C++ programavimo kalba

Klasės, klasių savybės, vardų erdvės

(3 paskaita)

OOP

Struktūrinio programavimo modelio problema:

- Didelės programos tampa labai sudėtingos t.y. egzistuoja tūkstančiai kintamųjų ir funkcijų vardų ir sunku juos kontroliuoti;
- Duomenys dažniausiai globalūs ir gali būti pasiekiami visų funkcijų;
- Duomenys nesurišti su funkcijomis blogai atvaizduoja realų pasaulį.

Realaus pasaulio modeliavimas:

- Realiame pasaulyje egzistuoja objektai (žmonės, automobiliai…)
- Objektai turi atributus-požymius (automobilis: spalva, galia, durų skaičius …)
- Objektai turi elgseną t.y. jie atlieka tam tikrą veiksmą, priklausomai nuo situacijos (automobilis sustoja, paspaudus stabdžius….)

OOP

OOP pagrindinė idėja: duomenys ir funkcijos, kurios operuoja tais duomenimis apjungti į vieną vienetą, vadinamą **objektu**.

Objekto funkcijos, kurių dažniausia būna ne viena, vadinamos **metodais**.

Objekto duomenys dažniausiai pasiekiami (nuskaitomi, modifikuojami ir t.t.) tik per metodus. Tai reiškia, kad duomenys yra paslėpti nuo atsitiktinio modifikavimo. Kitaip dar sakoma, kad duomenys yra **inkapsuliuoti** (*encapsulated*).

C++ programą paprastai sudaro tam tikras skaičius objektų, kurie komunikuoja tarpusavyje iškviesdami vienas kito metodus.

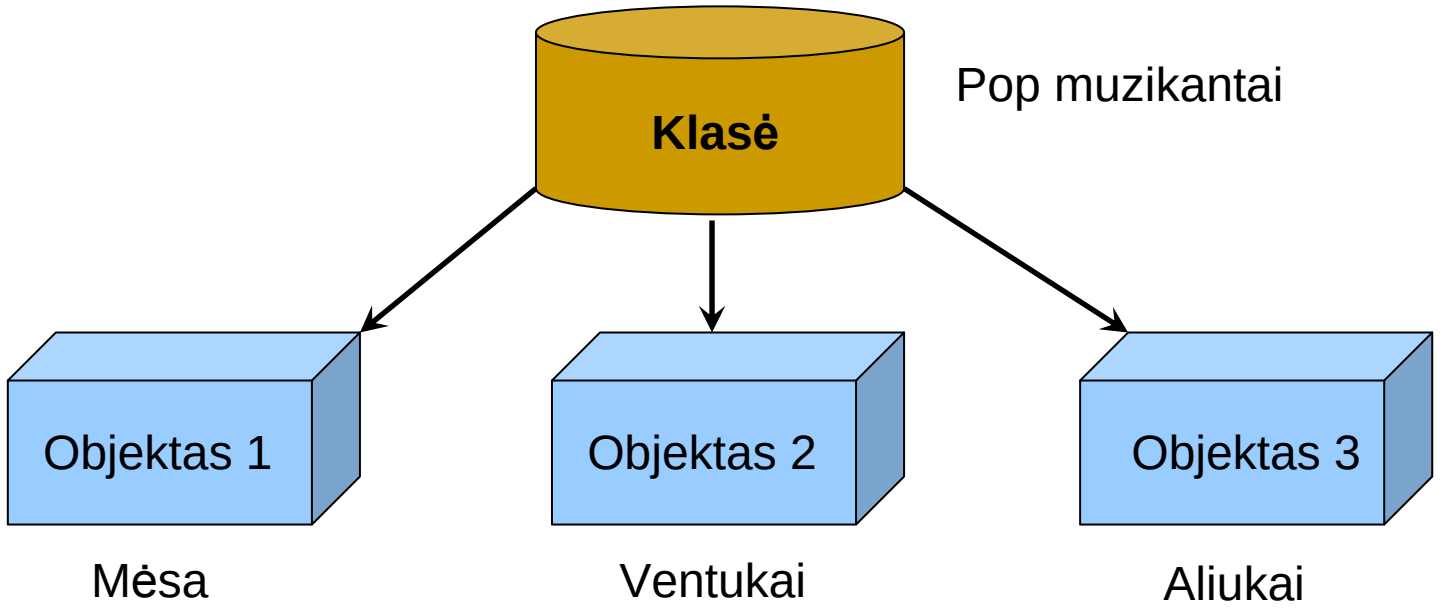
Objektai

- **Fiziniai objektai**
  - Automobiliai kelyje
  - Lėktuvai oro uosto modelyje
- **Kompiuterio aplinka**
  - Langai
  - Meniu
  - Grafiniai objektai (linijos, stačiakampiai, apskritimai)
- **Žmogiški objektai**
  - Darbuotojai
  - Studentai
  - Pirkėjai
  - Pardavėjai
- **Kompiuteriniai žaidimai**
  - Automobiliai lenktynėse
  - Kariai ir priešai kovos lauke

Klasės

Klasė – tai šablonas, pagal kurį kuriamas objektas. Ji apibrėžia kokie duomenys ir kokios funkcijos bus tos klasės objekte.

