

T04. Tekstiniai failai. Klasės, metodai.

2 ak. val.

Temos klausimai

1. Tekstinis duomenų failas.
2. Klasės konstruktorius.
3. Metodai. Duomenų perdavimas ir grąžinimas.



Tekstinis duomenų failas

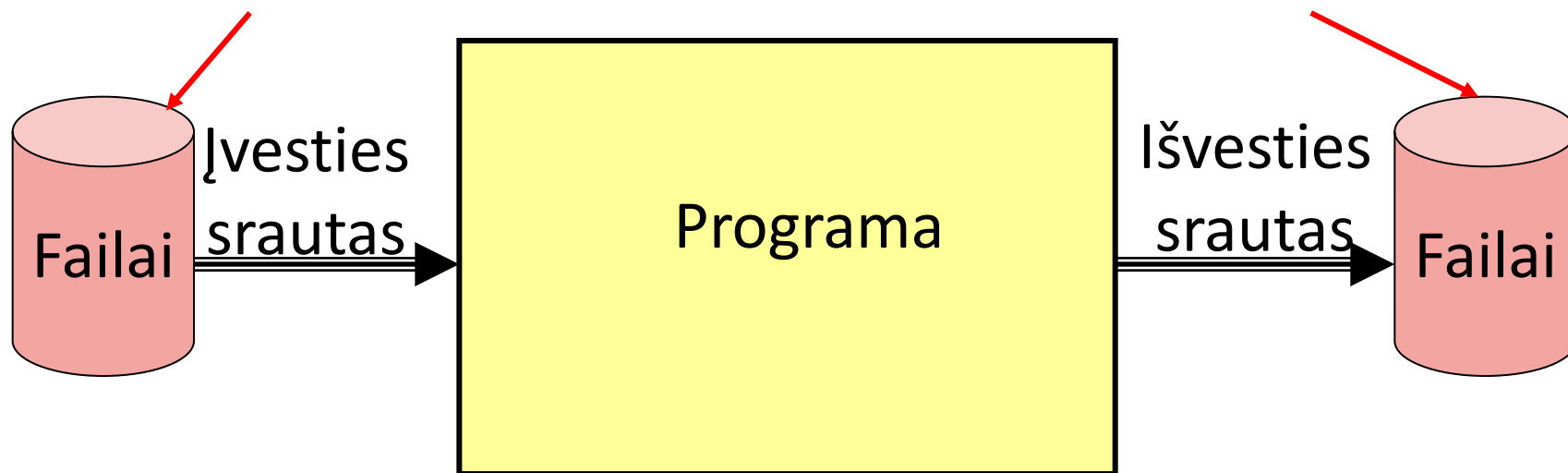
Įvestis ir išvestis srautu

Kad kiekvieno skaičiavimo metu nereikėtų nurodinėti programos duomenų, jie gali būti įvedami iš tekstinio failo. Duomenų failas programoje susiejamas su įvesties srautu.

Programos rezultatai gali būti surašomi į tekstinį failą. Tuomet juos galėsime stebėti ne vien programos darbo metu. Rezultatų failas programoje susiejamas su išvesties srautu.

Ivesties ir išvesties srautai

Išoriniai informacijos nešėjai
(diskas, atmintukas, klaviatūra, ekranas, ...)



Duomenų failas

Failas turi būti paruoštas prieš programos vykdymą.

Į failą surašomos įvedamų kintamųjų reikšmės, vardai nerašomi.

Duomenys faile skiriami skyrikliai (dažniausiai tarpais) arba <Enter>.

Duomenų išdėstymo tvarka turi atitikti kintamųjų tvarką įvesties sakiniuose.

Rezultatų failas

Failas suformuojamas programos vykdymo metu.

Į failą rekomenduotina išvesti ne tik kintamųjų reikšmes, bet ir paaiškinimus.

Informacijos išdėstymo tvarka nusakoma išvedimo sakiniuose.

Būtiniai įvesties veiksmai

- Prijungti įvedimo ir išvedimo klasių biblioteką:

using System.IO;

- Failo vardo paskelbimas

const string CFd = "..\\..\\Duomenys.txt";

const string CFr = "..\\..\\Rezultatai.txt";

..\\..\\Rezultatai.txt santykinis kelias iki katalogo, kuriame yra failas. Duomenų failą galima sukurti bet kuriame projekto kataloge, bet geriausiai tame, kur yra .cs failas.

