ktu informatikos fakultetas 1922

T04. Žodynas, surikiuotas žodynas, aibė, surikiuotas sąrašas (dictionary, sorted dictionary, hash set, sorted list)

4 ak. val.

ktu informatikos fakultetas 1922

Temos klausimai

1. Žodyno klasė (`Dictionary(Tkey, Tvalue) Class (System.Collections.Generic)`).
2. Rikiuoto žodyno klasė (Sorted`Dictionary(Tkey, Tvalue) Class (System.Collections.Generic)`).
3. Surikiuoto sąrašo klasė (`SortedList Class(Tkey, TValue) (System.Collections.Generic)`).
4. Surikiuoto sąrašo klasė (`SortedList Class (System.Collections)`).
5. Aibės klasė (`HashSet(T) Class (System.Collections.Generic)`).

`IDictionary<Tkey, TValue>`
sąsaja 1/3

ktu informatikos fakultetas 1922

1. Ši sąsaja yra pagrindinė `System.Collections.Generic` klasėms, kurių elementai – raktas/reikšmė pora.
2. Kiekvienas raktas/reikšmė poros elementas yra saugomas `KeyValuePair<Tkey, TValue>` objekte.
3. Kiekvieno elemento raktas - unikalus. Priklausomai nuo tolimesnės realizacijos jis (ne)gali įgyti reikšmę `null`.
4. Elementų reikšmės nėra unikalios.

IDictionary<Tkey, TValue> sąsaja 2/3

5. Konteinerius galima peržiūrėti naudojant enumeratorius. Tam naudojamas `foreach` ciklas. Šio ciklo parametras – būtinai `KeyValuePair<Tkey, TValue> tipo.`
6. Nėra numatyta jokia konkreti elementų rikiavimo tvarka.
7. Kiekviena klasė, paveldinti šią sąsają, turi turėti savo lyginimo metodą `Compare()`.
8. Savo raktų palyginimo metodų kūrimui naudojamos `IEqualityComparer<T>` ir `IComparer<T>` sąsajos. Pirmoji jų leidžia tik tikrinti lygu/nelygu. Antroji naudojama, kai reikia raktus rikiuoti ir rūšiuoti.

IDictionary<Tkey, TValue>
sąsaja 3/3

9. Šią sąsają paveldi:
   - žodynas `Dictionary<Tkey, TValue>`;
   - rikiuotas žodynas `SortedDictionary<Tkey, TValue>`;
   - rikiuotas sąrašas `SortedList<Tkey, TValue>`.
10. Rečiau naudojama `System.Collections` sąsaja `IDictionary`. Kiekvienas raktas/reikšmė poros elementas yra saugomas `DictionaryEntry` objekte. Šią sąsają paveldi maišos lentelė `HashTable`.

ktu informatikos fakultetas 1922

Dictionary<Tkey, TValue> Class (System.Collections.Generic)

Žodynas `Dictionary<Tkey, TValue>` 1/2

Žodynas – asociatyvusis konteineris.
Išrinkimui naudojamas raktas.
Raktas – tai gali būti eilutė, skaičius. Gali būti ir savo sukurtos klasės objektas.
Duomenys saugomi medžio tipo struktūroje.
Greitas įterpimas, pašalinimas, paieška.

Žodynas `Dictionary<Tkey, TValue>` 2/2

Žodynas `Dictionary<Tkey, TValue>` turi:

- 7 užklotus konstruktorius.
- 5 savybes (properties).
- 15 metodų.
- 23 sąsajos realizacijas (Explicit Interface Implementations).
- Didelę aibę užklotų metodų (>130).

Žodyno `Dictionary<Tkey, TValue>` aprašas 1/2

Žodyno `Dictionary<Tkey, TValue>` aprašai:

1. `Dictionary<Tkey, TValue> Pavadinimas = new Dictionary<Tkey, TValue> ();`

Sukuriamas numatytos talpos ***tuščias*** (Count = 0) žodynas. Raktų palyginimui naudoja standartinį lyginimo metodą (equality comparer).

2. `Dictionary<Tkey, TValue> Pavadinimas1 = new Dictionary<Tkey, TValue> (Pavadinimas);`

Sukuriamas naujas žodynas **`Pavadinimas1`**, kuriame yra bet kokio kito tipo žodyno **`Pavadinimas`** elementai. Raktų palyginimui naudoja standartinį lyginimo metodą.

3. `Dictionary<Tkey, TValue> Pavadinimas2 = new Dictionary<Tkey, TValue> (IEqualityComparer objektas);`

Sukuriamas numatytos talpos tuščias žodynas. Naudojamas užklotas raktų palyginimo metodas.

Žodyno `Dictionary<Tkey, TValue>` aprašas 2/2

4. `Dictionary<Tkey, TValue> Pavadinimas3 = new Dictionary<Tkey, TValue> (Pavadinimas, IEqualityComparer objektas);`

Sukuriamas naujas žodynas **`Pavadinimas3`**, kuriame yra bet kokio kito tipo žodyno **`Pavadinimas`** elementai. Raktų palyginimui naudojamas užklotas raktų palyginimo metodas.

5. `Dictionary<Tkey, TValue> Pavadinimas4 = new Dictionary<Tkey, TValue> (talpa);`

Sukuriamas naujas tuščias nurodytos talpos žodynas **`Pavadinimas4.`** Raktų palyginimui naudoja standartinį lyginimo metodą.

6. `Dictionary<Tkey, TValue> Pavadinimas5 = new Dictionary<Tkey, TValue> (talpa, IEqualityComparer objektas);`

Sukuriamas nurodytos talpos tuščias žodynas. Naudojamas užklotas raktų palyginimo metodas.

ktu informatikos fakultetas 1922

Dažniausiai naudojami žodyno `Dictionary<Tkey, TValue>` metodai

Metodas arba savybė	Aprašas
Add(raktas, reikšmė)	Įterpia elementą į žodyną.
Clear()	Pašalina visus žodyno elementus.
ContainsKey(raktas)	Grąžina true, jei ieškomas raktas yra, priešingu atveju – false.
ContainsValue(reikšmė)	Grąžina true, jei ieškoma reikšmė yra, priešingu atveju – false.
Count	Savybė, kuri grąžina žodyno elementų skaičių.
Equals(objektas)	Grąžina true, jei nurodytas objektas lygus nagrinėjamam objektui, priešingu atveju – false. (Paveldėta iš Object)
GetEnumerator()	Grąžina žodyno enumeratorių.
Item[raktas]	Paima arba įdeda rakto apibrėžtą reikšmę.
Keys	Pateikia žodyno raktus.
Remove(raktas)	Išmeta iš žodyno nurodyto rakto elementą.
TryGetValue(raktas, reikšmė)	Grąžina nurodyto rakto reikšmę.
Values	Pateikia žodyno reikšmes.

Žodyno `Dictionary<Tkey, TValue>` pavyzdžiai (1/23)

```
// spausdina žodyno <string, int> reikšmes.
// Savybės Key ir Value
public static void Spausdinti(Dictionary<string, int> zodynas)
{
    foreach (KeyValuePair<string, int> pora in zodynas)
    {
        Console.WriteLine("     {0,-10}  -  {1}",
                pora.Key.ToString(), pora.Value.ToString());
    }
    Console.WriteLine();
}
```

Žodyno `Dictionary<Tkey, TValue>` pavyzdžiai (2/23)

```
// spausdina žodyno <string, Asmuo> reikšmes.
// Savybės Key ir Value
public static void Spausdinti(Dictionary<string, Asmuo> zodynas)
{
    foreach (KeyValuePair<string, Asmuo> pora in zodynas)
    {
        Console.WriteLine("     {0,-10}  -  {1}",
                pora.Key.ToString(), pora.Value.ToString());
    }
    Console.WriteLine();
}
```

Žodyno `Dictionary<Tkey, TValue>` pavyzdžiai (3/23)

```
// spausdina žodyno <Asmuo, string> reikšmes.
// Savybės Key ir Value
public static void Spausdinti(Dictionary<Asmuo, string> zodynas)
{
    foreach (KeyValuePair<Asmuo, string> pora in zodynas)
    {
        Console.WriteLine("     {0,-10}  -  {1}",
                pora.Key.ToString(), pora.Value.ToString());
    }
    Console.WriteLine();
}
```

Žodyno `Dictionary<Tkey, Tvalue>` pavyzdžiai (4/23)

```
// Sukuria ir inicializuoja žodyną
Dictionary<string, int> pirmas =
                  new Dictionary<string, int>();
pirmas.Add("vienas", 1);
pirmas.Add("du", 2);
pirmas.Add("trys", 3);
pirmas.Add("keturi", 4);
pirmas.Add("penki", 5);
Console.WriteLine("   Žodynas pirmas");
Console.WriteLine("Žodyno narių kiekis:    {0}", pirmas.Count);
Console.WriteLine("   Reikšmės:");
Spausdinti(pirmas);
```

Žodynas pirmas

vienas	1
du	2
trys	3
keturi	4
penki	5

```
   Žodynas pirmas
Žodyno narių kiekis:    5
   Reikšmės:
      vienas       -  1
      du           -  2
      trys         -  3
      keturi       -  4
      penki        -  5

Press any key to continue . . .
```

Žodyno `Dictionary<Tkey, Tvalue>` pavyzdžiai (5/23)

```
// Sukuria ir inicializuoja žodyną
Dictionary<string, int> antras =
          new Dictionary<string, int>(pirmas);

antras.Add("šeši", 6);
Console.WriteLine("   Žodynas antras");
Console.WriteLine("Žodyno narių kiekis:    {0}", antras.Count);
Console.WriteLine("   Reikšmės:");
Spausdinti(antras);
```

Žodynas antras

vienas	1
du	2
trys	3
keturi	4
penki	5
šeši	6

```
   Žodynas antras
Žodyno narių kiekis:    6
   Reikšmės:
      vienas        -  1
      du            -  2
      trys          -  3
      keturi        -  4
      penki         -  5
      šeši          -  6

Press any key to continue . . .
```

Žodyno `Dictionary<Tkey, TValue>` pavyzdžiai (6/23)

```
public class Asmuo
{
    private string vardas;
    private int amžius;
    public Asmuo(string vardas, int amžius) // Konstruktorius
    {
        this.vardas = vardas;
        this.amžius = amžius;
    }
    public override string ToString()
    {
        return this.vardas + "  " + this.amžius;
    }
    public override bool Equals(object objektas)
    {
        Asmuo stud = objektas as Asmuo;
        return stud.vardas == vardas && stud.amžius == amžius;
    }
    // Užklotas metodas GetHashCode()
    public override int GetHashCode()  {return base.GetHashCode();}
}
```

Žodyno `Dictionary<Tkey, Tvalue>` pavyzdžiai (7/23)

```
// Sukuria ir inicializuoja žodyną
Asmuo stud = new Asmuo("Jonas", 25);
Asmuo stud1 = new Asmuo("Petras", 26);
Asmuo stud2 = new Asmuo("Juozas", 38);
Dictionary<string, Asmuo> naujas =
           new Dictionary<string, Asmuo>();
naujas.Add("vienas", stud);
naujas.Add("du", stud1);
naujas.Add("trys", stud2);
Console.WriteLine("   Žodynas naujas");
Console.WriteLine("Žodyno narių kiekis:    {0}", naujas.Count);
Console.WriteLine("   Reikšmės:");
Spausdinti(naujas);
```

Žodynas naujas

vienas	Jonas 25
du	Petras 26
trys	Juozas 38

```
   Žodynas naujas
Žodyno narių kiekis:    3
   Reikšmės:
     vienas       -  Jonas  25
     du           -  Petras  26
     trys         -  Juozas  38

Press any key to continue . . .
```

Žodyno `Dictionary<Tkey, Tvalue>` pavyzdžiai (8/23)

```
// ContainsKey()
Console.WriteLine();
string raktas = "vienas";
if (naujas.ContainsKey(raktas))
    Console.WriteLine("rado raktą: {0}", raktas);
else Console.WriteLine("nerado rakto: {0}", raktas);
// rezultatai – kitoje skaidrėje
```

Žodynas naujas

vienas	Jonas 25
du	Petras 26
trys	Juozas 38

Žodyno `Dictionary<Tkey, TValue>` pavyzdžiai (9/23)

```
// ContainsValue()
Asmuo stud3 = new Asmuo("Juozas", 55);
if (naujas.ContainsValue(stud3))
    Console.WriteLine("rado reikšmę: {0}  {1}",
        stud3.vardas, stud3.amžius);
else Console.WriteLine("nerado reikšmės: {0}  {1}",
                       stud3.vardas, stud3.amžius);
if (naujas.ContainsValue(stud2))
    Console.WriteLine("rado reikšmę: {0}  {1}", stud2.vardas,
        stud2.amžius);
else Console.WriteLine("nerado reikšmės: {0}  {1}",
                       stud2.vardas, stud2.amžius);
```

```
rado raktą: vienas
nerado reikšmės: Juozas  55
rado reikšmę: Juozas  38

Press any key to continue . . .
```

Žodyno `Dictionary<Tkey, TValue>` pavyzdžiai (10/23)

```
// Žodynas, kurio raktas - objektas
Dictionary<Asmuo, string> naujas1 =
           new Dictionary<Asmuo, string>();
naujas1.Add(stud, "vienas");
naujas1.Add(stud1, "du");
naujas1.Add(stud2, "trys");
Console.WriteLine("   Žodynas naujas1");
Console.WriteLine("Žodyno narių kiekis:    {0}", naujas1.Count);
Console.WriteLine("   Reikšmės:");
Spausdinti(naujas1);
```

Žodynas naujas1

Jonas 25	vienas
Petras 26	du
Juozas 38	trys

```
   Žodynas naujas1
Žodyno narių kiekis:    3
   Reikšmės:
      Jonas  25   -  vienas
      Petras  26  -  du
      Juozas  38  -  trys

Press any key to continue . . .
```

Žodyno `Dictionary<Tkey, TValue>` pavyzdžiai (11/23)

```
// ContainsKey()
if (naujas1.ContainsKey(stud3))
    Console.WriteLine("rado raktą: {0}  {1}", stud3.vardas,
                      stud3.amžius);
else Console.WriteLine("nerado rakto: {0}  {1}",
    stud3.vardas, stud3.amžius);
if (naujas1.ContainsKey(stud2))
    Console.WriteLine("rado raktą: {0}  {1}", stud2.vardas,
                      stud2.amžius);
else Console.WriteLine("nerado rakto: {0}  {1}",
    stud2.vardas, stud2.amžius);
Console.WriteLine();
```

Žodynas naujas1

Jonas 25	vienas
Petras 26	du
Juozas 38	trys

```
nerado rakto: Juozas  55
rado raktą: Juozas  38

Press any key to continue . . .
```

Žodyno `Dictionary<Tkey, TValue>` pavyzdžiai (12/23)

```
// Nauja palyginimo klasė
class IComparer : IEqualityComparer<Asmuo>
{
    public bool Equals(Asmuo b1, Asmuo b2)
    {
        return b1.amžius == b2.amžius;
    }

    public int GetHashCode(Asmuo zmogus)
    {
        return base.GetHashCode();
    }
}
```

