas stud1, Studentas stud2)
    {
        int poz = String.Compare(stud1.pavVrd, stud2.pavVrd,
                                   StringComparison.CurrentCulture);

        return poz < 0;
    }
}
```

Klasė **Studentas** (papildyta užklotais operatoriais \geq ir \leq)

```
class Studentas
{
    ...
    // užklotas operatorius >=
    public static bool operator >=(Studentas stud1, Studentas stud2)
    {
        int poz = String.Compare(stud1.pavVrd, stud2.pavVrd,
                                   StringComparison.CurrentCulture);
        if ((stud1.pazym > stud2.pazym) ||
            ((stud1.pazym == stud2.pazym) && (poz > 0))) return true;
        else return false;
    }
    // užklotas operatorius <=
    public static bool operator <=(Studentas stud1, Studentas stud2)
    {
        int poz = String.Compare(stud1.pavVrd, stud2.pavVrd,
                                   StringComparison.CurrentCulture);
        if ((stud1.pazym < stud2.pazym) ||
            ((stud1.pazym == stud2.pazym) && (poz > 0))) return true;
        else return false;
    }
}
```


Klasė **Studentas** (papildyta užklotu operatoriumi +)

```
class Studentas
{
    ...
    // užklotas operatorius +
    public static Studentas operator +(Studentas stud1, Studentas stud2)
    {
        Studentas laik = new Studentas(); // objektas rezultatų kaupimui
        laik.pazym = stud1.pazym + stud2.pazym;
        return laik;
    }
}
```

Užklotų operatorių naudojimas

```
// Gražina, masyvo Studentai(kiek) pažymių sumą
static int SumaPazymiu(Studentas[] Studentai, int kiek)
{
    Studentas suma = new Studentas();
    for (int i = 0; i < kiek; i++)
    {
        suma = suma + Studentai[i];
        //arba suma += Studentai[i];
    }
    return suma.ImtiPazym();
}
```

Naudojamas užklotas
operatorius +

Užklotų operatorių naudojimas

```
static void Main(string[] args)
{
    Studentas stud1 = new Studentas("Pavardenis Vardenis", 7);
    Studentas stud2 = new Studentas("Pavardaitis Vardaitis", 10);

    if (stud1 > stud2)
        Console.WriteLine("Pirmo studento pavardė DIDESNĖ.");
    else
        Console.WriteLine("Pirmo studento pavardė MAŽESNĖ arba LYGI.");
}
```

Naudojamas užklotas operatorius >

Užkloto operatoriaus panaudojimo pavyzdys (1/4)

Pavyzdžiui, studentų konteineryje reikia surasti nurodytą studentą, t.y. atlikti paiešką.

Konteinerinę klasę **MasyvasStudentai** papildykime studento paieškos metodu **StudentoVieta()**.

Užkloto operatoriaus panaudojimo pavyzdys

(2/4)

```
class MasyvasStudentai
{
    const int Cn = 50;           // studentų masyvo dydis
    private Studentas [] Studentai; // studentų objektų masyvas
    private int kiek;           // studentų skaičius

    public MasyvasStudentai()
    {
        kiek = 0;
        Studentai = new Studentas[Cn];
    }

    public Studentas ImtiStudenta(int i)    { return Studentai[i]; }
    public int ImtiKiek()                   { return kiek; }
    public void DetiStudenta(Studentas obj) { Studentai[kiek++] = obj; }
    public int StudentoVieta(Studentas stud) { ... }
}
```

Užkloto operatoriaus panaudojimo pavyzdys

(3/4)

// Gražina nurodyto studento stud indeksą,

// priešingu atveju gražina -1

```
public int StudentoVieta(Studentas stud)
```

```
{  
    for (int i = 0; i < kiek; i++)  
    {  
        if (Studentai[i] == stud)  
            return i;  
    }  
    return -1;  
}
```

Naudojamas klasės
Studentas
užklotas operatorius
==

Užkloto operatoriaus panaudojimo pavyzdys

(4/4)