Duomenų įvedimas iš failo (1/5)

Nekeičiant koduotės (**while** ciklas)

Failo
vardas

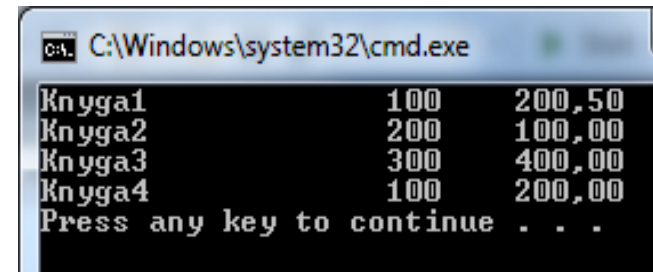
Skaitymo ciklas per eilutes:

```
using (StreamReader reader = new StreamReader(CFd))  
{  
    string line;  
    while ((line = reader.ReadLine()) != null)  
    {  
        string[] parts = line.Split(';');  
        ...// įvedami duomenys, skiriami tarpusavyje ;  
    }  
}
```

Duomenų įvedimas iš failo (2/5)

```
string CFd = "..\\..\\Duomenys.txt";
using (StreamReader reader = new StreamReader(CFd))
{
    string line;
    while ((line = reader.ReadLine()) != null)
    {
        string[] dalys = line.Split(';');
        string prekė = dalys[0];
        int kiekis = Convert.ToInt32(dalys[1]);
        double kaina = Convert.ToDouble(dalys[2]);
        Console.WriteLine("{0,-15} {1,5:d} {2,8:f2}", prekė, kiekis, kaina);
    }
}
```

Knyga1; 100; 200,5;
Knyga2; 200; 100;
Knyga3; 300; 400;
Knyga4; 100; 200;



```
C:\Windows\system32\cmd.exe
Knyga1      100      200,50
Knyga2      200      100,00
Knyga3      300      400,00
Knyga4      100      200,00
Press any key to continue . . .
```

Nekeičiant koduotės (**foreach** ciklas)

Skaitymo ciklas per eilutes:

```
string[] lines = File.ReadAllLines(CFd);  
foreach (string line in lines)  
{  
    string[] parts = line.Split(';');  
    ...    // įvedami duomenys  
}
```

Duomenų įvedimas iš failo (4/5)

Nekeičiant koduotės (**for** ciklas)

Skaitymo ciklas per eilutes:

```
string[] lines = File.ReadAllLines(CFd);  
for (int i=0; i < lines.Length; i++)  
{  
    string line = lines[i];  
    ...    // įvedami duomenys  
}
```

Duomenų įvedimas iš failo (5/5)

Keičiant koduotę

Skaitymo ciklas per eilutes:

```
using (StreamReader reader = new StreamReader(CFd,
    Encoding.GetEncoding(1257)))
{
    string line;
    while ((line = reader.ReadLine()) != null)
    {
        string[] parts = line.Split(';');
        ...    // įvedami duomenys
    }
}
```

Duomenų failo pavyzdys

Duomenų faile naudojamas skyriklis ;

Prieš skaičius gali būti bet koks tarpų kiekis. Jie nereikšminiai. Tarpai prieš tekstą – reikšminiai. Realių skaičių trupmeninę dalį atskiriame kableliais.

```
Jonaitis;Jonas;    6; 7,5;  
Aleksaitė;Alina;  6; 9,5;  
Petraitis;Petras;  5; 8,5;  
Antanaitis;Antanas; 6; 5,5;  
Juozaitis;Juozas;  5; 8,5;  
Rimaitis;Rimas;   6; 7,5;  
Rasaitė;Rasa;     5; 6,0;
```

Duomenų išvedimas į failą (1/2)

Naujas failas

Failo
vardas

```
using (var fr = File.CreateText(CFr))  
{  
    fr.WriteLine(...);  
    ... // išvedami rezultatai  
}
```

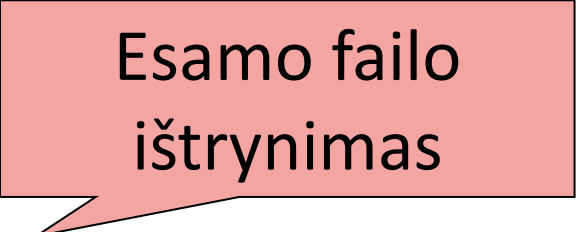
Duomenų išvedimas į failą (2/2)

Failo papildymas

```
using (var fr = File.AppendText(CFr))  
{  
    fr.WriteLine (...);  
    ... // išvedami rezultatai  
}
```

Jeigu informaciją reikia rašyti naujame faile, tai pagrindiniame metode reikia ištrinti kataloge egzistuojantį failą:

```
if (File.Exists(CFr))  
    File.Delete(CFr);
```

A red speech bubble with a black outline, pointing towards the File.Delete(CFr); line of code.

Esamo failo
ištrynimasis



Klasės konstruktorius

Objektų savybių reikšmės

Naujai sukurtuose objektuose pradinės savybių reikšmės neapibrėžtos, todėl prieš pirmąjį jų panaudojimą jas reikia nustatyti, pavyzdžiui, įvedant. Tačiau kartais reikšmės nekinta.

Apibrėžti reikšmes galima naudojant sąsajų metodus:

preke1.DėtiKainą(11.5);

pirkimas1.DėtiKiekį(15);

pirkimas2.DėtiKiekį(8.5);

Tačiau tam patogiau panaudoti objektų konstruktorių.

Klasės konstruktorius

Konstruktorius – tai specialus metodas, kviečiamas kuriant objektą ir nustatant jo savybių pradines reikšmes.

