Základy programování v C++ 22. cvičení Letmý úvod do objektově orientovaného programování (v C++) II. Kopírovací konstruktor

Zuzana Petříčková

8. prosince 2019

Přehled

Terminologie

- **třída (class)** ... objektový datový typ
- **instance (objekt, object)** ... proměnná objektového typu

Složky třídy

- **atribut** ... datová složka
- **metoda** ... funkce
  - **konstruktor** ... postará se o vytvoření a inicializaci nové instance
  - **destruktor** ... postará se o zrušení instance

1. **instanční** ... metoda pro práci s jednotlivými instancemi
2. **třídní** ... metoda pro práci s třídou jako celek

stručně:

- na rozdíl od struktury definujeme u třídy zároveň **atributy** i **metody**
- **třída** ale přináší další nové možnosti

Konstruktor a destruktor

- speciální metody, které slouží (k vytvoření a) inicializaci instance třídy (**konstruktor**) a k zrušení instance třídy (**destruktor**)
- nevoláme je přímo, volají se automaticky při vzniku / zániku instance třídy

Co bývá v konstruktoru:

- inicializace datových složek (atributů)
- vytvoření (alokace) dynamických datových složek
- příp. otevření souboru ap.

Co bývá v destruktoru:

- zrušení (dealokace) dynamických datových složek
- příp. zavření souboru ap.

Příklad: třída s dynamickými datovými složkami

```
class Vektor
{
  float *a;
  const int n;
 public:
  Vektor(int _n): n(_n)
  {
        a = new float[n];
        for (int i = 0; i < n; i++)
              a[i] = 0;
  }
  ~Vektor()
  {
        delete[] a;
  }
};
```

Kopírovací konstruktor (copy-constructor)

- specifický konstruktor, slouží k vyvoření kopie instance
- jeho jedinným parametrem je reference na danou třídu
- použije se:
  - při předávání parametrů objektového typu hodnotou

```
void Vektor::pricti1(Vektor v) { ... }
```

- v deklaracích typu:

```
Vektor x(8);     // zde se vola bezny konstruktor
...
Vektor y = x;    // inicializace jinou instanci tridy
Vektor z(x);     // prime volani kopirovaciho konstruktoru
```

- pokud nedeklarujeme kopírovací destruktor, překladač vytvoří vlastní (tzv. implicitní)

Kopírovací konstruktor (copy-constructor)

- specifický konstruktor, slouží k vyvoření kopie instance
- pokud nedeklarujeme kopírovací destruktor, překladač vytvoří vlastní (tzv. implicitní):
  - neobjektové atributy přenese („překopíruje")
  - objektové složky překopíruje pomocí jejich kopírovacích konstruktorů
  - vytváří tzv. **mělkou** kopii (v případě dynamických složek se překopíruje jen ukazatel)

Kopírovací konstruktor

- pokud instance obsahuje dynamicky alokovanou paměť, nebude implicitní konstruktor pracovat správně:
  - chyba se často projevuje velmi zákeřně a na nečekaných místech

  → **vlastní kopírovací konstruktor**
  - postará se o přidělení odpovídající dynamické paměti a překopírování obsahu
  - kopírováním získáme tzv. **hlubokou** kopii

```
Vektor::Vektor(Vektor &v) : n(v.n)
{
    a = new float[n];
    for (int i = 0; i < n; i++)
        a[i] = v.a[i];
}
```

Operátor přiřazení

- implicitní operátor přiřazení pro třídu, která má (pouze) veřejné atributy
- podobně jako kopírovací konstruktor vytváří tzv. **mělkou** kopii (v případě dynamických složek se překopíruje jen ukazatel) → pokud má třída dynamické složky, je na místě **přetížit operátor přiřazení**
  - tak, abychom kopírováním získali **hlubokou** kopii

```
Vektor& operator=(const Vektor& v)
{
    for (int i = 0; ((i < n) && (i < v.n)); i++)
        a[i] = v.a[i];
    return *this;
}
```

Příklad na procvičení I : Chytré pole

- Přeměňte strukturu **ChytrePole** z 16. cvičení na třídu se soukromými datovými složkami.
  - Deklarace třídy bude v hlavičkovém souboru **ChytrePole.h** a definiční deklarace metod budou v souboru **ChytrePole.cpp**.
  - Přidejte třídě **ChytrePole** konstruktor a destruktor (na základě původních funkcí **vytvor(...)** a **zrus(...)**).
  - Implementujte konstantní metodu **vypis()** (na základě původní funkce)
  - Implementuje metody **pridejNaKonec()** a **smazPrvek(...)** (na základě původních funkcí **pridej(...)** a **smaz(...)**)
  - Implementujte kopírovací konstruktor, aby vytvářel **hlubokou** kopii chytrého pole. Vyzkoušejte, že funguje.
  - Přetěžte pro chytré pole operátor přiřazení, aby vytvářel **hlubokou** kopii. Vyzkoušejte, že funguje.
  - Přetěžte pro chytré pole operátor indexace []. Vyzkoušejte, že funguje.

Příklad na procvičení II : Spojový seznam

Na rozmyšlenou (dobrovolně)

- Přeměňte struktury **Prvek** a **Seznam** ze cvičení na třídy se soukromými datovými složkami.
  - Přidejte třídě **Seznam** konstruktor a destruktor (na základě původních funkcí **vytvor()** a **zrus()**).
  - Implementujte pro seznam kopírovací konstruktor, aby vytvářel **hlubokou** kopii seznamu.
  - Přetěžte pro seznam operátor přiřazení, aby také vytvářel **hlubokou** kopii.
  - Přetěžte pro seznam operátor indexace, aby vracel data uložená v i-tém prvku seznamu.
  - Vyzkoušejte, že metody fungují.