

T02. Dinaminis masyvas (List)

2 ak. val.

Temos klausimai

1. Kolekcijos (Collections).
2. Dinaminis masyvas (List).
3. Dinaminio masyvo savybės ir metodai.
4. Veiksmų su dinaminiu masyvu pavyzdžiai.



Kolekcijos(ang. collections)

Klasių kolekcijos (1/4)

C# leidžia naudotis klasėmis, vadinamomis kolekcijomis (angl. *collections*).

Šios klasės dar vadinamos kolekcijos klasėmis (*collection classes*). Jos saugoja duomenų rinkinius.

Kiekvieną šios klasės objektą galima laikyti objektų kolekcija (analogija konteineriui).

Pvz.: kortų kolekcija žaidimuose, dainų kolekcija kompiuteryje, paveikslų kolekcija, tam tikros komandos žaidėjai ir pan.

Klasių kolekcijos (2/4)

Klasių kolekcijos leidžia saugoti įvairių egzistuojančių duomenų struktūrų elementus, nesirūpinant šių kolekcijų realizacija.

Tai leidžia pakartotinai naudoti šias klases. Šių klasių metodai yra pilnai patikrinti (ištestuoti), veikia greitai, minimizuoja atminties sąnaudas.

Tai leidžia programuotojams greičiau ir efektyviau rašyti programas.

Klasių kolekcijos (3/4)

Klasių kolekcijos (toliau kolekcijos) turi (teikia) efektyvius metodus, kurie organizuoja, saugo ir išrenka duomenis: nereikalauja žinių apie tai, kaip tie duomenys yra atmintyje saugomi.

Masyvai (**array**) taip pat saugoja duomenis (objektus). Tačiau juos naudojam tuomet, kai masyvo dydis yra **žinomas** ir programos darbo metu **nekinta** (fiksuotas). Tokius masyvus vadiname *fiksuoto ilgio masyvais*.

Kolekcijos leidžia lanksčiai dirbti su duomenimis (objektais), jų apimtis gali būti nežinoma ir **kisti dinamiškai**: didėti ar mažėti. Kolekcijas dar galima vadinti *dinaminiais masyvais*.

Klasių kolekcijos (4/4)

Norint naudotis šių klasių kolekcijomis reikia, kad programoje būtų aprašytos atitinkamos vardų sritys (**namespace**).

Pavyzdžiui, vardų sritis `System.Collections` turi kolekcijas, kurios gali saugoti nuorodas į objektus (pvz., jau žinoma klasė `ArrayList`).

Daugelis naujausių programų naudoja kolekcijas, kurios priklauso `System.Collections.Generic` vardų sričiai, kur yra aprašytos **bendrinės klasės** (angl. *Generic classes*).

Čia yra aprašyta bendrinė klasė `List<T>`,

kur `T` – rezervuota vieta, kuriant naują sąrašą ir nurodant duomenų elementų, kuriuos saugos kolekcija, tipą (panašiai kaip fiksuoto dydžio masyve nurodomas tipas).



Dinaminis masyvas (ang. List)

Dinaminio masyvo **List** <T> aprašas

Dinaminio masyvo **List** aprašas:

```
List<Tipas> Pavadinimas = new List<Tipas>();
```

Taip sukuriamas *tuščias* (Count = 0) dinaminis masyvas.

Pavyzdžiai:

```
List<int> skaiciai = new List<int>();
```

```
// arba
```

```
//List<int> skaiciai;
```

```
//skaiciai = new List<int>();
```

```
List<char> simboliai = new List<char>();
```

```
List<string> spalvos = new List<string>();
```

```
List<bool> Pozymiai = new List<bool>();
```

```
List<Studentas> studentai = new List<Studentas>();
```

Dinaminio masyvo **List** <T> aprašas

Dinaminio masyvo **List** aprašas:

List<**Tipas**> Pavadinimas = **new List**<**Tipas**>(Int32);
Taip sukuriamas *tuščias* (Count = 0) dinaminis masyvas, tačiau su nurodytu skliaustuose talpumu (Capacity =).