Konstruktorius neturi tipo, vardas sutampa su klasės vardu, jis negrąžina reikšmės. Konstruktorius dalyvauja kuriant objektus pagal klasės aprašymą:

```
Klasė objektas = new Klasė();
```

Jei konstruktorius nepaskelbtas, kompiliatorius pats sukuria numatytąjį. Jį jau naudojome, to nežinodami.

Konstruktoriaus sudarymo taisyklės

Konstruktorius skirtas:

Išskirti atmintį objekto savybėms

Suteikti pradines reikšmes objekto savybėms

Konstruktoriaus parametrams, kamieno sudarymui galioja tos pačios taisyklės, kaip ir kitiems metodams.

Tiesiogiai niekada nekviečiamas, nors įmanoma.

Konstruktorių užklojimas

Jeigu reikia ir konstruktoriaus be parametrų, o jo dažniausiai reikia, tai turime skelbti ir tokį konstruktorių.

Kai klasėje turime keletą konstruktorių, besiskiriančių parametrais, tai vadinama konstruktorių užklojimu.

Jeigu vartotojas parašo savo konstruktorių, numatytasis nekuriamas.

Konstruktoriai (1/4)

Konstruktorius be parametų:

```
public Klasė()
```

```
{
```

```
}
```

```
...
```

```
Klasė objektas = new Klasė();
```

Sukuria objektą su standartinėmis reikšmėmis.

Konstruktoriai (2/4)

Konstruktorius be parametų su numatytomis reikšmėmis:

```
public Klasė()
```

```
{
```

```
// klasės savybių reikšmių priskirimas
```

```
...
```

```
}
```

```
...
```

```
Klasė objektas = new Klasė();
```

Sukuria objektą su numatytomis reikšmėmis.

Konstruktoriai (3/4)

Konstruktorius su parametrais:

```
public Klasė(Parametrai)
```

```
{
```

```
// klasės savybėms priskiriamos parametru
```

```
// reikšmės
```

```
...
```

```
}
```

```
...
```

```
Klasė objektas = new Klasė(Argumentai);
```

Sukuria objektą su parinktomis reikšmėmis.

Konstruktoriai (4/4)

Konstruktorius su parametrais ir numatytomis reikšmėmis :

```
public Klasė(Parametras1 [= reikšmė], ...,
             Parametrasn = reikšmė)
```

Gali būti
praleista.

```
{
// klasės savybėms priskiriamos parametru
// reikšmės.
```

```
...
```

```
}
```

```
...
```

```
Klasė objektas = new Klasė(Argumentas1, ...,
[Argumenrasn]);
```

Sukuria objektą su parinktomis reikšmėmis.

Konstruktorius be parametru (1/3)

```
/** Konstruktorius be parametru */
```

```
public Prekė()
```

```
{
```

```
}
```

...

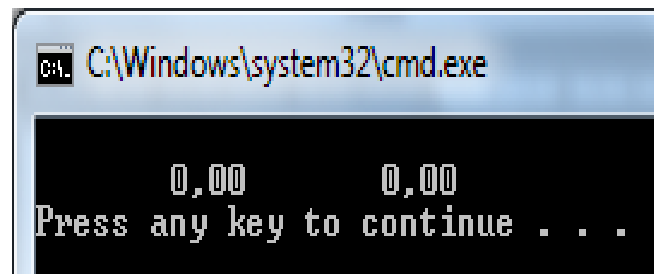
```
Prekė pr1 = new Prekė(); // sukuriamas objektas
```

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

```
Console.WriteLine("{0}    {1,8:f}    {2,8:f}",
```

```
    pr1.ImtiPavadinimą(), pr1.ImtiKainą(),
```

```
    pr1.ImtiKiekį());
```



```
C:\Windows\system32\cmd.exe

0,00    0,00
Press any key to continue . . .
```

Konstruktorius be parametų (2/3)

```
/** Konstruktorius be parametų su numatytomis reikšmėmis */
```

```
public Prekė()
```

```
{
```

```
    pavad = "Knyga4";
```

```
    kaina = 40;
```

```
    kiekis = 250;
```

```
}
```

```
...
```

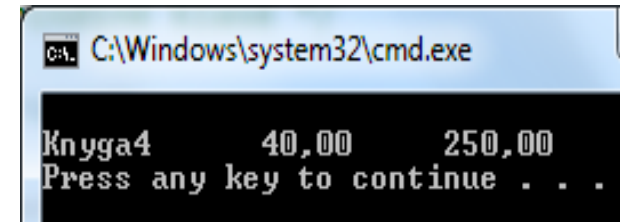
```
Prekė pr1 = new Prekė(); // sukuriamas objektas
```

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

```
Console.WriteLine("{0}    {1,8:f}    {2,8:f}",
```

```
    pr1.ImtiPavadinimą(), pr1.ImtiKainą(),
```

```
    pr1.ImtiKiekį());
```



```
C:\Windows\system32\cmd.exe
Knyga4      40,00      250,00
Press any key to continue . . .
```

Konstruktorius be parametų (3/3)

/ Konstruktorius be parametų su numatytomis reikšmėmis */**

public Prekė()

{

 pavadin = "Knyga4";

 kaina = 40;

// kiekio reikšmė pagal nutylėjimą

}

...