Žodyno `Dictionary<Tkey, TValue>` pavyzdžiai (13/23)

```
// Sukuria ir inicializuoja žodyną
IComparer kitas = new IComparer();
Dictionary<Asmuo, string> naujas2 =
           new Dictionary<Asmuo, string>(naujas1, kitas);
// Tikrina ir gali būti vykdymo klaida
Asmuo stud4 = new Asmuo("Rimas", 25);
// naujas2.Add(stud4, "penki");
// Duos klaidą dėl IComparer kitas. Klaida dėl 25
// naujas2.Add(stud1, "du");   // Duos klaidą dėl IComparer kitas
Console.WriteLine("   Žodynas naujas2");
Console.WriteLine("Žodyno narių kiekis:    {0}", naujas2.Count);
Console.WriteLine("   Reikšmės:");
Spausdinti(naujas2);
```

```
   Žodynas naujas2
Žodyno narių kiekis:    3
   Reikšmės:
      Jonas  25   -  vienas
      Petras  26  -  du
      Juozas  38  -  trys

Press any key to continue . . .
```

Žodynas naujas1

Jonas 25	vienas
Petras 26	du
Juozas 38	trys

Žodynas naujas2

Jonas 25	vienas
Petras 26	du
Juozas 38	trys

Žodyno `Dictionary<Tkey, TValue>` pavyzdžiai (14/23)

```
// Sukuria ir inicializuoja žodyną
IComparer kitas = new IComparer();
Dictionary<Asmuo, string> naujas3 =
            new Dictionary<Asmuo, string>(kitas);
Asmuo stud4 = new Asmuo("Rimas", 25);
naujas3.Add(stud4, "penki");
// naujas3.Add(stud, "du"); // Duos klaidą dėl IComparer kitas
naujas3.Add(stud2, "trys");
Console.WriteLine("   Žodynas naujas3");
Console.WriteLine("Žodyno narių kiekis:    {0}", naujas3.Count);
Console.WriteLine("   Reikšmės:");
Spausdinti(naujas3);
```

stud2

Juozas 38

Žodynas naujas3

Rimas 25	penki
Juozas 38	trys

```
   Žodynas naujas3
Žodyno narių kiekis:    2
   Reikšmės:
     Rimas  25   -  penki
     Juozas  38  -  trys

Press any key to continue . . .
```

Žodyno `Dictionary<Tkey, TValue>` pavyzdžiai (15/23)

```
// panaudojamas kitas standartinis Comparer
Dictionary<string, string> naujas4 =
          new Dictionary<string, string>(
              StringComparer.CurrentCultureIgnoreCase);
```

Žodyno `Dictionary<Tkey, TValue>` pavyzdžiai (16/23)

```
// Sukuria ir inicializuoja žodyną
Dictionary<Asmuo, string> naujas5 =
            new Dictionary<Asmuo, string>(10);
naujas5.Add(stud4, "penki");
naujas5.Add(stud, "du");
naujas5.Add(stud2, "trys");
Console.WriteLine("   Žodynas naujas5");
Console.WriteLine("Žodyno narių kiekis:    {0}", naujas5.Count);
Console.WriteLine("   Reikšmės:");
Spausdinti(naujas5);
```

Žodynas naujas5

Rimas 25	penki
Jonas 25	du
Juozas 38	trys

```
   Žodynas naujas5
Žodyno narių kiekis:    3
   Reikšmės:
     Rimas  25   -  penki
     Jonas  25   -  du
     Juozas  38  -  trys

Press any key to continue . . .
```

Žodyno `Dictionary<Tkey, TValue>` pavyzdžiai (17/23)

ktu informatikos fakultetas 1922

```
// Sukuria ir inicializuoja žodyną
Dictionary<Asmuo, string> naujas5 =
            new Dictionary<Asmuo, string>(10);
naujas5.Add(stud4, "penki");
naujas5.Add(stud, "du");
naujas5.Add(stud2, "trys");
Asmuo stud5 = new Asmuo("Rimas", 25);
naujas5.Add(stud5, "penki");
Console.WriteLine("   Žodynas naujas5");
Console.WriteLine("Žodyno narių kiekis:    {0}", naujas5.Count);
Console.WriteLine("   Reikšmės:");
Spausdinti(naujas5);
```

```
   Žodynas naujas5
Žodyno narių kiekis:    4
   Reikšmės:
      Rimas  25    -  penki
      Jonas  25    -  du
      Juozas  38   -  trys
      Rimas  25    -  penki

Press any key to continue . . .
```

Žodynas naujas5

Rimas 25	penki
Jonas 25	du
Juozas 38	trys
Rimas 25	penki

Kodėl klaida?

Žodyno `Dictionary<Tkey, TValue>` pavyzdžiai (18/23)

```
// Sukuria ir inicializuoja žodyną
Dictionary<Asmuo, string> naujas6 =
            new Dictionary<Asmuo, string>(10, kitas);
naujas6.Add(stud4, "penki");
// naujas6.Add(stud, "du");   // Būtų klaida
naujas6.Add(stud2, "trys");
Console.WriteLine("   Žodynas naujas6");
Console.WriteLine("Žodyno narių kiekis:    {0}", naujas6.Count);
Console.WriteLine("   Reikšmės:");
Spausdinti(naujas6);
```

Žodynas naujas6

Rimas 25	penki
Juozas 38	trys

```
   Žodynas naujas6
Žodyno narių kiekis:    2
   Reikšmės:
     Rimas  25   -  penki
     Juozas  38  -  trys

Press any key to continue . . .
```

Žodyno `Dictionary<Tkey, TValue>` pavyzdžiai (19/23)

```
// GetEnumerator() naudojamas skaityti, bet ne modifikuoti.
// Geriau ciklas foreach
var enumerator = naujas6.GetEnumerator();
Console.WriteLine("Enumeratoriaus pavyzdys");
Console.WriteLine("   Žodynas naujas6 ");
while (enumerator.MoveNext())
{
    object item = enumerator.Current;
    Console.WriteLine("    {0}", item);
}
Console.WriteLine();
```

Nuoroda prieš pradinį žodyno elementą. Prieš darbą būtinai pereiti prie sekančio elemento.

```
Enumeratoriaus pavyzdys
   Žodynas naujas6
    [Rimas  25, penki]
    [Juozas  38, trys]

Press any key to continue . . .
```

Žodyno `Dictionary<Tkey, TValue>` pavyzdžiai (20/23)

```
// Remove(Key)
naujas6.Remove(stud2);
var enumerator = naujas6.GetEnumerator();
Console.WriteLine("   Žodynas naujas6 po išmetimo ");
while (enumerator.MoveNext())
{
    object item = enumerator.Current;
    Console.WriteLine("    {0}", item);
}
Console.WriteLine();
```

Žodynas naujas6 prieš išmetimą

Rimas 25	penki
Juozas 38	trys

```
Žodynas naujas6 po išmetimo
 [Rimas  25, penki]

Press any key to continue . . .
```

Žodyno `Dictionary<Tkey, TValue>` pavyzdžiai (21/23)

```
// TryGetValue()
string reiksme = "";
if (naujas5.TryGetValue(stud, out reiksme))
   Console.WriteLine("Raktui {0} {1} rasta reikšmė  {2}.",
                     stud.vardas, stud.amžius, reiksme);
else  Console.WriteLine("Raktui {0} {} nerasta reikšmė.",
                        stud.vardas, stud.amžius);

if (naujas5.TryGetValue(stud3, out reiksme))
   Console.WriteLine("Raktui {0} {1} rasta reikšmė  {2}.",
                     stud3.vardas, stud3.amžius, reiksme);
else  Console.WriteLine("Raktui {0} {1} nerasta reikšmė.",
                        stud3.vardas, stud3.amžius);
```

Žodynas naujas5

Rimas 25	penki
Jonas 25	du
Juozas 38	trys

```
Raktui Jonas 25 rasta reikšmė  du.
Raktui Juozas 55 nerasta reikšmė.
Press any key to continue . . .
```

Žodyno `Dictionary<Tkey, TValue>` pavyzdžiai (22/23)

```
// Item[]
Console.WriteLine("Raktas = {0} {1}", stud.vardas, stud.amžius);
Console.WriteLine("Žodyno elementas prieš pakeitimą {0}.",
naujas5[stud]);
naujas5[stud] = "septyni";
Console.WriteLine("Žodyno elementas po pakeitimo {0}.",
naujas5[stud]);
```

Žodynas naujas5

Rimas 25	penki
Jonas 25	du
Juozas 38	trys

```
Raktas = Jonas 25
Žodyno elementas prieš pakeitimą du.
Žodyno elementas po pakeitimo septyni.
Press any key to continue . . .
```

Žodyno `Dictionary<Tkey, TValue>` pavyzdžiai (23/23)

```
var didz = pirmas.Max(elem => elem.Key);
Console.WriteLine("Max raktas: {0}", didz);
var didz1 = pirmas.Max(elem => elem.Value);
Console.WriteLine("Max reikšmė: {0}", didz1);
var didz2 = pirmas.Average(elem => elem.Value);
Console.WriteLine("Reikšmių vidurkis: {0}", didz2);
```

Žodynas pirmas

vienas	1
du	2
trys	3
keturi	4
penki	5

```
Max raktas: vienas
Max reikšmė: 5
Reikšmių vidurkis: 3
Press any key to continue . . .
```

ktu informatikos fakultetas 1922

SortedDictionary<Tkey, TValue> Class (System.Collections.Generic)

Rikiuotas žodynas `SortedDictionary<Tkey, TValue>` 1/2

Rikiuotas žodynas – asociatyvusis konteineris.
Išrinkimui naudojamas raktas.
Raktas – tai gali būti eilutė, skaičius, savo sukurtos klasės objektas.
Duomenys saugomi medžio tipo struktūroje.
Daugumoje atvejų dirba lėčiau nei paprastas žodynas.

Rikiuotas žodynas `SortedDictionary<Tkey, TValue>` 2/2

Rikiuotas žodynas `SortedDictionary<Tkey, TValue>` turi:

- 4 užklotus konstruktorius.
- 5 savybes (properties).
- 14 metodų.
- 22 sąsajos realizacijas (Explicit Interface Implementations).
- Didelę aibę užklotų metodų (>130).

ktu informatikos fakultetas 1922

Rikiuoto žodyno `SortedDictionary<Tkey, TValue>` aprašas 1/2

Rikiuoto žodyno `SortedDictionary<Tkey, TValue>` aprašai:

1. `SortedDictionary<Tkey, TValue> Pavadinimas = new SortedDictionary<Tkey, TValue> ();`

Sukuriamas numatytos talpos ***tuščias*** (Count = 0) rikiuotas žodynas. Raktų palyginimui naudoja standartinį lyginimo metodą (**comparer, bet ne equality comparer,** kaip paprastas žodynas).

2. `SortedDictionary<Tkey, TValue> Pavadinimas1 = new SortedDictionary<Tkey, TValue> (Pavadinimas);`

Sukuriamas naujas rikiuotas žodynas **`Pavadinimas1`**, kuriame yra bet kokio kito tipo žodyno **`Pavadinimas`** elementai. Raktų palyginimui naudoja standartinį lyginimo metodą.

Rikiuoto žodyno `SortedDictionary<Tkey, TValue>` aprašas 2/2

3. `SortedDictionary<Tkey, TValue> Pavadinimas2 = new SortedDictionary<Tkey, TValue> (IComparer objektas);`

Sukuriamas numatytos talpos tuščias rikiuotas žodynas. Naudojama užklotą raktų palyginimo metodą.

4. `SortedDictionary<Tkey, TValue> Pavadinimas3 = new SortedDictionary<Tkey, TValue> (Pavadinimas, IComparer objektas);`

Sukuriamas naujas rikiuotas žodynas **`Pavadinimas3`**, kuriame yra bet kokio kito žodyno **`Pavadinimas`** elementai. Raktų palyginimui naudojamas užklotas raktų palyginimo metodas.

ktu informatikos fakultetas 1922

Dažniausiai naudojami rikiuoto žodyno `SortedDictionary<Tkey, TValue>` metodai

Metodas arba savybė	Aprašas
Add(raktas, reikšmė)	Įterpia elementą į žodyną.
Clear()	Pašalina visus žodyno elementus.
ContainsKey(raktas)	Grąžina true, jei ieškomas raktas yra, priešingu atveju – false.
ContainsValue(reikšmė)	Grąžina true, jei ieškoma reikšmė yra, priešingu atveju – false.
CopyTo(Masyvas, indeksas)	Kopijuoja žodyno raktų ir reikšmių poras į reikiamos struktūros masyvą, pradedant nurodytu masyvo indeksu.
Count	Savybė, kuri grąžina žodyno elementų skaičių.
Equals(objektas)	Grąžina true, jei nurodytas objektas lygus nagrinėjamam objektui, priešingu atveju – false. (Paveldėta iš Object)
GetEnumerator()	Grąžina žodyno enumeratorių.
Item[raktas]	Paima arba įdeda rakto apibrėžtą reikšmę.
Keys	Pateikia žodyno raktus.
Remove(raktas)	Išmeta iš žodyno nurodyto rakto elementą.
TryGetValue(raktas, reikšmė)	Grąžina nurodyto rakto reikšmę.
Values	Pateikia žodyno reikšmes.

ktu informatikos fakultetas 1922

Rikiuoto žodyno SortedDictionary<Tkey, TValue> pavyzdžiai (1/17)

```
// spausdina rikiuoto žodyno <string, int> reikšmes.
// Savybės Key ir Value
public static void Spausdinti(SortedDictionary<string, int>zod)
{
    foreach (KeyValuePair<string, int> pora in zod)
    {
        Console.WriteLine("     {0,-10}  -  {1}",
                pora.Key.ToString(), pora.Value.ToString());
    }
    Console.WriteLine();
}
```

ktu informatikos fakultetas 1922

Rikiuoto žodyno SortedDictionary<Tkey, TValue> pavyzdžiai (2/17)

```
// spausdina rikiuoto žodyno <string, Asmuo> reikšmes.
// Savybės Key ir Value
public static void Spausdinti(SortedDictionary<string, Asmuo>zod)
{
    foreach (KeyValuePair<string, Asmuo> pora in zod)
    {
        Console.WriteLine("     {0,-10}  -  {1}",
                pora.Key.ToString(), pora.Value.ToString());
    }
    Console.WriteLine();
}
```

Rikiuoto žodyno SortedDictionary<Tkey, TValue> pavyzdžiai (3/17)

```
// spausdina rikiuoto žodyno <Asmuo, string> reikšmes.
// Savybės Key ir Value
public static void Spausdinti(SortedDictionary<Asmuo, string>zod)
{
    foreach (KeyValuePair<Asmuo, string> pora in zod)
    {
        Console.WriteLine("     {0,-10}  -  {1}",
                pora.Key.ToString(), pora.Value.ToString());
    }
    Console.WriteLine();
}
```

Rikiuoto žodyno SortedDictionary<Tkey, TValue> pavyzdžiai (4/17)

```
// spausdina masyvo, į kurį perrašytas rikiuotas žodynas, reikšmes.
// kiek - masyvo elementų kiekis
public static void Spausdinti(KeyValuePair<string, Asmuo> [] masyvas, int kiek)
{
    int i = 0;
    foreach (KeyValuePair<string, Asmuo> pora in masyvas)
    {
        if (i < kiek)
        {
            Console.WriteLine("     {0,-10}  -  {1}", pora.Key.ToString(),
                              pora.Value.ToString());
            i++;
        }
    }
    Console.WriteLine();
}
```

