

# T11. Objektų rikiavimo ir paieškos algoritmai

2 ak. val.

# Temos klausimai

1. Objektų rinkinio rikiavimas.
2. Išrinkimo (atkarpos, *MinMax*) rikiavimo algoritmas.
3. Paieškos rikiuotame masyve algoritmas.
4. Dvejainės paieškos algoritmas.



# *Objektų rinkinio rikiavimas*

# Rikiavimo uždavinys

Išdėstyti objektų masyvo elementus taip, kad nurodytos objektų kintamųjų reikšmės sudarytų *didėjančią* (*mažėjančią*) seką.

Rikiuojant reikia nurodyti kokį objektų kintamąjį (-uosius) naudosite rikiavimui.

Rikiavimą galima atlikti naudojant tiek skaitines, tiek ir simbolines kintamųjų reikšmes.

# Pavyzdys (1/6)

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

Perskaitykite duomenis į objektų masyvą (konteinerį) ir juos išspausdinkite rezultatų faile.

Surikiuokite objektų masyvą (konteinerį) **pagal pažymius** nuo geriausio iki blogiausio (*mažėjančia tvarka*).

# Pavyzdys (2/6)

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

# Pavyzdys (3/6)

## Studentų sąrašas

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

# Pavyzdys (4/6)

## Studentų sąrašas

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

**Pastaba:** sąrašas surikiuotas pagal pažymius (mažėjimo tvarka).



# Pavyzdys (5/6)

## Studentų sąrašas

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

**Pastaba:** sąrašas surikiuotas pagal pavardes (didėjimo tvarka).

# Pavyzdys (6/6)

## Studentų sąrašas

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

**Pastaba:** sąrašas surikiuotas pagal pažymius ir pavardes.



# *Išrinkimo rikiavimo algoritmas*

# Išrinkimo rikiavimo algoritmas

Sutrumpintai vadinamas **Minmax** rikiavimo algoritmu.

Tarkime, reikia skaičių masyvo **A(n)** elementus išdėstyti taip, kad skaitinės reikšmės sudarytų *didėjančią* (*mažėjančią*) seką.

111 34 9 17 -55 99

-55 9 17 34 99 111

Didėjanti seka

111 99 34 17 9 -55

Mažėjanti seka

# Išrinkimo rikiavimo algoritmas didėjančia tvarka

1. Mažiausią reikšmę surandame visame masyve, t. y. nuo nulinio iki  $(n-1)$ -ojo elementų. Sukeičiame vietomis surastojo elemento ir nulinio elemento reikšmes.
2. Surandame mažiausią reikšmę turintį elementą masyvo intervale nuo 1-ojo iki  $(n-1)$ -ojo elementų. Sukeičiame vietomis surastojo elemento ir 1-ojo elemento reikšmes.
3. Surandame mažiausią reikšmę turintį elementą masyvo intervale nuo 2-ojo iki  $(n-1)$ -ojo elementų. Sukeičiame vietomis surastojo elemento ir 2-ojo elemento reikšmes.

*Taip darome tol, kol intervale lieka tik viena reikšmė.*

Atlikus visus veiksmus masyvo elementų reikšmės bus išdėstytos **didėjimo** tvarka.

# Išrinkimo rikiavimo algoritmas didėjančia tvarka

Mažiausią reikšmę surandame visame masyve, t. y. nuo nulinio iki n-1-ojo elemento.

<b>111</b>	<b>34</b>	<b>9</b>	<b>17</b>	<b>-55</b>	<b>99</b>
0	1	2	3	4	5



Sukeičiame vietomis surastąjį elementą ir nulinio elemento reikšmę.

<b>-55</b>	<b>34</b>	<b>9</b>	<b>17</b>	<b>111</b>	<b>99</b>
0	1	2	3	4	5

# Išrinkimo rikiavimo algoritmas didėjančia tvarka

Mažiausią reikšmę surandame visame masyve, t. y. nuo pirmojo iki n-1-ojo elemento.