Apibrėžiant klasę, nekuriamas objektas, klasė tik aprašo objektą.



Paveldimumas

Realiame pasaulyje žmogus daug ką suklasifikavo į klasės ir subklases:

- Gyvūnai: pirmuonys, amebos, vabzdžiai, paukščiai, ropliai, žuvys, žinduoliai.
- Autotransportas: motociklas, lengvieji automobiliai, sunkvežimiai, autobusai, traktoriai.

Toks skirstymas pasižymi **paveldimumu** t.y. žemesnės klasės objektas paveldi aukštesnės klasės objekto savybes. (pvz. visos autotransporto priemonės turi variklį…)

Panašiai OOP egzistuoja **bazinės** klasės ir **išvestinės** klasės.

Bazinės klasės dažnai apibrėžia bendras objekto charakteristikas, o išvestinės klasės jas papildo.

Polimorfizmas ir perkrovimas

Operatorių ir funkcijų panaudojimas įvairiems tikslams, priklausomai nuo to su kuo atliekamas veiksmas, vadinamas **polimorfizmu.**

Polimorfiniai gali būti operatoriai arba funkcijos. Tuomet jie vadinami perkrautais.

Perkrautos funkcijos – tai tokios vienoje klasėje apibrėžtos funkcijos, kurios turi tą patį vardą, bet skirtingą argumentų sąrašą.

Perkrauti operatoriai – tai operatoriai, kurie gali atlikti atitinkamus veiksmus priklausomai nuo operando tipo.

OOP pagrindinės savybės: inkapsuliacija, paveldimumas, polimorfizmas.

Klasės apibrėžimas

C++ kalboje struktūra, jungianti savyje kintamuosius, skirtus duomenims saugoti, ir funkcijas, kurios naudoja tik tuos kintamuosius, vadinama **klase**.

Klasės kintamieji vadinami **duomenimis**, o funkcijos – **metodais**.

Objektiniame programavime priimta, kad duomenimis tiesiogiai gali naudotis tik tos klasės metodai.

Klasės aprašo struktūra:

class [*Klasės_vardas*]
{ *Duomenų elementai;*
public: *Metodai*;
} [objektų sąrašas];

Pavyzdys

```
#include <iostream>
using namespace std;
class small {
      private: int somedata;
      public: void setdata(int d)
                    {somedata = d; }
                 void showdata()
                   {cout<<“Duomenys =“<<somedata<<endl;  }
     };
void main()
{ small s1, s2;
  s1.setdata(109);  s2.setdata(111);
  s1.showdata(); s2.showdata();
}
```

Metodų aprašymas

Metodai klasėje gali būti pilnai aprašyti (prieš tai buvusioje skaidrėje). Tokius metodų aprašus tikslinga turėti, jeigu jų tekstas yra trumpas.

Dažniausiai metodų aprašai iškeliami už klasės ribų. Tuomet klasėje rašomas tik metodo prototipas (metodas deklaruojamas).

Metodo aprašo klasės išorėje sintaksė:

[Reikšmės tipas] [**Klasės Vardas**] :: [**Metodo vardas**] (Parametrų sąrašas)
{ Programos tekstas }

Pavyzdžiai

```
int AnyClass::get_a()
{cout<< a<<endl;
 return a;
}

void AnyClass::set_a(int num)
{a = num;
}
```

Klasės elementų požymiai

Klasės elementai (duomenys ir metodai) gali turėti požymius. Požymis klasėje galioja tol, kol bus sutiktas kito požymio užrašas.

Jeigu požymio užrašo nėra, tuomet pagal nutylėjimą bus priimtas private visiems elementams iki pirmojo sutikto požymio užrašo, jeigu jis iš viso bus.

C++ klasės elementų požymiai:

- **private** (lokalusis). Duomenys ir metodai prieinami tik klasės metodams.
- **public** (globalusis). Klasės elementai prieinami tiek klasės metodams tiek ir išorinėms funkcijoms.
- **protected** (apsaugotasis). Klasės elementai prieinami klasėje, kuri paveldi duotąją klasę. Paveldėtoje klasėje jie galioja *private* teisėmis.