Pavyzdžiai:

```
List<int> Skaiciai = new List<int>(1000);  
    // arba  
    // List<int> Skaiciai;  
    // Skaiciai = new List<int>(1000);  
List<char> Simboliai = new List<char>(1000000);  
List<string> Spalvos = new List<string>(256);  
List<bool> Pozymiai = new List<bool>(10000);  
List<Studentas> Studentai = new List<Studentas>(CMax);
```

Dažniausiai naudojami dinaminio masyvo

List <T> metodai

Metodas arba savybė	Aprašas
Add	Prideda elementą List'o pabaigoje.
Capacity	Savybė, kuri parodo List'o dydį.
Clear	Pašalina visus List'o elementus.
Contains	Grąžina true, jei ieškoma reikšmė yra, priešingu atveju – false.
Count	Savybė, kuri grąžina List'o elementų skaičių.
IndexOf	Grąžina pirmos ieškomos reikšmės elemento indeksą.
Insert	Įterpia į List'ą reikšmę nurodytoje indeksu vietoje.
Remove	Pašalina pirmąją nurodytą sutiktą reikšmę.
RemoveAt	Pašalina nurodyto indekso elementą.
RemoveRange	Pašalina nurodytą skaičių elementų, nurodant pirmojo elemento indeksą.
Sort	List'ą surikiuoja.
TrimExcess	Capacity padaro lygų Count.

Dinaminio masyvo `List <T>` pavyzdžiai (1/11)

```
// Dinaminio masyvo užpildymas reikšmėmis ir
spausdinimas
for (int i = 1; i <= 10; i++)
    Skaiciai.Add(i);
for (int i = 0; i < Skaiciai.Count; i++)
    Console.WriteLine("{0, 3:d}", Skaiciai[i]);
Console.WriteLine();
foreach (int skaicius in Skaiciai)
    Console.WriteLine("{0, 3:d}", skaicius);
Console.WriteLine();
Console.WriteLine("Count: {0, 2:d}", Skaiciai.Count);
Console.WriteLine("Capacity: {0, 2:d}",
                    Skaiciai.Capacity);
```

1	2	3	4	5	6	7	8	9	10
1	2	3	4	5	6	7	8	9	10
Count: 10									
Capacity: 16									

Dinaminio masyvo **List** <T> pavyzdžiai (2/11)

```
// Fiksuoto dydžio masyvo reikšmių surašymas į List'ą  
int[] Pirminiai = new int[10] { 2, 3, 5, 7, 11, 13,  
17, 19, 23, 29 };
```

```
Skaiciai = new List<int>(Pirminiai);
```

```
foreach (int skaicius in Skaiciai)  
    Console.WriteLine("{0, 3:d}", skaicius);  
Console.WriteLine();  
Console.WriteLine("Count: {0, 2:d}", Skaiciai.Count);  
Console.WriteLine("Capacity: {0, 2:d}",  
                    Skaiciai.Capacity);
```

```
2  3  5  7 11 13 17 19 23 29  
Count: 10  
Capacity: 10
```

Dinaminio masyvo `List <T>` pavyzdžiai (3/11)

```
// skaičiaus paieška
int sk = 5; // galima įvesti ir dialogo būdu
if (Skaiciai.Contains(sk) == true)
    Console.WriteLine("Skaičius {0, 2:d} yra List'e.", sk);
else
    Console.WriteLine("Skaičiaus {0, 2:d} List'e nėra.", sk);
//-----
// skaičiaus (pirmo) indekso paieška
int index = Skaiciai.IndexOf(sk);
if (index > -1)
    Console.WriteLine("Skaičiaus {0, 2:d} indeksas {1, 2:d}.",
        sk, index);
else
    Console.WriteLine("Skaičiaus {0, 2:d} List'e nėra.", sk);
```

```
2  3  5  7 11 13 17 19 23 29
...
Skaičius  5 yra List'e.
Skaičiaus  5 indeksas  2.
```

Dinaminio masyvo `List <T>` pavyzdžiai (4/11)

```
// skaičiaus (paskutinio) indekso paieška
int sk = 5; // galima įvesti ir dialogo būdu
int index = Skaiciai.LastIndexOf(sk);
if (index > -1)
    Console.WriteLine("Skaičiaus {0, 2:d} indeksas {1, 2:d}.",
                      sk, index);
else
    Console.WriteLine("Skaičiaus {0, 2:d} List'e nėra.", sk);
```