<b>-55</b>	<b>34</b>	<b>9</b>	<b>17</b>	<b>111</b>	<b>99</b>
0	1	2	3	4	5



Sukeičiame vietomis surastąjį elementą ir pirmojo elemento reikšmę.

<b>-55</b>	<b>9</b>	<b>34</b>	<b>17</b>	<b>111</b>	<b>99</b>
0	1	2	3	4	5

# Išrinkimo rikiavimo algoritmas didėjančia tvarka

Mažiausią reikšmę surandame visame masyve, t. y. nuo antrojo iki n-1-ojo elemento.

<b>-55</b>	<b>9</b>	<b>34</b>	<b>17</b>	<b>111</b>	<b>99</b>
0	1	2	3	4	5



Sukeičiame vietomis surastąjį elemento ir antrojo elemento reikšmes.

<b>-55</b>	<b>9</b>	<b>17</b>	<b>34</b>	<b>111</b>	<b>99</b>
0	1	2	3	4	5



# Išrinkimo rikiavimo algoritmas didėjančia tvarka

Mažiausią reikšmę surandame visame masyve, t. y. nuo trečiojo iki n-1-ojo elemento.

<b>-55</b>	<b>9</b>	<b>17</b>	<b>34</b>	<b>111</b>	<b>99</b>
0	1	2	3	4	5



Sukeičiame vietomis surastąjį elementą ir trečiojo elemento reikšmes.

<b>-55</b>	<b>9</b>	<b>17</b>	<b>34</b>	<b>111</b>	<b>99</b>
0	1	2	3	4	5

# Išrinkimo rikiavimo algoritmas didėjančia tvarka

Mažiausią reikšmę surandame visame masyve, t. y. nuo ketvirtojo iki n-1-ojo elemento.

<b>-55</b>	<b>9</b>	<b>17</b>	<b>34</b>	<b>111</b>	<b>99</b>
0	1	2	3	4	5



Sukeičiame vietomis surastąjį elementą ir ketvirtojo elemento reikšmes.

<b>-55</b>	<b>9</b>	<b>17</b>	<b>34</b>	<b>99</b>	<b>111</b>
0	1	2	3	4	5

# Išrinkimo rikiavimo algoritmas didėjančia tvarka

Mažiausią reikšmę surandame visame masyve, t. y. nuo penktojo iki n-1-ojo elemento.

<b>-55</b>	<b>9</b>	<b>17</b>	<b>34</b>	<b>99</b>	<b>111</b>
0	1	2	3	4	5



Sukeičiame vietomis surastąjį elementą ir penktojo elemento reikšmes.

<b>-55</b>	<b>9</b>	<b>17</b>	<b>34</b>	<b>99</b>	<b>111</b>
0	1	2	3	4	5

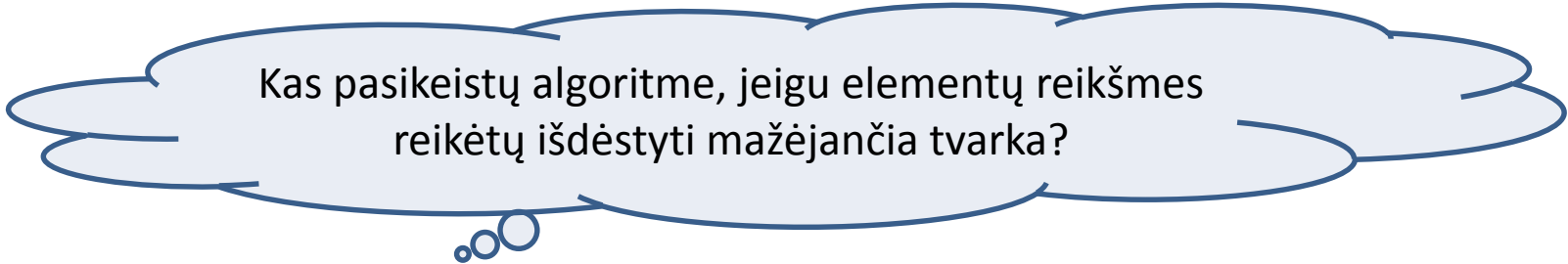
# Išrinkimo rikiavimo algoritmas didėjančia tvarka

Masyvo **Mas(kiek)** elementų reikšmių rikiavimui atlikti reikalingi du ciklai – išorinis ir vidinis (**ciklas cikle**).