Rikiuoto žodyno SortedDictionary<Tkey, Tvalue> pavyzdžiai (5/17)

```
// Sukuria ir inicializuoja žodyną
SortedDictionary<string, int> pirmas =
                    new SortedDictionary<string, int>();
pirmas.Add("vienas", 1);
pirmas.Add("du", 2);
pirmas.Add("trys", 3);
pirmas.Add("keturi", 4);
pirmas.Add("penki", 5);
Console.WriteLine("   Žodynas pirmas");
Console.WriteLine("Žodyno narių kiekis:    {0}", pirmas.Count);
Console.WriteLine("   Reikšmės:");
Spausdinti(pirmas);
```

Žodynas pirmas

vienas	1
du	2
trys	3
keturi	4
penki	5

Palyginkite reikšmių užrašymo eiliškumą.

```
   Žodynas pirmas
Žodyno narių kiekis:    5
   Reikšmės:
      du          -  2
      keturi      -  4
      penki       -  5
      trys        -  3
      vienas      -  1

Press any key to continue . . .
```

Rikiuoto žodyno SortedDictionary<Tkey, TValue> pavyzdžiai (6/17)

```
// Sukuria ir inicializuoja žodyną
SortedDictionary<string, int> antras =
            new SortedDictionary<string, int>(pirmas);
antras.Add("šeši", 6);
Console.WriteLine("   Žodynas antras");
Console.WriteLine("Žodyno narių kiekis:    {0}", antras.Count);
Console.WriteLine("   Reikšmės:");
Spausdinti(antras);
```

Žodynas antras

vienas	1
du	2
trys	3
keturi	4
penki	5
šeši	6

```
   Žodynas antras
Žodyno narių kiekis:    6
   Reikšmės:
      du          -  2
      keturi      -  4
      penki       -  5
      šeši        -  6
      trys        -  3
      vienas      -  1

Press any key to continue . . .
```

Rikiuoto žodyno SortedDictionary<Tkey, Tvalue> pavyzdžiai (7/17)

```
public class Asmuo : IComparable<Asmuo>
{
    private string vardas;
    private int amžius;
    public Asmuo(string vardas, int amžius) // Konstruktorius
    {
        this.vardas = vardas;
        this.amžius = amžius;
    }
    public override string ToString()
     return this.vardas + "  " + this.amžius;
    }
    public override bool Equals(object objektas)
    {
        Asmuo stud = objektas as Asmuo;
        return stud.vardas == vardas && stud.amžius == amžius;
    }
// Tęsinys – kitoje skaidrėje
}
```

Paveldėjimas, reikalingas parašyti savo rakto lyginimo metodą.

Rikiuoto žodyno SortedDictionary<Tkey, Tvalue> pavyzdžiai (8/17)

```
public class Asmuo : IComparable<Asmuo>
{
    // Tęsinys...
    public override bool Equals(object objektas)
    {
        Asmuo stud = objektas as Asmuo;
        return stud.vardas == vardas && stud.amžius == amžius;
    }
    public int CompareTo(Asmuo kitas)
    {
        int poz = String.Compare(this.vardas, kitas.vardas,
                                 StringComparison.CurrentCulture);
        if (poz > 0) return 1;
        if (poz < 0) return -1;
        else
            if (this.amžius > kitas.amžius) return 1;
            else if (this.amžius < kitas.amžius) return -1;
                 else return 0;
    }
}
```

Užklotas metodas (būtinai CompareTo), reikalingas parašyti savo rakto lyginimo metodą.

Rikiuoto žodyno SortedDictionary<Tkey, Tvalue> pavyzdžiai (9/17)

```
// Sukuria ir inicializuoja žodyną
Asmuo stud = new Asmuo("Jonas", 25);
Asmuo stud1 = new Asmuo("Petras", 26);
Asmuo stud2 = new Asmuo("Juozas", 38);
SortedDictionary<string, Asmuo> naujas =
            new SortedDictionary<string, Asmuo>();
naujas.Add("vienas", stud);
naujas.Add("du", stud1);
naujas.Add("trys", stud2);
Console.WriteLine("   Žodynas naujas");
Console.WriteLine("Žodyno narių kiekis:    {0}", naujas.Count);
Console.WriteLine("   Reikšmės:");
Spausdinti(naujas);
```

Žodynas naujas

vienas	Jonas 25
du	Petras 26
trys	Juozas 38

```
   Žodynas naujas
Žodyno narių kiekis:    3
   Reikšmės:
      du          -  Petras  26
      trys        -  Juozas  38
      vienas      -  Jonas  25

Press any key to continue . . .
```

Rikiuoto žodyno SortedDictionary<Tkey, TValue> pavyzdžiai (10/17)

```
// Žodynas, kurio raktas - objektas
SortedDictionary<Asmuo, string> naujas1 =
            new SortedDictionary<Asmuo, string>();
naujas1.Add(stud, "vienas");
naujas1.Add(stud1, "du");
naujas1.Add(stud2, "trys");
Console.WriteLine("   Žodynas naujas1");
Console.WriteLine("Žodyno narių kiekis:    {0}", naujas1.Count);
Console.WriteLine("   Reikšmės:");
Spausdinti(naujas1);
```

Žodynas naujas1

Jonas 25	vienas
Petras 26	du
Juozas 38	trys

```
   Žodynas naujas1
Žodyno narių kiekis:    3
   Reikšmės:
      Jonas  25   -  vienas
      Juozas  38  -  trys
      Petras  26  -  du

Press any key to continue . . .
```

Rikiuoto žodyno `SortedDictionary<Tkey, TValue>` pavyzdžiai (11/17)

```
// Nauja palyginimo klasė
    public class IComparer : IComparer<Asmuo>
    {
        public bool Equals(Asmuo b1, Asmuo b2)
        {
            return b1.amžius == b2.amžius;
        }
        public int Compare(Asmuo kitas, Asmuo kitas1)
        {
            if (kitas.amžius > kitas1.amžius)
                return 1;
            if (kitas.amžius < kitas1.amžius)
                return -1;
            else return 0;
        }
        public int GetHashCode(Asmuo zmogus)
        {
            return base.GetHashCode();
        }
    }
```

Kita klasė, nei paprasto žodyno atveju.

Paprastam žodynui šio metodo nereikėjo.

Rikiuoto žodyno SortedDictionary<Tkey, TValue> pavyzdžiai (12/17)

```
// Sukuria ir inicializuoja žodyną
IComparer kitas = new IComparer();
SortedDictionary<Asmuo, string> naujas2 =
            new SortedDictionary<Asmuo, string>(naujas1, kitas);
// Tikrina ir gali būti vykdymo klaida
Asmuo stud4 = new Asmuo("Rimas", 25);
// naujas2.Add(stud4, "penki");
// Duos klaidą dėl IComparer kitas. Klaida dėl 25
// naujas2.Add(stud1, "du");   // Duos klaidą dėl IComparer kitas
Console.WriteLine("   Žodynas naujas2");
Console.WriteLine("Žodyno narių kiekis:    {0}", naujas2.Count);
Console.WriteLine("   Reikšmės:");
Spausdinti(naujas2);
```

```
   Žodynas naujas2
Žodyno narių kiekis:    3
   Reikšmės:
      Jonas  25   -  vienas
      Petras  26  -  du
      Juozas  38  -  trys

Press any key to continue . . .
```

Žodynas naujas1

Jonas 25	vienas
Petras 26	du
Juozas 38	trys

Žodynas naujas2

Jonas 25	vienas
Petras 26	du
Juozas 38	trys

Rikiuoto žodyno SortedDictionary<Tkey, TValue> pavyzdžiai (13/17)

```
// Sukuria ir inicializuoja žodyną
IComparer kitas = new IComparer();
SortedDictionary<Asmuo, string> naujas3 =
            new SortedDictionary<Asmuo, string>(kitas);
Asmuo stud4 = new Asmuo("Rimas", 25);
naujas3.Add(stud4, "penki");
// naujas3.Add(stud, "du"); // Duos klaidą dėl IComparer kitas
naujas3.Add(stud2, "trys");
Console.WriteLine("   Žodynas naujas3");
Console.WriteLine("Žodyno narių kiekis:    {0}", naujas3.Count);
Console.WriteLine("   Reikšmės:");
Spausdinti(naujas3);
```

stud2

Juozas 38

Žodynas naujas3

Rimas 25	penki
Juozas 38	trys

```
   Žodynas naujas3
Žodyno narių kiekis:    2
   Reikšmės:
     Rimas  25   -  penki
     Juozas  38  -  trys

Press any key to continue . . .
```

ktu informatikos fakultetas 1922

Rikiuoto žodyno `SortedDictionary<Tkey, TValue>` pavyzdžiai (14/17)

```
// panaudojamas kitas standartinis Comparer
SortedDictionary<string, string> naujas4 =
          new SortedDictionary<string, string>(
              StringComparer.CurrentCultureIgnoreCase);
```

Rikiuoto žodyno SortedDictionary<Tkey, TValue> pavyzdžiai (15/17)

```
// CopyTo()
Console.WriteLine("   Žodynas naujas:");
Spausdinti(naujas);
KeyValuePair<string, Asmuo>[] Masyvas =
              new KeyValuePair<string, Asmuo>[naujas.Count];
naujas.CopyTo(Masyvas, 0);
Console.WriteLine("   Žodynas naujas perrašytas į masyvą:");
Spausdinti(Masyvas, Masyvas.Count());
```

```
   Žodynas naujas:
     du          -  Petras  26
     trys        -  Juozas  38
     vienas      -  Jonas  25

   Žodynas naujas perrašytas į masyvą:
     du          -  Petras  26
     trys        -  Juozas  38
     vienas      -  Jonas  25

Press any key to continue . . .
```

Rikiuoto žodyno SortedDictionary<Tkey, TValue> pavyzdžiai (16/17)

```
// paprasto žodyno sukūrimas iš rikiuoto
Console.WriteLine("   Žodynas naujas");
Console.WriteLine("   Reikšmės:");
Spausdinti(naujas);
Dictionary<string, Asmuo> naujas7 =
            new Dictionary<string, Asmuo>(naujas);
//naujas7.Add("vienas", stud);   // kartojasi, būtų klaida
// rikiuoto žodyno sukūrimas iš paprasto
SortedDictionary<string, Asmuo> naujas8 =
new SortedDictionary<string, Asmuo>(naujas7);
Console.WriteLine("   Žodynas naujas8");
Spausdinti(naujas8);
```

```
   Žodynas naujas
   Reikšmės:
     du             -  Petras  26
     trys           -  Juozas  38
     vienas         -  Jonas  25

   Žodynas naujas8
   Reikšmės:
     du             -  Petras  26
     trys           -  Juozas  38
     vienas         -  Jonas  25

Press any key to continue . . .
```

ktu informatikos fakultetas 1922

Rikiuoto žodyno `SortedDictionary<Tkey, TValue>` pavyzdžiai (17/17)

Kitų metodų naudojimas nesiskiria nuo jų panaudojimo paprastam žodynui.

SortedList<Tkey, TValue> Class (System.Collections.Generic)

Rikiuotas sąrašas
`SortedList<Tkey, TValue>`
1/3

Rikiuotas sąrašas – asociatyvusis konteineris.
Išrinkimui naudojamas raktas.
Raktas – tai gali būti eilutė, skaičius, savo sukurtos klasės objektas.
Duomenys saugomi vidiniame masyve.
Greitas įterpimas, pašalinimas, paieška.

Rikiuotas sąrašas `SortedList<Tkey, TValue>` 2/3

Rikiuotas sąrašas `SortedList<Tkey, TValue>` turi:

- 6 užklotus konstruktorius.
- 6 savybes (properties).
- 17 metodų.
- 23 sąsajos realizacijas (Explicit Interface Implementations).
- Didelę aibę užklotų metodų (>130).

ktu informatikos fakultetas 1922

Rikiuotas sąrašas `SortedList<Tkey, TValue>` 3/3

Rikiuoto sąrašo `SortedList<Tkey, TValue>` ir rikiuoto žodyno `SortedDictionary<Tkey, TValue>` palyginimas:

- sąrašas naudoja mažiau atminties;
- žodynas greičiau atlieka įterpimo ir išmetimo operacijas nerikiuotiem duomenims;
- jei visas sąrašas sukuriamas vienu kartu, jis yra greitesnis;
- sąrašas – rikiuotas masyvas, žodynas – medis.
- sąrašas elementų radimui naudojama dvejetainė paieška.

ktu informatikos fakultetas 1922

Rikiuoto sąrašo SortedList<Tkey, TValue> aprašas 1/2

Rikiuoto sąrašo SortedList<Tkey, TValue> aprašai:

1. `SortedList<Tkey, TValue> Pavadinimas = new SortedList<Tkey, TValue> ();`

Sukuriamas numatytos talpos ***tuščias*** (Count = 0) rikiuotas sąrašas. Raktų palyginimui naudoja standartinį lyginimo metodą (comparer).

2. `SortedList<Tkey, TValue> Pavadinimas1 = new SortedList<Tkey, TValue> (Pavadinimas);`

Sukuriamas naujas rikiuotas sąrašas **Pavadinimas1**, kuriame yra bet kokio kito tipo žodyno **Pavadinimas** elementai. Naudojamas standartinis raktų lyginimo metodas.

3. `SortedList<Tkey, TValue> Pavadinimas2 = new SortedList<Tkey, TValue> (IComparer objektas);`

Sukuriamas numatytos talpos tuščias žodynas. Naudojamas užklotas raktų palyginimo metodas.

ktu informatikos fakultetas 1922

Rikiuoto sąrašo SortedList<Tkey, TValue> aprašas 2/2

4. `SortedList<Tkey, TValue> Pavadinimas3 = new SortedList<Tkey, TValue> (Pavadinimas, IComparer objektas);`

Sukuriamas naujas rikiuotas sąrašas **Pavadinimas3**, kuriame yra bet kokio kito tipo žodyno **Pavadinimas** elementai. Raktų palyginimui naudojamas užklotas raktų palyginimo metodas.

5. `SortedList, TValue> Pavadinimas4 = new SortedList<Tkey, TValue> (talpa);`

Sukuriamas naujas tuščias nurodytos talpos rikiuotas sąrašas **Pavadinimas4.** Raktų palyginimui naudoja standartinį lyginimo metodą.

6. `SortedList<Tkey, TValue> Pavadinimas5 = new SortedList<Tkey, TValue> (talpa, IComparer objektas);`

Sukuriamas nurodytos talpos tuščias rikiuotas sąrašas. Naudojamas užklotas raktų palyginimo metodas.