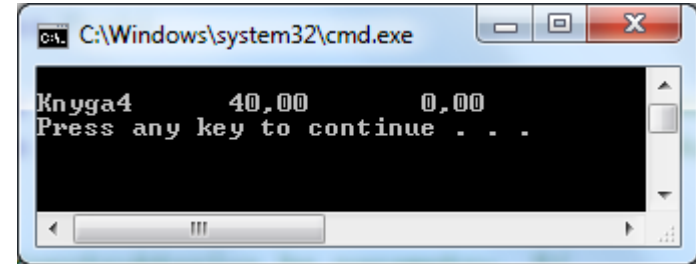
Prekė pr1 = **new** Prekė(); **// sukuriamas objektas**

Console.WriteLine();

Console.WriteLine("{0} {1,8:f} {2,8:f}",

 pr1.ImtiPavadinimą(), pr1.ImtiKainą(),

 pr1.ImtiKiekį());



Konstruktorius su parametrais (1/2)

```
/** Konstruktorius su parametrais */
```

```
public Prekė(string pavad, double kaina, double kiekis)
```

```
{
```

```
    this.pavad = pavad;
```

```
    this.kaina = kaina;
```

```
    this.kiekis = kiekis;
```

```
}
```

```
...
```

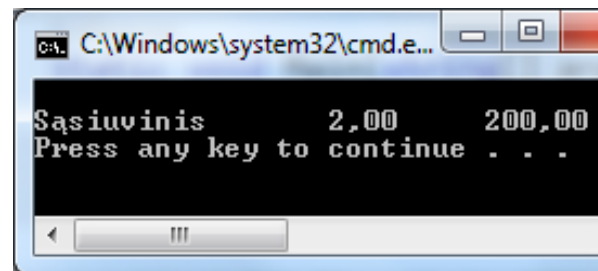
```
Prekė pr2 = new Prekė("Sąsiuvinis", 2, 200); // naujas objektas
```

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

```
Console.WriteLine("{0}    {1,8:f}    {2,8:f}",
```

```
    pr2.ImtiPavadinimą(), pr2.ImtiKainą(),
```

```
    pr2.ImtiKiekį());
```



Konstruktorius su parametrais (2/2)

/ Konstruktorius su parametrais su numatytomis reikšmėmis */**

```
public Prekė(string pavad = "Knyga3", double kaina = 15,  
            double kiekis = 20)
```

```
{
```

```
    this.pavad = pavad;
```

```
    this.kaina = kaina;
```

```
    this.kiekis = kiekis;
```

```
}
```

```
...
```

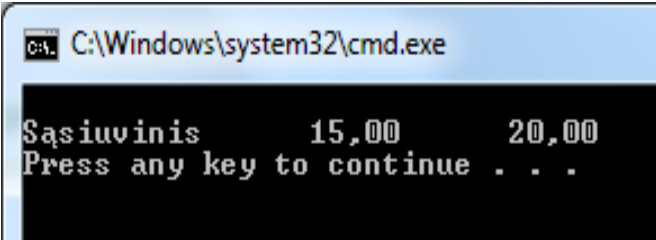
```
Prekė pr2 = new Prekė("Sąsiuvinis"); // sukuriamas objektas
```

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

```
Console.WriteLine("{0}    {1,8:f}    {2,8:f}",
```

```
                pr2.ImtiPavadinimą(), pr2.ImtiKainą(),
```

```
                pr2.ImtiKiekį());
```



```
C:\Windows\system32\cmd.exe  
Sąsiuvinis      15,00      20,00  
Press any key to continue . . .
```

Pavyzdys 1/7

Duota informacija apie prekę: pavadinimas, vieneto kaina, kiekis.
Sukurti:

- klasę duomenims saugoti;
- klasės konstruktorius: be parametų ir su parametrais;
- sąsajos metodus Dėti() kainai ir kiekiui;
- sąsajos metodus Imti() pavadinimui, kainai ir kiekiui;
- klasės metodą sumai skaičiuoti;
- pagrindinį metodą Main(). Išspręsti uždavinį, įvedus duomenis panaudojant konstruktorių. Rasti sumą. Pakeisti kiekį sąsajos metodo pagalba ir dar kartą apskaičiuoti sumą.

Pavyzdys 2/7

```
/** Klasė prekės duomenims saugoti
@class Prekė */
public class Prekė
{
    private string pavad;    // prekės pavadinimas
    private double kaina,    // prekės vieneto kaina
                kiekis;      // prekės kiekis

    // konstruktoriai
    // sąsajos metodai
    // sumos radimo metodas
    ...
}
```


Pavyzdys 3/7

```
/** Konstruktorius be parametru */
```

```
public Prekė()
```

```
{
```

```
}
```

```
/** Konstruktorius su parametrais */
```

```
public Prekė(string pavad, double kaina, double kiekis)
```

```
{
```

```
    this.pavad = pavad;
```

```
    this.kaina = kaina;
```

```
    this.kiekis = kiekis;
```

```
}
```

