Builder.cz - http://www.builder.cz
Autor: Radim Dostál
Rubrika: C/C++
Datum vydání: 28.12. 2000
Url: http://www.builder.cz/art/cpp/cpp_obp_impl.html

Vytváření tříd, instance třídy, zasílání zpráv v C++

Vytváření tříd, instance třídy, zasílání zpráv

V minulém článku jsem vysvětlil základní pojmy z OOP. Nyní bych chtěl ukázat jak vše implementovat v C++. V tomto článku, i ve všech ostatních, již nebudu vysvětlovat význam jednotlivých pojmů, ale jen ukážu jak je implementovat v C++. Budete-li mít nějaké nejasnosti, doporučuji vám si přečíst můj předchozí článek.

Jak vytvořit (definovat) třídu?

K vytvoření třídy slouží klíčové slovo `class`. Definice vypadá následovně:

```
class NázevTřídy [: předchůdci třídy]
{ Položky třídy };
```

Tato definice třídy není jediná. Mně se zdá nejlepší, proto ji budu dále používat. Třída je v C++ velmi podobná struktuře a také jsou k dispozici stejné možnosti pro vytváření třídy a struktury.

V tomto článku se dále nebudeme zabývat nepovinnou částí `[: předchůdci třídy]` . Týká se dědičnosti, které chci věnovat dva jiné články. Položky třídy jsou proměnné, nebo metody. Pojmu metoda se v C++ také někdy říká členská funkce (Member function). Metoda je vlastně funkce, která není globální, ale je volatelná jen v souvislosti s instancemi dané třídy. Položky mohou být soukromé (`private`), chráněné (`protected`) a veřejné (`public`). Soukromé položky mohou být proměnné, ke kterým mají přístup pouze metody dané třídy, nebo metody, které mohou být volány jen jinými metodami dané třídy. Naopak veřejné položky může využívat kdokoliv. Veřejné položky tvoří rozhraní objektu dané třídy. Soukromé položky jsou zapouzdřeny, a přístup k nim se provádí jen přes veřejné rozhranní. Jako veřejné položky by neměli být proměnné, protože by docházelo k porušení zapouzdření. V praxi se ale na zapouzdření někdy moc nehledí. Chráněné položky jsou vlastně soukromé položky, které se někdy chovají jako veřejné. Jako veřejné se chovají k tak zvaným přátelským funkcím a vůči instancím tak zvaných přátelských tříd. Použití chráněných položek je vlastně také porušení zapouzdření. Budu se snažit chráněné položky nepoužívat, ale v praxi se v C++ někdy zapouzdření porušit musí.

Příklad třídy:

```
class MojePrvniTrida
{
   private: /* Následující položky jsou soukromé.*/
     int Cislo;
     char Znak;
   public: /* Následující položky jsou veřejné.*/
     int VratMiTvojeCislo();
     void NastavSiCislo(int noveCislo);
};
```

V této definici třídy jsem deklaroval hlavičky metod. Mohl jsem napsat těla metod hned v definici třídy. U větších tříd by takový zápis nebyl příliš přehledný. Také každá metoda, jejíž tělo je definováno v definici třídy je překládána, jako `inline`. Význam klíčového slova `inline` je stejný jako v C. Tedy funkce (nebo metoda) nebude volána, ale překladač vloží její tělo na místo jejího zavolání. Inline metoda je něco jako-by makro. Na rozdíl od makra, které je vkládáno na úrovni zdrojového textu, je inline funkce vkládána na úrovni přeloženého binárního kódu. Jen bych ještě měl upozornit, že slovo `inline` není pro překladač nijak závazné a pokud překladač "neuzná" za vhodné funkci překládat jako `inline`, tak ji přeloží normálně.