Dažniausiai naudojami rikiuoto sąrašo `SortedList<Tkey, TValue>` metodai 1/2

Metodas arba savybė	Aprašas
Add(raktas, reikšmė)	Įterpia elementą į sąrašą.
Capacity	Savybė, kuri nustato arba grąžina sąrašo talpą.
Clear()	Pašalina visus sąrašo elementus.
ContainsKey(raktas)	Grąžina true, jei ieškomas raktas yra, priešingu atveju – false.
ContainsValue(reikšmė)	Grąžina true, jei ieškoma reikšmė yra, priešingu atveju – false.
Count	Savybė, kuri grąžina sąrašo elementų skaičių.
Equals(objektas)	Grąžina true, jei nurodytas objektas lygus nagrinėjamam objektui, priešingu atveju – false. (Paveldėta iš Object)
GetEnumerator()	Grąžina sąrašo enumeratorių.
IndexOfKey(raktas)	Ieško nurodyto rakto ir, jei randa, grąžina indeksą >= 0.
IndexOfValue(reikšmė)	Ieško nurodytos reikšmės ir, jei randa, grąžina indeksą >= 0 pirmos rastos reikšmės.
Item[raktas]	Paima arba įdeda rakto apibrėžtą reikšmę.

Dažniausiai naudojami rikiuoto sąrašo `SortedList<Tkey, TValue>` metodai 2/2

Metodas arba savybė	Aprašas
Keys	Pateikia sąrašo raktus.
Remove(raktas)	Išmeta iš sąrašo nurodyto rakto elementą.
RemoveAt(indeksas)	Išmeta iš sąrašo nurodyto indekso elementą.
TrimExcess()	Pakeičia sąrašo talpą pagal jos elementų kiekį, suapvalinant iki artimiausio skaičiavimuose numatyto dydžio.
TryGetValue(raktas, reikšmė)	Grąžina nurodyto rakto reikšmę.
Values	Pateikia sąrašo reikšmes.

ktu informatikos fakultetas 1922

Rikiuoto sąrašo SortedList<Tkey, TValue> pavyzdžiai (1/20)

```
// spausdina rikiuoto sąrašo <string, int> reikšmes.
// Savybės Key ir Value
public static void Spausdinti(SortedList<string, int>zod)
{
    foreach (KeyValuePair<string, int> pora in zod)
    {
        Console.WriteLine("     {0,-10}  -  {1}",
                pora.Key.ToString(), pora.Value.ToString());
    }
    Console.WriteLine();
}
```

ktu informatikos fakultetas 1922

Rikiuoto sąrašo SortedList<Tkey, TValue> pavyzdžiai (2/20)

```
// spausdina rikiuoto sąrašo<string, Asmuo> reikšmes.
// Savybės Key ir Value
public static void Spausdinti(SortedList<string, Asmuo>zod)
{
    foreach (KeyValuePair<string, Asmuo> pora in zod)
    {
        Console.WriteLine("     {0,-10}  -  {1}",
                pora.Key.ToString(), pora.Value.ToString());
    }
    Console.WriteLine();
}
```

ktu informatikos fakultetas 1922

Rikiuoto sąrašo SortedList<Tkey, TValue> pavyzdžiai (3/20)

```
// spausdina rikiuoto sąrašo<Asmuo, string> reikšmes.
// Savybės Key ir Value
public static void Spausdinti(SortedList<Asmuo, string>zod)
{
    foreach (KeyValuePair<Asmuo, string> pora in zod)
    {
        Console.WriteLine("     {0,-10}  -  {1}",
                pora.Key.ToString(), pora.Value.ToString());
    }
    Console.WriteLine();
}
```

Rikiuoto sąrašo `SortedList<Tkey, TValue>` pavyzdžiai (4/20)

```
// Sukuria ir inicializuoja sąrašą
SortedList<string, int> pirmas = new SortedList<string, int>();
pirmas.Add("vienas", 1);
pirmas.Add("du", 2);
pirmas.Add("trys", 3);
pirmas.Add("keturi", 4);
pirmas.Add("penki", 5);
Console.WriteLine("   Sąrašas pirmas");
Console.WriteLine("Sąrašo narių kiekis:    {0}", pirmas.Count);
Console.WriteLine("Sąrašo talpa:    {0}", pirmas.Capacity);
Console.WriteLine("   Reikšmės:");
Spausdinti(pirmas);
```

Sąrašas pirmas

vienas	1
du	2
trys	3
keturi	4
penki	5

Palyginkite reikšmių užrašymo eiliškumą.

```
   Sąrašas pirmas
Sąrašo narių kiekis:    5
Sąrašo talpa:    8
   Reikšmės:
      du          -  2
      keturi      -  4
      penki       -  5
      trys        -  3
      vienas      -  1

Press any key to continue . . .
```

Rikiuoto sąrašo SortedList<Tkey, TValue> pavyzdžiai (5/20)

```
// Sukuria ir inicializuoja sąrašą
SortedList<string, int> antras =
           new SortedList<string, int>(pirmas);
antras.Add("šeši", 6);
Console.WriteLine("   Sąrašas antras");
Console.WriteLine("Sąrašo narių kiekis:    {0}", antras.Count);
Console.WriteLine("   Reikšmės:");
Spausdinti(antras);
```

Sąrašas antras

vienas	1
du	2
trys	3
keturi	4
penki	5
šeši	6

```
   Sąrašas antras
Sąrašo narių kiekis:    6
   Reikšmės:
      du          -  2
      keturi      -  4
      penki       -  5
      šeši        -  6
      trys        -  3
      vienas      -  1

Press any key to continue . . .
```

ktu informatikos fakultetas 1922

Rikiuoto sąrašo SortedList<Tkey, TValue> pavyzdžiai (6/20)

```
public class Asmuo : IComparable<Asmuo>
{
    private string vardas;
    private int amžius;
    public Asmuo(string vardas, int amžius) // Konstruktorius
    {
        this.vardas = vardas;
        this.amžius = amžius;
    }
    public override string ToString()
     return this.vardas + "  " + this.amžius;
    }
    public override bool Equals(object objektas)
    {
        Asmuo stud = objektas as Asmuo;
        return stud.vardas == vardas && stud.amžius == amžius;
    }
// Tęsinys – kitoje skaidrėje
}
```

Paveldėjimas, reikalingas parašyti savo rakto lyginimo metodą.

ktu informatikos fakultetas 1922

Rikiuoto sąrašo `SortedList<Tkey, TValue>` pavyzdžiai (7/20)

```
public class Asmuo : IComparable<Asmuo>
{
    // Tęsinys...
    public int CompareTo(Asmuo kitas)
    {
        int poz = String.Compare(this.vardas, kitas.vardas,
                                 StringComparison.CurrentCulture);
        if (poz > 0) return 1;
        if (poz < 0) return -1;
        else
            if (this.amžius > kitas.amžius) return 1;
            else if (this.amžius < kitas.amžius) return -1;
                 else return 0;
    }
    public override int GetHashCode() // Užklotas metodas GetHashCode()
    {
        return base.GetHashCode();
    }
}
```

Užklotas metodas (būtinai CompareTo), reikalingas parašyti savo rakto lyginimo metodą.

Rikiuoto sąrašo SortedList<Tkey, TValue> pavyzdžiai (8/20)

```
// Sukuria ir inicializuoja sąrašą
Asmuo stud = new Asmuo("Jonas", 25);
Asmuo stud1 = new Asmuo("Petras", 26);
Asmuo stud2 = new Asmuo("Juozas", 38);
SortedList<string, Asmuo> naujas =
            new SortedList<string, Asmuo>();
naujas.Add("vienas", stud);
naujas.Add("du", stud1);
naujas.Add("trys", stud2);
Console.WriteLine("   Sąrašas naujas");
Console.WriteLine("Sąrašo narių kiekis:    {0}", naujas.Count);
Console.WriteLine("   Reikšmės:");
Spausdinti(naujas);
```

Sąrašas naujas

vienas	Jonas 25
du	Petras 26
trys	Juozas 38

```
   Sąrašas naujas
Sąrašo narių kiekis:    3
   Reikšmės:
     du          -  Petras  26
     trys        -  Juozas  38
     vienas      -  Jonas  25

Press any key to continue . . .
```

Rikiuoto sąrašo SortedList<Tkey, TValue> pavyzdžiai (9/20)

```
// Sąrašas, kurio raktas - objektas
SortedList<Asmuo, string> naujas1 =
            new SortedList<Asmuo, string>();
naujas1.Add(stud, "vienas");
naujas1.Add(stud1, "du");
naujas1.Add(stud2, "trys");
Console.WriteLine("   Sąrašas naujas1");
Console.WriteLine("Sąrašo narių kiekis:    {0}", naujas1.Count);
Console.WriteLine("   Reikšmės:");
Spausdinti(naujas1);
```

Sąrašas naujas1

Jonas 25	vienas
Petras 26	du
Juozas 38	trys

```
   Sąrašas naujas1
Sąrašo narių kiekis:    3
   Reikšmės:
      Jonas  25   -  vienas
      Juozas  38  -  trys
      Petras  26  -  du

Press any key to continue . . .
```

Rikiuoto sąrašo `SortedList<Tkey, TValue>` pavyzdžiai (10/20)

```
// Nauja palyginimo klasė
public class IComparer : IComparer<Asmuo>
{
    public bool Equals(Asmuo b1, Asmuo b2)
    {
        return b1.amžius == b2.amžius;
    }
    public int Compare(Asmuo kitas, Asmuo kitas1)
    {
        if (kitas.amžius > kitas1.amžius)
            return 1;
        if (kitas.amžius < kitas1.amžius)
            return -1;
        else return 0;
    }
    public int GetHashCode(Asmuo zmogus)
    {
        return base.GetHashCode();
    }
}
```

Ta pati klasė, kaip ir rikiuoto žodyno atveju.

Toks pat metodas, kaip ir rikiuotam žodynui.

Rikiuoto sąrašo SortedList<Tkey, Tvalue> pavyzdžiai (11/20)

```
// Sukuria ir inicializuoja sąrašą
IComparer kitas = new IComparer();
SortedList<Asmuo, string> naujas2 =
            new SortedList<Asmuo, string>(naujas1, kitas);
// Tikrina ir gali būti vykdymo klaida
Asmuo stud4 = new Asmuo("Rimas", 25);
// naujas2.Add(stud4, "penki");
// Duos klaidą dėl IComparer kitas. Klaida dėl 25
// naujas2.Add(stud1, "du");   // Duos klaidą dėl IComparer kitas
Console.WriteLine("   Sąrašas naujas2");
Console.WriteLine("Sąrašo narių kiekis:    {0}", naujas2.Count);
Console.WriteLine("   Reikšmės:");
Spausdinti(naujas2);
```

```
   Sąrašas naujas2
Sąrašo narių kiekis:    3
   Reikšmės:
      Jonas  25   -  vienas
      Petras  26   -  du
      Juozas  38   -  trys

Press any key to continue . . .
```

Sąrašas naujas1

Jonas 25	vienas
Petras 26	du
Juozas 38	trys

Sąrašas naujas2

Jonas 25	vienas
Petras 26	du
Juozas 38	trys

Rikiuoto sąrašo SortedList<Tkey, Tvalue> pavyzdžiai (12/20)

```
// Sukuria ir inicializuoja sąrašą
IComparer kitas = new IComparer();
SortedList<Asmuo, string> naujas3 =
            new SortedList<Asmuo, string>(kitas);
Asmuo stud4 = new Asmuo("Rimas", 25);
naujas3.Add(stud4, "penki");
// naujas3.Add(stud, "du"); // Duos klaidą dėl IComparer kitas
naujas3.Add(stud2, "trys");
Console.WriteLine("   Sąrašas naujas3");
Console.WriteLine("Sąrašo narių kiekis:    {0}", naujas3.Count);
Console.WriteLine("   Reikšmės:");
Spausdinti(naujas3);
```

stud2

Juozas 38

Sąrašas naujas3

Rimas 25	penki
Juozas 38	trys

```
   Sąrašas naujas3
Sąrašo narių kiekis:    2
   Reikšmės:
     Rimas  25   -  penki
     Juozas  38  -  trys

Press any key to continue . . .
```

ktu informatikos fakultetas 1922

Rikiuoto sąrašo `SortedList<Tkey, TValue>` pavyzdžiai (13/20)

```
// panaudojamas kitas standartinis Comparer
SortedList<string, string> naujas4 =
          new SortedList<string, string>(
              StringComparer.CurrentCultureIgnoreCase);
```

Rikiuoto sąrašo SortedList<Tkey, Tvalue> pavyzdžiai (14/20)

```
// Sukuria ir inicializuoja sąrašą
SortedList<Asmuo, string> naujas5 =
            new SortedList<Asmuo, string>(10);
naujas5.Add(stud4, "penki");
//naujas5.Add(stud, "du"); Duoda klaidą dėl 25
naujas5.Add(stud2, "trys");
Asmuo stud5 = new Asmuo("Rimas", 25);
//naujas5.Add(stud5, "penki"); Duoda klaidą dėl 25
Console.WriteLine("   Sąrašas naujas5");
Console.WriteLine("Sąrašo narių kiekis:    {0}", naujas5.Count);
Console.WriteLine("   Reikšmės:");
Spausdinti(naujas5);
```

Sąrašas naujas5

Jonas 25	du
Juozas 38	trys

```
   Sąrašas naujas5
Sąrašo narių kiekis:    2
   Reikšmės:
     Juozas  38  -  trys
     Rimas  25   -  penki

Press any key to continue . . .
```

Rikiuoto sąrašo SortedList<Tkey, Tvalue> pavyzdžiai (15/20)

```
// Sukuria ir inicializuoja Sąrašą
SortedList<Asmuo, string> naujas6 =
            new SortedList<Asmuo, string>(10, kitas);
naujas6.Add(stud4, "penki");
// naujas6.Add(stud, "du");   // Būtų klaida
naujas6.Add(stud2, "trys");
Console.WriteLine("   Sąrašas naujas6");
Console.WriteLine("Sąrašo narių kiekis:    {0}", naujas6.Count);
Console.WriteLine("   Reikšmės:");
Spausdinti(naujas6);
```

Kodėl klaida?