Pastaba: šiuo atveju metodo `LastIndexOf()` grąžinama reikšmė sutampa su metodo `IndexOf()` grąžinama reikšme, nes skaičius 5 masyve yra tik vienas.

```
2  3  5  7 11 13 17 19 23 29
...
Skaičiaus  5 indeksas  2.
```

Dinaminio masyvo `List <T>` pavyzdžiai (5/11)

```
// Skaičiaus įterpimas nurodytoje vietoje (List'o ribose)
int index = Skaiciai.Count; // 0<= index <= Skaiciai.Count
int sk = 31;
```

```
Skaiciai.Insert(index, sk);
```

```
foreach (int skaicius in Skaiciai)
    Console.WriteLine("{0, 3:d}", skaicius);
Console.WriteLine();
Console.WriteLine("Count: {0, 2:d}", Skaiciai.Count);
```

```
2  3  5  7 11 13 17 19 23 29
...
2  3  5  7 11 13 17 19 23 29 31
Count: 11
```


Dinaminio masyvo `List <T>` pavyzdžiai (6/11)

```
// Skaičiaus (pirmo iš eilės sutikto) pašalinimas
```

```
int sk = 11;
```

```
Skaiciai.Remove(sk);
```

```
foreach (int skaicius in Skaiciai)
```

```
    Console.Write("{0, 3:d}", skaicius);
```

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

```
Console.WriteLine("Count: {0, 2:d}", Skaiciai.Count);
```

```
2  3  5  7 11 13 17 19 23 29
...
2  3  5  7 13 17 19 23 29
Count: 9
```

Dinaminio masyvo `List <T>` pavyzdžiai (7/11)

```
// Nurodyto indekso skaičiaus pašalinimas
int index = 3; // 0 <= index <= Skaiciai.Count

Skaiciai.RemoveAt(index);

foreach (int skaicius in Skaiciai)
    Console.Write("{0, 3:d}", skaicius);
Console.WriteLine();
Console.WriteLine("Count: {0, 2:d}", Skaiciai.Count);
```

```
2  3  5  7 11 13 17 19 23 29
...
2  3  5 11 13 17 19 23 29
Count: 9
```

Dinaminio masyvo `List <T>` pavyzdžiai (8/11)

```
// Nurodyto elementų skaičiaus, pradedant nurodytu indeksu,  
pašalinimas
```

```
int index = 1;    // 0 <= index <= Skaiciai.Count  
int kiek = 3;
```

```
Skaiciai.RemoveRange(index, kiek);
```

```
foreach (int skaicius in Skaiciai)  
    Console.WriteLine("{0, 3:d}", skaicius);  
Console.WriteLine();  
Console.WriteLine("Count: {0, 2:d}", Skaiciai.Count);
```

```
2  3  5  7 11 13 17 19 23 29  
...  
2 11 13 17 19 23 29  
Count:  7
```

Dinaminio masyvo `List <T>` pavyzdžiai (9/11)

```
// List'o apsuikimas
```

```
Skaiciai.Reverse();
```

```
foreach (int skaicius in Skaiciai)
```

```
    Console.WriteLine("{0, 3:d}", skaicius);
```

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

```
// List'o rikiavimas (didėjančia tvarka)
```

```
Skaiciai.Sort();
```

```
foreach (int skaicius in Skaiciai)
```

```
    Console.WriteLine("{0, 3:d}", skaicius);
```

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

2	3	5	7	11	13	17	19	23	29
...									
29	23	19	17	13	11	7	5	3	2
2	3	5	7	11	13	17	19	23	29

Dinaminio masyvo `List <T>` pavyzdžiai (10/11)

```
// List'o elementų reikšmių surašymas į masyvą
int[] skaičiaiA = skaičiai.ToArray();
```

```
for (int i = 0; i < skaičiaiA.Count(); i++)
    Console.WriteLine("{0, 3:d}", skaičiaiA[i]);
Console.WriteLine();
```

```
foreach (int skaicius in skaičiaiA)
    Console.WriteLine("{0, 3:d}", skaicius);
Console.WriteLine();
```

2	3	5	7	11	13	17	19	23	29
...									
2	3	5	7	11	13	17	19	23	29
2	3	5	7	11	13	17	19	23	29