Pavyzdys 4/7

```
/** įrašo prekės kainą */  
public void DėtiKainą(double kaina)  
{  
    this.kaina = kaina;  
}  
  
/** įrašo prekės kiekį */  
public void DėtiKiekį(double kiekis)  
{  
    this.kiekis = kiekis;  
}
```

Pavyzdys 5/7

```
/** grąžina prekės pavadinimą */  
public string ImtiPavadinimą() { return pavad; }
```

```
/** grąžina prekės kainą */  
public double ImtiKainą() { return kaina; }
```

```
/** grąžina prekės kiekį */  
public double ImtiKiekį() { return kiekis; }
```

Pavyzdys 6/7

```
// randa prekės sumą  
public double RastiSumą()  
{  
    double suma;  
    suma = kiekis * kaina;  
    return suma; // grąžina atsakymą  
}
```

Pavyzdys 7/7

```
/** Pagrindinė klasė */
```

```
class Program
```

```
{
```

```
    static void Main(string[] args)
```

```
    {
```

```
        Prekė pr = new Prekė("Sąsiuvinis", 2, 150); // sukuriamas objektas
```

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

```
        Console.WriteLine("{0}    {1,8:f}    {2,8:f}", pr.ImtiPavadinimą(),  
                           pr.ImtiKainą(), pr.ImtiKiekį());
```

```
        Console.WriteLine("Prekės suma {0} ", pr.RastiSumą());
```

```
        pr.DėtiKiekį(600);
```

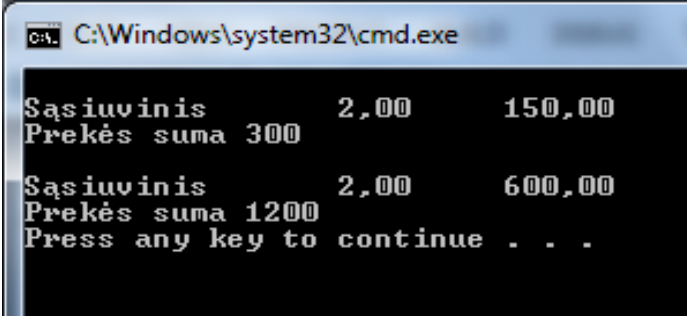
```
        Console.WriteLine();
```

```
        Console.WriteLine("{0}    {1,8:f}    {2,8:f}", pr.ImtiPavadinimą(),  
                           pr.ImtiKainą(), pr.ImtiKiekį());
```

```
        Console.WriteLine("Prekės suma {0} ", pr.RastiSumą());
```

```
    }
```

```
}
```



```

C:\Windows\system32\cmd.exe
Sąsiuvinis      2,00      150,00
Prekės suma 300
Sąsiuvinis      2,00      600,00
Prekės suma 1200
Press any key to continue . . .
  
```



Metodai. Duomenų perdavimas ir grąžinimas.

Pagrindinės klasės metodai

Programa gali būti pakankamai ilga ir nevaizdi. Joje gali būti pasikartojantys fragmentai. Tokių programų struktūrizavimui yra naudojami metodai.

Metodas – tai iškelta ir atskirai aprašyta programos dalis. Jis leidžia kartą aprašius panaudoti jį daug kartų. Tam rašomi kreipiniai į metodą.

Pagrindinės klasės metodo aprašymas ir panaudojimas tapatus metodo panaudojimui klasėje.

C++ kalboje tokie metodai vadinami funkcijomis.

Metodo parametrai

Paskirtis – perduoti metodui duomenų reikšmes ir grąžinti rezultatus.

Parametrų sąrašas metodo teksto antraštėje:

(Tipas1 Vardas1, Tipas2 Vardas2, . . .)

Jei parametrų nėra, naudojami tušti skliaustai.

Tai parametrų perdavimas pagal reikšmę. Šiame būde kuriama parametro kopija ir pokyčiai metodo viduje neturi įtakos originaliam kintamajam.

Nepatartina metoduose naudoti globalių kintamųjų.

Sąvokos, susijusios su metodu

. . .

```
double a = 3.2, b = 8.6;
```

```
double c = Suma(a, b);
```

. . .

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

```
static double Suma(double x, double y)
```

```
{
```

```
    double z = x + y;
```

```
    return z;
```

```
}
```

Kreipinys į metodą

Metodo argumentai

Metodo parametrai

Metodo antraštė

Metodo kamienas

Rezultatų grąžinimas iš metodo

Per metodo vardą (sakiny **return**).

Per metodo kreipinio argumentus (**out**, **ref**).

Metodo grąžinamos reikšmės tipas:

- Bet kuris bazinis C# tipas.
- Bet kuris programuotojo anksčiau aprašytas tipas.
- **void**.

return sakiny

- Reikšmei grąžinti per metodo vardą naudojamas sakiny:
return reiškinys;
- Grąžinamo reiškinio tipas turi atitikti metodo grąžinamos reikšmės tipą.
- Metode gali būti keli **return** sakiniai.
- Įvykdžius **return** sakinį, išeinama iš metodo.

Metodas (1/3)

Metodai aprašomi savo sukurtose klasėse arba pagrindinėje klasėje. Aprašyme skiriasi tik antraštės eilutės modifikatoriai.