Žodynas naujas6

Rimas 25	penki
Juozas 38	trys

```
   Sąrašas naujas6
Sąrašo narių kiekis:    2
   Reikšmės:
     Rimas  25   -  penki
     Juozas  38  -  trys

Press any key to continue . . .
```

Rikiuoto sąrašo SortedList<Tkey, Tvalue> pavyzdžiai (16/20)

```
// IndexOfKey(,
int sk = naujas6.IndexOfKey(stud4);
if (sk >= 0) Console.WriteLine("rado raktą: {0}  {1}. Jo " +
                                "indeksas {2}",
                        stud4.vardas, stud4.amžius, sk);
else Console.WriteLine("nerado rakto: {0}  {1}", stud4.vardas,
                         stud4.amžius);
sk = naujas6.IndexOfKey(stud3);
if (sk >= 0) Console.WriteLine("rado raktą: {0}  {1}. Jo " +
                                "indeksas {2}",
                       stud3.vardas, stud3.amžius, sk);
else Console.WriteLine("nerado rakto: {0}  {1}", stud3.vardas,
stud3.amžius);
Console.WriteLine();
```

```
rado raktą: Rimas  25. Jo indeksas 0
nerado rakto: Juozas  55

Press any key to continue . . .
```

Sąrašas naujas6

Rimas 25	penki
Juozas 38	trys

Rikiuoto sąrašo SortedList<Tkey, Tvalue> pavyzdžiai (17/20)

```
// IndexOfValue()
int sk1 = naujas6.IndexOfValue("trys");
if (sk1 >= 0) Console.WriteLine("rado reikšmę: trys. Jos " +
                                "indeksas {0}", sk1);
else Console.WriteLine("nerado reikšmės: trys");
sk1 = naujas6.IndexOfValue("šeši");
if (sk1 >= 0) Console.WriteLine("rado reikšmę: šeši. Jos " +
                                "indeksas {0}", sk1);
else Console.WriteLine("nerado reikšmės: šeši");
Console.WriteLine();
```

Sąrašas naujas6

Rimas 25	penki
Juozas 38	trys

```
rado reikšmę: trys. Jos indeksas 1
nerado reikšmės: šeši

Press any key to continue . . .
```

Rikiuoto sąrašo SortedList<Tkey, Tvalue> pavyzdžiai (18/20)

```
// Remove(Key)
Console.WriteLine("   Sąrašas naujas3");
Console.WriteLine("   Reikšmės:");
Spausdinti(naujas3);
naujas3.Remove(stud1);  // Jei neranda, klaidos neduoda
var enumerator1 = naujas3.GetEnumerator(); // GetEnumerator()
Console.WriteLine("   Sąrašas naujas3 po išmetimo ");
while (enumerator1.MoveNext())
{
    object item = enumerator1.Current;
    Console.WriteLine("    {0}", item);
}
Console.WriteLine();
```

stud1

Petras 26

```
   Sąrašas naujas3
   Reikšmės:
     Rimas  25   -  penki
     Juozas  38  -  trys

   Sąrašas naujas3 po išmetimo
    [Rimas  25, penki]
    [Juozas  38, trys]

Press any key to continue . . .
```

Rikiuoto sąrašo SortedList<Tkey, Tvalue> pavyzdžiai (19/20)

```
// RemoveAt(Indeksas)
Console.WriteLine("   Sąrašas naujas6");
Console.WriteLine("   Reikšmės:");
Spausdinti(naujas6);
int sk2 = 0; // metamo elemento indeksas
naujas6.RemoveAt(sk2); // Jei nėra tokio indekso, klaida
var enumerator2 = naujas6.GetEnumerator();
Console.WriteLine("   Sąrašas naujas6 po išmetimo ");
while (enumerator2.MoveNext())
{
    object item = enumerator2.Current;
    Console.WriteLine("    {0}", item);
}
Console.WriteLine();
```

```
   Sąrašas naujas6
   Reikšmės:
     Rimas  25   -  penki
     Juozas  38  -  trys

   Sąrašas naujas6 po išmetimo
    [Juozas  38, trys]

Press any key to continue . . .
```

Rikiuoto sąrašo `SortedList<Tkey, TValue>` pavyzdžiai (20/20)

```
// TrimExcess()
Console.WriteLine("Sąrašo pirmas talpa:    {0}",
                  pirmas.Capacity);
pirmas.TrimExcess();
Console.WriteLine("Sąrašo pirmas talpa po korekcijos:    {0}",
                   pirmas.Capacity);
```

Kitų metodų naudojimas nesiskiria nuo jų panaudojimo rikiuotam žodynui.

```
Sąrašo pirmas talpa:    8
Sąrašo pirmas talpa po korekcijos:    5
Press any key to continue . . .
```

SortedList Class (System.Collections)

ktu informatikos fakultetas 1922

Rikiuotas sąrašas `SortedList` 1/2

Rikiuotas sąrašas (*System.Collections*) – asociatyvusis konteineris.
Išrinkimui naudojamas raktas.
Raktas – tai gali būti eilutė, skaičius, savo sukurtos klasės objektas.
Duomenys saugomi dviejuose vidiniuose masyvuose.
Greitas įterpimas, pašalinimas, paieška.

ktu informatikos fakultetas 1922

Rikiuotas sąrašas `SortedList` 2/2

Rikiuotas sąrašas `SortedList` turi:

- 6 užklotus konstruktorius.
- 9 savybes (properties).
- 25 metodų.
- 1 sąsajos realizaciją (Explicit Interface Implementations).
- 4 užklotus metodus.

SortedList Class (System.Collections) 1/2

1. Kolekcijoje *System.Collections* yra rikiuoto sąrašo konteineris – `SortedList`. Šio sąrašo elementai saugomi dviejuose vienmačiuose masyvuose: atskirai raktai, atskirai reikšmės. Masyvų elementai susieti indeksu. Kiekvieną jų sudaro pora: raktas/reikšmė. Kolekcijoje *System.Collections.Generic* yra rikiuoto sąrašo konteineris – `SortedList<Tkey, TValue>`. Šio sąrašo elementai saugomi viename vienmačiame masyve, kurio elementai `KeyValuePair<Tkey, TValue>` tipo.
2. `SortedList` tipo konteinerių elementus galima persirašyti į savo programos `KeyValuePair<Tkey, TValue>` tipo vieną masyvą. Galima susikurti ir du masyvus: atskirai raktams bei reikšmėms, ir atitinkamus sąrašo elementus persirašyti į juos. `SortedList<Tkey, TValue>` konteinerių elementų persirašyti į savo masyvą/masyvus negalima.

SortedList Class (System.Collections) 2/2

3. `SortedList` tipo konteineriams negalima naudoti užklotų metodų, tokių kaip `Sum()`, `Max()`, `Average()` ir t.t.
4. `SortedList` tipo konteineriuose galima naudoti indeksus.
5. Kolekcijoje *System.Collections* yra maišos lentelė `HashTable`. Jos elementai – pora: raktas/reikšmė. `SortedList` tipo konteinerių panaudojimas – lėtesnis nei maišos lentelių `HashTable` dėl elementų rikiavimo. Privalumas – galima dirbti ir su indeksais.

ktu informatikos fakultetas 1922

Rikiuoto sąrašo `SortedList` aprašas 1/2

Rikiuoto sąrašo `SortedList` aprašai:

1. `SortedList Pavadinimas = new SortedList ();`

Sukuriamas numatytos talpos (16 elementų) ***tuščias*** (Count = 0) rikiuotas sąrašas. Raktų palyginimui naudoja standartinį lyginimo metodą (`IComparable` sąsajos).

2. `SortedList Pavadinimas1 = new SortedList (Pavadinimas);`

Sukuriamas naujas rikiuotas sąrašas **`Pavadinimas1`**, kuriame yra bet kokio kito tipo žodyno **`Pavadinimas`** elementai. Pradinė talpa – nukopijuotų elementų kiekis. Naudojamas standartinis raktų lyginimo metodas (`IComparable` sąsajos).

3. `SortedList Pavadinimas2 = new SortedList (IComparer objektas);`

Sukuriamas numatytos talpos (16 elementų) tuščias žodynas. Naudojamas užklotas raktų palyginimo metodas.

ktu informatikos fakultetas 1922

Rikiuoto sąrašo `SortedList` aprašas 2/2

4. `SortedList Pavadinimas3 = new SortedList (Pavadinimas, IComparer objektas);`

Sukuriamas naujas rikiuotas sąrašas **`Pavadinimas3`**, kuriame yra bet kokio kito tipo žodyno **`Pavadinimas`** elementai. Pradinė talpa – nukopijuotų elementų kiekis. Raktų palyginimui naudojamas užklotas raktų palyginimo metodas.

5. `SortedList Pavadinimas4 = new SortedList (talpa);`

Sukuriamas naujas tuščias nurodytos talpos rikiuotas sąrašas **`Pavadinimas4.`** Raktų palyginimui naudoja standartinį lyginimo metodą (`IComparable` sąsajos).

6. `SortedList Pavadinimas5 = new SortedList (IComparer objektas, talpa);`

Sukuriamas nurodytos talpos tuščias rikiuotas sąrašas. Naudojamas užklotas raktų palyginimo metodas.

ktu informatikos fakultetas 1922

Dažniausiai naudojami rikiuoto sąrašo `SortedList` metodai 1/2

Metodas arba savybė	Aprašas
Add(raktas, reikšmė)	Įterpia elementą į sąrašą.
Capacity	Savybė, kuri nustato arba grąžina sąrašo talpą.
Clear()	Pašalina visus sąrašo elementus.
Contains(objektas)	Grąžina true, jei ieškomas raktas yra, priešingu atveju – false.
ContainsKey(objektas)	Grąžina true, jei ieškomas raktas yra, priešingu atveju – false.
ContainsValue(objektas)	Grąžina true, jei ieškoma reikšmė yra, priešingu atveju – false.
CopyTo(Masyvas, indeksas)	Kopijuoja sąrašo elementus į vienmatį masyvą, pradedant nurodytu indeksu masyve.
Count	Savybė, kuri grąžina sąrašo elementų skaičių.
Equals(objektas)	Grąžina true, jei nurodytas objektas lygus nagrinėjamam objektui, priešingu atveju – false. (Paveldėta iš Object)

ktu informatikos fakultetas 1922

Dažniausiai naudojami rikiuoto sąrašo `SortedList` metodai 2/2

Metodas arba savybė	Aprašas
GetByIndex(indeksas)	Grąžina nurodyto indekso sąrašo reikšmę.
GetEnumerator()	Grąžina sąrašo enumeratorių.
GetKey(indeksas)	Grąžina nurodyto indekso sąrašo elemento raktą.
GetKeyList()	Grąžina sąrašo elementų raktus.
GetValueList()	Grąžina sąrašo elementų reikšmes.
IndexOfKey(raktas)	Ieško nurodyto rakto ir, jei randa, grąžina indeksą >= 0.
IndexOfValue(reikšmė)	Ieško nurodytos reikšmės ir, jei randa, grąžina indeksą >= 0 pirmos rastos reikšmės.
IsFixedSize	Grąžina savybę, ar sąrašas yra fiksuoto ilgio.
IsReadOnly	Grąžina savybę, ar sąrašas yra tik skaitomas.

ktu informatikos fakultetas 1922

Dažniausiai naudojami rikiuoto sąrašo `SortedList` metodai 2/2

Metodas arba savybė	Aprašas
Item[raktas]	Paima arba įdeda rakto apibrėžtą reikšmę.
Keys	Pateikia sąrašo raktus.
Remove(raktas)	Išmeta iš sąrašo nurodyto rakto elementą.
RemoveAt(indeksas)	Išmeta iš sąrašo nurodyto indekso elementą.
SetByIndex(indeksas, objektas)	Pakeičia nurodyto indekso sąrašo elemento reikšmę.
TrimSize()	Pakeičia sąrašo talpą pagal jos elementų kiekį, suapvalinant iki artimiausio skaičiavimuose numatyto dydžio.
Values	Pateikia sąrašo reikšmes.

Rikiuoto sąrašo SortedList pavyzdžiai (1/19)

```
// spausdina sąrašo reikšmes.
// Metodai GetKey(), GetByIndex()
public static void Spausdinti(SortedList sarasas)
{
    for (int i = 0; i < sarasas.Count; i++)
    {
        Console.WriteLine("     {0,-10}  -  {1}",
                          sarasas.GetKey(i).ToString(),
                          sarasas.GetByIndex(i).ToString());
    }
    Console.WriteLine();
}
```

Rikiuoto sąrašo SortedList pavyzdžiai (2/19)

```
// Sukuria ir inicializuoja sąrašą
SortedList pirmas = new SortedList();
pirmas.Add("vienas", 1);
pirmas.Add("du", 2);
pirmas.Add("trys", 3);
pirmas.Add("keturi", 4);
pirmas.Add("penki", 5);
Console.WriteLine("   Sąrašas pirmas");
Console.WriteLine("Sąrašo narių kiekis:    {0}", pirmas.Count);
Console.WriteLine("Sąrašo talpa:    {0}", pirmas.Capacity);
Console.WriteLine("   Reikšmės:");
Spausdinti(pirmas);
```

Sąrašas pirmas

vienas	1
du	2
trys	3
keturi	4
penki	5

Palyginkite reikšmių užrašymo eiliškumą.

```
   Sąrašas pirmas
Sąrašo narių kiekis:    5
Sąrašo talpa:    16
   Reikšmės:
      du          -  2
      keturi      -  4
      penki       -  5
      trys        -  3
      vienas      -  1

Press any key to continue . . .
```

Rikiuoto sąrašo SortedList pavyzdžiai (3/19)

```
// Sukuria ir inicializuoja sąrašą
SortedList antras = new SortedList(pirmas);
antras.Add("šeši", 6);
Console.WriteLine("   Sąrašas antras");
Console.WriteLine("Sąrašo narių kiekis:    {0}", antras.Count);
Console.WriteLine("   Reikšmės:");
Spausdinti(antras);
```

Sąrašas antras

vienas	1
du	2
trys	3
keturi	4
penki	5
šeši	6

```
   Sąrašas antras
Sąrašo narių kiekis:    6
   Reikšmės:
      du          -  2
      keturi      -  4
      penki       -  5
      šeši        -  6
      trys        -  3
      vienas      -  1

Press any key to continue . . .
```

Rikiuoto sąrašo `SortedList` pavyzdžiai (4/19)

```
public class Asmuo  : IComparable
{
    private string vardas;
    private int amžius;
    public Asmuo(string vardas, int amžius) // Konstruktorius
    {
        this.vardas = vardas;
        this.amžius = amžius;
    }
    public override string ToString()
     return this.vardas + "  " + this.amžius;
    }
    public override bool Equals(object objektas)
    {
        Asmuo stud = objektas as Asmuo;
        return stud.vardas == vardas && stud.amžius == amžius;
    }
// Tęsinys – kitoje skaidrėje
}
```

Paveldėjimas, reikalingas parašyti savo rakto lyginimo metodą. Koks skirtumas lyginant su Generic sąrašu?

Rikiuoto sąrašo `SortedList` pavyzdžiai (5/19)

```
public class Asmuo : IComparable
{
    // Tęsinys...
    public int CompareTo(object obj)
    {
        if (obj == null) return 1;
        Asmuo kitas = obj as Asmuo;
        int poz = String.Compare(this.vardas, kitas.vardas,
                                 StringComparison.CurrentCulture);
        if (poz > 0) return 1;
        if (poz < 0) return -1;
        else
            if (this.amžius > kitas.amžius) return 1;
            else if (this.amžius < kitas.amžius) return -1;
            else return 0;
    }
    public override int GetHashCode() // Užklotas metodas GetHashCode()
    {
        return base.GetHashCode();
    }
}
```

Užklotas metodas (būtinai CompareTo), reikalingas parašyti savo rakto lyginimo metodą.

