ktu informatikos fakultetas 1922

T03. Dėklas, eilė (stack, queue)

3 ak. val.

ktu informatikos fakultetas 1922

Temos klausimai

1. Kolekcijos C#, STL (`Standard Template Library`) C++.
2. Dėklo klasė (`Stack Class (System.Collections)`).
3. Dėklo klasė (`Stack(T) Class (System.Collections.Generic)`).
4. Eilės klasė (`Queue Class (System.Collections)`).
5. Eilės klasė (`Queue(T) Class (System.Collections.Generic)`).

ktu informatikos fakultetas 1922

Kolekcijos C#, STL (Standard Template Library) C++

ktu informatikos fakultetas 1922

STL paskirtis

- STL – tai apibendrintų algoritmų, klasių-konteinerių ir priemonių, pasiekti jų elementus, bei pagalbinių funkcijų rinkinys.
- Visi STL konteineriai yra šabloninės klasės, todėl gali būti panaudoti įvairių tipų duomenims saugoti ir apdoroti.
- STL – tai aukštesnis C++ abstrakcijos lygis, kolekcijos (`System.Collections` arba `System.Collections`. Generic) – tai aukštesnis C# abstrakcijos lygis.

ktu informatikos fakultetas 1922

STL privalumai

- Patikimumas.

Tiek STL algoritmai, tiek STL konteinerių metodai yra patikrinti ilgametėje praktikoje.

- Darbo ir laiko sąnaudų taupymas.

Kuriant savo konteinerius (masyvus, sąrašus, ...), reikėtų programuoti ir jų metodus.

- Universalumas.

STL naudotojas gali kurti savo funkcijas (metodus), tinkančias įvairių tipų STL konteineriams (vektoriams, sąrašams, aibėms, simbolių eilutėms, ...).

ktu informatikos fakultetas 1922

STL sudėtis

Pagrindiniai STL komponentai yra:

- konteineriai;
- algoritmai;
- iteratoriai.

C# iteratoriai nenaudojami, o yra įvesta enumeratoriaus sąvoka. Tai yra nuoroda į kolekcijos elementą.

ktu informatikos fakultetas 1922

Konteinerių paskirtis

- Pradinė STL idėja yra konteineris.
- Konteineris – tai šabloninė klasė duomenų rinkiniams saugoti (kartu su metodais).
- Tos pačios konteinerio klasės objektuose gali būti saugomi skirtingų tipų duomenys, tačiau visi vieno objekto duomenys yra to paties tipo.

Konteinerių tipai (C#, C++)

Konteineriai yra skirstomi į:

• nuosekliuosius, kuriuose kiekvienas elementas turi savo numerį,

• asociatyviuosius, kuriuose elementai pasiekiami, naudojant raktą,

• specializuotuosius, sudaromus iš pirmųjų dviejų tipų konteinerių.

ktu informatikos fakultetas 1922

Nuoseklieji konteineriai

Pagrindiniai:

- vector;
- list;
- deque.

Specializuoti nuoseklieji:

- stack;
- queue;
- priority_queue.

Asociatyvieji konteineriai

Pagrindiniai:

- dictionary;
- hashset;
- sortedlist.

Išrinkimui naudojami raktai.

Raktas – tai gali būti eilutė, tai gali būti skaičius.

Duomenys saugomi medžio tipo struktūroje.

Greitas įterpimas, pašalinimas, paieška.

ktu informatikos fakultetas 1922

Stack Class (System.Collections)

ktu informatikos fakultetas 1922

Dėklas Stack

Dėklas įgyvendina **LIFO** principą:

Last In First Out – paskutinis įėjęs, pirmas išeina.

Dėkle galimos tik 3 operacijos:

- elemento įkėlimas į dėklo viršų (**push**);
- elemento išmetimas iš dėklo viršaus (**pop**);
- viršutinio elemento pažiūrėjimas be išmetimo iš dėklo (**peek**).

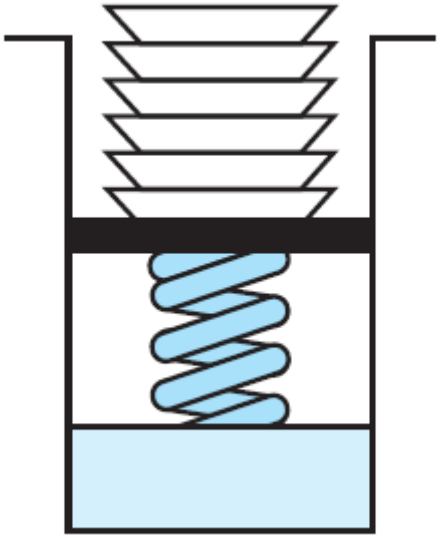
Kitokių operacijų (paieškos, rikiavimo, indekso panaudojimo) dėkle **negalima atlikti**.

Tai labai greitos operacijos.

Šios klasės dėkle galima talpinti skirtingų tipų objektus.

ktu informatikos fakultetas 1922

Dėklo pavyzdys



ktu informatikos fakultetas 1922

Dėklas `Stack`

Dėklas `Stack` turi:

- 3 užklotus konstruktorius.
- 3 savybes (properties).
- 16 metodų.
- 4 užklotus metodus.

Dėklo `Stack` aprašas

Dėklo `Stack` aprašai:

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

Sukuriamas ***tuščias*** (Count = 0) dėklas.

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

Taip sukuriamas naujas dėklas **`Pavadinimas1`**, kuriame yra dėklo **`Pavadinimas`** elementai, surašyti atvirkščia tvarka. Naujo dėklo talpa lygi anksčiau sukurtojo talpai.

3. `Stack Pavadinimas2 = new Stack(talpa);`

Sukuriamas ***tuščias*** (Count = 0) dėklas. Jo talpa – didesnis dydis iš numatytojo ir nurodyto apraše (talpa).

Dažniausiai naudojami dėklo `Stack` metodai

ktu informatikos fakultetas 1922

Metodas arba savybė	Aprašas
Clear()	Pašalina visus dėklo elementus.
Contains(objektas)	Grąžina true, jei ieškoma reikšmė yra, priešingu atveju – false.
CopyTo(masyvas, indeksas)	Kopijuoja dėklo reikšmes į vienmatį masyvą, pradedant nurodytu masyvo indeksu.
Count	Savybė, kuri grąžina dėklo elementų skaičių.
Equals(objektas)	Grąžina true, jei nurodytas objektas lygus nagrinėjamam objektui, priešingu atveju – false. (Paveldėta iš Object)
GetEnumerator()	Grąžina dėklo enumeratorių.
GetType()	Nurodo nagrinėjamo elemento tipą. (Paveldėta iš Object)
Peek()	Grąžina viršutinį dėklo elementą jo neišmetant.
Pop()	Grąžina viršutinį dėklo elementą ir jį išmeta iš dėklo.
Push(objektas)	Įterpia objektą į dėklo viršų.
ToArray()	Kopijuoja dėklą į masyvą.

ktu informatikos fakultetas 1922

`Object` tipas

Visi C# tipai: baziniai bei vartotojo apibrėžti, reikšmės ar nuorodos tipo, yra paveldėti iš Object klasės. Bet kokio tipo reikšmė gali būti priskirta `object` tipo kintamajam. Reikšmės tipo konvertavimas į `object` tipą, vadinamas uždarymu į dėžutę (boxing), o priešingas veiksmas – išėmimu iš dėžutės (unboxing).

```
int sk;
object objektas;
sk = 125;

objektas= sk;          // Boxing
sk= (int)objektas;     // Unboxin
```

Dėklo `Stack` pavyzdžiai (1/11)

```
// Dėklo spausdinimo metodas
static void Spausdinti(Stack stekas)
{
    foreach (Object obj in stekas)
        Console.Write("    {0}", obj);
    Console.WriteLine();
}
```

Dėklo Stack pavyzdžiai (2/11)

```
// Sukuria ir inicializuoja dėklą
Stack stekas = new Stack();
stekas.Push("Vyksta");
stekas.Push("paskaita");
stekas.Push("!");
// Spausdina dėklo savybes ir elementus.
Console.WriteLine("   Dėklas");
Console.WriteLine("Dėklo narių kiekis:    {0}",
                  stekas.Count);
Console.Write("   Reikšmės:");
Spausdinti(stekas);
Console.WriteLine();
```


```
   Dėklas
Dėklo narių kiekis:    3
   Reikšmės:    !    paskaita    Vyksta

Press any key to continue . . .
```

Dėklo Stack pavyzdžiai (3/11)

```
// Sukuria ir inicializuoja dėklą
Stack stekas1 = new Stack(stekas);
stekas1.Push(25.8);
// Spausdina dėklo savybes ir elementus.
Console.WriteLine("   Dėklas1");
Console.WriteLine("Dėklo narių kiekis:    {0}",
                  stekas1.Count);
Console.Write("   Reikšmės:");
Spausdinti(stekas1);
Console.WriteLine();
```

Dėklas1


```
   Dėklas1
Dėklo narių kiekis:    4
   Reikšmės:    25,8    Vyksta    paskaita    !

Press any key to continue . . .
```

Dėklo `Stack` pavyzdžiai (4/11)

```
// Sukuria ir inicializuoja dėklą
Stack stekas2 = new Stack(10);
// Spausdina dėklo savybes
Console.WriteLine("   Dėklas2");
Console.WriteLine("Dėklo narių kiekis:    {0}",
                  stekas2.Count);
```

```
   Dėklas2
Dėklo narių kiekis:    0
Press any key to continue . . .
```

Dėklo Stack pavyzdžiai (5/11)