```
static void Main(string[] args)
{
    ...
    MasyvasStudentai TestasMas = new MasyvasStudentai();
    SkaitytiStudKont(CFd1, TestasMas);
    SpausdintiStudKont(CFr, TestasMas, "Studentų sąrašas");
    Studentas stud = new Studentas("Juozaitis Juozas");
    int index = TestasMas.StudentoVieta(stud);
    if (index >= 0)
        Console.WriteLine("Ieškomas studentas: {0} {1}\n",
                           TestasMas.ImtiStudenta(index).ImtiPavv(),
                           TestasMas.ImtiStudenta(index).ImtiPazym());
    else
        Console.WriteLine("Studento {0} masyve nėra!",
                           stud.ImtiPavv());
}
```



*Nario įterpimas į nerikiuotą ir rikiuotą
masyvą*

Naujas narys masyve $A(n)$

Prieš papildant masyvą nauju nariu (**naujas**) arba įterpiant naują narį (**naujas**) į masyvą, **reikia patikrinti, ar masyvas nepilnas**, t.y.

ar $n < C_n$ (masyvo dydis)?

0	1	2			n-2	n-1			C_n-1
4	-3	6	10	25	-7	1			

Masyvo $A(n)$ papildymas

Papildant masyvą nauju nariu reikalinga naujojo nario reikšmė (**naujas**) ir papildymo vieta **k**.

Galimi atvejai:

- kai duomenys netvarkingi
- kai duomenys tvarkingi (surikiuoti)

Kai duomenys netvarkingi naują narį galima įrašyti masyvo pabaigoje arba nurodytoje vietoje **k**.

Kai duomenys tvarkingi, naują narį reikalinga įrašyti toje vietoje, kur jis tinka (reikalinga vietos **k** paieška).

0	1	2		k		n-1			Cn-1
4	-3	6	10	25	-7	1			

Masyvo papildymas nauju nariu

Masyvo pabaigoje: **n**

A[n] = naujas;

n = n+1;

Kintamasis **naujas** yra to paties tipo, kaip ir masyvo **A(n)** elementai.

0	1	2			n-2	n-1	n		Cn-1
4	-3	6	10	25	-7	1			

naujas: 99

0	1	2				n-2	n-1		Cn-1
4	-3	6	10	25	-7	1	99		

Naujo nario įterpimas **nerikiuotame** masyve

Įterpimo vieta: **k**

```
A[n] = A[k];
A[k] = naujas;
n = n+1;
```

0	1	2		k		n-1	n		C _{n-1}
4	-3	6	10	25	-7	1			

naujas: 99

0	1	2		k		n-2	n-1		C _{n-1}
4	-3	6	10	99	-7	1	25		

Įterpimo vieta: k

```
for (int i = n; i > k; i--)  
    A[i] = A[i-1];  
A[k] = naujas;  
n = n + 1;
```

0	1	2		k		$n-1$	n		C_{n-1}
-7	-3	1	4	6	10	25			

naujas: 5



0	1	2		k		$n-2$	$n-1$		C_{n-1}
-7	-3	1	4	5	6	10	25		

```
// Grąžina reikšmės naujas įterpimo vietą rikiuotame masyve A(n)
static int IterpimoVietaRikiuotame(int[] A, int n,
                                   int naujas)
{
    int i;
    for (i = 0; (i < n) && (A[i] < naujas); i++)
        { }
    // Kitas variantas:
    //int i = 0;
    //while (i < n && A[i] < naujas)
    //    i++;
    return i;
}
```

0	1	2		k		n-1			Cn-1
-7	-3	1	4	6	10	25			



Nario išmetimas iš nerikiuoto ir rikiuoto masyvo

Nario šalinimas iš masyvo

Šalinant narį iš masyvo reikia žinoti šalinamojo nario vietą **k**.

Galimi atvejai:

- kai duomenys netvarkingi
- kai duomenys tvarkingi (surikiuoti)

Kai duomenys netvarkingi šalinamojo nario vietoje **k** galima įrašyti masyvo paskutiniojo (**n-1**) nario reikšmę.

Kai duomenys tvarkingi masyvo nariai esantys dešiniau šalinamojo nario **k** perstumiami į kairę (nesuardoma tvarka).

0	1	2		k		n-1			Cn-1
4	-3	6	10	25	-7	1			

Išmetamo nario indeksas: k

$A[k] = A[n - 1];$

$n = n - 1;$

0	1	2		k		$n-1$			C_{n-1}
4	-3	6	10	25	-7	1			

0	1	2		k	$n-1$				C_{n-1}
4	-3	6	10	1	-7				

Nario išmetimas iš **rikiuoto** masyvo

Išmetamo nario indeksas: **k**

```
for (int i = k; i < n - 1; i++)
```

```
    A[i] = A[i+1];
```

```
n = n - 1;
```

0	1	2		k		n-1			Cn-1
-7	-3	1	4	6	10	25			



0	1	2		k	n-1				Cn-1
-7	-3	1	4	10	25				

Narių išmetimas iš masyvo (1/2)

Išmetami nariai, **netenkinantys sąlygos**, pvz. **neigiami**:

```
int m = 0;
for (int i = 0; i < n; i++)
    if (A[i] >= 0)
        A[m++] = A[i];
n = m;
```

0	1	2				n-1			Cn-1
4	-3	6	10	25	-7	1			

0	1	2		n-1					Cn-1
4	6	10	25	1					

Narių išmetimas iš masyvo (2/2)

Išmetami nariai, **netenkinantys sąlygas**, pvz. **neigiami**:

```
int m = 0;
for (int i = 0; i < n; i++)
    if (A[i] >= 0)
        A[m++] = A[i];
n = m;
```

Išmetami nariai, **tenkinantys sąlygą**, pvz. **neigiami**:

```
for (int i = 0; i < n; i++)
    if (A[i] < 0) {
        for (int j = i; j < n - 1; j++)
            A[j] = A[j+1];
        n = n - 1;
        i = i - 1;
    }
```

Palyginkite šiuos
programos
fragmentus.
Kuris
efektyvesnis?

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

Daroma prielaida, kad kontrolinį darbą studentams buvo leista rašyti (gerinti įvertinimus) kiek norima kartų.

Reikia:

- suformuoti studentų rinkinį be studentų pasikartojimų, t.y., jeigu studentas perrašinėjo kontrolinį darbą keletą kartų, tai palikti geriausią įvertinimą.
- pašalinti iš rinkinio studentus, kurių pažymys (įvertinimas) mažesnis už **x**.

Jonaitis Jonas;	8;
Petraitis Petras;	7;
Antanaitis Antanas;	10;
Giedraitis Giedrius;	5;
Onaitytė Ona;	8;
Juozaitis Juozas;	4;
Ramunaitė Ramunė;	5;
Petraitis Petras;	6;
Giedraitis Giedrius;	6;
Juozaitis Juozas;	6;
Onaitytė Ona;	7;
Giedraitis Giedrius;	7;

```
class Studentas
```

```
{  
    private string pavVrd;    // studento pavardė ir vardas  
    private int pazym;       // pažymys (įvertinimas)  
    ...                      // konstruktorius ir sąsajos metodai  
}
```

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

```
// Konteinerinė klasė
```

```
class MasyvasStudentai
```

```
{  
    const int Cn = 500;      // studentų masyvo dydis  
    private Studentas [] Studentai; // studentų masyvas  
    private int kiek;        // studentų skaičius  
    ...                      // konstruktorius ir sąsajos metodai  
    ...                      // palyginimo (==) operatorius  
}
```

```
class MasyvasStudentai
```

```
{ ...
```

```
// Palieka masyve Studentai(kiek) studentus, su didesniais įvertinimais,
```

```
// jeigu tas pats studentas masyve įrašytas ne vieną kartą
```

```
public void SutrauktiVienodus()
```

```
{
```

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

```
        for (int j = i + 1; j < kiek; j++)
```

```
            if (Studentai[i] == Studentai[j])
```

```
            {
```

```
                if (Studentai[j].ImtiPazym() >
```

```
                    Studentai[i].ImtiPazym())
```

```
                    Studentai[i] = Studentai[j];
```

```
                    Studentai[j] = Studentai[kiek - 1];
```

```
                    kiek = kiek - 1;
```

```
                    j = j - 1;
```

```
            }
```

```
    }
```

```
    ...
```

```
}
```

Naudojamas
užklotas ==
operatorius

Perkeliamas
paskutinis
studentas


```
class MasyvasStudentai
```

```
{    ...
```

```
// Pašalina iš masyvo Studentai(kiek) studentus, kurių pažymiai mažesni už x
```

```
public void Šalinti(int x)
```

```
{
```

```
    int m = 0;
```

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

```
        if (Studentai[i].ImtiPazym() >= x)
```

```
            Studentai[m++] = Studentai[i];
```

```
    kiek = m;
```

```
}
```

```
    ...
```

```
}
```

```
static void Main(string[] args)
{
    MasyvasStudentai TestasMas = new MasyvasStudentai();
    SkaitytiStudKont(CFd1, TestasMas);
    SpausdintiStudKont(CFr, TestasMas, "Studentų sąrašas");

    TestasMas.SutrauktiVienodus();
    SpausdintiStudKont(CFr, TestasMas, "Studentų sąrašas (po)");

    TestasMas.Šalinti(7);
    if (TestasMas.ImtiKiek() > 0)
        SpausdintiStudKont(CFr, TestasMas, "Studentų sąrašas (po)");
    else
        Console.WriteLine("Studentų sąrašas tuščias!");
}
```

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
8	Petraitis Petras	6
9	Giedraitis Giedrius	6
10	Juozaitis Juozas	6
11	Onaitytė Ona	7
12	Giedraitis Giedrius	7

...

...

Studentų sąrašas (po)

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

Studentų sąrašas (po)

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

Šioje temoje susipažinote:

1. Kokie galimi veiksmai su objektais.
2. Kaip užklojami dviviečiai operatoriai.
3. Kaip naudojami užkloti operatoriai.
4. Kaip įterpiamas naujas narys į nerikiuotą/rikiuotą rinkinį.
5. Kaip išmetamas narys iš nerikiuoto/rikiuoto rinkinio.
6. Kaip sujungti panašius narius.



Klausimai?