Rikiuoto sąrašo SortedList pavyzdžiai (6/19)

```
// Sukuria ir inicializuoja sąrašą
Asmuo stud = new Asmuo("Jonas", 25);
Asmuo stud1 = new Asmuo("Petras", 26);
Asmuo stud2 = new Asmuo("Juozas", 38);
SortedList naujas = new SortedList();
naujas.Add("vienas", stud);
naujas.Add("du", stud1);
naujas.Add("trys", stud2);
Console.WriteLine("   Sąrašas naujas");
Console.WriteLine("Sąrašo narių kiekis:    {0}", naujas.Count);
Console.WriteLine("   Reikšmės:");
Spausdinti(naujas);
```

Sąrašas naujas

vienas	Jonas 25
du	Petras 26
trys	Juozas 38

```
   Sąrašas naujas
Sąrašo narių kiekis:    3
   Reikšmės:
     du          -  Petras  26
     trys        -  Juozas  38
     vienas      -  Jonas  25

Press any key to continue . . .
```

Rikiuoto sąrašo SortedList pavyzdžiai (7/19)

```
// Sąrašas, kurio raktas - objektas
SortedList naujas1 = new SortedList();
naujas1.Add(stud, "vienas");
naujas1.Add(stud1, "du");
naujas1.Add(stud2, "trys");
Console.WriteLine("   Sąrašas naujas1");
Console.WriteLine("Sąrašo narių kiekis:    {0}", naujas1.Count);
Console.WriteLine("   Reikšmės:");
Spausdinti(naujas1);
```

Sąrašas naujas1

Jonas 25	vienas
Petras 26	du
Juozas 38	trys

```
   Sąrašas naujas1
Sąrašo narių kiekis:    3
   Reikšmės:
      Jonas  25   -  vienas
      Juozas  38  -  trys
      Petras  26  -  du

Press any key to continue . . .
```

Rikiuoto sąrašo SortedList pavyzdžiai (8/19)

```
// Nauja palyginimo klasė
public class IComparer_n : IComparer
{
    public bool Equals(Asmuo b1, Asmuo b2)
    {
        return b1.amžius == b2.amžius;
    }
    public int Compare(object obj, object obj1)
    {
        if (obj == null) return 1;
        Asmuo kitas = obj as Asmuo;
        Asmuo kitas1 = obj1 as Asmuo;
        if (kitas.amžius > kitas1.amžius) return 1;
        if (kitas.amžius < kitas1.amžius) return -1;
        else return 0;
    }
    public int GetHashCode(Asmuo zmogus)
    {
        return base.GetHashCode();
    }
}
```

Palyginkite su užrašu Generic rikiuoto sąrašo atveju.

Palyginkite su metodu CompareTo() klasėje Asmuo .

Rikiuoto sąrašo SortedList pavyzdžiai (9/19)

```
// Sukuria ir inicializuoja sąrašą
IComparer kitas = new IComparer_n();
SortedList naujas2 = new SortedList(naujas1, kitas);
// Tikrina ir gali būti klaida
Asmuo stud4 = new Asmuo("Rimas", 25);
//naujas2.Add(stud4, "penki");// Duos klaidą dėl IComparer kitas.
// Klaida dėl 25
//naujas2.Add(stud1, "du");  // Duos klaidą dėl IComparer kitas
Console.WriteLine("   Sąrašas naujas2");
Console.WriteLine("Sąrašo narių kiekis:    {0}", naujas2.Count);
Console.WriteLine("   Reikšmės:");
Spausdinti(naujas2);
```

Palyginkite su užrašu Generic rikiuoto sąrašo atveju.

```
   Sąrašas naujas2
Sąrašo narių kiekis:    3
   Reikšmės:
      Jonas  25   -  vienas
      Petras  26   -  du
      Juozas  38   -  trys

Press any key to continue . . .
```

Sąrašas naujas1

Jonas 25	vienas
Petras 26	du
Juozas 38	trys

Sąrašas naujas2

Jonas 25	vienas
Petras 26	du
Juozas 38	trys

Rikiuoto sąrašo SortedList pavyzdžiai (10/19)

```
// Sukuria ir inicializuoja sąrašą
IComparer kitas = new IComparer_n();
SortedList naujas3 = new SortedList(kitas);
//Asmuo stud4 = new Asmuo("Rimas", 25);
naujas3.Add(stud4, "penki");
//naujas3.Add(stud, "du");   // Duoda klaidą dėl IComparer kitas
naujas3.Add(stud2, "trys");
Console.WriteLine("   Sąrašas naujas3");
Console.WriteLine("Sąrašo narių kiekis:    {0}", naujas3.Count);
Console.WriteLine("   Reikšmės:");
Spausdinti(naujas3);
```

stud2

Juozas 38

Sąrašas naujas3

Rimas 25	penki
Juozas 38	trys

```
   Sąrašas naujas3
Sąrašo narių kiekis:    2
   Reikšmės:
     Rimas  25   -  penki
     Juozas  38  -  trys

Press any key to continue . . .
```

ktu informatikos fakultetas 1922

Rikiuoto sąrašo `SortedList` pavyzdžiai (11/19)

```
// panaudojamas kitas standartinis Comparer
SortedList naujas4 = new SortedList
                     (StringComparer.CurrentCultureIgnoreCase);
```

Rikiuoto sąrašo SortedList pavyzdžiai (12/19)

```
// Sukuria ir inicializuoja sąrašą
SortedList naujas5 = new SortedList(10);
naujas5.Add(stud4, "penki");
naujas5.Add(stud, "du"); // Neduoda klaidos dėl 25,
// nes kitas standartinio Comperer
naujas5.Add(stud2, "trys");
Asmuo stud5 = new Asmuo("Rimas", 25);
//naujas5.Add(stud5, "penki");  // Būtų klaida
Console.WriteLine("   Sąrašas naujas5");
Console.WriteLine("Sąrašo narių kiekis:    {0}", naujas5.Count);
Console.WriteLine("Sąrašo talpa:    {0}", naujas5.Capacity);
Console.WriteLine("   Reikšmės:");
Spausdinti(naujas5);
```

Sąrašas naujas5

Rimas 25	penki
Jonas 25	du
Juozas 38	trys

```
   Sąrašas naujas5
Sąrašo narių kiekis:    3
Sąrašo talpa:    10
   Reikšmės:
     Jonas  25   -  du
     Juozas  38  -  trys
     Rimas  25   -  penki

Press any key to continue . . .
```

Rikiuoto sąrašo SortedList pavyzdžiai (13/19)

```
// Sukuria ir inicializuoja sąraša
SortedList naujas6 = new SortedList(kitas, 10);
naujas6.Add(stud4, "penki");
//naujas6.Add(stud, "du");   //Būtų klaida
naujas6.Add(stud2, "trys");
Console.WriteLine("   Sąrašas naujas6");
Console.WriteLine("Sąrašo narių kiekis:    {0}", naujas6.Count);
Console.WriteLine("   Reikšmės:");
Spausdinti(naujas6);
```

Palyginkite su užrašu Generic rikiuoto sąrašo.

Žodynas naujas6

Rimas 25	penki
Juozas 38	trys

```
   Sąrašas naujas6
Sąrašo narių kiekis:    2
   Reikšmės:
     Rimas  25   -  penki
     Juozas  38  -  trys

Press any key to continue . . .
```

Rikiuoto sąrašo SortedList pavyzdžiai (14/19)

```
// GetEnumerator() naudojamas skaityti, bet ne
// modifikuoti. Geriau ciklas foreach
IDictionaryEnumerator enumerator = naujas6.GetEnumerator();
Console.WriteLine("Enumeratoriaus pavyzdys");
Console.WriteLine("   Sąrašas naujas6 ");
while (enumerator.MoveNext())
{
    Console.WriteLine("    {0,-15}  -  {1} ", enumerator.Key,
                      enumerator.Value);
}
Console.WriteLine();
```

Palyginkite su užrašu Generic rikiuoto sąrašo.

```
Enumeratoriaus pavyzdys
   Sąrašas naujas6
    Rimas  25          -  penki
    Juozas  38         -  trys

Press any key to continue . . .
```

Sąrašas naujas6

Rimas 25	penki
Juozas 38	trys

Rikiuoto sąrašo SortedList pavyzdžiai (15/19)

```
// IsFixedSize, IsReadOnly
if (naujas3.IsFixedSize)
    Console.WriteLine("Sąrašas naujas3 yra fiksuoto ilgio");
else Console.WriteLine("Sąrašas naujas3 nėra fiksuoto ilgio");
if (naujas3.IsReadOnly)
    Console.WriteLine("Sąrašas naujas3 yra tik skaitomas");
else Console.WriteLine("Sąrašas naujas3 nėra tik skaitomas");
// GetKeyList(), GetValueList
IList Raktai = naujas3.GetKeyList();   // Speciali klasė
IList Reikšmės = naujas3.GetValueList();
Console.WriteLine();
Console.WriteLine("   Sąrašas naujas3:");
for (int i = 0; i < naujas3.Count; i++)
    Console.WriteLine("    {0,-15}  -  {1}",
                      Raktai[i], Reikšmės[i]);
```

```
Sąrašas naujas3 nėra fiksuoto ilgio
Sąrašas naujas3 nėra tik skaitomas

   Sąrašas naujas3:
    Rimas  25        -  penki
    Juozas  38       -  trys
Press any key to continue . . .
```

Rikiuoto sąrašo SortedList pavyzdžiai (16/19)

```
// CopyTo()
Console.WriteLine("   Sąrašas naujas:");
Spausdinti(naujas);
DictionaryEntry[] Masyvas = new DictionaryEntry[naujas.Count];
naujas.CopyTo(Masyvas, 0);
Console.WriteLine("   Sąrašas naujas perrašytas į masyvą:");
for (int i = 0; i < Masyvas.Length; i++)
    Console.WriteLine("     {0,-10}  -  {1}", Masyvas[i].Key,
                      Masyvas[i].Value);
Console.WriteLine();
```

Palyginkite su užrašu Generic rikiuoto žodyno.

```
Sąrašas naujas3 nėra fiksuoto ilgio
Sąrašas naujas3 nėra tik skaitomas
   Sąrašas naujas:
     du          -  Petras  26
     trys        -  Juozas  38
     vienas      -  Jonas  25

   Sąrašas naujas perrašytas į masyvą:
     du          -  Petras  26
     trys        -  Juozas  38
     vienas      -  Jonas  25

Press any key to continue . . .
```

Rikiuoto sąrašo SortedList pavyzdžiai (17/19)

Atkreipkite dėmesį į klasę.

```
// Keys, Values
ICollection Raktai = naujas3.Keys;   // Speciali klasė
ICollection Reikšmės = naujas3.Values;
var enumerator = Raktai.GetEnumerator();
var enumerator1 = Reikšmės.GetEnumerator();
Console.WriteLine("   Sąrašo naujas3 raktai: ");
while (enumerator.MoveNext())
{
    object item = enumerator.Current;
    Console.WriteLine("    {0}     ", item);
}
Console.WriteLine("   Sąrašo naujas3 reikšmės: ");
while (enumerator1.MoveNext())
{
    object item = enumerator1.Current;
    Console.WriteLine("    {0}     ", item);
}
```

```
   Sąrašo naujas3 raktai:
    Rimas  25
    Juozas  38
   Sąrašo naujas3 reikšmės:
    penki
    trys
Press any key to continue . . .
```

Rikiuoto sąrašo `SortedList` pavyzdžiai (18/19)

```
// SetByIndex()
Console.WriteLine("   Sąrašas naujas3");
Console.WriteLine("   Reikšmės:");
Spausdinti(naujas3);
naujas3.SetByIndex(1, "vienas");
Console.WriteLine("   Sąrašas naujas3 po pakeitimo");
Console.WriteLine("   Reikšmės:");
Spausdinti(naujas3);
```

Galima tik pakeisti.

```
   Sąrašas naujas3
   Reikšmės:
     Rimas  25   -  penki
     Juozas  38  -  trys

   Sąrašas naujas3 po pakeitimo
   Reikšmės:
     Rimas  25   -  penki
     Juozas  38  -  vienas

Press any key to continue . . .
```

Rikiuoto sąrašo SortedList pavyzdžiai (19/19)

```
// TrimToSize()
Console.WriteLine("Sąrašo pirmas talpa:    {0}",
                  pirmas.Capacity);
pirmas.TrimToSize();
Console.WriteLine("Sąrašo pirmas talpa po korekcijos:    {0}",
                  pirmas.Capacity);
```

Kitų metodų naudojimas nesiskiria nuo jų panaudojimo kituose konteineriuose.

```
Sąrašo pirmas talpa:    16
Sąrašo pirmas talpa po korekcijos:    5
Press any key to continue . . .
```

ktu informatikos fakultetas 1922

HashSet<T> Class (System.Collections.Generic)

Maišos aibė HashSet<T> 1/2

Maišos aibė – asociatyvusis konteineris.
Išrinkimui naudojas raktas.
Raktas – tai gali būti eilutė, skaičius. Gali būti ir savo sukurtos klasės objektas.

Maišos aibė `HashSet<T>` 2/2

Maišos aibė `HashSet<T>` turi:

- 5 užklotus konstruktorius.
- 2 savybę (properties).
- 28 metodų.
- 4 sąsajos realizacijas (Explicit Interface Implementations).
- Didelę aibę užklotų metodų (>130).

Maišos aibės `HashSet<T>` aprašas 1/2

ktu informatikos fakultetas 1922

Maišos aibės `HashSet<T>` aprašai:

1. `HashSet<T> Pavadinimas = new HashSet<T> ();`

Sukuriama numatytos talpos ***tuščia*** (Count = 0) aibė. Aibės tipo elementų palyginimui naudoja standartinį lyginimo metodą (equality comparer).

2. `HashSet<T> Pavadinimas1 = new HashSet<T> (Pavadinimas);`

Sukuriama nauja reikiamos talpos aibė **`Pavadinimas1`**, kurioje yra aibės **`Pavadinimas`** elementai. Aibės tipo elementų palyginimui naudoja standartinį lyginimo metodą.

Maišos aibės `HashSet<T>` aprašas 2/2

ktu informatikos fakultetas 1922

3. `HashSet<T> Pavadinimas2 = new HashSet<T> (IComparer objektas);`

Sukuriama tuščia aibė. Aibės tipo elementų palyginimui naudojama užklotą palyginimo metodą.

4. `HashSet<T> Pavadinimas3 = new HashSet<T> (Pavadinimas, IComparer objektas);`

Sukuriama nauja reikiamos talpos aibė **`Pavadinimas3`**, kuriame yra aibės **`Pavadinimas`** elementai. Aibės tipo elementų palyginimui naudojama užklotą palyginimo metodą.

Dažniausiai naudojami maišos aibės `HashSet<T>` metodai 1/3

Metodas arba savybė	Aprašas
Add(elementas)	Įterpia elementą į aibę.
Clear()	Pašalina visus aibės elementus.
Contains(objektas)	Grąžina true, jei ieškomas objektas yra, priešingu atveju – false.
CopyTo(masyvas)	Kopijuoja aibės elementus į vienmatį masyvą.
CopyTo(masyvas, indeksas)	Kopijuoja aibės elementus į vienmatį masyvą, pradedant nurodytu masyvo indeksu.
CopyTo(masyvas, indeksas, kiekis)	Kopijuoja nurodytą kiekį aibės elementų į vienmatį masyvą, pradedant nurodytu masyvo indeksu.
Count	Savybė, kuri grąžina aibės elementų skaičių.
Equals(objektas)	Grąžina true, jei nurodytas objektas lygus nagrinėjamam objektui, priešingu atveju – false. (Paveldėta iš Object)
ExceptWith(Pavadinimas)	Iš aibės išmeta visus **Pavadinimas** aibės elementus.
GetEnumerator()	Grąžina aibės enumeratorių.