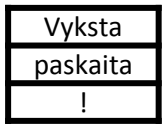
```
// Pop() – šalina viršutinį dėklo elementą
stekas1.Pop();
// Spausdina dėklo savybes ir elementus.
Console.WriteLine("   Dėklas1 po išmetimo");
Console.WriteLine("Dėklo narių kiekis:    {0}",
                  stekas1.Count);
Console.Write("Reikšmės:");
Spausdinti(stekas1);
Console.WriteLine();
```

```
   Dėklas1 po išmetimo
Dėklo narių kiekis:    3
Reikšmės:    Vyksta    paskaita    !

Press any key to continue . . .
```

Dėklo Stack pavyzdžiai (6/11)

```
// Peek() – pažiūri viršutinį dėklo elementą jo neišmetant
object eil = stekas1.Peek();
Console.WriteLine("Dėklo viršutinis narys:    {0}", eil);

// Contains() metodas
if (stekas1.Contains("paskaita"))
    Console.WriteLine("Žodis paskaita rastas");
else Console.WriteLine("Žodis paskaita nerastas ");
Console.WriteLine();
```

Dėklas1

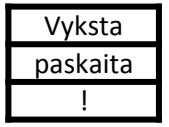



```
Dėklo viršutinis narys:    Vyksta
Žodis paskaita rastas

Press any key to continue . . .
```

Dėklo Stack pavyzdžiai (7/11)

```
// Masyvo spausdinimo metodas
static void Spausdinti1(string [] Mas)
{
    foreach (string obj in Mas)
        Console.Write("    {0}", obj);
    Console.WriteLine();
}
        . . .
Stack stekas3 = new Stack(stekas);
string[] Mas = new string[10];
// CopyTo() metodas
stekas3.CopyTo(Mas, 0);
Console.WriteLine("Masyvo reikšmės: ");
Spausdinti1(Mas);
Console.WriteLine();
```

Dėklas3




```
Masyvo reikšmės:
    Vyksta    paskaita    !

Press any key to continue . . .
```

Dėklo Stack pavyzdžiai (8/11)

```
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;}

    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();}
}
```

Dėklo Stack pavyzdžiai (9/11)

```
// Equals() metodas
Asmuo stud = new Asmuo("Jonas", 25);
stekas3.Push(stud);
Console.WriteLine("Papildytas dėklas3: ");
Spausdinti(stekas3);
if (stud.Equals(stekas3.Peek()))
    Console.WriteLine("Objektai sutampa ");
else Console.WriteLine("Objektai nesutampa");
Console.WriteLine();
```

Užklotas metodas

Kodėl atspausdintas tik vardas?

```
Papildytas dėklas3:
   Jonas    Vyksta    paskaita    !
Objektai sutampa

Press any key to continue . . .
```

Dėklo Stack pavyzdžiai (10/11)

```
// Masyvo spausdinimo metodas
static void Spausdinti2(object[] Mas)
{
    foreach (string obj in Mas)
        Console.Write("    {0}", obj);
    Console.WriteLine();
}
                                . . .
// ToArray() metodas
object  [] Mas1 = new object[10];
Mas1 = stekas.ToArray();
Console.WriteLine("Masyvo iš dėklo stekas reikšmės: ");
Spausdinti2(Mas1);
Console.WriteLine();
```

Tipas **string** negali būti

```
Masyvo iš dėklo stekas reikšmės:
    !    paskaita    Vyksta

Press any key to continue . . .
```

Dėklo Stack pavyzdžiai (11/11)

```
// GetEnumerator() naudojamas skaityti, bet ne
// modifikuoti. Geriau ciklas foreach
IEnumerator pirmas = stekas.GetEnumerator();

Console.WriteLine("Enumeratoriaus pavyzdys");
Console.WriteLine("   Dėklas stekas: ");
while (pirmas.MoveNext())
{
    object item = pirmas.Current;
    Console.Write("    {0}", item);
}
Console.WriteLine();
```

Nuoroda į elementą

Įmamas nagrinėjamas elementas

```
Enumeratoriaus pavyzdys
   Dėklas stekas:
     !     paskaita     Vyksta
Press any key to continue . . .
```

Dėklo `Stack` panaudojimo pavyzdys (1/2)

Duota aritmetinė išraiška. Patikrinti, ar teisingai sudėti skliaustai: ().

Dėklo `Stack` panaudojimo pavyzdys (2/2)

ktu informatikos fakultetas 1922

```
static void Main(string[] args)
{
    string eilute = "9 + (31 + 22 - (21+33)*41 - ((32+1)*(48-22)))+9/(99-88";
    Stack stack = new Stack();
    bool ats = true;
    for (int index = 0; index < eilute.Length; index++)
    {
        char ch = eilute[index];
        if (ch == '(') stack.Push(index);
        else if (ch == ')')
        {
            if (stack.Count == 0)
            {
                ats = false;
                break;
            }
            stack.Pop();
        }
    }
    if (stack.Count != 0) ats = false;
    Console.WriteLine("Ar išraiška teisinga? " +    ats);
}
```

```
Ar išraiška teisinga? False
Press any key to continue . . .
```

Dėklo klasės sukūrimas

Galima sukurti savo dėklo klasę. Tam galima naudoti:

1. fiksuoto ilgio masyvą;
2. dinaminį sąrašą.

Sukursime savo dėklo klasę, panaudojant fiksuoto ilgio masyvą. Realizuosime metodus: Capacity, Clean, Contains, Count, Pop, Peek, Push, ToArray.

```
class Stekas
{
    const int max = 20;
    Object[] stack;
    int i;
    int j;

    // Konstruktoriai
            . . .
    // Metodai
            . . .
}
```

Dėklo klasės konstruktoriai

```
public Stekas()
{
    stack = new Object[max];
    i = 0;          // Dėklo elementų kiekis
    j = max;        // Dėklo talpa
}

public Stekas(int n)
{
    stack = new Object[n];
    i = 0;
    j = n;
}

public Stekas(Stekas stekas1)
{
    j = max; stack = new Object[j];
    for (int k = 0; k < stekas1.Count(); k++)
    {
        stack[stekas1.Count()-k-1] = stekas1.ImtiElementą(k);
    }
    i = stekas1.Count();
}
```

Dėklo klasės metodai 1

```
public Object [] ToArray(out int n)
{
    Object [] stekas1 = new Object[i];
    for (int k = 0; k < i; k++)
        stekas1 [k] = stack [i-k-1];
    n = i;
    return stekas1;
}

public void Clean()
{
    i = 0;
}

public int Capacity()
{
    return stack.Length;
}
```

Dėklo klasės metodai 2

```
public int Count()
{
    return i;
}

public void Push(object item)
{
    if (!isStackFull())
    {
        stack[i++] = item;
    }
    else
    {
        Console.WriteLine("Dėklas pilnas");
        int kiekis = stack.Count();
        Array.Resize(ref stack, 2 * kiekis); // Keičia masyvo dydį
        j = 2*kiekis;
        stack[i++] = item;
    }
}
```

Dėklo klasės metodai 3

```
public bool isStackFull()
{
  if (i == j)
      return true;
  else
      return false;
}

public object Pop()
{
    if (stack.Length != 0)
        return stack[--i];
    Console.WriteLine("Dėklas tuščias");
    return -1;
}

public object Peek()
{
    if (stack.Length != 0)
        return stack[i - 1];
    return 0;
}
```

Dėklo klasės metodai 4

```
public object Contains(Object obj)
{
    for (int k = 0; k < i; k++)
    {
        if (stack[k].Equals(obj))
            return 1;
    }
    return -1;
}

public object ImtiElementą(int i)
{
     return stack[i];
}
```

Dėklo klasės testavimas

```
class Program
{
    static void Main(string[] args)
    {
        // Metodų panaudojimas
                . . .
    }
    // Spausdina dėklą
    public static void Spausdinti(Stekas stekas)
    {
        for (int i = 0; i < stekas.Count(); i++)
            Console.Write("    {0}", stekas.ImtiElementą(stekas.Count()-i-1));
        Console.WriteLine();
    }

    // Spausdina masyvą
    public static void Spausdinti2(object[] Mas, int kiekis)
    {
        for (int i = 0; i < kiekis; i++ )
            Console.Write("  {0}", Mas[i]);
        Console.WriteLine();
    }
}
```

Dėklo klasės metodų panaudojimo pavyzdys 1

```
Stekas naujas = new Stekas(5);
Console.WriteLine("Pradinis dėklo naujas dydys: {0}", naujas.Count());
Console.WriteLine("Pradinė dėklo naujas talpa: {0}", naujas.Capacity());
naujas.Push(10);
naujas.Push(16);
naujas.Push("žodis");
naujas.Push(8);
naujas.Push("tekstas1");
naujas.Push(18);
naujas.Push(118);
naujas.Push(1118);
naujas.Push("tekstas2");

naujas.Push("tekstas2");
naujas.Push(22);
naujas.Push(222);
naujas.Push(2222);
naujas.Push("tekstas3");
Console.WriteLine("Pradinis dėklas naujas:");
Spausdinti(naujas);
```

