Builder.cz - http://www.builder.cz
Autor: Radim Dostál
Rubrika: C/C++
Datum vydání: 14.06. 2001
Url: http://www.builder.cz/art/cpp/cpp_vyjimky.html

Výjimky v C++

Výjimky v C++

Dnes si povíme něco o výjimkách a o ošetření výjimek. Pod pojmem výjimka se rozumí nějaká výjimečná situace, která nastane ve volané metodě, nebo funkci. V jazyce C i C++ se často používá návratových hodnot funkcí, které vracejí úspěšnost provádění nějaké operace. Nejčastěji -1 se vrací v případě chyby, jinak se vrací 0. Chceme-li v posloupnosti několika řádků zdrojovém textu takto ošetřit všechny možné chyby, vznikne nám velice nepřehledný program plný vnořených příkazů `if`. Jinou možnost nám nabízí mechanizmus výjimek. Princip spočívá v tom, že označíme posloupnost příkazů do zvláštního bloku, říká se mu často hlídaný blok, ve kterém neošetřujeme žádné chyby. Právě v tomto bloku může vzniknout výjimka. Za tímto blokem označíme postupně jaké výjimky mohly v hlídaném bloku nastat, a jak je ošetřit. V C++ může být výjimkou proměnná jakéhokoliv primitivního datového typu, nebo instance jakékoliv třídy. Výjimka by v sobě měla nést nějakou informaci o situaci, která nastala a proč byla vyvolána.

Syntaxe výjimek v C++

Výjimka se vyvolá (vyvrhne) pomocí klíčového slova `throw`. Hlídaný blok se značí klíčovým slovem `try`. K odchycení vyvolaných výjimek slouží klíčové slovo `catch`. Vše uvedu na jednoduchém příkladu.

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

class Vyjimka
{ /* Třída výjimek */
         private:
                  string Duvod;
         public:
                  void nastav(string d) { Duvod = d; }
                  string dej() { return Duvod; }
};

ostream &operator<< (ostream &os, Vyjimka &v)
{
         return os << v.dej() << endl;
```

```
}

class Zlomek
{
        private:
                int C,J;
        public:
                void nastavCitatel(int c) { C = c;}
                void nastavJmenovatel(int j) {J = j;}
                double vydel() throw (Vyjimka);
       /*Z metody vyděl může být vyvržená výjimka - instance třídy Vyjimka*/
};

double Zlomek::vydel() throw (Vyjimka)
{ // začátek bloku 2
        int *i = new int;
        if (J == 0)
        { //začátek bloku 1
                string s("Nejde");
                Vyjimka v;
                v.nastav(s);
                throw v;
        } // konec bloku 1
        delete i;
        return ((double)C / J);
} // konec bloku 2

int main(void)
{
        Zlomek z1,z2;

        z1.nastavCitatel(10);
        z2.nastavCitatel(5);
        for(int i = 5; i > -5; i--)
        {
                z1.nastavJmenovatel(i);
                z2.nastavJmenovatel(i);
                try
                {       /* Zkusim: */
                        cout << "10 / " << i << " = " << z1.vydel() << endl;
                        cout << "5 / " << i << " = " << z2.vydel() << endl;
                }
                catch (Vyjimka v) // Zde je odchycení výjimky třídy Vyjimka
                {
                          cout << v << endl;
                }
                /*
```