Dažniausiai naudojami maišos aibės `HashSet<T>` metodai 2/3

Metodas arba savybė	Aprašas
IntersectWith(Pavadinimas)	Modifikuoja nagrinėjamą aibę, suformuoja aibių pjūvį.
IsProperSubsetOf(Pavadinimas)	Grąžina true, jei aibė **Pavadinimas** yra tikrinis nagrinėjamos aibės poaibis, priešingu atveju – false.
IsProperSupersetOf (Pavadinimas)	Grąžina true, jei aibė **Pavadinimas** yra tikrinis nagrinėjamos aibės viršaibis, priešingu atveju – false.
IsSubsetOf(Pavadinimas)	Grąžina true, jei aibė **Pavadinimas** yra nagrinėjamos aibės poaibis, priešingu atveju – false.
IsSupersetOf(Pavadinimas)	Grąžina true, jei aibė **Pavadinimas** yra nagrinėjamos aibės viršaibis, priešingu atveju – false.
Overlaps(Pavadinimas)	Grąžina true, jei aibė **Pavadinimas** ir nagrinėjama aibė turi bendrų elementų, priešingu atveju – false.
Remove(objektas)	Išmeta objektą iš aibės.
RemoveWhere(operatorius)	Iš aibės išmeta visus aibės elementus, kurie tenkina užkloto operatoriaus sąlygas.

Dažniausiai naudojami maišos aibės `HashSet<T>` metodai 3/3

Metodas arba savybė	Aprašas
SetEquals(Pavadinimas)	Grąžina true, jei aibė **Pavadinimas** ir nagrinėjama aibės yra lygios, priešingu atveju – false.
SymmetricExceptWith (Pavadinimas)	Modifikuoja nagrinėjamą aibę, paliekant tik elementus, kurie yra tik vienoje iš aibių, bet ne abiejose.
TrimExcess()	Pakeičia aibės talpą pagal jos elementų kiekį, suapvalinant iki artimiausio skaičiavimuose numatyto dydžio.
UnionWith(Pavadinimas)	Suformuoja dviejų aibių junginį.

Maišos aibės HashSet<T> pavyzdžiai (1/34)

```
// spausdina sveikų skaičių aibės reikšmes
public static void Spausdinti(HashSet<int> aibe)
{
    foreach (int elem in aibe)
    {
        Console.Write(" {0}  ", elem);
    }
    Console.WriteLine();
}
```

Maišos aibės HashSet<T> pavyzdžiai (2/34)

```
// spausdina eilutės tipo aibės reikšmes
public static void Spausdinti(HashSet<string> aibe)
{
    foreach (string elem in aibe)
    {
        Console.Write(" {0}  ", elem);
    }
    Console.WriteLine();
}
```

Maišos aibės HashSet<T> pavyzdžiai (3/34)

```
// spausdina objekto tipo aibės reikšmes
public static void Spausdinti(HashSet<Asmuo> aibe)
{
    foreach (Asmuo obj in aibe)
        Console.WriteLine("    {0,-12}   {1}",
                obj.vardas, obj.amžius);
    Console.WriteLine();
}
```

Maišos aibės HashSet<T> pavyzdžiai (4/34)

```
public class Asmuo
{
    private string vardas;
    private int amžius;
    public Asmuo(string vardas, int amžius) // Konstruktorius
    {
        this.vardas = vardas;
        this.amžius = amžius;
    }
    public override string ToString()
    {
        return this.vardas + "  " + this.amžius;
    }
    public override bool Equals(object objektas)
    {
        Asmuo stud = objektas as Asmuo;
        return stud.vardas == vardas && stud.amžius == amžius;
    }
    // Užklotas metodas GetHashCode()
    public override int GetHashCode()  {return base.GetHashCode();}
}
```

Maišos aibės HashSet<T> pavyzdžiai (5/34)

```
// Nauja palyginimo klasė
class IComparer : IEqualityComparer<Asmuo>
{
    public bool Equals(Asmuo b1, Asmuo b2)
    {
        return b1.amžius == b2.amžius;
    }

    public int GetHashCode(Asmuo zmogus)
    {
        return base.GetHashCode();
    }
}
```

Maišos aibės HashSet<T> pavyzdžiai (6/34)

```
// Sukuria ir inicializuoja aibę
HashSet<int> skaic = new HashSet<int>();

// formuoja aibę
for (int i = 0; i < 10; i++)
    skaic.Add(i);

Console.WriteLine("   Aibė skaic:");
Console.WriteLine("Aibės skaic narių kiekis:    {0}",
skaic.Count);
Spausdinti(skaic);
Console.WriteLine();}
```

```
   Aibė skaic:
Aibės skaic narių kiekis:    10
 0   1   2   3   4   5   6   7   8   9

Press any key to continue . . .
```

Maišos aibės HashSet<T> pavyzdžiai (7/34)

```
   Aibė nauja
Aibės nauja narių kiekis:    4
   Reikšmės:
    Jonas          25
    Petras         26
    Juozas         38
    Kazys          25

Press any key to continue . . .
```

```
Asmuo stud = new Asmuo("Jonas", 25);
Asmuo stud1 = new Asmuo("Petras", 26);
Asmuo stud2 = new Asmuo("Juozas", 38);
Asmuo stud3 = new Asmuo("Kazys", 25);
// Sukuria ir inicializuoja aibę
HashSet<Asmuo> nauja = new HashSet<Asmuo>();
nauja.Add(stud);
nauja.Add(stud1);
nauja.Add(stud2);
nauja.Add(stud3); // Kas būtų įterpus nauja.Add(stud3); ?
Console.WriteLine("   Aibė nauja");
Console.WriteLine("Aibės nauja narių kiekis:    {0}",
                  nauja.Count);
Console.WriteLine("   Reikšmės:");
Spausdinti(nauja);
```

Maišos aibės HashSet<T> pavyzdžiai (8/34)

```
// Sukuria ir inicializuoja aibę
HashSet<int> skaic1 = new HashSet<int>(skaic);

Console.WriteLine("   Aibė skaic1:");
Console.WriteLine("Aibės skaic1 narių kiekis:    {0}",
                  skaic1.Count);
Spausdinti(skaic1);
Console.WriteLine();
```

```
   Aibė skaic1:
Aibės skaic1 narių kiekis:    10
 0   1   2   3   4   5   6   7   8   9

Press any key to continue . . .
```

Maišos aibės HashSet<T> pavyzdžiai (9/34)

```
// Sukuria ir inicializuoja aibę
HashSet<Asmuo> nauja1 = new HashSet<Asmuo>(nauja);
Console.WriteLine("   Aibė nauja1");
Console.WriteLine("Aibės nauja1 narių kiekis:    {0}",
                  nauja1.Count);
Console.WriteLine("   Reikšmės:");
Spausdinti(nauja);
```

```
   Aibė nauja1
Aibės nauja1 narių kiekis:    4
   Reikšmės:
    Jonas          25
    Petras         26
    Juozas         38
    Kazys          25

Press any key to continue . . .
```

Maišos aibės HashSet<T> pavyzdžiai (10/34)

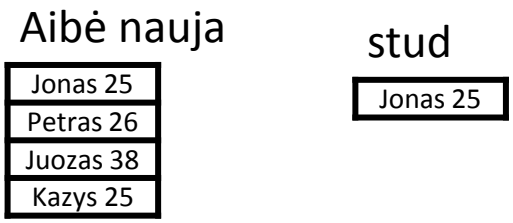
```
// Sukuria lyginimo klasės objektą
IComparer kitas = new IComparer();
// Sukuria ir inicializuoja aibę, pasikartojančių elementų
// neįtraukia, neišduodant vykdymo klaidos
HashSet<Asmuo> nauja2 = new HashSet<Asmuo>(kitas);
nauja2.Add(stud);
Asmuo stud4 = new Asmuo("Kazys", 25);
nauja2.Add(stud4);
nauja2.Add(stud4);
Console.WriteLine("   Aibė nauja2");
Console.WriteLine("Aibės nauja2 narių kiekis:    {0}",
                  nauja2.Count);
Console.WriteLine("   Reikšmės:");
Spausdinti(nauja2);
```

Neįdeda į aibę, bet neduoda klaidos. Kodėl?

```
   Aibė nauja2
Aibės nauja2 narių kiekis:    1
   Reikšmės:
    Jonas          25

Press any key to continue . . .
```

Maišos aibės HashSet<T> pavyzdžiai (11/34)

```
// Sukuria ir inicializuoja aibę
HashSet<Asmuo> nauja3 = new HashSet<Asmuo>(nauja, kitas);
// neįdeda, bet klaidos neduoda
nauja3.Add(stud); // neįdeda, bet klaidos neduoda
Console.WriteLine("   Aibė nauja3");
Console.WriteLine("Aibės nauja3 narių kiekis:    {0}",
                  nauja3.Count);
Console.WriteLine("   Reikšmės:");
Spausdinti(nauja3);
```




```
   Aibė nauja3
Aibės nauja3 narių kiekis:    3
   Reikšmės:
    Jonas          25
    Petras         26
    Juozas         38

Press any key to continue . . .
```

Maišos aibės HashSet<T> pavyzdžiai (12/34)

```
// Contains()
Asmuo stud5 = new Asmuo("Rita", 20);
Console.WriteLine("   Aibė nauja");
Console.WriteLine("Aibės nauja narių kiekis:    {0}",
                  nauja.Count);
Console.WriteLine("   Reikšmės:"); Spausdinti(nauja);
if (nauja.Contains(stud3))
    Console.WriteLine("rado reikšmę: {0}  {1}", stud3.vardas,
                      stud3.amžius);
else Console.WriteLine("nerado reikšmės: {0}  {1}",
                       stud3.vardas, stud3.amžius);
if (nauja.Contains(stud5))
    Console.WriteLine("rado reikšmę: {0}  {1}", stud5.vardas,
                      stud5.amžius);
else Console.WriteLine("nerado reikšmės: {0}  {1}",
                       stud5.vardas, stud5.amžius);
Console.WriteLine();
```

Maišos aibės HashSet<T> pavyzdžiai (13/34)

```
// Contains()
Console.WriteLine("   Aibė nauja3");
Console.WriteLine("Aibės nauja3 narių kiekis:    {0}",
                  nauja3.Count);
Console.WriteLine("   Reikšmės:");
Spausdinti(nauja3);
if (nauja3.Contains(stud3))
    Console.WriteLine("rado reikšmę: {0}  {1}", stud3.vardas,
                      stud3.amžius);
else Console.WriteLine("nerado reikšmės: {0}  {1}",
                       stud3.vardas, stud3.amžius);
if (nauja3.Contains(stud5))
    Console.WriteLine("rado reikšmę: {0}  {1}", stud5.vardas,
                      stud5.amžius);
else Console.WriteLine("nerado reikšmės: {0}  {1}",
                       stud5.vardas, stud5.amžius);
Console.WriteLine();
```

Maišosaibės HashSet<T> pavyzdžiai (14/34)

stud5

Rita 20

```
   Aibė nauja
Aibės nauja narių kiekis:    4
   Reikšmės:
    Jonas            25
    Petras           26
    Juozas           38
    Kazys            25

rado reikšmę: Kazys  25
nerado reikšmės: Rita  20

Press any key to continue . . .
```

Kodėl rado Kazys 25?

```
   Aibė nauja3
Aibės nauja3 narių kiekis:    3
   Reikšmės:
    Jonas            25
    Petras           26
    Juozas           38

rado reikšmę: Kazys  25
nerado reikšmės: Rita  20

Press any key to continue . . .
```

Maišos aibės HashSet<T> pavyzdžiai (15/34)

```
// spausdina int skaičių masyvą
public static void Spausdinti1(int[] Mas, int kiek)
{
    for (int i = 0; i < kiek; i++)
        Console.Write("    {0}", Mas[i]);
    Console.WriteLine();
}
            ...
// CopyTo()
int[] Mas = new int[20];
int kiek = skaic.Count;
skaic.CopyTo(Mas);
Console.WriteLine("Masyvo iš aibės skaic reikšmės: ");
Spausdinti1(Mas, kiek);
Console.WriteLine();
```

```
Masyvo iš aibės skaic reikšmės:
    0    1    2    3    4    5    6    7    8    9

Press any key to continue . . .
```

Maišos aibės HashSet<T> pavyzdžiai (16/34)

```
Console.WriteLine("Masyvo iš aibės skaic reikšmės: ");
Spausdinti1(Mas, kiek);
Console.WriteLine();

skaic.CopyTo(Mas, 2);
Console.WriteLine("Masyvo iš aibės skaic po antro įterpimo " +
                  "reikšmės: ");
kiek = kiek + 2;
Spausdinti1(Mas, kiek);
Console.WriteLine();
```

```
Masyvo iš aibės skaic reikšmės:
    0    1    2    3    4    5    6    7    8    9

Masyvo iš aibės skaic po antro įterpimo reikšmės:
    0    1    0    1    2    3    4    5    6    7    8    9

Press any key to continue . . .
```

Maišos aibės HashSet<T> pavyzdžiai (17/34)

```
skaic.CopyTo(Mas, 2);
Console.WriteLine("Masyvo iš aibės skaic po antro įterpimo " +
                  "reikšmės: ");
kiek = kiek + 2;
Spausdinti1(Mas, kiek);
Console.WriteLine();
skaic.CopyTo(Mas, 4, 2); // pakeičia masyvo 4 ir 5 elementus
Console.WriteLine("Masyvo iš aibės skaic po trečio įterpimo " +
                  "reikšmės: ");
Spausdinti1(Mas, kiek);
Console.WriteLine();
```

Kodėl nepakeistas?

```
Masyvo iš aibės skaic reikšmės:
    0    1    2    3    4    5    6    7    8    9

Masyvo iš aibės skaic po antro įterpimo reikšmės:
    0    1    0    1    2    3    4    5    6    7    8    9

Masyvo iš aibės skaic po trečio įterpimo reikšmės:
    0    1    0    1    0    1    4    5    6    7    8    9

Press any key to continue . . .
```

Maišos aibės HashSet<T> pavyzdžiai (18/34)

```
// Equals()
// GetEnumerator() naudojamas skaityti, bet ne modifikuoti.
// Geriau ciklas foreach
var enumerator = nauja.GetEnumerator();
Console.WriteLine("Enumeratoriaus pavyzdys");
Console.WriteLine("   Aibė nauja ");
Spausdinti(nauja);
int rastas = 0;
while (enumerator.MoveNext()) {
    object item = enumerator.Current;
    if (item.Equals(stud5)) {
        Console.WriteLine(" Elementas {0} aibėje surastas",
                          item);
        rastas = 1;
        break;
    }
}
if (rastas ==0)
Console.WriteLine("Elementas {0}  {1} aibėje nerastas", stud5.vardas,
                  stud5.amžius);
Console.WriteLine();
```

```
Enumeratoriaus pavyzdys
   Aibė nauja
    Jonas          25
    Petras         26
    Juozas         38
    Kazys          25

Elementas Rita  20 aibėje nerastas

Press any key to continue . . .
```

Maišos aibės HashSet<T> pavyzdžiai (19/34)

```
// Equals()
// GetEnumerator() naudojamas skaityti, bet ne modifikuoti.
// Geriau ciklas foreach
var enumerator = nauja.GetEnumerator();
Console.WriteLine("Enumeratoriaus pavyzdys");
Console.WriteLine("   Aibė nauja ");
Spausdinti(nauja);
int rastas = 0;
while (enumerator.MoveNext()) {
    object item = enumerator.Current;
    if (item.Equals(stud4)) {
        Console.WriteLine(" Elementas {0} aibėje surastas",
                          item);
        rastas = 1;
        break;
    }
}
if (rastas ==0)
Console.WriteLine("Elementas {0}  {1} aibėje nerastas", stud4.vardas,
                  stud4.amžius);
Console.WriteLine();
```

```
Enumeratoriaus pavyzdys
   Aibė nauja
    Jonas          25
    Petras         26
    Juozas         38
    Kazys          25

 Elementas Kazys  25 aibėje surastas

Press any key to continue . . .
```