Dėklas

tekstas3
2222
222
22
tekstas2
tekstas2
1118
118
18
tekstas1
8
žodis
16
10

Dėklo klasės metodų panaudojimo pavyzdys 2

```
Object[] masyvas = new Object[100];
int kiekis;
masyvas = naujas.ToArray(out kiekis);
Console.WriteLine("Masyvas iš dėklo  naujas");
Spausdinti2(masyvas, kiekis);

Console.WriteLine("Dėklo naujas dydys: {0}", naujas.Count());
Console.WriteLine("Dėklo naujas talpa: {0}", naujas.Capacity());
Console.WriteLine("Pažiūrėti dėklo naujas viršutinį elementą: {0}",
                  naujas.Peek());
Console.WriteLine("Išmesti dėklo naujas viršutinį elementą: {0}",
                  naujas.Pop());
Console.WriteLine("Pažiūrėti dėklo naujas viršutinį elementą: {0}",
                  naujas.Peek());
Console.WriteLine("Išmesti dėklo naujas viršutinį elementą: {0}",
                  naujas.Pop());
Console.WriteLine("Dėklo naujas dydys po abiejų išmetimų: {0}",
                  naujas.Count());
```

Dėklo klasės metodų panaudojimo pavyzdys 3

```
Object narys = "" + 10;
Console.WriteLine("Dėkle ieškosime elemento: {0}", narys);
Console.WriteLine("Ieško objekto dėkle: {0}", naujas.Contains(16));

Stekas naujas1 = new Stekas(naujas);
Console.WriteLine("Dėklo naujas1 dydys: {0}", naujas1.Count());
Console.WriteLine("Dėklo naujas1 talpa: {0}", naujas1.Capacity());
Console.WriteLine("Viršutinis dėklo elementas: {0}", naujas1.Peek());

masyvas = naujas1.ToArray(out kiekis);
Console.WriteLine("Masyvas");
Spausdinti2(masyvas, kiekis);

naujas1.Clean();
Console.WriteLine("Dėklo naujas1 dydys po išvalymo: {0}", naujas1.Count());
```

Dėklo klasės metodų panaudojimo rezultatai

```
Pradinis dėklo naujas dydys: 0
Pradinė dėklo naujas talpa: 5
Stekas pilnas
Stekas pilnas
Pradinis dėklas naujas:
    tekstas3    2222    222    22    tekstas2    tekstas2    1118    118    18
  tekstas1    8    žodis    16    10
Masyvas iš dėklo  naujas
  tekstas3  2222  222  22  tekstas2  tekstas2  1118  118  18  tekstas1  8  žodis
  16  10
Dėklo naujas dydys: 14
Dėklo naujas talpa: 20
Pažiūrėti dėklo naujas viršutinį elementą: tekstas3
Išmesti dėklo naujas viršutinį elementą: tekstas3
Pažiūrėti dėklo naujas viršutinį elementą: 2222
Išmesti dėklo naujas viršutinį elementą: 2222
Dėklo naujas dydys po abiejų išmetimų: 12
Dėkle ieškosime elemento: 10
Ieško objekto dėkle: 1
Dėklo naujas1 dydys: 12
Dėklo naujas1 talpa: 20
Viršutinis dėklo elementas: 10
Masyvas
  10  16  žodis  8  tekstas1  18  118  1118  tekstas2  tekstas2  22  222
Dėklo naujas1 dydys po išvalymo: 0
Press any key to continue . . .
```

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

ktu informatikos fakultetas 1922

Dėklas `Stack<T>`

Dėklas `Stack<T>` turi:

- 3 užklotus konstruktorius.
- 1 savybę (properties).
- 15 metodų.
- 5 sąsajos realizacijas (Explicit Interface Implementations).
- Didelę aibę užklotų metodų.

ktu informatikos fakultetas 1922

Dėklo `Stack<T>` aprašas

Dėklo `Stack<T>` aprašai:

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

Sukuriamas ***tuščias*** (Count = 0) dėklas.

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

Taip sukuriamas naujas dėklas **`Pavadinimas1`**, kuriame yra dėklo **`Pavadinimas`** elementai, surašyti atvirkščia tvarka. Naujo dėklo talpa lygi anksčiau sukurtojo talpai.

3. `Stack<T> Pavadinimas2 = new Stack<T>(talpa);`

Sukuriamas ***tuščias*** (Count = 0) dėklas. Jo talpa – didesnis dydis iš numatytojo ir nurodyto apraše (talpa).

Dažniausiai naudojami dėklo `Stack<T>` metodai

Metodas arba savybė	Aprašas
Clear()	Pašalina visus dėklo elementus.
Contains(objektas)	Grąžina true, jei ieškoma reikšmė yra, priešingu atveju – false.
CopyTo(masyvas, indeksas)	Kopijuoja dėklo reikšmes į vienmatį masyvą, pradedant nurodytu masyvo indeksu.
Count	Savybė, kuri grąžina dėklo elementų skaičių.
Equals(objektas)	Grąžina true, jei nurodytas objektas lygus nagrinėjamam objektui, priešingu atveju – false. (Paveldėta iš Object)
GetEnumerator()	Grąžina dėklo enumeratorių.
GetType()	Nurodo nagrinėjamo elemento tipą. (Paveldėta iš Object)
Peek()	Grąžina viršutinį dėklo elementą jo neišmetant.
Pop()	Grąžina viršutinį dėklo elementą ir jį išmeta iš dėklo.
Push(objektas)	Įterpia objektą į dėklo viršų.
ToArray()	Kopijuoja dėklą į masyvą.
TrimExcess()	Sumažina talpą, jei užpildyta mažiau nei 90% esamos talpos.

Dėklo Stack<T> pavyzdžiai (1/13)

ktu informatikos fakultetas 1922

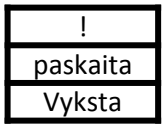
```
// Dėklo spausdinimo metodas
static void Spausdinti(Stack stekas)
{
    foreach (Object obj in stekas)
        Console.Write("    {0}", obj);
    Console.WriteLine();
}
```

Dėklo Stack<T> pavyzdžiai (2/13)

ktu informatikos fakultetas 1922

```
// Sukuria ir inicializuoja dėklą
Stack<string> stekas = new Stack<string>();
stekas.Push("Vyksta");
stekas.Push("paskaita");
stekas.Push("!");
// Spausdina dėklo savybes ir elementus.
Console.WriteLine("   Dėklas");
Console.WriteLine("Dėklo narių kiekis:    {0}",
                  stekas.Count);
Console.Write("   Reikšmės:");
Spausdinti(stekas);
Console.WriteLine();
```

Dėklas




```
   Dėklas
Dėklo narių kiekis:    3
   Reikšmės:    !    paskaita    Vyksta

Press any key to continue . . .
```

Dėklo Stack<T> pavyzdžiai (3/13)

ktu informatikos fakultetas 1922

```
// Sukuria ir inicializuoja dėklą
Stack<string> stekas1 = new Stack<string>(stekas);
stekas1.Push("" +25.8);
// Spausdina dėklo savybes ir elementus.
Console.WriteLine("   Dėklas1");
Console.WriteLine("Dėklo narių kiekis:    {0}",
                  stekas1.Count);
Console.Write("   Reikšmės:");
Spausdinti(stekas1);
Console.WriteLine();
```

Dėklas1


```
   Dėklas1
Dėklo narių kiekis:    4
   Reikšmės:    25,8    Vyksta    paskaita    !

Press any key to continue . . .
```

Dėklo Stack<T> pavyzdžiai (4/13)

ktu informatikos fakultetas 1922

```
// Sukuria ir inicializuoja dėklą
 Stack<string> stekas2 = new Stack<string>(10);
// Spausdina dėklo savybes
Console.WriteLine("   Dėklas2");
Console.WriteLine("Dėklo narių kiekis:    {0}",
                  stekas2.Count);
```

```
   Dėklas2
Dėklo narių kiekis:    0
Press any key to continue . . .
```

Dėklo Stack<T> pavyzdžiai (5/13)

ktu informatikos fakultetas 1922

```
// Pop() – šalina viršutinį dėklo elementą
stekas1.Pop();
// Spausdina dėklo savybes ir elementus.
Console.WriteLine("   Dėklas1 po išmetimo");
Console.WriteLine("Dėklo narių kiekis:    {0}",
                  stekas1.Count);
Console.Write("Reikšmės:");
Spausdinti(stekas1);
Console.WriteLine();
```

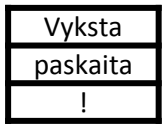
```
   Dėklas1 po išmetimo
Dėklo narių kiekis:    3
Reikšmės:    Vyksta    paskaita    !

Press any key to continue . . .
```

Dėklo Stack<T> pavyzdžiai (6/13)

ktu informatikos fakultetas 1922

```
// Peek() – pažiūri viršutinį dėklo elementą jo neišmetant
object eil = stekas1.Peek();
Console.WriteLine("Dėklo viršutinis narys:    {0}", eil);

// Contains() metodas
if (stekas1.Contains("paskaita"))
    Console.WriteLine("Žodis paskaita rastas");
else Console.WriteLine("Žodis paskaita nerastas ");
Console.WriteLine();
```

Dėklas1



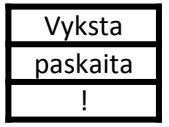

```
Dėklo viršutinis narys:    Vyksta
Žodis paskaita rastas

Press any key to continue . . .
```

Dėklo Stack<T> pavyzdžiai (7/13)

ktu informatikos fakultetas 1922

```
// Masyvo spausdinimo metodas
static void Spausdinti1(string [] Mas)
{
    foreach (string obj in Mas)
        Console.Write("    {0}", obj);
    Console.WriteLine();
}
        . . .
Stack stekas3 = new Stack(stekas);
string[] Mas = new string[10];
// CopyTo() metodas
stekas3.CopyTo(Mas, 0);
Console.WriteLine("Masyvo reikšmės: ");
Spausdinti1(Mas);
Console.WriteLine();
```

