Informatika 3

8

Špecialne prvky tried

Statické členy - dátové

- tento člen zdieľajú všetky objekty danej triedy (existuje iba jediná inštancia takéhoto objektu)
- existuje aj keď neexistuje žiaden objekt danej triedy

```h
#pragma once
#include <iostream>

using namespace std;

class Tovar
{
private:
    string nazov = "";
    int hmotnost = 0;
    static int celkovaHmotnost;
    static int celkovyPocet;
public:
    Tovar(const char* pnazov, int hmotn)
        : nazov(pnazov ? pnazov : ""), hmotnost(hmotn)
    {
        celkovaHmotnost += hmotn;
        celkovyPocet++;
    };
    ~Tovar() {
        celkovaHmotnost -= hmotnost;
        celkovyPocet--;
    };
    void Zobraz() {
        cout << nazov << ":\t" << hmotnost << " kg" << endl;
    };
};
```

```cpp
#include "Tovar.h"

int Tovar::celkovaHmotnost = 0;
int Tovar::celkovyPocet = 0;
```

```
Microsoft Visual Studio Debug Console
Vrtacka:        5 kg
Stolova pila:   7 kg
Kladivo:        12 kg
```

Statické členy - dátové

```
Tovar(const Tovar& zdroj)
    : Tovar{ zdroj.nazov.c_str(), zdroj.hmotnost }
{
}

Tovar& operator=(const Tovar& zdroj)
{
    celkovaHmotnost -= hmotnost;
    hmotnost = zdroj.hmotnost;
    celkovaHmotnost += hmotnost;
    nazov = zdroj.nazov;
    return *this;
}
```

KST

Statické členy - metódy

- nepatrí žiadnemu objektu, t.j. nemá „nultý“ parameter this

```
static void ZobrazCelkom()
{
    ZobrazPocet();
    ZobrazHmotnost();
}

static void ZobrazPocet();
static void ZobrazHmotnost();
```

```
#include "Tovar.h"

int Tovar::celkovaHmotnost = 0;
int Tovar::celkovyPocet = 0;


void Tovar::ZobrazPocet()
{
    cout << "Pocet: " << celkovyPocet << endl;
}

void Tovar::ZobrazHmotnost()
{
    cout << "Hmotnost: " << celkovaHmotnost << endl;
}
```

Statické členy - metódy

- volanie statickej metódy:

```
#include "Tovar.h"

int main()
{
    Tovar T1("Vrtacka", 5), T2("Stolova pila", 7), T3("Kladivo", 12);
    //....
    Tovar::ZobrazCelkom();
    Tovar::Info(T2);
    return 0;
}
```

- Rozdiel medzi statickou metódou a friend funkciou je iba v spôsobe deklarácie a volaní – z hľadiska funkčnosti sú rovnocenné

```
int main();

class Tovar
{
    friend int main();
}
```

```
#include "Tovar.h"

int main()
{
    Tovar T1("Vrtacka", 5), T2("Stolova pila", 7), T3("Kladivo", 12);
    T1.nazov = "XXXX";
```

Statické členy - metódy

- ak chceme pracovať v statickej metóde s nestatickými členmi, musíme jej poskytnúť objekt ako parameter

```
static void Info(Tovar& tovar)
{
    ZobrazHmotnost();
    cout << endl;
    if (tovar.hmotnost > 0)
        tovar.Zobraz();
}
```

```
int main()
{
    Tovar T1("Vrtacka", 5), T2("Stolova pila", 7), T3("Kladivo", 12);
    Tovar::Info(T2);
}
```

Polia objektov

- Trieda musí mať štandardný konštruktor

```cpp
#pragma once
#include <iostream>

using namespace std;

int main();

class Tovar
{
private:
    string nazov = "";
    int hmotnost = 0;
    static int celkovaHmotnost;
    static int celkovyPocet;
public:
    Tovar(const char* pnazov = nullptr, int hmotn = 0)
        : nazov(pnazov ? pnazov : ""), hmotnost(hmotn)
    {
        celkovaHmotnost += hmotn;
        celkovyPocet++;
    };
    // ...
};
```

```cpp
#include "Tovar.h"

int main()
{
    Tovar sklad[5];
```

Polia objektov

- Ak trieda nemá štandardný konštruktor, musíme zavolať konštruktor pre každý prvok poľa - musíme zadaj inicializačný zoznam

```h
#pragma once
#include <iostream>

using namespace std;

int main();

class Tovar
{
private:
    string nazov = "";
    int hmotnost = 0;
    static int celkovaHmotnost;
    static int celkovyPocet;
public:
    Tovar(const char* pnazov, int hmotn)
        : nazov(pnazov ? pnazov : ""), hmotnost(hmotn)
    {
        celkovaHmotnost += hmotn;
        celkovyPocet++;
    };

};
```

```cpp
#include "Tovar.h"

int main()
{
    Tovar sklad[3]
    {
        Tovar("Vrtacka", 5),
        Tovar("Stolova pila", 7),
        Tovar("Kladivo", 12)
    };
    return 0;

}
```

```cpp
int main()
{
    Tovar* sklad = new Tovar[3]
    {
        Tovar("Vrtacka", 5),
        Tovar("Stolova pila", 7),
        Tovar("Kladivo", 12)
    };
    return 0;
}
```

```cpp
#include "Tovar.h"

int main()
{
    Tovar sklad[]
    {
        Tovar("Vrtacka", 5),
        Tovar("Stolova pila", 7),
        Tovar("Kladivo", 12)
    };

    return 0;
}
```

Konštantné objekty

- z konštantných objektoch môžeme dátové členy iba čítať
- v konštantných objektoch je možné použiť iba metódy:
  - **konštruktor**
  - **deštruktor**

```
#include "Tovar.h"

const Tovar KonstTovar("Pilnik",1);
```

- metódy, ktoré majú v prototype za zoznamom parametrov kľúčové slovo *const* – je to označenie, že metóda nemodifikuje objekt

```
void Zobraz() const
{
    cout << nazov << ":\t" << hmotnost << " kg" << endl;
};
```

Vnorené triedy

- C++ umožňuje deklarovať triedy v rámci inej triedy
- uvoľňuje sa globálny menný priestor

```
#pragma once

class A {
private:
    class Vnorena { // Vnorena je private - nie je možné vytvoriť objekt mimo A
    public:
        int data;
        Vnorena(int x);
        int  Data();
    };
    Vnorena VnorenyObjekt_d;
public:
    Vnorena& VnorenyObjekt();
};
```

```
#include "A.h"

A::Vnorena::Vnorena(int x) : data(x)
{
}

int A::Vnorena::Data()
{
    return data;
}

A::Vnorena& A::VnorenyObjekt()
{
    return VnorenyObjekt_d;
}
```

Menné priestory

- uvoľňuje sa globálny menný priestor

```
using namespace std;
```

- Vlastný menný priestor

```
namespace mojpriestor {
  class A{
        ...
  };
}

mojpriestor::A a;
```