Výjimky

Výjimky umožňují přenést vykonávání programu z místa, kde nastala nežádoucí situace nebo došlo k chybě, do bloku, kde tato situace, či chyba je řešena nebo ošetřena. Přitom lze navíc předat informaci o typu a příčině výjimečné situace nebo chyby.

Mechanismus výjimky je založen na existenci dvou bloků. První blok, označovaný jako try-blok podle klíčového slova `try`, které je před ním, obsahuje vykonávaný kód. Tento kód může v libovolném svém místě ukončit své vykonávání generováním výjimky. Druhý blok, označovaný jako catch-blok podle klíčového slova `catch`, kterým tato část začíná, obsahuje kód, jež se provede v případě, že výjimka vygenerovaná v try-bloku je tímto blokem zachycena.

Výjimku generujeme v try-bloku příkazem `throw`, který předá tzv. objekt výjimky. Tímto objektem výjimky je hodnota výrazu nebo objekt nějaké třídy.

Za klíčovým slovem `catch`, kterým začíná část s druhým blokem, je v kulatých závorkách deklarace proměnné, jejíž datový typ odpovídá datovému typu zachycovaného objektu výjimky, nebo je tam trojice teček ..., což znamená, že blok zachytí libovolný typ výjimky.

Příklad.

```
float a,b,c;

try { if (b==0) throw "Nulovy delitel";
      c=a/b;
    }
catch (const char *e) { cout << e << endl; c=FLT_MAX; }
```

Výjimku může generovat i funkce, kterou v try-bloku voláme.

Příklad.

```
float a,b,c;

inline float podil(float a, float b)
{
  if (b==0) throw "Nulovy delitel";
  return a/b;
}

try { c = podil(a,b); }
catch (const char *e) { cout << e << endl; c=FLT_MAX; }
```

Pokud potřebujeme předat více údajů o výjimečném stavu nebo o chybě, sestavíme si třídu s příslušnými proměnnými, do kterých tyto údaje uložíme. K uložení údajů použijeme konstruktor třídy. Při výskytu výjimečného stavu se předá objekt třídy.

Příklad.

```
class Preteceni {
  public: char oper;
          unsigned char a,b;

          Preteceni(char o, unsigned char a, unsigned char b):
```

```
                                                oper(o),a(a),b(b) { }
};

unsigned char a=100,b=200,c;

try { unsigned u = (unsigned)a+b;
      if (u>UCHAR_MAX) throw Preteceni('+',a,b);
      c=u;
    }

catch (Preteceni &e) { cout << "Preteceni: "
                            << (unsigned)e.a << e.oper
                            << (unsigned)e.b;
                       c=UCHAR_MAX; }
```

Preteceni: 100+200

Za try-blokem může být více catch-bloků.

Příklad.

```
int vek; char *jmeno;

try { if (vek<0)      throw vek;
      if (jmeno=="") throw jmeno;
    }

catch (int v)          { cout << "Chybny vek: " << v; }
catch (const char *j) { cout << "Neuvedene jmeno"; }
catch (...)            { cout << "Jina vyjimka"; }
```

Generováním výjimky můžeme zabránit vytvoření v objektu v okamžiku, když už je volán jeho konstruktor.

Příklad.

Generováním výjimky v konstruktoru zabráníme, aby existoval větší počet objektů třídy, než je stanovený jejich maximální počet.

```
const int MAX_POCET=3;

class C { static int pocet;
  public: C() { if (pocet==MAX_POCET) throw pocet+1;
                ++pocet; }
          ~C() { --pocet; }
};

int C::pocet=0;

for (int i=0;i<=MAX_POCET;++i)
{
  try { new C; }
  catch (int p) { cout << "Objekt nevytvoren .. " << p; }
```

```
}
```
Objekt nevytvoren .. 4

Některé výjimky rovněž generuje prostředí, ve kterém náš program běží, v situacích, kdy dochází k určitým typům chyb při vykonávání programu.

Příklad. Generováním výjimky `bad_alloc` při přidělování paměti.

```
try { new char [INT_MAX]; }
catch (bad_alloc) { cout << "Chyba pridelovani pameti"; }
```

Chyba pridelovani pameti

Příklad. Generování výjimky `bad_typeid` při zjišťování datového typu. Tato výjimka vrací referenci na objekt třídy `exception`, která obsahuje funkci `what`, jež vrací řetězec s popisem chyby.

```
class A { };
class B { virtual void foo() =0; };

try { A *a = nullptr; cout << typeid(*a).name() << endl;
      B *b = nullptr; cout << typeid(*b).name() << endl;
    }
catch (bad_typeid &e) { cout << e.what() << endl; }
```

class A

Attempted a typeid of NULL pointer !