Dinaminio masyvo `List <T>` pavyzdžiai (11/11)

```
// Didžiausio ir mažiausio skaičiaus paieška
Console.WriteLine("Didžiausia reikšmė: {0}", skaiciai.Max());
Console.WriteLine("Mažiausia reikšmė: {0}", skaiciai.Min());

// List'o išvalymas
skaiciai.Clear(); // Count = 0, o Capacity lieka nepakitęs
Console.WriteLine("Count: {0, 2:d}", skaiciai.Count);
Console.WriteLine("Capacity: {0, 2:d}", skaiciai.Capacity);
```

```
2  3  5  7 11 13 17 19 23 29
...
Didžiausia reikšmė: 29
Mažiausia reikšmė: 2
Count:  0
Capacity: 10
```

Dinaminio masyvo `List<T>` pavyzdžiai (1/6)

```
List<string> spalvos = new List<string>();
spalvos.Add("Balta");
spalvos.Add("Juoda");
spalvos.Add("Raudona");
spalvos.Add("Geltona");
spalvos.Add("Mėlyna");
spalvos.Add("Žalia");
foreach (string spalva in spalvos)
    Console.WriteLine("{0, -10}", spalva);
Console.WriteLine("Count: {0, 2:d}", spalvos.Count);
Console.WriteLine("Capacity: {0, 2:d}", spalvos.Capacity);
```

```
Balta
Juoda
Raudona
Geltona
Mėlyna
Žalia
Count:  6
Capacity: 8
```

Dinaminio masyvo `List <T>` pavyzdžiai (2/6)

```
// spalvos paieška
string sp = "Geltona"; // galima įvesti ir dialogo būdu
if (Spalvos.Contains(sp) == true)
    Console.WriteLine("Spalva {0, -10} yra List'e.", sp);
else
    Console.WriteLine("Spalva {0, -10} List'e nėra.", sp);
```

```
Balta
Juoda
Raudona
Geltona
Mėlyna
Žalia
...
Spalva Geltona    yra List'e.
```


Dinaminio masyvo **List** <T> pavyzdžiai (3/6)

```
// spalvos indeksų paieška
```

```
int index = Spalvos.IndexOf(sp);
```

```
if (index > -1)
```

```
    Console.WriteLine("Spalvos {0, -10} indeksas {1, 2:d}.",  
                        sp, index);
```

```
else
```

```
    Console.WriteLine("Spalvos {0, -10} List'e nėra.", sp);
```

Balta

Juoda

Raudona

Geltona

Mėlyna

Žalia

...

Spalvos Geltona indeksas 3.

Dinaminio masyvo **List** <T> pavyzdžiai (4/6)

```
// spalvos įterpimas nurodytoje vietoje (List'o ribose)
int index = 2;           // 0 <= index <= spalvos.Count
string sp = "Ruda";
```

```
spalvos.Insert(index, sp);
```

```
foreach (string spalva in spalvos)
    Console.WriteLine("{0, -10}", spalva);
Console.WriteLine();
```

```
Balta
Juoda
Raudona
Geltona
Mėlyna
Žalia
...
```

```
Balta
Juoda
Ruda
Raudona
Geltona
Mėlyna
Žalia
```

Dinaminio masyvo **List** <T> pavyzdžiai (5/6)

```
// spalvos pašalinimas
```

```
string sp = "Juoda";
```

```
Spalvos.Remove(sp);
```

```
foreach (string spalva in Spalvos)
```

```
    Console.WriteLine("{0, -10}", spalva);
```

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

Balta
Juoda
Raudona
Geltona
Mėlyna
Žalia
...

Balta
Raudona
Geltona
Mėlyna
Žalia

Dinaminio masyvo `List <T>` pavyzdžiai (6/6)

```
// List'o rikiavimas didėjančia(alfabeto) tvarka
Spalvos.Sort();
```

```
foreach (string spalva in Spalvos)
    Console.WriteLine("{0, -10}", spalva);
Console.WriteLine();
```

```
Balta
Juoda
Raudona
Geltona
Mėlyna
Žalia
...
```

```
Balta
Geltona
Juoda
Mėlyna
Raudona
Žalia
```



Veiksmų su dinaminiu objektų masyvu pavyzdžiai

Dinaminis objektų masyvas (1/23)

Tekstiniame faile yra duomenys apie studentų testo rezultatus:

Pavardė ir vardas, pažymys.