- Savo sukurtų klasių modifikatoriai **private** arba **public** apsprendžia metodo matomumą.
- Pagrindinėje klasėje aprašomiems metodams naudojamas modifikatorius **static**.

Metodas (2/3)

Metodo, grąžinančio atsakymą per vardą, bendrasis pavidalas:

```
[Modifikatorius] tipas MetodoVardas(Parametrai)
{
    // Kamienas
    . . .
    return atsakymas;
}
```

Parametrų gali ir nebūti. Būtinai **return**.

Metodas (3/3)

Metodo, grąžinančio atsakymą per parametrus, bendrasis pavidalas:

```
[Modifikatorius] void MetodoVardas(Parametrai)  
{  
    // Kamienas  
    . . .  
}
```

Parametrų gali ir nebūti.

Metodų naudojimas

- Kreipinys į metodą, kurio tipas **void**, turi būti užrašytas atskiru sakiniu:

Vardas(Argumentų sąrašas);

- Kreipinys į metodą, kuris grąžina reikšmę, gali dalyvauti reiškinyje, pvz.:

kint = Vardas(Argumentų sąrašas);

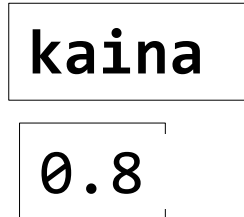
Argumentų sąrašas turi atitikti metodo parametrų sąrašą.

Rezultatų gražinimas per kreipinio argumentus

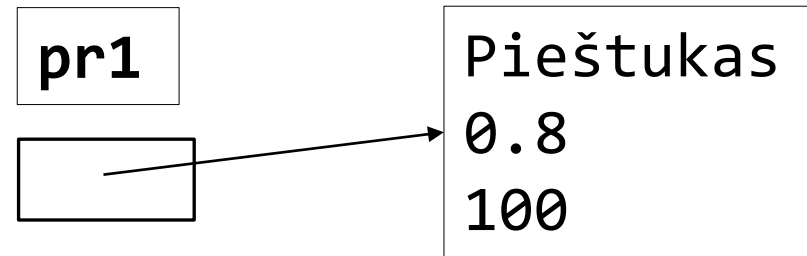
- Rašome prieš parametrus, kurių pakitusios reikšmės turi grįžti, **out** arba **ref**;
- **ref** naudojamas kintamiesiems, kurių pradinės reikšmės jau yra priskirtos iki kreipinio, taip pat tokiam kintamajam nėra būtina priskirti reikšmę metodo viduje;
- **out** naudojamas kintamiesiems, kurių pradinės reikšmės priskiriamos metodo viduje;
- Šie raktiniai žodžiai turi būti naudojami tiek antraštėje, tiek ir kreipinyje.

Reikšmės ir nuorodos tipai (1/7)

```
double kaina = 0.8;
```



```
Prekė pr1 = new Prekė();
```



Atminties ląstelė `kaina` saugo duomenis (bazinis tipas), o atminties ląstelė `pr1` – adresą į duomenis (išvestinis tipas).

Reikšmės ir nuorodos tipai (2/7)

Perduodant į metodą bazinių tipų kintamuosius (nenaudojant **ref** ir **out**), kuriami lokalūs kintamieji, kurių vardai yra metodo parametrų vardai. Argumentų reikšmės kopijuojamos į šiuos lokalius kintamuosius.

Pavyzdžiui,

- **static void BrėžtiLiniją (int kiek, char simb)** yra funkcija, brėžianti nurodyto ilgio ir išvaizdos liniją ekrane;
- **int k = 30; char s = '*';** yra du **Main()** metodo kintamieji, kurie nurodo linijos ilgį ir išvaizdą;
- **BrėžtiLiniją (k, s)** yra kreipinys į metodą, suformuojantis liniją ekrane.

Reikšmės ir nuorodos tipai (3/7)

Pagrindinio metodo **Main()** vykdymo metu atmintyje yra sukuriami du kintamieji ir jiems priskiriamos reikšmės.