Dėklas3




```
Masyvo reikšmės:
    Vyksta    paskaita    !

Press any key to continue . . .
```

Dėklo Stack<T> pavyzdžiai (8/13)

ktu informatikos fakultetas 1922

```
public class Asmuo
   {
       public string vardas { get; set; }
       public int amžius { get; set; }
       public Asmuo(string vardas, int amžius) // Konstruktorius
       {
           this.vardas = vardas;
           this.amžius = amžius;
       }

       public override string ToString() {return this.vardas;}

       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();}
   }
```

Dėklo Stack<T> pavyzdžiai (9/13)

ktu informatikos fakultetas 1922

```
// Objektų dėklo spausdinimo metodas
static void Spausdinti(Stack<Asmuo> stekas)
{
    foreach (Asmuo obj in stekas)
        Console.WriteLine("    {0,-12}   {1}", obj.vardas, obj.amžius);
    Console.WriteLine();
}

Stack<Asmuo> stekas4 = new Stack<Asmuo>();
Asmuo stud = new Asmuo("Jonas", 25);
Asmuo stud1 = new Asmuo("Petras", 25);
Asmuo stud2 = new Asmuo("Juozas", 38);
stekas4.Push(stud);
stekas4.Push(stud1);
stekas4.Push(stud2);
Console.WriteLine("Dėklas4: ");
Spausdinti(stekas4);
```

Dėklo Stack<T> pavyzdžiai (10/13)

ktu informatikos fakultetas 1922

```
// Equals()
if (stud.Equals(stekas4.Peek()))
    Console.WriteLine("Objektai sutampa ");
else Console.WriteLine("Objektai nesutampa");
            Console.WriteLine();
// Contains()
if (stekas4.Contains(stud))
    Console.WriteLine("Objektas: {0}   {1}   rastas",
                      stud.vardas, stud.amžius);
else Console.WriteLine("Objektas: {0}   {1}   nerastas",
                       stud.vardas, stud.amžius);
Console.WriteLine();
```

```
Dėklas4:
    Juozas          38
    Petras          25
    Jonas           25

Objektai nesutampa

Objektas: Jonas   25   rastas

Press any key to continue . . .
```

Dėklo Stack<T> pavyzdžiai (11/13)

ktu informatikos fakultetas 1922

```
// GetEnumerator() Naudojamas skaityti, bet ne modifikuoti.
IEnumerator pirmas = stekas4.GetEnumerator(); //nuoroda
Console.WriteLine("Enumeratoriaus pavyzdys");
Console.WriteLine("   Dėklas stekas4: ");
while (pirmas.MoveNext())
{
    object item = pirmas.Current;
    Asmuo stud3 = (Asmuo)item;
    Console.WriteLine("    {0,-12}   {1}", stud3.vardas,
                      stud3.amžius);
}
Console.WriteLine();
```

```
Dėklas4:
Enumeratoriaus pavyzdys
   Dėklas stekas4:
    Juozas          38
    Petras          25
    Jonas           25

Press any key to continue . . .
```

Dėklo Stack<T> pavyzdžiai (12/13)

ktu informatikos fakultetas 1922

```
// Masyvo spausdinimo metodas
static void Spausdinti2(Asmuo[] Mas)
{
    foreach (Asmuo obj in Mas)
        Console.WriteLine("    {0,-12}   {1}", obj.vardas,
                          obj.amžius);
    Console.WriteLine();
}
                              . . .
// ToArray()
Asmuo[] Mas1 = new Asmuo[20];
Mas1 = stekas4.ToArray();
Console.WriteLine("Masyvo iš dėklo stekas4 reikšmės: ");
Spausdinti2(Mas1);
Console.WriteLine();
```

```
Masyvo iš dėklo stekas4 reikšmės:
    Juozas          38
    Petras          25
    Jonas           25

Press any key to continue . . .
```

Dėklo Stack<T> pavyzdžiai (13/13)

ktu informatikos fakultetas 1922

```
// Kai kurie paveldėti metodai
int suma = stekas4.Sum(elem => elem.amžius);
Console.WriteLine("Dėklo stekas4 elementų suma {0} ",suma);

int didžiausias = stekas4.Max(elem => elem.amžius);
Console.WriteLine("Dėklo stekas4 didžiausias elementas {0} ", didžiausias);

int mažiausias = stekas4.Min(elem => elem.amžius);
Console.WriteLine("Dėklo stekas4 mažiausias elementas {0} ", mažiausias);

double vidurkis = stekas4.Average(elem => elem.amžius);
Console.WriteLine("Dėklo stekas4 elementų vidurkis {0, 8:f} ", vidurkis);
```

```
Dėklas stekas4:
 Juozas          38
 Petras          25
 Jonas           25
```

```
Dėklo stekas4 elementų suma 88
Dėklo stekas4 didžiausias elementas 38
Dėklo stekas4 mažiausias elementas 25
Dėklo stekas4 elementų vidurkis    29,33
Press any key to continue . . .
```

ktu informatikos fakultetas 1922

Queue Class (System.Collections)

ktu informatikos fakultetas 1922

Eilė Queue

Eilė įgyvendina **FIFO** principą:

First In First Out – pirmas įėjęs, pirmas išeina.

Eilėje galimos tik 3 operacijos:

- elemento įkėlimas į eilės pabaigą (**enqueue**);
- elemento išmetimas iš eilės pradžios (**dequeue**);
- pirmo elemento pažiūrėjimas be išmetimo iš eilės (**peek**).

Kitokių operacijų (paieškos, rikiavimo, indekso panaudojimo) eilėje **negalima atlikti**.

Tai labai greitos operacijos.

Šios klasės eilėje galima talpinti skirtingų tipų objektus.

ktu informatikos fakultetas 1922

Eilė **Queue**

Eilė **Queue** turi:

- 4 užklotus konstruktorius.
- 3 savybes (properties).
- 17 metodų.
- 4 užklotus metodus.

Eilės `Queue` aprašas 1

Eilės `Queue` aprašai:

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

Sukuriama ***tuščia*** (Count = 0) eilė.

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

Taip sukuriama nauja eilė **`Pavadinimas1`**, kurioje yra eilės **`Pavadinimas`** elementai. Naujos eilės talpa lygi anksčiau sukurtos talpai.

Eilės `Queue` aprašas 2

3. `Queue Pavadinimas2 = new Queue(talpa);`

Sukuriama ***tuščia*** (Count = 0) eilė. Jos talpa – didesnis dydis iš numatytojo ir nurodyto apraše (talpa).

4. `Queue Pavadinimas3 = new Queue(talpa, augimo_faktorius);`

Sukuriama ***tuščia*** (Count = 0) eilė. Jos talpa – didesnis dydis iš numatytojo ir nurodyto apraše (talpa). Augimo faktorius – realus dydis iš intervalo [1.0; 10.0]. Tipas – `float`.

Dažniausiai naudojami eilės **Queue** metodai

ktu informatikos fakultetas 1922

Metodas arba savybė	Aprašas
Clear()	Pašalina visus eilės elementus.
Contains(objektas)	Grąžina true, jei ieškoma reikšmė yra, priešingu atveju – false.
CopyTo(masyvas, indeksas)	Kopijuoja eilės reikšmes į vienmatį masyvą, pradedant nurodytu masyvo indeksu.
Count	Savybė, kuri grąžina eilės elementų skaičių.
Dequeue()	Grąžina pirmą eilės elementą ir jį išmeta iš eilės.
Enqueue(objektas)	Įkelia objektą į eilės pabaigą.
Equals(objektas)	Grąžina true, jei nurodytas objektas lygus nagrinėjamam objektui, priešingu atveju – false. (Paveldėta iš Object)
GetEnumerator()	Grąžina eilės enumeratorių.
GetType()	Nurodo nagrinėjamo elemento tipą. (Paveldėta iš Object)
Peek()	Grąžina pirmą eilės elementą jo neišmetant.
ToArray()	Kopijuoja eilę į masyvą.

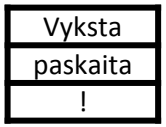
Eilės Queue pavyzdžiai (1/12)

```
// Eilės spausdinimo metodas
static void Spausdinti(Queue eil)
// Gali būti (IEnumerable eil)
{
    foreach (Object obj in eil)
        Console.Write("    {0}", obj);
    Console.WriteLine();
}
```

Eilės Queue pavyzdžiai (2/12)

```
// Sukuria ir inicializuoja eilę
Queue eilė = new Queue();
eilė.Enqueue("Vyksta");
eilė.Enqueue("paskaita");
eilė.Enqueue("!");

// Spausdina eilės savybes ir elementus.
Console.WriteLine("   Eilė");
Console.WriteLine("Eilės narių kiekis:    {0}", eilė.Count);
Console.Write("   Reikšmės:");
Spausdinti(eilė);
```

Eilė




```
   Eilė
Eilės narių kiekis:    3
   Reikšmės:    Vyksta    paskaita    !
Press any key to continue . . .
```

Eilės Queue pavyzdžiai (3/12)

```
// Sukuria ir inicializuoja eilę
Queue eilė1 = new Queue(eilė);
eilė1.Enqueue(25.8);
// Spausdina eilės savybes ir elementus.
Console.WriteLine("   Eilė1");
Console.WriteLine("Eilė1 narių kiekis:    {0}",
                  eilė1.Count);
Console.Write("   Reikšmės:");
Spausdinti(eilė1);
Console.WriteLine();
```

Eilė1

Vyksta
paskaita
!
25,8

```
   Eilė1
Eilė1 narių kiekis:    4
   Reikšmės:    Vyksta    paskaita    !    25,8

Press any key to continue . . .
```