**Išoriniame cikle**  $[0; \text{kiek}-2]$  fiksuojama mažiausio elemento reikšmės vieta **i**.

**Vidiniame cikle**  $[i+1; \text{kiek}-1]$  surandama mažiausio elemento vieta (indeksas) **minInd**.

Po to sukeičiamos **i**-ojo ir **minInd** elementų reikšmės vietomis.



Kas pasikeistų algoritme, jeigu elementų reikšmės  
reikėtų išdėstyti mažėjančia tvarka?

# Išrinkimo rikiavimo metodas skaičių masyvui

// surikiuoja masyvą Mas(kiek) skaičių didėjimo tvarka

static void MinMax(int[] Mas, int kiek)

{

int pag = 0; // pagalbinis kintamasis reikšmių sukeitimui

int minInd; // mažiausios reikšmės elemento indeksas

for (int i = 0; i < kiek - 1; i++)

{

minInd = i;

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

{

if (Mas[j] < Mas[minInd])

minInd = j;

}

pag = Mas[i];

Mas[i] = Mas[minInd];

Mas[minInd] = pag;

}

}

Mažiausios reikšmės  
elemento indekso  
paieškos ciklas

Dviejų elementų  
reikšmių sukeitimas  
vietomis



# *Išrinkimo rikiavimo algoritmo taikymo pavyzdys*

# Klasė Studentas

```
class Studentas
```

```
{
```

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

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

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

```
    {
```

```
        pavVrd = pavv;
```

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

```
    }
```

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

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

```
    ... // užkloti metodai ir užkloti operatoriai
```

```
}
```

# Klasė **Studentas** (papildyta užklotais operatoriais < ir >) (1/3)

```
class Studentas
{
    ...
    // užklotas operatorius < (lyginami pažymiai)
    public static bool operator <(Studentas stud1, Studentas stud2)
    {
        return stud1.pazym < stud2.pazym;
    }
    // užklotas operatorius > (lyginami pažymiai)
    public static bool operator >(Studentas stud1, Studentas stud2)
    {
        return stud1.pazym > stud2.pazym;
    }
}
```



# Klasė **Studentas** (papildyta užklotais operatoriais < ir >) (2/3)

```
class Studentas
```

```
{
```

```
    ...
```

```
    // Užklotas operatorius < (lyginamos pavardės)
```

```
    public static bool operator <(Studentas stud1, Studentas stud2)
```

```
    {
```

```
        int poz = String.Compare(stud1.pavVrd, stud2.pavVrd,  
                                StringComparison.CurrentCulture);
```

```
        return poz < 0;
```

```
    }
```

```
    // Užklotas operatorius > (lyginamos pavardės)
```

```
    public static bool operator >(Studentas stud1, Studentas stud2)
```

```
    {
```

```
        int poz = String.Compare(stud1.pavVrd, stud2.pavVrd,  
                                StringComparison.CurrentCulture);
```

```
        return poz > 0;
```

```
    }
```

```
}
```

# Klasė **Studentas** (papildyta užklotais operatoriais $\geq$ ir $\leq$ ) (3/3)

```
class Studentas
{
    ...
    // užklotas operatorius  $\geq$  (lyginami pažymiai ir pavardės)
    public static bool operator  $\geq$ (Studentas stud1, Studentas stud2)
    {
        int poz = String.Compare(stud1.pavVrd, stud2.pavVrd,
                                   StringComparison.CurrentCulture);
        return (stud1.pazym > stud2.pazym) ||
            ((stud1.pazym == stud2.pazym) && (poz > 0));
    }
    // užklotas operatorius  $\leq$  (lyginami pažymiai ir pavardės)
    public static bool operator  $\leq$ (Studentas stud1, Studentas stud2)
    {
        int poz = String.Compare(stud1.pavVrd, stud2.pavVrd,
                                   StringComparison.CurrentCulture);
        return (stud1.pazym < stud2.pazym) ||
            ((stud1.pazym == stud2.pazym) && (poz > 0));
    }
}
```