Skaityti duomenis į dinaminį objektų masyvą (List'ą) ir pademonstruoti veiksmus su šio masyvo elementais, naudojant kolekcijos List metodus.

Duomenų failo pavyzdys:

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

Dinaminis objektų masyvas (2/23)

```
class Studentas
{
    // Savybė: studento pavardė ir vardas
    public string PavVrd { get; set; }
    // Savybė: pažymys (įvertinimas)
    public int Pazym { get; set; }
    // Konstruktorius
    public Studentas(string pavv, int pazym)
    {
        PavVrd = pavv;
        this.Pazym = pazym;
    }
    ...
}
```

Dinaminis objektų masyvas (3/23)

```
class Studentas
{
    ...
    // užklotas metodas ToString()
    public override string ToString()
    {
        string eilute;
        eilute = string.Format("{0, -20}    {1, 2}",
                                PavVrd, Pazym);
        return eilute;
    }
    ...
}
```


Dinaminis objektų masyvas (4/23)

```
class Studentas
{
    ...
    // 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();
    }
    ...    // papildysime vėliau, esant poreikiui
}
```

Dinaminis objektų masyvas (5/23)

```
static List<Studentas> skaitytiStudList(string fv)
{
    // studentų objektų dinaminis masyvas
    List<Studentas> StudList = new List<Studentas>();
    using (StreamReader srautas = new StreamReader(fv,
                                                    Encoding.GetEncoding(1257)))
    {
        string eilute; // viena duomenų failo eilutė
        while ((eilute = srautas.ReadLine()) != null)
        {
            string[] eilDalis = eilute.Split(';');
            string pavVrd = eilDalis[0];
            int pazym = int.Parse(eilDalis[1]);
            Studentas studentas = new Studentas(pavVrd, pazym);
            StudList.Add(studentas);
        }
    }
    return StudList;
}
```

Sukuriamas
dinaminis objektų
masyvas

Naudojamas metodas Add()

Dinaminis objektų masyvas (6/23)

```
static void spausdintiStudList(string fv, List<Studentas> StudList,
                                string antraste)
{
    const string virsus =
        "-----\r\n"
        + " Nr.   Pavardė ir vardas      Pažymys \r\n"
        + "-----";

    // Vietoj Append galima naudoti ir CreateNew
    using (var fr = new StreamWriter(File.Open(fv, FileMode.Append),
                                                Encoding.GetEncoding(1257)))
    {
        fr.WriteLine("\n " + antraste);
        fr.WriteLine(virsus);
        for (int i = 0; i < StudList.Count; i++)
        {
            studentas stud = StudList[i];
            fr.WriteLine("{0, 3}   {1}", i + 1, stud);
        }
        fr.WriteLine("-----\n");
    }
}
```

Kreipinys į
dinaminio masyvo
elementą

Dinaminis objektų masyvas (7/23)

```
const string CFd = "..\\..\\Studentai.txt"; // duomenų failo vardas
const string CFr = "..\\..\\Rezultatai.txt"; // rezultatų failo vardas
static void Main(string[] args)
{
    // Studentų sąrašo sudarymas ir spausdinimas
    List<Studentas> StudList = SkaitytiStudList(CFd);
    SpausdintiStudList(CFr, StudList, "Studentų testo rezultatai");

    ... // Toliau bus papildoma
}
```

Studentų testo rezultatai		

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

Dinaminis objektų masyvas (8/23)

```
// Studento paieška
```

```
Console.WriteLine("Užrašykite studento (-ės) pavardę ir vardą: ");
```

```
string pavVrd = Console.ReadLine();
```

```
Studentas stud = new Studentas(pavVrd, -1);
```

```
int index = StudList.IndexOf(stud);
```

Kreipinys į
dinaminio masyvo
metodą **IndexOf()**

```
if (index > -1)
```

```
    Console.WriteLine("Studento (-ės) numeris: {0, 2:d}.", index+1);
```

```
else
```

```
    Console.WriteLine("Tokio studento nėra.");
```

```
Užrašykite studento (-ės) pavardę ir vardą:  
Antanaitis Antanas  
Studento (-ės) numeris: 3.
```