Eilės Queue pavyzdžiai (4/12)

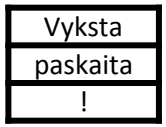
```
// Sukuria ir inicializuoja eilę
Queue eilė2 = new Queue(10);
eilė2.Enqueue("25.8");
// Spausdina eilės savybes ir elementus.
Console.WriteLine("   Eilė2");
Console.WriteLine("Eilė2 narių kiekis:    {0}",
                  eilė2.Count);
Console.Write("   Reikšmės:");
Spausdinti(eilė2);
Console.WriteLine();
```

```
   Eilė2
Eilė2 narių kiekis:    1
   Reikšmės:    25.8

Press any key to continue . . .
```

Eilės Queue pavyzdžiai (5/12)

```
// Sukuria ir inicializuoja eilę
Queue eilė3 = new Queue(2, 2);
eilė3.Enqueue("Vyksta");
eilė3.Enqueue("paskaita");
eilė3.Enqueue("!");
// Spausdina eilės savybes ir elementus.
Console.WriteLine("   Eilė3");
Console.WriteLine("Eilė3 narių kiekis:    {0}",
                  eilė3.Count);
Console.Write("   Reikšmės:");
Spausdinti(eilė3);
Console.WriteLine();
```

Eilė3

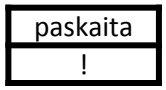



```
   Eilė3
Eilė3 narių kiekis:    3
   Reikšmės:    Vyksta    paskaita    !

Press any key to continue . . .
```

Eilės Queue pavyzdžiai (6/12)

```
// Dequeue()
Console.WriteLine("Išmeta iš eilės3:     {0}",
                  eilė3.Dequeue());
// Spausdina eilės savybes ir elementus.
Console.Write("   Reikšmės po išmetimo:");
Spausdinti(eilė3);
Console.WriteLine();
```

Eilė3

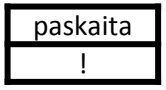



```
Išmeta iš eilės3:     Vyksta
   Reikšmės po išmetimo:    paskaita    !

Press any key to continue . . .
```

Eilės Queue pavyzdžiai (7/12)

```
// Peek()
Console.WriteLine("Pažiūri pirmą eilės3 elementą:   {0}",
                  eilė3.Peek());
// Spausdina eilės savybes ir elementus.
Console.Write("   Eilė3 reikšmės:");
Spausdinti(eilė3);
Console.WriteLine();
// Contains()
if (eilė3.Contains("paskaita"))
    Console.WriteLine("Žodis paskaita rastas");
else Console.WriteLine("Žodis paskaita nerastas ");
Console.WriteLine();
```

Eilė3

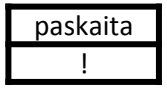



```
Pažiūri pirmą eilės3 elementą:   paskaita
   Eilė3 reikšmės:    paskaita    !

Žodis paskaita rastas

Press any key to continue . . .
```

Eilės Queue pavyzdžiai (8/12)

```
// Masyvo spausdinimo metodas
static void Spausdinti1(string [] Mas)
{
    foreach (string obj in Mas)
        Console.Write("    {0}", obj);
    Console.WriteLine();
}
        . . .
// CopyTo()
string[] Mas = new string[10];
eilė3.CopyTo(Mas, 0);
Console.WriteLine("Masyvo iš eilė3 reikšmės: ");
Spausdinti1(Mas);
Console.WriteLine();
```

Eilė3




```
Masyvo iš eilė3 reikšmės:
    paskaita    !

Press any key to continue . . .
```

Eilės Queue pavyzdžiai (9/12)

```
public class Asmuo
{
    public string vardas { get; set; }
    public int amžius { get; set; }
    public Asmuo(string vardas, int amžius) // Konstruktorius
    {
        this.vardas = vardas;
        this.amžius = amžius;
    }

    public override string ToString() {return this.vardas;}

    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();}
}
```

Eilės Queue pavyzdžiai (10/12)

ktu informatikos fakultetas 1922

```
// Objektų eilės spausdinimo metodas
static void Spausdinti3(Queue eil)
{
    foreach (Asmuo obj in eil)
        Console.WriteLine("    {0,-12}   {1}", obj.vardas,
                          obj.amžius);
    Console.WriteLine();
}
                . . .
// Equals()
Queue eilė4 = new Queue();
Asmuo stud = new Asmuo("Jonas", 25);
Asmuo stud1 = new Asmuo("Petras", 25);
eilė4.Enqueue(stud);
eilė4.Enqueue(stud1);
Console.WriteLine("   Eilė4 reikšmės: ");
Spausdinti3(eilė4);
if (stud.Equals(eilė4.Peek())) Console.WriteLine("Objektai sutampa ");
else Console.WriteLine("Objektai nesutampa");
Console.WriteLine();
```

```
   Eilė4 reikšmės:
    Jonas           25
    Petras          25

Objektai sutampa

Press any key to continue . . .
```

Užklotas metodas

Eilės Queue pavyzdžiai (11/12)

```
// Masyvo spausdinimo metodas
static void Spausdinti2(object[] Mas)
{
    foreach (string obj in Mas)
        Console.Write("    {0}", obj);
    Console.WriteLine();
}
                                . . .
// ToArray()
object[] Mas1 = new object[10];
Mas1 = eilė3.ToArray();
Console.WriteLine("Masyvo iš eilės3 reikšmės: ");
Spausdinti2(Mas1);
Console.WriteLine();
```

Tipas `string` negali būti

```
Masyvo iš eilės3 reikšmės:
    paskaita    !

Press any key to continue . . .
```

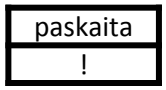
Eilės Queue pavyzdžiai (12/12)

```
// GetEnumerator() naudojamas skaityti, bet ne
// modifikuoti. Geriau ciklas foreach
IEnumerator pirmas = eilė3.GetEnumerator();

Console.WriteLine("Enumeratoriaus pavyzdys");
Console.WriteLine("   Eilė3: ");
while (pirmas.MoveNext())
{
    object item = pirmas.Current;
    Console.Write("    {0}", item);
}
Console.WriteLine();
```

Nuoroda į elementą

Imamas nagrinėjamas elementas

Eilė3




```
Enumeratoriaus pavyzdys
   Eilė3:
    paskaita    !
Press any key to continue . . .
```

Eilės **Queue** panaudojimo pavyzdys (1/19)

ktu informatikos fakultetas 1922

Ūkiai veža žemės ūkio kultūras: rugius, kviečius, miežius ir avižas, saugoti į sandėlį. Tekstiniame faile *Masinos.txt* pateikta informacija: *ūkio pavadinimas, mašinos tipas, kultūra, kiekis, atvykimo prie sandėlio laikas.* Informacija surikiuota atvykimo laiko didėjimo tvarka. Tekstiniame faile *Auto.txt* pateikta informacija apie mašinas: *mašinos tipas, aptarnavimo laikas*. Dėl įvairių atsitiktinių priežasčių mašinos aptarnavimo laikas gali būti kitoks, nei numatyta. Apskaičiuokite, kiek ir kokios kultūros buvo išsaugota sandėlyje per pamainą, jei žinoma pamainos pradžia ir pabaiga. Jei mašina spėja įvažiuoti į sandėlį iki pamainos pabaigos, ji bus iškrauta. Panaudojant atsitiktinių skaičių generatorių, įveskite į skaičiavimus laiko korekcijas ir sumodeliuokite programos darbą. Dėl trukdžių ir kitų faktorių aptarnavimo laikas gali ir didėti, ir mažėti intervale [pr, gal].

Eilės **Queue** panaudojimo pavyzdys (2/19)

ktu informatikos fakultetas 1922

Duomenų failas *Masinos.txt*:

```
Ūkis1;m1;rugiai;6,2;7:20:00
Ūkis2;m1;kviečiai;4;7:50:00
Ūkis3;m2;rugiai;5;9:20:00
Ūkis4;m3;rugiai;10;10:20:00
Ūkis5;m1;kviečiai;5;11:20:00
Ūkis6;m1;rugiai;5;12:20:00
```

Duomenų failas *Auto.txt*:

```
m1;25
m2;30
m3;22
m4;32
m5;45
```

Eilės **Queue** panaudojimo pavyzdys (3/19)

ktu informatikos fakultetas 1922

Rezultatų failas *Rezultatai.txt*:

Krovinių atvežimo grafikas

Nr.	Organizacija	Auto tipas	Kultūra	Kiekis(t)	Atvežimo laikas
1	Ūkis1	m1	rugiai	6,20	07:20:00
2	Ūkis2	m1	kviečiai	4,00	07:50:00
3	Ūkis3	m2	rugiai	5,00	09:20:00
4	Ūkis4	m3	rugiai	10,00	10:20:00
5	Ūkis5	m1	kviečiai	5,00	11:20:00
6	Ūkis6	m1	rugiai	5,00	12:20:00

Auto aptarnavimų lentelė

Nr.	Auto tipas	Aptarnavimo trukmė(min)
1	m1	25
2	m2	30
3	m3	22
4	m4	32
5	m5	45

Per pamainą priimtų kultūrų kiekių lentelė

Nr.	Kultūros tipas	Kiekis(tonos)
1	rugiai	26,20
2	kviečiai	9,00
3	miežiai	0,00
4	avižos	0,00