# Konteinerinė klasė (papildyta rikiavimo metodu)

```
// Konteinerinė klasė studentų duomenims aprašyti
```

```
class MasyvasStudentai
```

```
{
```

```
    const int Cn = 50;           // studentų masyvo dydis
```

```
    private Studentas [] Studentai; // studentų objektų masyvas
```

```
    private int kiek;           // studentų skaičius
```

```
    public MasyvasStudentai()
```

```
    {
```

```
        kiek = 0;
```

```
        Studentai = new Studentas[Cn];
```

```
    }
```

```
    public Studentas ImtiStudenta(int i) { return Studentai[i]; }
```

```
    public int ImtiKiek() { return kiek; }
```

```
    public void DetiStudenta(Studentas obj) { Studentai[kiek++] = obj; }
```

```
    public void MinMax() { ... }
```

```
}
```

# Rikiavimo metodas

```
class MasyvasStudentai
```

```
{ ...
```

```
// Rikiuoja studentų masyvą pagal užklotame operat. nurodytą požymį
```

```
public void MinMax()
```

```
{
```

```
    int maxInd;        // elemento su didžiausia reikšme indeksas
```

```
    for (int i = 0; i < kiek - 1; i++)
```

```
    {
```

```
        maxInd = i;
```

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

```
        {
```

```
            if (Studentai[j] > Studentai[maxInd])  
                maxInd = j;
```

```
        }
```

```
        Studentas stud = Studentai[i];
```

```
        Studentai[i] = Studentai[maxInd];
```

```
        Studentai[maxInd] = stud;
```

```
    }
```

```
}
```

```
}
```

Objektų palyginimui  
naudojamas klasės  
**Studentas** užklotas  
operatorius >

# Main() metodas

```
static void Main(string[] args)
{
    MasyvasStudentai TestasMas = new MasyvasStudentai();

    SkaitytiStudKont(CFd1, TestasMas);
    SpausdintiStudKont(CFr, TestasMas, "Studentų sąrašas");

    TestasMas.MinMax();
    SpausdintiStudKont(CFr, TestasMas,
                       "Studentų sąrašas (surikiuotas)");
}
```

Rikiavimo rezultatas yra pateiktas anksčiau (žiūr. 6, 7 ir 9 skaidrės).



# *Nuoseklios paieškos surikiuotame masyve algoritmas*

Pavyzdžiui, jeigu rikiuotame (*didėjimo tvarka*) skaičių masyve ieškome skaičiaus **30**,

		Masyvo dydis: 7						
Mas	----->	-8	5	12	45	56	78	99
		0	1	2	3	4	5	6

tai einant nuo masyvo pradžios ir sutikus reikšmę **45** galime nutraukti paiešką, nes ieškomos reikšmės tikrai toliau nebus.

**Išvada:** Paiešką galima nutraukti, kai ji tampa beprasmė.

# Nuosekli paieška rikiuotame masyve

```
// Gražina surasto elemento indeksą, arba
// -1, jei tokios reikšmės sk masyve A(n) nebuvo
static int IndeksasRikiuotame(int[] Mas, int kiek, int sk)
{
    for (int i = 0; i < kiek; i++)
    {
        if (Mas[i] == sk)
            return i;    // elementas surastas
        else if (Mas[i] > sk)
            return -1;    // elementas nerastas, toliau ieškoti netikslinga
    }
    return -1;            // elementas nerastas
}
```

**Pastaba:** masyvas Mas(kiek) yra surikiuotas reikšmių didėjimo tvarka.



# Nuosekli paieška rikiuotame objektų masyve

```
// Ieško rikiuotame pagal pavardes masyve Studentai(kiek) studento stud
// Gražina surasto studento indeksą; jei neranda gražina -1
static int NuosekliPaieška(Studentas[] Studentai, int kiek, Studentas stud)
{
    for (int i = 0; i < kiek; i++)
    {
        if (Studentai[i] == stud) // naudojamas užklotas operatorius ==
            return i;           // studentas surastas
        else
            if (Studentai[i] >= stud) // naudojamas užklotas operatorius >=
                return -1; // studentas nerastas, toliau ieškoti netikslinga
    }
    return -1; // studentas nerastas
}
```

