C++; delete Java;

Část 5: I/O operace a streamy

Kennny

srpen 2017

I/O operace

- STL zaobaluje vstupně-výstupní operace do tzv. streamů
- dva streamy již známe - `std::cout` a `std::cin`
- mají přetížený operátor « (výstup) a/nebo » (vstup)

Stream

- kromě jiných části hierarchie nás zajímají dva předci streamů
  - `std::istream` - předek vstupního streamu
  - `std::ostream` - předek výstupního streamu
- již lze odvodit, že
  - `std::cout` je globální instance `std::ostream`
  - `std::cin` je globální instance `std::istream`
- od těchto předků pak dědí další varianty, nás budou zajímat zejména varianty pro práci s řetězcem a souborem

Vlastní přetížení operátoru

- je vhodné připomenout, že si můžeme přetížit operátor« a » jak potřebujeme pro práci s `std::istream` a `std::ostream`
- připomeňme například ten pro výpis souřadnic vektoru

```
std::ostream& operator<<(std::ostream& os, Vektor const& v)
{
    os << "(" << v.x << ",␣" << v.y << ")";
    return os;
}
```

- nutno dodat, že operaci definujeme nad `std::ostream`, tedy bude fungovat jak nad řetězcovým, tak souborovým, tak jakýmkoliv jiným výstupním streamem

Input stream

- při čtení je výchozí oddělovač zakončení řádku
- lze vynutit jiné použitím `std::getline` namísto operátoru »

```
std::getline(stream, targetstring, ';');
```

- má přetíženy operátory pro všechny primitivní typy a string

Řetězcový stream

- `#include <sstream>`
- `std::istringstream` a `std::ostringstream` - implementace `std::istream` a `std::ostream` nad řetězcem
- `std::stringstream` - vícenásobná dědičnost od obou předků
- pro vyzvednutí řetězce z `std::ostringstream`
  - metoda `str()`, vrací instanci `std::string`
  - pozor, nutno zkopírovat, po zaniknutí streamu zaniká i instance stringu

Příklad

- Prostor pro příklad 05_a_stringstream

Souborový stream

- `#include <fstream>`
- `std::ifstream` a `std::ofstream` - implementace `std::istream` a `std::ostream` nad souborem
- `std::fstream` - vícenásobná dědičnost od obou předků
- módy otevření souboru (bitmaska, lze skládat `|`)
  - `std::ios::in` - pro čtení ("r")
  - `std::ios::out` - pro zápis ("w")
  - `std::ios::binary` - binárně ("b")
  - `std::ios::app` - nepřepisovat, připojit za konec ("a")
- RAII struktury - destruktor zavírá soubor

Souborový stream

- pro textové čtení a zápis operátory « a »
- pro binární čtení a zápis metody `read()` a `write()`

Příklad

- Prostor pro příklad 05_b_files

Manipulátory výstupů

- `#include <iomanip>`
- vkládají se operátory « nebo » (podle typu streamu)
- parametrické a bezparametrické manipulátory

Číselný vstup/výstup

- prefixovat číslo základem? `std::showbase` ; rozmysleli jsme si to? `std::noshowbase`
- změna základu čísla: `std::setbase()` (podporuje 8, 10 a 16), případně:
  - `std::hex`, `std::dec`, `std::oct`

```
std::cout << std::hex  << std::showbase<< 127;
// vystup: 0x7F
```

- formát čísla s des. tečkou: `std::fixed`, `std::scientific`
- přesnost čísla s des. tečkou (počet des. míst): `std::setprecision`

```
std::cout << std::scientific << std::setprecision(2) << 12.3531;
// vystup: 1.23e+01
```

Další

- znaky čísla mají být velké? `std::uppercase` ; rozmysleli jsme si to? `std::nouppercase`
- chceme všechna čísla stejně dlouhá? `std::setw`
- chceme neobsazené cifry vypnit znakem? `std::setfill`
- bool hodnoty řetězcem? `std::boolalpha` ; rozmysleli jsme si to? `std::noboolalpha`
- a další... `http://en.cppreference.com/w/cpp/io/manip`

Příklad

- Prostor pro příklad 05_c_iomanip

Konec 5. části

```
exit(0);
```