Eilės Queue panaudojimo pavyzdys (4/19)

ktu informatikos fakultetas 1922

```
// krovinio klasė
class Krovinys //: IComparable<Krovinys>
{
    public string pav { get; set; }         // organizacijos pavadinimas
    public string tipas { get; set; }       // transporto priemonės tipas
    public string kultura { get; set; }     // atvešta žemės ūkio kultūra
    public double kiekis { get; set; }      // kiekis(t)
    public TimeSpan laikas { get; set; }    // atvežimo laikas

    // Konstruktorius be parametrų
    public Krovinys() {   }
    // Konstruktorius su parametrais
    public Krovinys(string pav, string tipas, string kultura, double kiekis, TimeSpan laikas )
    {
        this.pav = pav;
        this.tipas = tipas;
        this.kultura = kultura;
        this.kiekis = kiekis;
        this.laikas = laikas;
    }
    // Užklotas metodas ToString()
    public override string ToString()
    {
        string eilute;
        eilute = string.Format("{0, -12}     {1, -10} {2, -8}     {3, 6:f}      {4} ",
                               pav, tipas, kultura, kiekis, laikas);
        return eilute;
    }
}
```

Eilės Queue panaudojimo pavyzdys (5/19)

ktu informatikos fakultetas 1922

```
// mašinos klasė
class Auto
{
    public string tipas { get; set; }       // transporto priemonės tipas
    public int trukme { get; set; }         // aptarnavimo laikas
    // Konstruktorius be parametrų
    public Auto()
    {
    }

    // Konstruktorius su parametrais
    public Auto(string tipas, int trukme)
    {
        this.tipas = tipas;
        this.trukme = trukme;
    }

    // Užklotas metodas ToString()
    public override string ToString()
    {
        string eilute;
        eilute = string.Format("  {0, -8}        {1} ", tipas, trukme);
        return eilute;
    }
}
```

Eilės Queue panaudojimo pavyzdys (6/19)

ktu informatikos fakultetas 1922

```
// žemės ūkio kultūros klasė
class Kultura
{
    public string pav { get; set; }       // žemės ūkio kultūros pavadinimas
    public double kiekis { get; set; }    // žemės ūkio kultūros kiekis
    // Konstruktorius be parametrų
    public Kultura()
    {
    }

    // Konstruktorius su parametrais
    public Kultura(string pav, double kiekis)
    {
        this.pav = pav;
        this.kiekis = kiekis;
    }

    // Užklotas metodas ToString()
    public override string ToString()
    {
        string eilute;
        eilute = string.Format("{0, -8}                {1,6:f} ", pav, kiekis);
        return eilute;
    }
}
```

Eilės Queue panaudojimo pavyzdys (7/19)

ktu informatikos fakultetas 1922

```
// konstantos
const string CFd =  "..\\..\\Masinos.txt";     // krovinių duomenų failo vardas
const string CFd1 = "..\\..\\Auto.txt";        // auto duomenų failo vardas
const string CFr  = "..\\..\\Rezultatai.txt";  // rezultatų failo vardas
const int pr = -10;  // paklaidos intervalo apatinis rėžis
const int gal = 15;  // paklaidos intervalo viršutinis rėžis
static TimeSpan darboPradzia = new TimeSpan(8, 00, 00);
static TimeSpan darboPabaiga = new TimeSpan(13, 00, 00);
```

Eilės Queue panaudojimo pavyzdys (8/19)

ktu informatikos fakultetas 1922

```
// skaito atvežamų kultūrų failą
static List<Krovinys> SkaitytiKrovinysList(string fv)
{
    // Krovinių objektų dinaminis masyvas
    List<Krovinys> KrovinysList = new List<Krovinys>();
    using (StreamReader srautas = new StreamReader(fv,
           Encoding.GetEncoding(1257)))
    {
        string eilute;  // viena duomenų failo eilutė
        while ((eilute = srautas.ReadLine()) != null)
        {
            string[] eilDalis = eilute.Split(';');
            string pav = eilDalis[0];
            string tipas = eilDalis[1];
            string kultura = eilDalis[2];
            double kiekis = double.Parse(eilDalis[3]);
            TimeSpan laikas = TimeSpan.Parse(eilDalis[4]);
            Krovinys naujas = new Krovinys(pav, tipas, kultura, kiekis, laikas);
            KrovinysList.Add(naujas);
       }
    }
    return KrovinysList;
}
```

Eilės Queue panaudojimo pavyzdys (9/19)

ktu informatikos fakultetas 1922

```
// skaito mašinų aptarnavimo failą
static List<Auto> SkaitytiAutoList(string fv)
{
    // Auto objektų dinaminis masyvas
    List<Auto> AutoList = new List<Auto>();
    using (StreamReader srautas = new StreamReader(fv,
           Encoding.GetEncoding(1257)))
    {
        string eilute;  // viena duomenų failo eilutė
        while ((eilute = srautas.ReadLine()) != null)
        {
            string[] eilDalis = eilute.Split(';');
            string tipas = eilDalis[0];
            int trukme = int.Parse(eilDalis[1]);
            Auto naujas = new Auto(tipas, trukme);
            AutoList.Add(naujas);
        }
    }
    return AutoList;
}
```

Eilės Queue panaudojimo pavyzdys (10/19)

ktu informatikos fakultetas 1922

```
// spausdina atvežamų kultūrų lentelę
static void SpausdintiKroviniuList(string fv, List<Krovinys> KrovinysList,
                                   string antraste)
{
    const string virsus =
    "-------------------------------------------------------------------\r\n"
    + " Nr.  Organizacija   Auto tipas   Kultūra    Kiekis(t)  Atvežimo laikas   \r\n"
    + "-------------------------------------------------------------------";
    // Vietoj Append galima naudoti ir CreateNew
    using (var fr = new StreamWriter(File.Open(fv, FileMode.Append),
                                     Encoding.GetEncoding(1257)))
    {
        fr.WriteLine("\n " + antraste);
        fr.WriteLine(virsus);
        for (int i = 0; i < KrovinysList.Count; i++)
        {
            Krovinys krov = KrovinysList[i];
            fr.WriteLine("{0, 3}   {1}", i + 1, krov);
        }
        fr.WriteLine("-----------------------------------------" +
                     "-------------------------\n");
    }
}
```

Eilės Queue panaudojimo pavyzdys (11/19)

ktu informatikos fakultetas 1922

```
// spausdina mašinų aptarnavimo lentelę
static void SpausdintiAutoList(string fv, List<Auto> AutoList,
                               string antraste)
{
    const string virsus =
         "-----------------------------------------\r\n"
         + " Nr.  Auto tipas   Aptarnavimo trukmė(min)  \r\n"
         + "-----------------------------------------";
    // Vietoj Append galima naudoti ir CreateNew
    using (var fr = new StreamWriter(File.Open(fv, FileMode.Append),
                                     Encoding.GetEncoding(1257)))
   {
       fr.WriteLine("\n " + antraste);
       fr.WriteLine(virsus);
       for (int i = 0; i < AutoList.Count; i++)
       {
           Auto krov = AutoList[i];
           fr.WriteLine("{0, 3}   {1}", i + 1, krov);
       }
       fr.WriteLine("-----------------------------------------\n");
    }
}
```

Eilės Queue panaudojimo pavyzdys (12/19)

ktu informatikos fakultetas 1922

```
// spausdina atvežtų kultūrų kiekių lentelę
static void SpausdintiKulturosList(string fv, List<Kultura> KulturosList,
                                   string antraste)
{
    const string virsus =
          "-----------------------------------------\r\n"
          + " Nr.  Kultūros tipas        Kiekis(tonos)  \r\n"
          + "-----------------------------------------";
    // Vietoj Append galima naudoti ir CreateNew
    using (var fr = new StreamWriter(File.Open(fv, FileMode.Append),
                                     Encoding.GetEncoding(1257)))
    {
        fr.WriteLine("\n " + antraste);
        fr.WriteLine(virsus);
        for (int i = 0; i < KulturosList.Count; i++)
        {
            Kultura krov = KulturosList[i];
            fr.WriteLine("{0, 3}   {1}", i + 1, krov);
        }
        fr.WriteLine("-----------------------------------------\n");
    }
}
```

Eilės Queue panaudojimo pavyzdys (13/19)

ktu informatikos fakultetas 1922

```
// ieško ir grąžina aptarnavimo trukmę reikiamo tipo mašinai
static void Paieska(List<Auto> AutoList, Krovinys elementas, out int rastaTrukme)
{
    rastaTrukme = -1;
    for (int i = 0; i < AutoList.Count; i++)
    {
        if (AutoList[i].tipas == elementas.tipas)
        {
            rastaTrukme = AutoList[i].trukme;
            break;
        }
    }
}
```

Eilės Queue panaudojimo pavyzdys (14/19)

ktu informatikos fakultetas 1922

```
// papildo atvežtos kultūros bendrą kiekį
static void Sumavimas(List<Kultura> KulturosList, Krovinys elementas)
{
    for (int i = 0; i < KulturosList.Count; i++)
    {
        if (KulturosList[i].pav == elementas.kultura)
        {
            KulturosList[i].kiekis = KulturosList[i].kiekis + elementas.kiekis;
            break;
        }
    }
}

// įvertina ir grąžina laiko paklaidą
static TimeSpan LaikoPaklaida(int pr, int galas, int rastaTrukme)
{
    Random rnd = new Random(); // generuoja atsitiktinius dydžius
    string eilute = "00:hh:00";
    int paklaida = rastaTrukme + rnd.Next(pr, gal); // įvertinta paklaida
    string vv = "" + paklaida;
    string vv1 = eilute.Replace("hh", vv); // trukmė eilutės formate
    return TimeSpan.Parse(vv1);
}
```