Dinaminis objektų masyvas (9/23)

```
// Studento paieška
```

```
Console.WriteLine("Užrašykite studento (-ės) pavardę ir vardą: ");
```

```
string pavVrd = Console.ReadLine();
```

```
Studentas stud = new Studentas(pavVrd, -1);
```

```
StudList.Remove(stud);
```

Kreipinys į
dinaminio masyvo
metodą **Remove()**

```
spausdintiStudList(CFr, StudList, "Sąrašas po pašalinimo");
```

Studentų testo rezultatai

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

Sąrašas po pašalinimo

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	Ramunaitė Ramunė	5

Dinaminis objektų masyvas (10/23)

```
// Nurodyto indekso studento šalinimas  
int index = 5; // 0 <= index <= StudList.Count
```

```
StudList.RemoveAt(index);
```

Kreipinys į dinaminio
masyvo metodą
RemoveAt()

```
SpausdintiStudList(CFr, StudList, "Sąrašas po pašalinimo");
```

Studentų testo rezultatai

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

Sąrašas po pašalinimo

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	Ramunaitė Ramunė	5

Dinaminis objektų masyvas (11/23)

Naudojama **Lambda**
išraiška

// studentų šalinimas

StudList.RemoveAll(item => item.Pazym < 6);

Kreipinys į dinaminio
masyvo metodą
RemoveAll()

SpausdintiStudList(CFr, StudList, "Sąrašas (po pašalinimo)");

Studentų testo rezultatai

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

Sąrašas (po pašalinimo)

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

Dinaminis objektų masyvas (12/23)

```
// Pašalina visus studentus, kurių pažymys mažesnis už riba
static void PasalintiVisus(List<Studentas> StudList, int riba)
{
    for (int i = 0; i < StudList.Count; i++)
        if (StudList[i].Pazym < riba){
            StudList.RemoveAt(i);
            i--;
        }
    ...
    PasalintiVisus(StudList, 6);
    SpausdintiStudList(CFr, StudList, "Sąrašas (po pašalinimo)");
}
```

Studentų testo rezultatai

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

Sąrašas (po pašalinimo)

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

Dinaminis objektų masyvas (13/23)

```
// Naujo studento įterpimas
```

```
Studentas stud = new Studentas("Gerutis Geras", 10);
```

```
int index = StudList.Count; // 0 <= index <= StudList.Count
```

```
StudList.Insert(index, stud);
```

```
SpausdintiStudList(CFr, StudList, "Sąrašas po įterpimo");
```

Studentų testo rezultatai

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

Sąrašas po įterpimo

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	Gerutis Geras	10

Dinaminis objektų masyvas (14/23)

Naudojama **Lambda**
išraiška

```
// kiekio skaičiavimas
```

```
int KiekNepazang = StudList.Count(item => item.Pazym < 5);
```

```
Console.WriteLine("Nepažangių studentų: {0, 2:d}", KiekNepazang);
```

Nepažangių studentų: 1

Studentų testo rezultatai

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

Dinaminis objektų masyvas (15/23)

```
// Apskaičiuoja, kiek yra studentų, kurių pažymys < riba
static int KiekNepazangiu(List<Studentas> StudList, int riba)
{
    int kiek = 0;
    foreach (Studentas stud in StudList)
    {
        if (stud.Pazym < riba)
            kiek++;
    }
    return kiek;
}
...
int KiekNepazang = KiekNepazangiu(StudList, 5);
Console.WriteLine("Nepažangių studentų: {0, 2:d}", KiekNepazang);
```

Nepažangių studentų: 1

Dinaminis objektų masyvas (16/23)

Naudojama **Lambda**
išraiška

Kreipinys į
dinaminio masyvo
metodą **Sum()**

```
// sumos skaičiavimas
```

```
int sumaPaz = StudList.Sum(elem => elem.Pazym);
```

```
Console.WriteLine("Pažymių suma: {0, 2:d}", sumaPaz);
```

Pažymių suma: 47

Studentų testo rezultatai

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

Dinaminis objektų masyvas (17/23)

Kreipinys į
dinaminio masyvo
metodą **Sum()**

Naudojama **Lambda**
išraiška

// vidurkio skaičiavimas

```
double vidurkis = StudList.Average(elem => elem.Pazym);
```

