C++; delete Java;

Část 8: pseudonáhodná čísla

Kennny

srpen 2017

Generátory

- je k dispozici několik generátorů
- tyto generátory se vyznačují tím, že každý generuje "jinou náhodu"
- pro nás je důležité jen to, že existují
- `std::random_device` - obaluje zdroj náhody, např. `/dev/urandom` pokud je k dispozici, apod.
- může být ale pomalý, hodí se např. jen pro inicializaci generátoru pseudonáhodných čísel

Generátory

- v STL existuje více generátorů pseudonáh. čísel
  - `std::default_random_engine` - alias pro jiný generátor
  - `std::mersenne_twister_engine` - Mersenne Twister generátor
  - `std::linear_congruental_engine` - lineární kongruentní generátor
- ty mají často ale své parametry
- na to v STL bylo myšleno a byly vytvořeny aliasy s ověřenou parametrizací
  - `std::mt19937` - většinou výchozí, mersenne twister
  - `std::minstd_rand`
  - `std::knuth_b`
  - ...

Rozložení

- samotný generátor nestačí
- jím generované hodnoty musí projít ještě generátorem rozložení
- k dispozici jsou standardní rozložení
  - `std::uniform_int_distribution` - rovnoměrné celočíselné
  - `std::uniform_real_distribution` - rovnoměrné desetinné
  - `std::normal_distribution` - normální
  - `std::poisson_distribution` - Poissonovo
  - `std::chi_squared_distribution` - Chí-kvadrát
  - a spousty dalších
- šablonové typy

Příklad

- prakticky vždy si vystačíme s konstrukcí podobnou této

```
// klidne jen jeden na cely program
std::random_device rd;

// vytvoreni PRNG generatoru, jako seed je pouzita hodnota
// z random device
std::default_random_engine eng(rd());
std::uniform_int_distribution<int> dist(1, 6);

int hraciKostka()
{
    return dist(eng);
}
```

Příklad

- Prostor pro příklad 08_a_random

Konec 8. části

```
exit(0);
```