Eilės Queue panaudojimo pavyzdys (15/19)

ktu informatikos fakultetas 1922

```
// skaičiavimo metodas. Algoritmo aprašymas už metodo kodo.
static void Skaiciavimai(List<Krovinys> KroviniuList, List<Auto> AutoList,
                         List<Kultura> KulturosList, out int apKiekis)
{
    TimeSpan laikas = new TimeSpan(0, 0, 0);         // ciklo parametras
    TimeSpan intervalas = new TimeSpan(0, 1, 0);     // ciklo žingsnis
    TimeSpan aptarnavimoPradzia = new TimeSpan();
    TimeSpan aptarnavimoPabaiga = new TimeSpan();
    Krovinys elementas = new Krovinys();
    int rastaTrukme;    // normatyvinė mašinos aptarnavimo trukmė
    Queue<Krovinys> eile = new Queue<Krovinys>();

    int i = 0;    // Nagrinėjamo krovinio numeris rinkinyje
    // į eilę surašomos mašinos, atvykusios iki darbo pradžios
    while (i < KroviniuList.Count && KroviniuList[i].laikas < darboPradzia)
    {
        eile.Enqueue(KroviniuList[i]);
        i++;
```

Eilės Queue panaudojimo pavyzdys (16/19)

ktu informatikos fakultetas 1922

```
// skaičiavimo metodo tęsinys
    apKiekis = 0;
    int poz = 1;  // papildomas kintamasis: 0 yra aptarnaujama mašina, 1 - nėra
    if (i > 0) {
        // apskaičiuojama pirmos mašinos aptarnavimo pradžia ir pabaiga
        aptarnavimoPradzia = darboPradzia + LaikoPaklaida(0, gal, 0);
        Paieska(AutoList, KroviniuList[0], out rastaTrukme);
        aptarnavimoPabaiga = darboPradzia + LaikoPaklaida(pr, gal, rastaTrukme);
        poz = 0;
    }

    // skaičiavimo ciklas
    for (laikas = darboPradzia; laikas <= darboPabaiga; laikas = laikas + intervalas)
    {
        // skaičiavimai cikle...
```

Eilės Queue panaudojimo pavyzdys (17/19)

ktu informatikos fakultetas 1922

```
// skaičiavimo metodo tęsinys 2

        // skaičiavimai cikle
        if (laikas == aptarnavimoPabaiga)
        {
            // veiksmai su baigta aptarnauti mašina
            apKiekis++;
            Sumavimas(KulturosList, eile.Peek());
            elementas = eile.Dequeue();
            if (eile.Count != 0 && eile.Peek().laikas <= laikas)
            {
                // Jei yra, imama aptarnauti nauja mašina
                poz = 0;
                aptarnavimoPradzia = aptarnavimoPabaiga + LaikoPaklaida(0, gal, 0);
                Paieska(AutoList, eile.Peek(), out rastaTrukme);
                aptarnavimoPabaiga = aptarnavimoPradzia + LaikoPaklaida(pr, gal,
                 rastaTrukme);
             }
             else poz = 1; // eilė tuščia, nėra ką aptarnauti
         }
```

Eilės Queue panaudojimo pavyzdys (18/19)

ktu informatikos fakultetas 1922

```
// skaičiavimo metodo tęsinys 3
        // skaičiavimai tame pačiame cikle
        if (poz == 1)
        {
            if (eile.Count != 0 && eile.Peek().laikas <= laikas)
            {
                // jei sandėlis neturėjo darbo (poz = 1) ir atvyko mašina,
                // ji imama aptarnavimui
                aptarnavimoPradzia = laikas + LaikoPaklaida(0, gal, 0);
                Paieska(AutoList, eile.Peek(), out rastaTrukme);
                aptarnavimoPabaiga = aptarnavimoPradzia + LaikoPaklaida(pr,
                    gal, rastaTrukme);
                poz = 0;
            }
        }
    } // ciklo pabaiga
}
```

Eilės Queue panaudojimo pavyzdys (19/19)

ktu informatikos fakultetas 1922

```
static void Main(string[] args)
{
    int apKiekis = 0;
    Kultura[] array = new Kultura[4]
    {
        new Kultura("rugiai", 0),
        new Kultura("kviečiai", 0),
        new Kultura("miežiai", 0),
        new Kultura("avižos", 0),
    };
    List<Kultura> KulturosList = new List<Kultura>(array);
    if (File.Exists(CFr)) File.Delete(CFr);
    // Krovinių sąrašo sudarymas ir spausdinimas
    List<Krovinys> KroviniuList = SkaitytiKrovinysList(CFd);
    SpausdintiKroviniuList(CFr, KroviniuList, "Krovinių atvežimo grafikas");
    // Auto sąrašo sudarymas ir spausdinimas
    List<Auto> AutoList = SkaitytiAutoList(CFd1);
    SpausdintiAutoList(CFr, AutoList, "Auto aptarnavimų lentelė");
    Skaiciavimai(KroviniuList, AutoList, KulturosList, out apKiekis);
    SpausdintiKulturosList(CFr, KulturosList,
                           "Per pamainą priimtų kultūrų kiekių lentelė");
    Console.WriteLine("Aptarnauta mašinų  {0}", apKiekis);
}
```

Skaičiavimo metodo algoritmas 1

1. Į laukimo eilę surašomos visos mašinos, atvykusios prie sandėlio iki jo darbo pradžios.
2. Jei iki darbo pradžios eilėje jau yra mašinų, apskaičiuojama pirmos jų aptarnavimo laiko pradžia ir pabaiga.
3. Vykdomas ciklas: nuo pamainos pradžios iki pamainos pabaigos žingsniu - viena minutė.
   - Ciklo skaičiavimai...

Skaičiavimo metodo algoritmas 2

Ciklo skaičiavimai...

- Jei nagrinėjamas laikas sutampa su mašinos atvykimo laiku, mašina įtraukiama į laukimo eilę.
- Jei nagrinėjamas laikas sutampa su mašinos aptarnavimo pabaigos laiku, krovinio žemės ūkio kultūra įtraukiama į aptarnautų kultūrų rinkinį, o iškrauta mašina šalinama iš eilės. Tikrinama, ar nagrinėjamu laiku laukimo eilėje yra bent viena mašina. Jei taip, ji paimama aptarnavimui. Papildomas kintamasis poz = 0. Jei eilė tuščia, poz = 1.
- Jei sandėlis neturėjo darbo (poz = 1) ir atvyko mašina, ji imama aptarnavimui. Kintamasis poz = 0.

ktu informatikos fakultetas 1922

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

Eilė Queue<T>

Kaip ir **Queue** klasės eilėje, **Queue<T>** klasės eilė įgyvendina **FIFO** principą:

First In First Out – pirmas įėjęs, pirmas išeina.

Eilėje galimos tik 3 operacijos:

- elemento įkėlimas į eilės pabaigą (**enqueue**);
- elemento išmetimas iš eilės pradžios (**dequeue**);
- pirmo elemento pažiūrėjimas be išmetimo iš eilės (**peek**).

Kitokių operacijų (paieškos, rikiavimo, indekso panaudojimo) eilėje **negalima atlikti**.

Šios klasės eilėje negalima talpinti skirtingų tipų objektų.

ktu informatikos fakultetas 1922

Eilė Queue<T>

Eilė **Queue<T>** turi:

- 3 užklotus konstruktorius.
- 1 savybę (properties).
- 15 metodų.
- 5 sąsajos realizacijas (Explicit Interface Implementations).
- Didelę aibę užklotų metodų (> 130).

Eilės `Queue<T>` aprašas

Eilės `Queue` aprašai:

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

Sukuriama ***tuščia*** (Count = 0) eilė.

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

Taip sukuriama nauja eilė **`Pavadinimas1`**, kurioje yra eilės **`Pavadinimas`** elementai. Naujos eilės talpa lygi anksčiau sukurtos talpai.

3. `Queue Pavadinimas2 = new Queue(talpa);`

Sukuriama ***tuščia*** (Count = 0) eilė. Jos talpa – nurodyta apraše (talpa).

Dažniausiai naudojami eilės Queue<T> metodai

ktu informatikos fakultetas 1922

Metodas arba savybė	Aprašas
Clear()	Pašalina visus eilės elementus.
Contains(objektas)	Grąžina true, jei ieškoma reikšmė yra, priešingu atveju – false.
CopyTo(masyvas, indeksas)	Kopijuoja eilės reikšmes į vienmatį masyvą, pradedant nurodytu masyvo indeksu.
Count	Savybė, kuri grąžina eilės elementų skaičių.
Dequeue()	Grąžina pirmą eilės elementą ir jį išmeta iš eilės.
Enqueue(objektas)	Įkelia objektą į eilės pabaigą.
Equals(objektas)	Grąžina true, jei nurodytas objektas lygus nagrinėjamam objektui, priešingu atveju – false. (Paveldėta iš Object)
GetEnumerator()	Grąžina eilės enumeratorių.
GetType()	Nurodo nagrinėjamo elemento tipą. (Paveldėta iš Object)
Peek()	Grąžina pirmą eilės elementą jo neišmetant.
ToArray()	Kopijuoja eilę į masyvą.

