C++; delete Java;

Část 3: Výjimky

Kennny

srpen 2017

Výjimka

- v C++ je výjimka pouze mechanismus
- neváže se na konkrétní datový typ
- syntaxe podobná jako jinde, pouze s menšími úpravami

```
try
{
    throw std::overflow_error("Preteceni");
}
catch (std::invalid_argument& e)
{
    //
}
catch (...)
{
    //
}
```

Výjimka

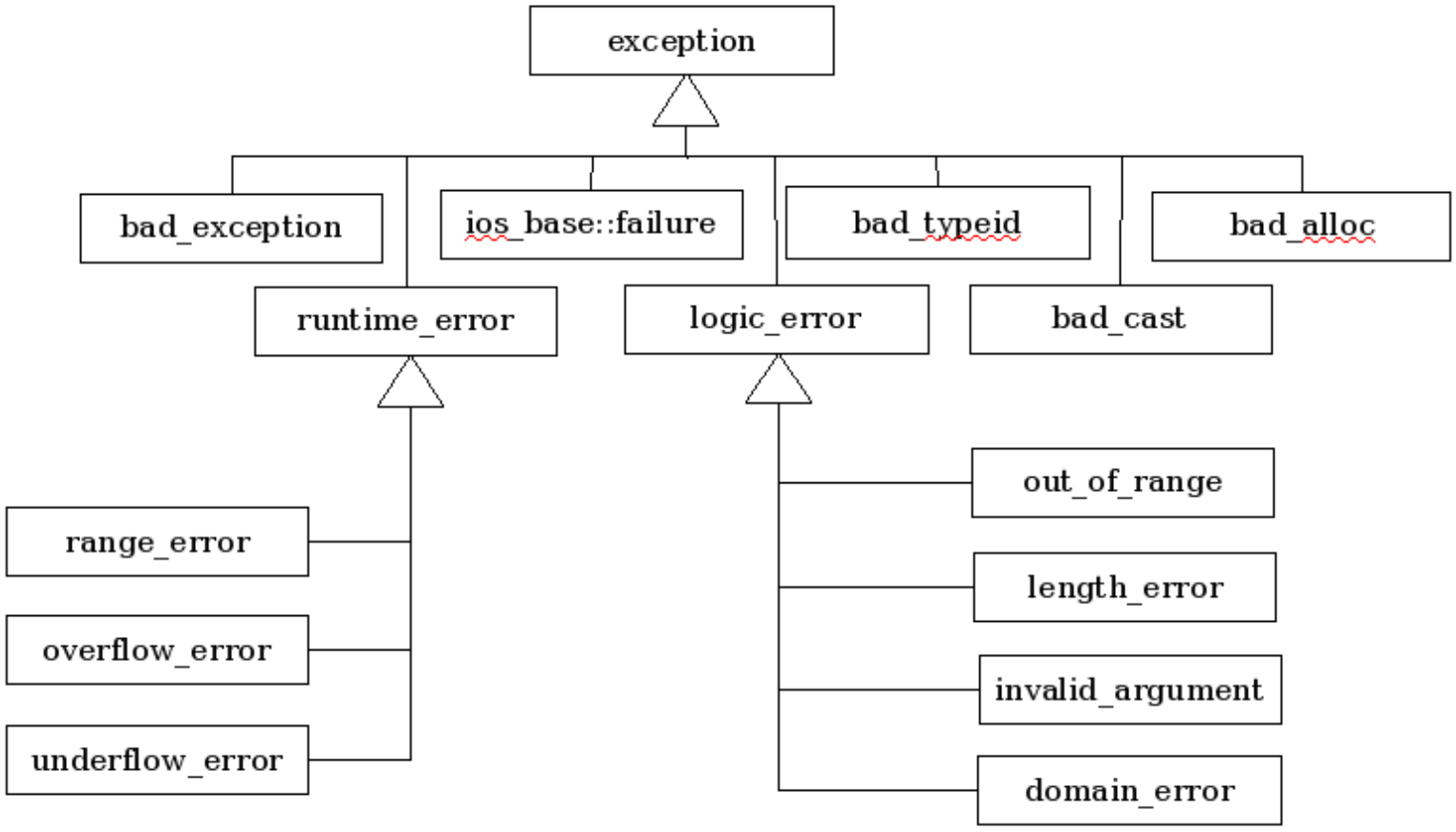
- instance se vytvoří ve scope příslušného `catch` bloku
- pomocí `throw` vyhazujeme staticky alokované (throw by-value)
- pomocí `catch` je lepší odchytávat reference (catch by-reference)
  - odchytávání hodnotou nezohledňuje polymorfismus
- za `throw` může být libovolný datový typ

```
try
{
    throw int(5);
}
catch (int& e)
{
    //
}
```

Standardní výjimky

- STL obsahuje standardní výjimky
- ty používají všechny STL třídy, je možné (někdy i vhodné) je používat také
- popřípadě si oddědit vlastní
- specifikátor `throw()` specifikující typy, co metoda může vyhodit
  - nedoporučuje se
  - maximálně čistý `throw()` kde víme, že nikdy výjimka nevznikne

std::exception



Obrázek: Hierarchie základních STL výjimek

std::exception

- base class pro všechny STL výjimky
- konstruktor a metoda `what`, která vrací řetězec s chybou
- jedná se ale o kostru - konstruktor bez parametru, `what` vrací prázdný řetězec
- potomci definují vlastní konstruktor a přepisují `what`

Vlastní výjimková třída

```
class CustomException : public std::exception
{
    public:
        CustomException(std::string& what)
        {
            m_what = "Custom exception: " + what;
        }

        const char* what() const throw()
        {
            return m_what.c_str();
        };
    private:
        std::string m_what;
};
```

Poznámky

- výjimku lze "odhodit" v catch bloku do vyšších handlerů

```
catch (std::exception& e)
{
    throw;    // ta sama instance
    throw e; // vytvori novou instanci kopii
}
```

- odchycení všech výjimek (resp. zbytku): syntaxe třech teček

```
catch (...)
{
    // handler
}
```

Praktiky

- využít existující STL výjimku nebo oddědit vlastní výjimku od `std::exception`
- nepoužívat výjimky pokud to není nutné - nestandardní flow
- odchytávat pokud možno co nejspecifičtější výjimky (znalost situace), ale zbytečně neriskovat
- *throw by value, catch by reference*; nealokovat dynamicky

Příklad

- Prostor pro příklad 03_a_exceptions

Konec 3. části

```
exit(0);
```