```
                catch (Jina_vyjimka j)
                {
                        .....
                }
                */
        }
        return 0;
}
```

Podívejme se podrobněji na tento příklad. Nejprve jsme vytvořili nějakou třídu s názvem `Výjimka`. Dále třídu Zlomek. V deklaraci jedné metody se zde objevilo další klíčové slovo `throw`. Za slovem `throw` následuje seznam typů výjimek, které mohou být z dané metody (nebo i funkce) vyvrženy. V případě naší metody `vyděl` se jedná o výjimky třídy `Výjimka`. Nedoporučuji používat jako výjimky primitivní datové typy, i když syntaxe jazyka do umožňuje. V případě, že atribut `J` třídy `zlomek` bude v momentě zavolání metody `vyděl` roven nule, dojde k vyvolání výjimky. V tomto případě nejprve vytvořím instanci třídy `Výjimka`, a poté ji příkazem `throw vyvrhnu.` Při vyvržení výjimky dojde k okamžitému opuštění aktuálního bloku daného závorkami `{ }`. V našem příkladě jsem jej poznámkami označil jako blok 1. Výjimka byla z tohoto bloku vyvržena. Při opuštění tohoto bloku dojde k zavolání destruktorů všech lokálních instancí tak, jako by se jednalo o korektní opuštění bloku. V našem případě `s` a `v`. Nemusíte se obávat, že byla zlikvidována instance v, kterou bude v budoucnu používat. Budete totiž pracovat s její kopií. Po opuštění bloku se výjimka šíří dále. Dojde k vyvržení výjimky z "vnějšího" bloku, v naše případě označeného jako blok 2. V tomto bloku budou opět korektně zlikvidovány všechny lokální instance. V našem případě bude zlikvidován ukazatel, ale nedojde k uvolnění paměti, na kterou ukazuje. Později ukážu, jak tento problém řešit.
Takto vše pokračuje, dokud nebude opuštěn hlídaný blok označený klíčovým slovem `try`. Při vyvržení výjimky z hlídaného bloku se zjistí, jestli za hlídaným blokem existuje pro tento typ výjimky odchycení `catch`. Jestliže ne, výjimka je vyvržená dále z aktuálního bloku. Jestliže ano, dojde k vykonání tohoto bloku označeného `catch`. Po odchycení a ošetření výjimky program normálně pokračuje příkazy za blokem `catch`. Jak jsem se již zmínil, při vyvolání výjimky dojde k likvidaci všech lokálních proměnných. Je-li ale lokální proměnnou ukazatel, nebo reference, nedojde k uvolnění dat, na které "ukazují". Do našeho příkladu jsem úmyslně vložil do metody `vyděl` ukazatel na `int`. Při vyvolání výjimky se řádek `delete` neprovede. Paměť na kterou ukazatel ukazuje zůstane neuvolněná. Bylo by vhodné vyvrženou výjimku ošetřit na více místech. (V mém jednoduchém příkladě bych mohl jednoduše před slovo `throw` napsat `delete i`, já ale chci ukázat jak ošetřit výjimku na více místech.) Opravme metodu `vyděl` následovně:

```
double Zlomek::vydel() throw (Vyjimka)
{
        int *i = new int;
        try
        {
                if (J == 0)
                { //začátek bloku 1
                        string s("Nejde");
                        Vyjimka v;
                        v.nastav(s);
                        throw v;
                } // konec bloku 1
                delete i;
```

```
        }
        catch (Vyjimka v)
        { /* Ošetřím, co můžu */
                delete i;
                throw; /* V tomto případě má stejný význam jako throw v; */
        }
        return ((double)C / J);
}
```

Výjimku jsem odchytil ještě v metodě `vyděl`. Uvolnil jsem paměť, na kterou ukazuje ukazatel `i`, a poté jsem výjimku opět vyvrhl. Jsme-li v bloku `catch` a chceme-li odchycenou výjimku poslat dále, nemusíme uvádět její název.

Podívejme se ještě podrobněji na deklarace metod, ze kterých může být vyvržena výjimka. Metoda `vyděl` je deklarována: `double vydel() throw (Vyjimka);`. Znamená to, že z metody `vyděl` může být vyvržena výjimka třídy `Výjimka`. Je-li více typů výjimek, které mohou být vyvrženy, oddělí se čárkou. Například `double vydel() throw (Vyjimka,Jina1,Jina2);` Není-li v deklaraci metody uvedeno klíčové slovo `throw`, znamená to, že z metody může být vyvržena JAKÁKOLIV výjimka. Pro ty, kteří znají Javu to může být trochu matoucí, protože v Javě je to přesně naopak (tedy žádná). Chceme-li v C++ deklarovat metodu, z níž nemůže být vyvržena výjimka, za deklaraci připíšeme `throw ()`. Ještě jen zbývá dodat, že seznam výjimek, které mohou být vyvrženy je součástí názvu metody, nebo funkce. Musí tedy být uveden i v deklaraci, i v definici.

Pro dnešek by to o výjimkách mohlo stačit. Všem, kterým mechanizmus výjimek není jasný doporučuji můj příklad podrobně projít v debuggeru. Příště dokončíme téma výjimek. Podíváme se na situace, kdy výjimky vytvářejí dědičnou hierarchii, jak odchytit jakoukoliv výjimku a co se stane není-li výjimka ošetřená a opustí tělo funkce main.