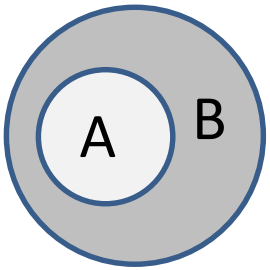
Maišos aAibės HashSet<T> pavyzdžiai (20/34)

```
Console.WriteLine("   Reikšmės nauja3:");
Spausdinti(nauja3);
Console.WriteLine();
Console.WriteLine("   Reikšmės nauja:");
Spausdinti(nauja);
Console.WriteLine();
// ExceptWith()
nauja3.ExceptWith(nauja);
if (nauja3.Count == 0)
    Console.WriteLine("   Aibė nauja3 tuščia \n");
else
{
    Console.WriteLine("   Reikšmės nauja3 po išmetimo:");
    Spausdinti(nauja3);
    Console.WriteLine();
}
```

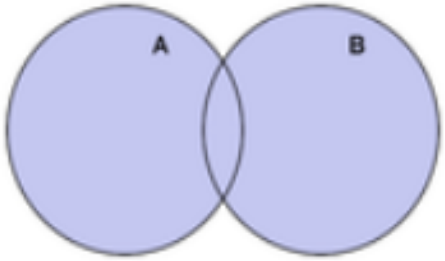
```
   Reikšmės nauja3:
    Jonas          25
    Petras         26
    Juozas         38


   Reikšmės nauja:
    Jonas          25
    Petras         26
    Juozas         38
    Kazys          25


   Aibė nauja3 tuščia

Press any key to continue . . .
```

Maišos aibės HashSet<T> pavyzdžiai (21/34)

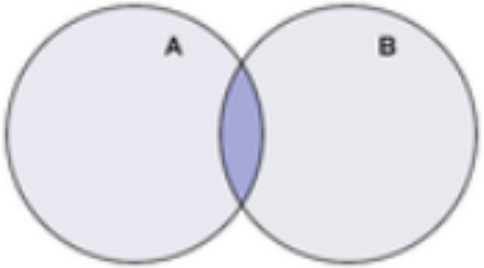
1. Aibės objektai vadinami **elementais** ar nariais.
2. Dvi aibės yra lygios ($A = B$), jei abiejų aibių elementai sutampa.
3. Jei kiekvienas aibės *A* elementas yra ir aibės *B* elementas, aibė *A* yra aibės *B* **poaibis** ir tai žymima $A \subseteq B$. Šiuo atveju aibė *B* yra aibės *A* **viršaibis**.
4. Jei tenkinama sąlyga, kad aibė *A* nelygi *B (*tačiau, kai kurie aibės *B* elementai sutampa su aibės *A* elementais), tai aibė *A* yra aibės *B* **tikrasis (griežtasis) poaibis** ir žymima $A \subset B$. Šiuo atveju aibė *B* yra aibės *A* **tikrasis viršaibis**.



Tikrasis poaibis



Aibių junginys $A \cup B$



Aibių sankirta (pjūvis) $A \cap B$

Maišos aibės HashSet<T> pavyzdžiai (22/34)

```
   Reikšmės nauja3:
    Jonas           25
    Petras          26
    Juozas          38
    Rita            20


   Reikšmės nauja:
    Jonas           25
    Petras          26
    Juozas          38
    Zigmas          40


   Reikšmės nauja3: aibių pjūvis
    Jonas           25
    Petras          26
    Juozas          38


Press any key to continue . . .
```

```
// nauja3.IntersectWith()
Console.WriteLine("   Reikšmės nauja3:");
Spausdinti(nauja3);
Console.WriteLine();
Console.WriteLine("   Reikšmės nauja:");
Spausdinti(nauja);
Console.WriteLine();
nauja3.IntersectWith(nauja);
if (nauja3.Count == 0)Console.WriteLine("   Aibė nauja3 tuščia");
else
{
    Console.WriteLine("   Reikšmės nauja3: aibių pjūvis");
    Spausdinti(nauja3);
    Console.WriteLine();
}
```

Maišos aibės HashSet<T> pavyzdžiai (23/34)

```
// IsProperSubsetOf()
Console.WriteLine("   Reikšmės nauja6:");
Spausdinti(nauja6);
Console.WriteLine();
Console.WriteLine("   Reikšmės nauja:");
Spausdinti(nauja);
Console.WriteLine();
if (!nauja6.IsProperSubsetOf(nauja))
    Console.WriteLine("   nauja6 nėra tikrinis aibės nauja " +
                      "poaibis\n");
else
{
    Console.WriteLine("   nauja6 yra tikrinis aibės nauja " +
                      "poaibis");
    Console.WriteLine();
}
```

Maišos aibės HashSet<T> pavyzdžiai (24/34)

```
   Reikšmės nauja6:
    Jonas           25
    Petras          26
    Juozas          38


   Reikšmės nauja:
    Jonas           25
    Petras          26
    Juozas          38


   nauja6 nėra tikrinis aibės nauja poaibis
Press any key to continue . . .
```

```
   Reikšmės nauja6:
    Jonas           25
    Petras          26
    Juozas          38


   Reikšmės nauja:
    Jonas           25
    Petras          26
    Juozas          38
    Zigmas          40


   nauja6 yra tikrinis aibės nauja poaibis
Press any key to continue . . .
```

Maišos aibės HashSet<T> pavyzdžiai (25/34)

```
// IsProperSupersetOf()
Console.WriteLine("   Reikšmės nauja6:");
Spausdinti(nauja6);
Console.WriteLine();
Console.WriteLine("   Reikšmės nauja:");
Spausdinti(nauja);
Console.WriteLine();
if (!nauja.IsProperSupersetOf(nauja6))
    Console.WriteLine("   nauja nėra tikrinis aibės nauja6 " +
                      "viršaibis\n");
else
{
    Console.WriteLine("   nauja yra tikrinis aibės nauja6 viršaibis");
    Console.WriteLine();
}
```

Maišos aibės HashSet<T> pavyzdžiai (26/34)

```
Reikšmės nauja6:
 Jonas          25
 Petras         26
 Juozas         38


Reikšmės nauja:
 Jonas          25
 Petras         26
 Juozas         38


nauja nėra tikrinis aibės nauja6 viršaibis
Press any key to continue . . .
```

```
Reikšmės nauja6:
 Jonas          25
 Petras         26
 Juozas         38


Reikšmės nauja:
 Jonas          25
 Petras         26
 Juozas         38
 Zigmas         40


nauja yra tikrinis aibės nauja6 viršaibis
Press any key to continue . . .
```

Maišos aibės HashSet<T> pavyzdžiai (27/34)

```
    Reikšmės nauja3:
     Jonas          25
     Petras         26
     Juozas         38
     Rita           20

    Reikšmės nauja:
     Jonas          25
     Petras         26
     Juozas         38

    nauja yra aibės nauja3 poaibis
    nauja3 yra aibės nauja viršaibis
Press any key to continue . . .
```

```
Console.WriteLine("   Reikšmės nauja3:");
Spausdinti(nauja3);
Console.WriteLine();
Console.WriteLine("   Reikšmės nauja:");
Spausdinti(nauja);
Console.WriteLine();
if (!nauja.IsSubsetOf(nauja3))
    Console.WriteLine("   nauja nėra aibės nauja3 poaibis");
else
    Console.WriteLine("   nauja yra aibės nauja3 poaibis");
if (!nauja3.IsSupersetOf(nauja))
    Console.WriteLine("   nauja3 nėra aibės nauja viršaibis");
else
    Console.WriteLine("   nauja3 yra aibės nauja viršaibis");
Console.WriteLine();
```

Maišos aibės HashSet<T> pavyzdžiai (28/34)

```
Console.WriteLine("   Reikšmės nauja7:");
Spausdinti(nauja7);
nauja3.Add(stud5);
Console.WriteLine("   Reikšmės nauja3:");
Spausdinti(nauja3);
Console.WriteLine();
Console.WriteLine("   Reikšmės nauja:");
Spausdinti(nauja);
Console.WriteLine();   // Overlaps()
if (nauja3.Overlaps(nauja))
    Console.WriteLine("   aibės nauja3 ir nauja turi bendrų " +
                      "elementų\n");
else
    Console.WriteLine("   aibės nauja3 ir nauja neturi bendrų " +
                      "elementų\n");
if (nauja7.Overlaps(nauja))
    Console.WriteLine("   aibės nauja7 ir nauja turi bendrų elementų");
else
   Console.WriteLine("   aibės nauja7 ir nauja neturi bendrų elementų");
Console.WriteLine();
```

```
   Reikšmės nauja7:
    Zigmas          40

   Reikšmės nauja3:
    Jonas           25
    Petras          26
    Juozas          38
    Rita            20


   Reikšmės nauja:
    Jonas           25
    Petras          26
    Juozas          38


   aibės nauja3 ir nauja turi bendrų elementų

   aibės nauja7 ir nauja neturi bendrų elementų


Press any key to continue . . .
```

Maišos aibės HashSet<T> pavyzdžiai (29/34)

```
// Remove()
Console.WriteLine("   Reikšmės nauja7:");
Spausdinti(nauja7);
Console.WriteLine("   elementas stud6: {0}  {1} ", stud6.vardas,
                  stud6.amžius);
nauja7.Remove(stud6);
if (nauja7.Count == 0)
    Console.WriteLine("   Aibė nauja7 po išmetimo tuščia \n");
else
{
    Console.WriteLine("   Reikšmės nauja7 po išmetimo:");
    Spausdinti(nauja7);
    Console.WriteLine();
}
```

```
   Reikšmės nauja7:
    Zigmas          40

   elementas stud6: Zigmas  40
   Aibė nauja7 po išmetimo tuščia

Press any key to continue . . .
```

Maišos aibės HashSet<T> pavyzdžiai (30/34)

```
// metų intervalo patikrinimas
public static bool MetuIntervalas(Asmuo elementas)
{
    return elementas.amžius >= 10 && elementas.amžius <= 30;
}
                                        ...

// RemoveWhere()
Console.WriteLine("   Reikšmės nauja:");
Spausdinti(nauja);
nauja.RemoveWhere(MetuIntervalas);
Console.WriteLine("   Išmetami asmenys, kurių metai iš [10; 30] ");
if (nauja.Count == 0) Console.WriteLine("   Aibė nauja po išmetimo tuščia \n");
else
{
    Console.WriteLine("   Reikšmės nauja po išmetimo:");
    Spausdinti(nauja);
    Console.WriteLine();
}
```

```
   Reikšmės nauja:
    Jonas          25
    Petras         26
    Juozas         38

   Išmetami asmenys, kurių metai iš [10; 30]
   Reikšmės nauja po išmetimo:
    Juozas         38


Press any key to continue . . .
```

Maišos aibės HashSet<T> pavyzdžiai (31/34)

```
   Reikšmės nauja:
    Jonas           25
    Petras          26
    Juozas          38

   Reikšmės nauja8:
    Jonas           25
    Petras          26
    Juozas          38

   aibės nauja8 ir nauja yra lygios

   Reikšmės nauja8 po papildymo:
    Jonas           25
    Petras          26
    Juozas          38
    Rita            20

   aibės nauja8 ir nauja nėra lygios
Press any key to continue . . .
```

```
// SetEquals()
Console.WriteLine("   Reikšmės nauja:");
Spausdinti(nauja);
Console.WriteLine("   Reikšmės nauja8:");
Spausdinti(nauja8);
if (nauja8.SetEquals(nauja))
    Console.WriteLine("   aibės nauja8 ir nauja yra lygios\n");
else
    Console.WriteLine("   aibės nauja8 ir nauja nėra lygios\n");
nauja8.Add(stud5);
Console.WriteLine("   Reikšmės nauja8 po papildymo:");
Spausdinti(nauja8);
if (nauja8.SetEquals(nauja))
    Console.WriteLine("   aibės nauja8 ir nauja yra lygios\n");
else
    Console.WriteLine("   aibės nauja8 ir nauja nėra lygios\n");
```

Maišos aibės HashSet<T> pavyzdžiai (32/34)

```
   Reikšmės nauja1:
    Jonas           25
    Petras          26
    Juozas          38
    Zigmas          40

   Reikšmės nauja4:
    Jonas           25
    Petras          26
    Juozas          38
    Rita            20

   Reikšmės nauja4 po modifikavimo:
    Zigmas          40
    Rita            20


Press any key to continue . . .
```

```
// SymmetricExceptWith()
Console.WriteLine("   Reikšmės nauja1:");
Spausdinti(nauja1);
Console.WriteLine("   Reikšmės nauja4:");
Spausdinti(nauja4);
nauja4.SymmetricExceptWith(nauja1);
if (nauja4.Count == 0)
    Console.WriteLine("   Aibė nauja4 po modifikavimo tuščia \n");
else
{
    Console.WriteLine("   Reikšmės nauja4 po modifikavimo:");
    Spausdinti(nauja4);
    Console.WriteLine();
}
```

Maišos aibės HashSet<T> pavyzdžiai (33/34)

```
    Reikšmės nauja1:
     Jonas              25
     Petras             26
     Juozas             38
     Zigmas             40

    Reikšmės nauja4:
     Jonas              25
     Petras             26
     Juozas             38
     Rita               20

    Aibių junginys:
     Jonas              25
     Petras             26
     Juozas             38
     Rita               20
     Zigmas             40


Press any key to continue . . .
```

```
// UnionWith()
Console.WriteLine("   Reikšmės nauja1:");
Spausdinti(nauja1);
Console.WriteLine("   Reikšmės nauja4:");
Spausdinti(nauja4);
nauja4.UnionWith(nauja1);
if (nauja4.Count == 0)
    Console.WriteLine("   Aibių junginys tuščias \n");
else
{
    Console.WriteLine("   Aibių junginys:");
    Spausdinti(nauja4);
    Console.WriteLine();
}
```

Maišos aibės HashSet<T> pavyzdžiai (34/34)

```
Console.WriteLine("   Reikšmės nauja:");
Spausdinti(nauja);
var didz = nauja.Max(elem => elem.amžius);
Console.WriteLine("    Max amžius: {0}.", didz);
var didz1 = nauja.Average(elem => elem.amžius);
Console.WriteLine("    Amžius reikšmių vidurkis: {0, 5:f}.", didz1);
```

```
   Reikšmės nauja:
    Jonas          25
    Petras         26
    Juozas         38

    Max amžius: 38.
    Amžius reikšmių vidurkis: 29,67.
Press any key to continue . . .
```

HashTable Class (System.Collections)

Kolekcijoje *System.Collections* yra specialus maišos aibių konteineris – `HashTable`. Šios aibės elementai panašūs į žodyno `Dictionary<Tkey, TValue>` elementus. Kiekvieną jų sudaro pora: raktas/reikšmė. Žodynus `Dictionary<Tkey, TValue>` galima tiesiogiai perrašyti į šias aibes, naudojant konstruktorių.

`HashTable` klasė – senesnė `Dictionary<Tkey, TValue>` klasės versija.

Klausimai?