Atmintis

k

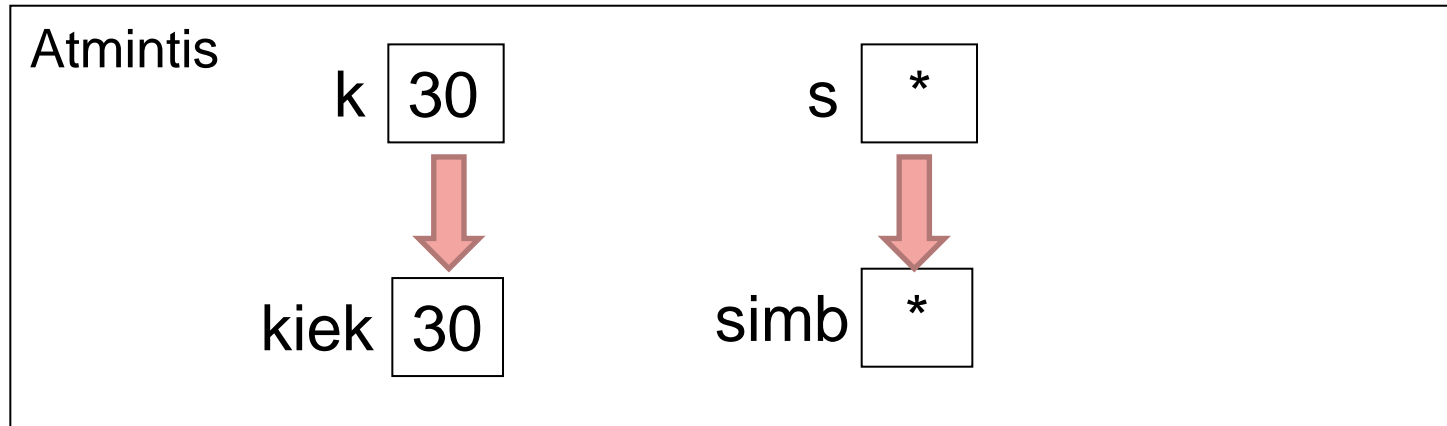
30

s

*

Reikšmės ir nuorodos tipai (4/7)

Kreipimosi į metodą **static void BrėžtiLiniją** (**int kiek**, **char simb**) metu atmintyje sukuriami lokalūs kintamieji, kurių vardai yra metodo parametrų vardai. Argumentų reikšmės kopijuojamos į šiuos lokalius kintamuosius.



Reikšmės ir nuorodos tipai (5/7)

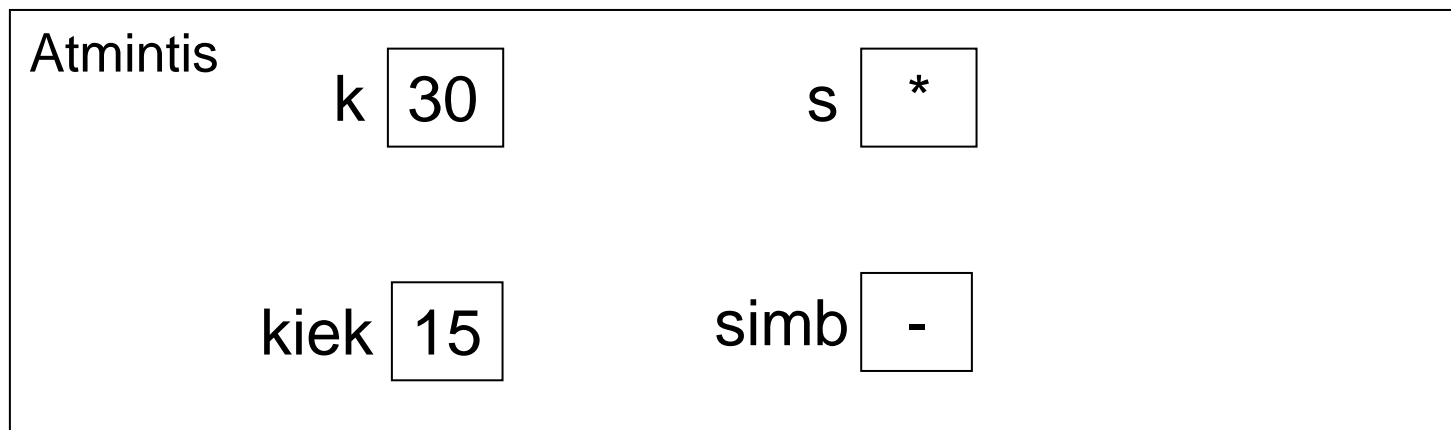
Metodas atlieka savo darbą.

Tarkime, kad metode parašome sakinius:

```
kiek = 15;
```

```
simb = '-';
```

Lokalių kintamųjų su šiais vardais (kiek, simb) reikšmės pasikeis.



Reikšmės ir nuorodos tipai (6/7)

Įvykdžius metodą baigia galioti vardai **kiek** ir **simb**, lokalūs kintamieji panaikinami, o **Main()** metodas naudoja kintamuosius **k** ir **s**.

Reikšmių pakeitimas metode **BrėžtiLiniją()** neturėjo įtakos kintamųjų, kurių reikšmės buvo kreipinio argumentais, reikšmėms.

Atmintis

k

30

s

*

Reikšmės ir nuorodos tipai (7/7)

Bazinių tipų kintamieji, metodų parametrų sąraše naudojant modifikatorius **ref** ir **out**, perduodami naudojant nuorodą. Tai nagrinėsime vėliau.

Išvestinių tipų kintamieji perduodami nuoroda.

Pavyzdys (1/5)

Prieš tai nagrinėtą pavyzdį pertvarkykite:

- pagrindinėje klasėje sukurkite duomenų skaitymo iš failo metodą;
- pagrindinėje klasėje sukurkite duomenų spausdinimo į konsolę metodą;
- pagrindinėje klasėje sukurkite metodą, kuris apskaičiuoja visą sumą;
- pertvarkykite pagrindinį metodą **Main()**;
- pagrindiniame metode **Main()** įveskite duomenis, juos atspausdinkite, raskite minimalią kainą ir visą sumą;
- sukurkite duomenų failą.

