Builder.cz - http://www.builder.cz
Autor: Radim Dostál
Rubrika: C/C++
Datum vydání: 20.06. 2001
Url: http://www.builder.cz/art/cpp/cpp_exception.html

Výjimky v C++ - výjimky tvoří dědičnou hierarchii

Výjimky v C++ - výjimky tvoří dědičnou hierarchii

V minulém článku jsme si ukázali základní princip výjimek. Dnes se podíváme na situaci, kdy výjimky jsou odvozeny z jedné společné nadtřídy.

Odchycení jakékoliv výjimky

Nejprve se ale podívejme, jak odchytit jakoukoliv výjimku. Za klíčovým slovem `catch` místo typu a lokálního názvu výjimky uvedeme 3 tečky. Například:

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

class V1 {};
class V2 {};

int main()
{
	for(int i = 0; i < 5; i++)
	{
		try
		{
			if (i % 2 == 0)
			{
				V1 v;
				throw v;
			}
			else
			{
				V2 v;
				throw v;
			}
		}
		catch(...)
```

```
                /* Odchytím jak výjimky třídy V1, tak i výjimky třídy V2.*/
                  {
                          cout << "Chyceno" << endl;
                  }
        }
}
```

Tento příklad nedává moc smysl, ale jasně ukazuje odchycení jakékoliv výjimky. Protože při vyvržení, i odchycení výjimky dochází často ke kopírování vyvržené výjimky na zásobník, je určitě efektivnější pracovat s referencemi, nebo ukazateli na výjimky. Já v dalším textu budu používat ukazatele.

Výjimky tvoří dědičnou hierarchii

Je velmi výhodné používat výjimky, které tvoří dědičnou hierarchii. Představme si třídu výjimek. Může být i abstraktní. Z této třídy dědí jiné výjimky, které jsou speciálním případem výjimky. Vytvořme si jako názornou ukázku jednoduchou hierarchii výjimek.

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

class Vyjimka
{
        private:
                string Text;
        public:
                Vyjimka(string s):Text(s){}
                string dejText() { return Text; }
};

class DeleniNulou : public Vyjimka
{
        private:
                int Cislo;
        public:
                DeleniNulou(string s, int i):Vyjimka(s),Cislo(i) {}
                int dejCislo() { return Cislo; }
};

class Preteceni : public Vyjimka
{
        public:
                Preteceni(string s):Vyjimka(s) {}
};
```

Vytvořili jsme tři výjimky. Nyní si ukážeme jak odchytávat výjimky, které mají společného předka. Problém je v tom, že musíme nejprve odchytit nejkonkrétnější výjimky a postupně odchytávat jejich nadtřídy. Dokončím příklad:

```
int main()
{
        for(short int p = 5; p >= 0; p--)
        {
                try
                {
                        if (p == 0) /* Vyvrhnu ukazatel */
                                throw new DeleniNulou("Deleni nulou",10);
                        cout << "10 / " << p << " = " << 10.0 / p << endl;
                        if ((short int ) (p * 10000) < p)
                                throw new Preteceni("Pretekl short int");
                        if (p == 3)
                                throw new Vyjimka("Jen tak, pro ukazku");
                }
                catch (Preteceni *v) /* Odchytávám ukazatel */
                { /* Došlo k přetečení */
                        cout << "Při násobení: " << v->dejText() << endl;
                        delete v;
                }
                catch (DeleniNulou *v)
                { /* Došlo k dělení nulou */
        cout << v->dejText() << " nelze " << v->dejCislo() << "/0" << endl;
                delete v;
                }
                catch (Vyjimka *v) /* Nadtřídu odchytím až jako poslední */
                { /* Nějaká výjimka */
                        cout << v->dejText() << endl;
                        delete v;
                }
        }
        return 0;
}
```

V příkladu jsme vyvrhli vždy ukazatel na třídu `výjimka`, nebo na nějakého potomka této třídy. Ukazatel na tuto třídu jsme museli odchytit až jako poslední. Stačí si uvědomit, že ukazatel na potomka je zároveň také ukazatelem na předka. Znamená to, že kdyby jsme odchytávali ukazatel na třídu `výjimka` jako první, došlo by zde vždy k odchycení a další výjimky by nebyly odchyceny nikdy. Můžete se o tom sami přesvědčit, když příklad upravíte tak, že nejprve bude blok `catch (Vyjimka *v)` a potom ostatní odchytávací bloky. Tento problém nastává samozřejmě vždy, nejen při používání ukazatelů jako v našem příkladě.

Všimněte si v příkladě řádků `delete v;`. Zde vždy zlikviduji výjimku, která byla vyvržená. Často se místo vyvrhnutí ukazatelů na výjimky a odchycení

ukazatelů pracuje s referencemi. Uvedu jen jednoduchý příklad:

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

class V
{};

int main()
{
	try
	{
		V v;
		throw v;
	}
	catch (V &v)
	{
		cout << "Chyceno" << endl;
	}
	return 0;
}
```

Pro dnešek by to mohlo stačit, příště se podíváme na některé doplňující informace k výjimkám. Podíváme se například co se stane, jestliže výjimka opustí tělo funkce main, když selže operátor new a podobně.