## Pastaba:

1. Objektų masyvas Studentai(kiek) yra *surikiuotas pagal pavardes* (didėjimo tvarka).
2. Užkloti klasės **Studentas** operatoriai **==** ir **>=** lygina dviejų studentų pavardes.

# Nuoseklios paieškos taikymo rikiuotame objektų masyve pavyzdys

Pavyzdžiui, surikiuotame pagal pavardes (*didėjimo tvarka*) studentų masyve :

## Studentų sąrašas

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

reikia surasti du studentus: **Jonaitis Jonas** ir **Jonauskas Julius**.

Atlikus paiešką:

pirmasis studentas bus surastas ir paieška bus nutraukta 3-iame cikle, o antrasis studentas nebus surastas, paieška bus nutraukta 4-ame cikle (toliau ieškoti nėra prasmės).



# *Dvejtainė paieška surikiuotame masyve*

# Dvejtainės paieškos iliustracija

Sakykim, Afrikoje yra liūtas – jį reikia sugauti.

Dalinam Afriką į dvi lygias dalis ir, tarkim, nustatom, kad liūtas yra kairėje Afrikos pusėje.

Šią dalį dalinam į dvi lygias dalis ir nustatom, kad liūtas yra kairėje (dešinėje) Afrikos pusėje.

Taip dalinant per pusę dalį, kur yra liūtas, jis galiausiai atsidurs narve.

Dvejetainę paiešką galima atlikti **tik surikiuotame masyve**.

Žodinis algoritmo aprašymas:

Pirmiausia ieškoma reikšmė palyginama su elemento reikšme, esančia masyvo viduryje:

- jeigu lyginamos reikšmės sutampa, paieška baigiama;
- jeigu nesutampa, masyvas padalijamas į dvi dalis ir toliau ieškoma toje dalyje, kurioje galėtų būti ieškoma reikšmė. Tai nustatoma iš palyginimo rezultatų: ar ieškoma reikšmė buvo didesnė, ar mažesnė už viduriniają.

Paiešką kartojant, paieškos sritis kaskart dvigubai sumažėja, kol randama reikšmė, arba kol paieškos srities nelieka ir reikšmė nerandama.

Sakykim, **pi** žymi masyvo  $A(n)$  pradžią (0),  
**gi** – masyvo pabaigą ( $n-1$ ), o  
**vi** – masyvo vidurį  $vi = (pi + gi) / 2$ .

Cikle (kol **pi** ≤ **gi**) lyginant masyvo vidurio elemento reikšmę  $A[vi]$  su ieškoma reikšme, gaunama viena iš trijų situacijų:

- 1) ieškoma reikšmė sutampa su  $A[vi]$ : surasta, grąžinama **vi**.
- 2) ieškoma reikšmė mažiau už  $A[vi]$ : imama *kairė dalis*, **gi** priskirti  $vi-1$ .
- 3) ieškoma reikšmė daugiau už  $A[vi]$ : imama *dešinė dalis*, **pi** priskirti  $vi+1$ .

# Dvejtainė paieška (1/4)

ieškom = 15

0	1	2	3	4	5	6	7	8
3	4	5	9	10	13	15	29	45

Diagram illustrating the search process for the value 15 in a sorted array. The array is indexed from 0 to 8. The current search range is from  $pi$  (index 0) to  $gi$  (index 8). The middle element  $A[vi]$  (index 4, value 10) is compared to the target 15. Since  $A[vi] < 15$ , the search range is updated to  $pi = vi + 1$ .

$A[vi] < 15$ , tai  $pi = vi + 1$ .