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

vidurkis: 6,71

Studentų testo rezultatai

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

Dinaminis objektų masyvas (18/23)

Naudojama **Lambda**
išraiška

Kreipinys į
dinaminio masyvo
metodą **Sum()**

```
// vidurkio skaičiavimas
int sumaPaz = StudList.Sum(item => item.Pazym);
double vidurkis;
if (StudList.Count > 0)
    vidurkis = (double)sumaPaz / StudList.Count;
else
    vidurkis = 0.0;
Console.WriteLine("vidurkis: {0, 7:f2}", vidurkis);
```

vidurkis: 6,71

Studentų testo rezultatai

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

Dinaminis objektų masyvas (19/23)

Naudojama **Lambda**
išraiška

Kreipiniai į dinaminio
masyvo metodus
Max() ir **Min()**

```
// Maksimumo ir minimumo ieškojimas
```

```
int maxPaz = StudList.Max(elem => elem.Pazym);
```

```
Console.WriteLine("Geriausias įvertinimas: {0, 2:d}", maxPaz);
```

```
int minPaz = StudList.Min(elem => elem.Pazym);
```

```
Console.WriteLine("Blogiausias įvertinimas: {0, 2:d}", minPaz);
```

```
Geriausias įvertinimas: 10
Blogiausias įvertinimas:  4
```

Studentų testo rezultatai

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

Dinaminis objektų masyvas (20/23)

/ Studentų dinaminio masyvo rikiavimui reikia modifikuoti klasės Studentas aprašą ir papildyti klasę metodu CompareTo(). */*

```
class Studentas : IComparable<Studentas>  
{
```

Modifikuotas klasės
aprašas

...

// Reikalinga rikiavimui (Sort metodui) pagal pavardes

```
public int CompareTo(Studentas kitas)
```

```
{
```

```
    int poz = String.Compare(this.PavVrd, kitas.PavVrd,  
                             StringComparison.CurrentCulture);
```

```
    if (poz > 0)  
        return 1;
```

```
    if (poz < 0)  
        return -1;
```

```
    else  
        return 0;
```

```
}
```

```
}
```

Dinaminis objektų masyvas (21/23)

```
// studentų sąrašo rikiavimas
```

```
StudList.Sort();           // rikiuoja pagal pavardes
```

```
SpausdintiStudList(CFr, StudList, "Surikiuotas sąrašas");
```

```
StudList.Reverse();        // sąrašą apsuks
```

```
SpausdintiStudList(CFr, StudList, "Apsuktas sąrašas");
```

Surikiuotas sąrašas

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

Apsuktas sąrašas

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

Dinaminis objektų masyvas (22/23)

/ Studentų dinaminio masyvo rikiavimui reikia modifikuoti klasės Studentas aprašą ir papildyti klasę metodu CompareTo(). */*

```
class Studentas : IComparable<Studentas>  
{
```

Modifikuotas klasės
aprašas

...

*// Reikia rikiavimui (Sort metodei) pagal pažymius
// ir pavardes*

```
public int CompareTo(Studentas kitas)  
{
```

```
    int poz = String.Compare(this.PavVrd, kitas.PavVrd,  
                             StringComparison.CurrentCulture);
```

```
    if ((this.Pazym < kitas.Pazym) ||  
        ((this.Pazym == kitas.Pazym) && (poz > 0)))  
        return 1;
```

```
    else  
        return -1;
```

```
}
```

```
}
```

Dinaminis objektų masyvas (23/23)

```
// studentų sąrašo rikiavimas
```

```
StudList.Sort();           // rikiuoja pagal pažymius ir pavardes
```

```
SpausdintiStudList(CFr, StudList, "Surikiuotas sąrašas");
```

```
StudList.Reverse();        // sąrašą apsuks
```

```
SpausdintiStudList(CFr, StudList, "Apsuktas sąrašas");
```

Surikiuotas sąrašas

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

Apsuktas sąrašas

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

Šioje temoje susipažinote:

1. Kolekcijomis (Collections).
2. Dinaminiu masyvu (List).
3. Dinaminio masyvo savybėmis bei metodais.
4. Veiksmais dinaminiam masyvui.



Klausimai?