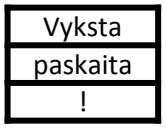
Eilės Queue<T> pavyzdžiai (1/13)

```
// Eilės spausdinimo metodas
public static void Spausdinti(Queue<string> eil)
// Gali būti (IEnumerable eil)
{
    foreach (Object obj in eil)
        Console.Write("    {0}", obj);
    Console.WriteLine();
}
```

Eilės Queue<T> pavyzdžiai (2/13)

```
// Sukuria ir inicializuoja eilę
Queue<string> eilė = new Queue<string>();
eilė.Enqueue("Vyksta");
eilė.Enqueue("paskaita");
eilė.Enqueue("!");

// Spausdina eilės savybes ir elementus.
Console.WriteLine("   Eilė");
Console.WriteLine("Eilės narių kiekis:    {0}", eilė.Count);
Console.Write("   Reikšmės:");
Spausdinti(eilė);
```

Eilė




```
   Eilė
Eilės narių kiekis:    3
   Reikšmės:    Vyksta    paskaita    !
Press any key to continue . . .
```

Eilės Queue<T> pavyzdžiai (3/13)

```
// Sukuria ir inicializuoja eilę
Queue<string> eilė1 = new Queue<string>(eilė);
eilė1.Enqueue(""+25.8);
// Spausdina eilės savybes ir elementus.
Console.WriteLine("   Eilė1");
Console.WriteLine("Eilė1 narių kiekis:    {0}",
                  eilė1.Count);
Console.Write("   Reikšmės:");
Spausdinti(eilė1);
Console.WriteLine();
```

Eilė1

Vyksta
paskaita
!
25,8

```
   Eilė1
Eilė1 narių kiekis:    4
   Reikšmės:    Vyksta    paskaita    !    25,8

Press any key to continue . . .
```

Eilės Queue<T> pavyzdžiai (4/13)

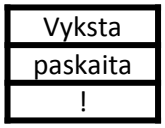
```
// Sukuria ir inicializuoja eilę
Queue<string> eilė2 = new Queue<string>(10);
eilė2.Enqueue("25.8");
// Spausdina eilės savybes ir elementus.
Console.WriteLine("   Eilė2");
Console.WriteLine("Eilė2 narių kiekis:    {0}",
                  eilė2.Count);
Console.Write("   Reikšmės:");
Spausdinti(eilė2);
Console.WriteLine();
```

```
   Eilė2
Eilė2 narių kiekis:    1
   Reikšmės:    25.8

Press any key to continue . . .
```

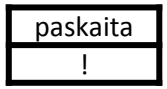
Eilės Queue<T> pavyzdžiai (5/13)

```
// Sukuria ir inicializuoja eilę
Queue<string> eilė3 = new Queue<string>();
eilė3.Enqueue("Vyksta");
eilė3.Enqueue("paskaita");
eilė3.Enqueue("!");
// Dequeue()
Console.WriteLine("Išmeta iš eilės3:     {0}",
                  eilė3.Dequeue());
// Spausdina eilės savybes ir elementus.
Console.Write("   Reikšmės po išmetimo:");
Spausdinti(eilė3);
Console.WriteLine();
```

Eilė3 prieš išmetimą



Eilė3 po išmetimo



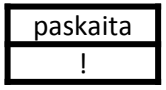

```
Išmeta iš eilės3:     Vyksta
   Reikšmės po išmetimo:    paskaita    !

Press any key to continue . . .
```

Eilės Queue<T> pavyzdžiai (6/13)

ktu informatikos fakultetas 1922

```
// Peek()
Console.WriteLine("Pažiūri pirmą eilės3 elementą:   {0}",
                  eilė3.Peek());
// Spausdina eilės savybes ir elementus.
Console.Write("   Eilė3 reikšmės:");
Spausdinti(eilė3);
Console.WriteLine();
// Contains()
if (eilė3.Contains("paskaita"))
    Console.WriteLine("Žodis paskaita rastas");
else Console.WriteLine("Žodis paskaita nerastas ");
Console.WriteLine();
```

Eilė3



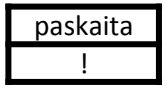

```
Pažiūri pirmą eilės3 elementą:   paskaita
   Eilė3 reikšmės:    paskaita    !

Žodis paskaita rastas

Press any key to continue . . .
```

Eilės Queue<T> pavyzdžiai (7/13)

ktu informatikos fakultetas 1922

```
// Masyvo spausdinimo metodas
static void Spausdinti1(string [] Mas)
{
    foreach (string obj in Mas)
        Console.Write("    {0}", obj);
    Console.WriteLine();
}
        . . .
// CopyTo()
string[] Mas = new string[10];
eilė3.CopyTo(Mas, 0);
Console.WriteLine("Masyvo iš eilė3 reikšmės: ");
Spausdinti1(Mas);
Console.WriteLine();
```

Eilė3




```
Masyvo iš eilė3 reikšmės:
    paskaita    !

Press any key to continue . . .
```

Eilės Queue<T> pavyzdžiai (8/13)

ktu informatikos fakultetas 1922

```
public class Asmuo
{
    public string vardas { get; set; }
    public int amžius { get; set; }
    public Asmuo(string vardas, int amžius) // Konstruktorius
    {
        this.vardas = vardas;
        this.amžius = amžius;
    }

    public override string ToString() {return this.vardas;}

    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();}
}
```

Eilės Queue<T> pavyzdžiai (9/13)

ktu informatikos fakultetas 1922

```
// Objektų eilės spausdinimo metodas
static void Spausdinti3(Queue<Asmuo> eil)
{
    foreach (Asmuo obj in eil)
        Console.WriteLine("    {0,-12}   {1}", obj.vardas,
                          obj.amžius);
    Console.WriteLine();
}
                . . .
// Equals()
Queue<Asmuo> eilė4 = new Queue<Asmuo>();
Asmuo stud = new Asmuo("Jonas", 25);
Asmuo stud1 = new Asmuo("Petras", 25);
eilė4.Enqueue(stud);
eilė4.Enqueue(stud1);
Console.WriteLine("   Eilė4 reikšmės: ");
Spausdinti3(eilė4);
if (stud.Equals(eilė4.Peek())) Console.WriteLine("Objektai sutampa ");
else Console.WriteLine("Objektai nesutampa");
Console.WriteLine();
```

```
   Eilė4 reikšmės:
    Jonas           25
    Petras          25

Objektai sutampa

Press any key to continue . . .
```

Užklotas metodas

Eilės Queue<T> pavyzdžiai (10/13)

```
// Masyvo spausdinimo metodas
public static void Spausdinti2(string[] Mas)
{
    foreach (string obj in Mas)
        Console.Write("    {0}", obj);
    Console.WriteLine();
}
                                . . .
// ToArray()
string[] Mas1 = new string[10];
Mas1 = eilė3.ToArray();
Console.WriteLine("Masyvo iš eilės3 reikšmės: ");
Spausdinti2(Mas1);
Console.WriteLine();
```

```
Masyvo iš eilės3 reikšmės:
    paskaita    !

Press any key to continue . . .
```

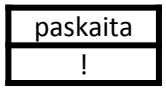
Eilės Queue<T> pavyzdžiai (11/13)

```
// GetEnumerator() naudojamas skaityti, bet ne
// modifikuoti. Geriau ciklas foreach
IEnumerator pirmas = eilė3.GetEnumerator();

Console.WriteLine("Enumeratoriaus pavyzdys");
Console.WriteLine("   Eilė3: ");
while (pirmas.MoveNext())
{
    object item = pirmas.Current;
    Console.Write("    {0}", item);
}
Console.WriteLine();
```

Nuoroda į elementą

Imamas nagrinėjamas elementas

Eilė3




```
Enumeratoriaus pavyzdys
   Eilė3:
    paskaita    !
Press any key to continue . . .
```

Dėklo Queue<T> pavyzdžiai (12/13)

ktu informatikos fakultetas 1922

```
Queue<Asmuo> eilė5 = new Queue<Asmuo>();
Asmuo stud = new Asmuo("Jonas", 25);
Asmuo stud1 = new Asmuo("Petras", 25);
Asmuo stud2 = new Asmuo("Juozas", 38);
eilė5.Enqueue(stud);
eilė5.Enqueue(stud1);
eilė5.Enqueue(stud2);
Console.WriteLine("   Eilė5 reikšmės: ");
Spausdinti3(eilė5);
```

Dėklo Queue<T> pavyzdžiai (13/13)

ktu informatikos fakultetas 1922

```
// Kai kurie paveldėti metodai
int suma = eilė5.Sum(elem => elem.amžius);
Console.WriteLine("Eilės eilė5 elementų suma {0} ",suma);

int didžiausias = eilė5.Max(elem => elem.amžius);
Console.WriteLine("Eilės eilė5 didžiausias elementas {0} ", didžiausias);

int mažiausias = eilė5.Min(elem => elem.amžius);
Console.WriteLine("Eilės eilė5 mažiausias elementas {0} ", mažiausias);

double vidurkis = eilė5.Average(elem => elem.amžius);
Console.WriteLine("Eilės eilė5 elementų vidurkis {0, 8:f} ", vidurkis);
```

```
   Eilė4 reikšmės:
    Jonas          25
    Petras         25
    Juozas         38

Eilės eilė4 elementų suma 88
Eilės eilė4 didžiausias elementas 38
Eilės eilė4 mažiausias elementas 25
Eilės eilė4 elementų vidurkis    29,33
Press any key to continue . . .
```

`Priority_queue` (STL/CLR)

`Priority_queue` – eilė su prioritetais.
Elementai joje išdėstyti prioritetų mažėjimo tvarka.
Kiekvienas talpinamas elementas turi savo prioritetą.
Pagal jį įkeliamas elementas talpinamas į atitinkamą eilės vietą, o ne į eilės galą. Iš eilės imamas elementas turi didžiausią prioritetą.
Galimos operacijos – kaip ir paprastoje eilėje.
Nei *System.Collections*, nei *System.Collections.Generic* prioritetinė klasė nėra realizuota.

Klausimai?