Nyní zbývá definovat těla metod:

```
int MojePrvniTrida::VratMiTvojeCislo() /* Operátor :: (čtyř-tečka) oznamuje překladači, že VratMiTvojeCislo() není globální funkce, ale členská metoda danné třídy.*/
{
   int a = 3; /* Ukázka lokální proměnné*/
   return Cislo;
}
```

V těle této metody je proměnná `a` lokální. Proměnná `Cislo` je proměnná, která je definována v třídě. Každá instance této třídy bude obsahovat svou proměnnou `Cislo`. V těle metod mohu také použít globální proměnné, kdyby nějaké byly.

```
void MojePrvniTrida::NastavSiCislo(int noveCislo)
{
   Cislo = noveCislo;
}
```

V čistě objektově orientovaném programu neexistují globální funkce, ale jen metody. V C++ je trochu problematické napsat čistý objektově orientovaný program. Už jen proto, že funkce main je globální funkcí, není metodou žádné třídy.

Instance třídy

Procesu vytváření a rušení tříd věnuji celý příští článek. Zatím jen pro jednoduchou ilustraci naší jednoduché třídy vytvoříme instanci následujícím způsobem: `MojePrvniTrida mojeprvniinstance;`. Tedy stejně jako by jsme definovali "proměnnou typu MojePrvniTrida".

Objektu se zašle zpráva (Vyvolá se jeho metoda.) následujícím způsobem:

```
Instance.NazevZpravy(argumenty - parametry);
```

Vyvolá se metoda, nebo se zašle zpráva? Jedná se jen o takové hraní si se slovíčky. V minulém článku jsem uvedl, že objekt na zprávu reaguje vyvoláním své metody. Podle teorie se objektům zasílají zprávy. Objekt na zprávu zareaguje voláním metody. Klidně tomu ale můžeme říkat zavolání metody. Nikomu snad nebude vadit, že přesně nedodržuji terminologii. Nyní příklad: (Napište před funkci main definici třídy a defici metod uvedené výše.)

```
#include <iostream.h>

void main(void)
{
   MojePrvniTrida a,b;
   a.Cislo = 3; /* CHYBA - proměnná Cislo je soukromá. Mají k ní přístup jen metody třídy MojePrvniTrida. Chcete-li program přeložit, odstraňte tento řádek. */
   a.NastavSiCislo(5);
   b.NastavSiCislo(4);
   cout << a.VratMiTvojeCislo() << endl;
   cout << b.VratMiTvojeCislo() << endl;
}
```

Ve svém prvním článku stejně jako zde používám zápis `cout << něco`, aniž bych vysvětlil o co jde. Vstupně výstupním operacím se chci věnovat v jednom svém článku podrobně. Zatím prosím berte jako fakt, že "to funguje" a že to něco vypíše. Nechcete-li psát něco, co neznáte, můžete zatím používat funkci `printf` z C knihovny `stdio.h`.

Implicitní parametr "this"

Každá metoda v C++ je volána s tak zvaným implicitním parametrem this. Tento parametr je ukazatelem na instanci třídy, jejíž položkou metoda je. Tento ukazatel ukazuje právě na instanci, pro kterou je volán. Tedy v některé metodě naší třídy je proměnná `Cislo` identická s proměnnou `this->Cislo`. Programátor může využít `this` při konfliktu jmen. Dopište prosím do definice třídy MojePrvniTrida mezi veřejné položky následující metodu: `void`

`pokus();`. Dále definujte globální proměnnou `int Cislo`. Nyní ukážu jak přistupovat k jednotlivým proměnným:

```
void MojePrvniTrida::pokus()
{
/* Nyní je globální proměnná Cislo překrytá proměnnou Cislo deklarovanou v třídě. */
   int Cislo;     /* Nyní jsem lokální proměnnou překryl i proměnnou v třídě.*/
   Cislo++;   /* Zvyšuji lokální proměnnou.*/
   this->Cislo++;   /* Zvyšuji instanční proměnnou deklarovanou v třídě.*/
   ::Cislo++;   /* Zvyšuji globální proměnnou. */
}
```

V příštím článku se podrobně podívám jak vytvářet a rušit instance. Povím něco o konstruktorech, destruktorech, operátorech new, delete, atd.

------ http://www.builder.cz ------