Palyginimas

Savybės	Klasės	Struktūros	Junginiai
Raktinis žodis deklaruojant	class	struct	union
Priėjimas prie duomenų (default)	private	public	public
Duomenų perdengimas	ne	ne	taip

Klasės savybės

- Klasės duomenų apribojimai:
  - Klasės duomenys negali būti apibrėžiami, naudojant modifikatorius auto, extern ir register;
  - Klasės duomenų tarpe negali būti tos pačios klasės objektas, tačiau gali būti rodyklė į objektą ar kitos klasės objektas.
- Paprastai klasės duomenys būna **private**, o funkcijos **public**.
- Klasė formuoja savo vardų erdvę, todėl norint iš vienos klasės kreiptis į kitos klasės kintamąjį reikia naudoti tokią sintaksę:

klasės_vardas::kintamojo_ar_funkcijos vardas

Pavyzdys

```
#include <iostream.h>
// Klasės aprašymas
class Studentas {
     char pav[20];
     float svoris;
public:
     void Skaito() {
          cout << "Iveskite pavarde: \n";
          cin >> pav;
          cout << "Iveskite svori:\n";
          cin >> svoris; };
     void Rodo() {
          cout << pav << " " <<
                  svoris << endl; };
};
```

```
// Pagrindinė programa
void main()
{
Studentas A, B;
A.Skaito();
B.Skaito();
B.Rodo();
A.Rodo();
}
```

Pavyzdys

```
#include <iostream>
using namespace std;

class Staciakampis {
    int x, y;
  public:
    void ivedimas (int,int);
    int plotas ()
      {return (x*y);}
};

void Staciakampis::ivedimas (int a, int b)
{
  x = a;
  y = b; }
```

```
int main () {
  Staciakampis rect;
  rect.ivedimas (3,4);
  cout << “Plotas: “;
  cout<<rect.plotas()<<endl;
  return 0;
}
```

Vardų erdvės

- C++ programavimo kalboje naudojami objektų, klasių, funkcijų vardai gali būti apjungti į atskiras vardų erdves. Tai leidžia programos kūrimo procesą skaidyti į atskirus smulkesnes užduotis, nesirūpinant apie pasikartojančių vardų naudojimą.

- Vardų erdvė sukūrimo formatas:

```
namespace pavadinimas
{
    objektai, klasės, funkcijos, kintamieji
}
```

```
namespace NAUJAS
{  int x;
   float y;  }
```

```
namespace Senas
{  int x;
   float y;  }
```

Vardų erdvės

Vardų erdvėje sukūrus kintamąjį ir norint į jį kreiptis iš kitos vardų erdvės, pvz. globalios, naudojama tokia sintaksė:

```
erdvės_pavadinimas::kintamojo_vardas
```

```
#include <iostream>
using namespace std;        // naudojama standartinė vardų erdvė
namespace first { int var = 5; }
namespace second { double var = 3.1416; }

int main ()
{  cout <<  first::var  <<  endl;
   cout <<  second::var << endl;
   return 0;
}
```

Vardų erdvės

Kaip panaudoti kintamuosius iš skirtingų vardų erdvių?

Atsakymas -> operatorius **using**

```
#include <iostream>
using namespace std;

namespace Vienas
{
  int x = 5;
  int y = 10;
}

namespace Du
{
  double x = 3.1416;
  double y = 2.7183;
}
```

```
int main () {
  using Vienas::x;
  using Du::y;
  cout << x << endl;    // x – iš erdvės Vienas
  cout << y << endl;    // y – iš erdvės Du
  cout << Vienas::y << endl;
  cout << Du::x << endl;
  return 0;
}
```

Pavyzdys

```
#include <iostream>
using namespace std;    // standartinė vardų erdvė

namespace Vienas
{
  int x = 5;
  int y = 10;
}

namespace Du
{
  double x = 3.1416;
  double y = 2.7183;
}
```

```
int main () {
  using namespace Vienas;
  cout << x << endl;
  cout << y << endl;
  cout << Du::x << endl;
  cout << Du::y << endl;
  return 0;
}
```

Vardų erdvė

Standartinė vardų erdvė – tai failų ir visa kas juose apibrėžta pavadinimų erdvė. Ji visada naudojama rašant programas, jei prijungiamas bent vienas C++ standarte apibrėžtas antraštės failas (iostream.h, string.h...)

Vardų erdvė galioja tik bloko ribose { } . Vienoje funcijoje galime naudoti kintuosius iš skirtingų vardų erdvių.

```
#include <iostream>
using namespace std;

namespace first
{   int x = 5; }

namespace second
{  double x = 3.1416; }
```

```
int main () {
 {
   using namespace first;
   cout << x << endl;
 }
 {   using namespace second;
   cout << x << endl;
 }
 return 0; }
```