Pavyzdys (2/5)

```
//-----  
/** Įveda trijų objektų duomenis iš failo  
@param failas - duomenų failo vardas  
@param pr1    - pirmas objektas  
@param pr2    - antras objektas  
@param pr3    - trečias objektas */  
//-----  
static void SkaitytiDuomenis(string failas, ref Prekė pr1, ref Prekė pr2, ref Prekė pr3)  
{  
    using (StreamReader reader = new StreamReader(failas))  
    {  
        for (int i = 0; i < 3; i++ )  
        {  
            string eilute = reader.ReadLine();  
            string[] dalys = eilute.Split(';');  
            string prekė = dalys[0];  
            double kaina = Convert.ToDouble(dalys[1]);  
            double kiekis = Convert.ToDouble(dalys[2]);  
            Prekė pr = new Prekė(prekė, kaina, kiekis);  
            if (i == 0)  
                pr1 = pr;  
            else if (i == 1)  
                pr2 = pr;  
            else pr3 = pr;  
        }  
    }  
}
```

Pavyzdys (3/5)

```
//-----  
/** Spausdina trijų objektų duomenis konsolėje  
@param failas - duomenų failo vardas  
@param pr1    - pirmas objektas  
@param pr2    - antras objektas  
@param pr3    - trečias objektas */  
//-----  
static void SpausdintiDuomenis(Prekė pr1, Prekė pr2, Prekė pr3)  
{  
    Console.WriteLine();  
    Console.WriteLine("{0} {1,8:f} {2,8:f}", pr1.ImtiPavadinimą(), pr1.ImtiKainą(), pr1.ImtiKiekį());  
    Console.WriteLine("Prekės suma {0} ", pr1.RastiSumą());  
  
    Console.WriteLine();  
    Console.WriteLine("{0} {1,8:f} {2,8:f}", pr2.ImtiPavadinimą(), pr2.ImtiKainą(), pr2.ImtiKiekį());  
    Console.WriteLine("Prekės suma {0} ", pr2.RastiSumą());  
  
    Console.WriteLine();  
    Console.WriteLine("{0} {1,8:f} {2,8:f}", pr3.ImtiPavadinimą(), pr3.ImtiKainą(), pr3.ImtiKiekį());  
    Console.WriteLine("Prekės suma {0} ", pr3.RastiSumą());  
}
```

Pavyzdys (4/5)

//-----

/** Randa trijų prekių visą sumą

@param suma1 - pirmo objekto suma

@param suma2 - antro objekto suma

@param suma3 - trečio objekto suma */

//-----

static double VisaSuma(double suma1, double suma2, double suma3)

{

double suma = suma1 + suma2 + suma3;

return suma;

}

Pavyzdys (5/5)

```

const string CFd = "..\\..\\Duomenys1.txt";
static void Main(string[] args)
{

    Prekė pr1 = new Prekė();
    Prekė pr2 = new Prekė();
    Prekė pr3 = new Prekė();

    SkaitytiDuomenis(CFd, ref pr1, ref pr2, ref pr3);
    SpausdintiDuomenis(pr1, pr2, pr3);
  
```

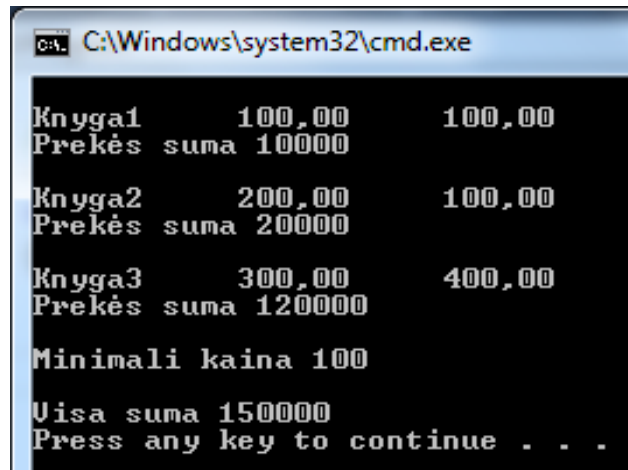
```

    double min = Math.Min(pr1.ImtiKainą(), Math.Min(pr2.ImtiKainą(), pr3.ImtiKainą()));
    Console.WriteLine();
    Console.WriteLine("Minimali kaina {0} ", min);

    Console.WriteLine();
    double bendSuma = VisaSuma(pr1.RastiSumą(), pr2.RastiSumą(), pr3.RastiSumą());
    Console.WriteLine("Visa suma {0} ", bendSuma);
}
  
```

```

Knyga1; 100; 100;
Knyga2; 200; 100;
Knyga3; 300; 400;
  
```



```

C:\Windows\system32\cmd.exe

Knyga1      100,00      100,00
Prekės suma 10000

Knyga2      200,00      100,00
Prekės suma 20000

Knyga3      300,00      400,00
Prekės suma 120000

Minimali kaina 100

Visa suma 150000
Press any key to continue . . .
  
```



Klausimai?