0	1	2	3	4	5	6	7	8
3	4	5	9	10	13	15	29	45

Diagram illustrating the search process for the value 15 in a sorted array. The array is indexed from 0 to 8. The current search range is from  $pi$  (index 5) to  $gi$  (index 8). The middle element  $A[vi]$  (index 6, value 15) is compared to the target 15. Since  $A[vi] == 15$ , the element is found.

$A[vi] == 15$ , elementas surastas

# Dvejtainė paieška (2/4)

ieškom = 5

0	1	2	3	4	5	6	7	8
3	4	5	9	10	13	15	29	45

Diagram illustrating the search process for the value 5 in a sorted array. The array is indexed from 0 to 8. The current search range is defined by  $pi$  (0) and  $gi$  (8). The middle element  $A[vi]$  (10) is compared to the target value 5. Since  $A[vi] > 5$ , the search range is updated to  $gi = vi - 1$ .

$A[vi] > 5$ , tai  $gi = vi - 1$ .

0	1	2	3	4	5	6	7	8
3	4	5	9	10	13	15	29	45

Diagram illustrating the search process for the value 5 in a sorted array. The array is indexed from 0 to 8. The current search range is defined by  $pi$  (0) and  $gi$  (3). The middle element  $A[vi]$  (4) is compared to the target value 5. Since  $A[vi] < 5$ , the search range is updated to  $pi = vi + 1$ .

$A[vi] < 5$ , tai  $pi = vi + 1$ .



# Dvejtainė paieška (3/4)

ieškom = 5

0	1	2	3	4	5	6	7	8
3	4	5	9	10	13	15	29	45



pi	gi
vi	

$A[vi] == 5$ , elementas surastas.

Savarankiškai: raskite masyvo elementą, kurio reikšmė yra 30.

# Dvejtainė paieška (4/4)

```
static int DvejtainėPaieška(int[] Mas, int kiek, int x)
{ // Intervalo pradžios, pabaigos ir vidurio indeksai
  int pi, gi, vi;
  pi = 0;  gi = n-1;
  while (pi <= gi) // kol nesusikirs intervalo rėžiai
  {
    vi = (pi + gi)/2;
    if (Mas[vi] == x)
      return vi; // elementas surastas
    else
      if (Mas[vi] < x) pi = vi + 1;
      else            gi = vi - 1;
  }
  return -1; // elementas nerastas
}
```

# Pavyzdys: dvejetainė paieška objektų masyve

```
static int DvejetainėPaieška(Studentas[] Studentai, int kiek, Studentas stud)
{
    int pi, gi, vi;    // intervalo pradžios, pabaigos ir vidurio indeksai
    pi = 0;    gi = kiek - 1;
    while (pi <= gi)    // kol nesusikirs intervalo rėžiai
    {
        vi = (pi + gi) / 2;
        int poz = String.Compare(Studentai[vi].ImtiPavv(), stud.ImtiPavv(),
                                   StringComparison.CurrentCulture);

        if (poz == 0)
            return vi;    // studentas surastas
        else
            if (poz < 0)
                pi = vi + 1;
            else
                gi = vi - 1;
    }
    return -1;    // studentas nerastas
}
```

**Pastaba:** Studentų masyvas Studentai(kiek) surikiuotas pagal pavardes (didėjimo tvarka). Dviejų studentų palyginimui pagal pavardes galima naudoti užklotą operatorių.

# Paieškos masyve darbo imlumo palyginimas

Reikia vertinti blogiausią atvejį – ieškomos reikšmės masyve nėra.

Jeigu masyvas:

**nerikiuotas** – reikia peržiūrėti visą masyvą.

**rikiuotas** – gali reikėti *nuosekliai* peržiūrėti visą masyvą.

**rikiuotas** – *dvejjetainė paieška* nutrauks peržiūrą, neperžiūrėjus viso masyvo.

# Šioje temoje susipažinsite su:

1. Objektų rinkinio rikiavimu.
2. Išrinkimo (atkarpos, *MinMax*) rikiavimo algoritmu.
3. Paieškos rikiuotame masyve algoritmu.
4. Dvejetainės paieškos algoritmu.



*Klausimai?*