Programowanie Współbieżne w C++ 4 fun && $

warsztaty

Bartek ’BaSz’ Szurgot

bartosz.1.szurgot@nokia.com

12 kwietnia 2016

Co potrzebne?

https://github.com/nokia-wroclaw/wroclaw_does_it

Pora na. . .

Intro

Wybrane wzorce projektowe

std::thread

- Wątek:
  - `std::thread::thread(func)`
  - `std::thread::join()`
  - `http://en.cppreference.com/w/cpp/thread/thread`

std::thread

- Wątek:
  - `std::thread::thread(func)`
  - `std::thread::join()`
  - `http://en.cppreference.com/w/cpp/thread/thread`
- ID:
  - `std::this_thread::get_id()`
  - `http://en.cppreference.com/w/cpp/thread/get_id`

std::thread

- Wątek:
  - `std::thread::thread(func)`
  - `std::thread::join()`
  - `http://en.cppreference.com/w/cpp/thread/thread`
- ID:
  - `std::this_thread::get_id()`
  - `http://en.cppreference.com/w/cpp/thread/get_id`
- Zadanie:
  - Uruchomić wątek
  - Wypisać *Hello ID thread!*
  - Zamiast ID – identyfikator wątku
  - Przyłączyć wątek

std::mutex

- Mutex:
  - `std::mutex::lock()`
  - `std::mutex::unlock()`
  - `http://en.cppreference.com/w/cpp/thread/mutex`

[1] Resource Acquisition Is Initialization

std::mutex

- Mutex:
  - `std::mutex::lock()`
  - `std::mutex::unlock()`
  - `http://en.cppreference.com/w/cpp/thread/mutex`
- Blokada RAII[1]:
  - `std::unique_lock`
  - `http://en.cppreference.com/w/cpp/thread/unique_lock`

[1] Resource Acquisition Is Initialization

std::mutex

- Mutex:
  - `std::mutex::lock()`
  - `std::mutex::unlock()`
  - `http://en.cppreference.com/w/cpp/thread/mutex`
- Blokada RAII[1]:
  - `std::unique_lock`
  - `http://en.cppreference.com/w/cpp/thread/unique_lock`
- Zadanie:
  - Wypisać *Hello ID thread!* z 4 wątków
  - Rozwiązać problem „przeplotu” podczas wypisywania
  - Użyć klasy RAII do blokady

[1] Resource Acquisition Is Initialization

std::condtion_variable

- Zmienna warunkowa:
  - `std::condtion_variable::wait()`
  - `std::condtion_variable::notify_all()`
  - `http://en.cppreference.com/w/cpp/thread/condition_variable`

std::condtion_variable

- Zmienna warunkowa:
  - `std::condtion_variable::wait()`
  - `std::condtion_variable::notify_all()`
  - `http://en.cppreference.com/w/cpp/thread/condition_variable`
- Zadanie:
  - Uruchomić 4 wątki
  - Zaczekać aż wszystkie wątki będą gotowe do działania
  - Wypisać *Hello ID thread!* z wszystkich wątków (bez przeplotu)

Pora na. . .

Intro

Wybrane wzorce projektowe

Kolejka

- Szablon klasy:

```
1 template<typename T>
2 struct MyTemplate
3 {
4   // ...
5   std::vector<T> data_;
6 };
```

[2] First-In First-Out

Kolejka

- Szablon klasy:

```
1 template<typename T>
2 struct MyTemplate
3 {
4   // ...
5   std::vector<T> data_;
6 };
```

- Zadanie:
  - Zaimplementować szablon kolejki FIFO[2]
  - Uruchomić 2 wątki
  - Przekazywać napisy (`std::string`) między wątkami co 1[s]
  - Pierwszy wątek dodaje napisy do kolejki
  - Drugi wątek odczytuje z kolejki i wypisuje na ekranie

[2] First-In First-Out

Pula wątków

- Poszukiwanie liczb pierwszych:

```
1 bool isPrime(const uint64_t n)
2 {
3   for(unsigned i=2; i<n; ++i)
4     if( (n%i)==0 )
5       return false;
6   return true;
7 }
```

Pula wątków

- Poszukiwanie liczb pierwszych:

```
1 bool isPrime(const uint64_t n)
2 {
3   for(unsigned i=2; i<n; ++i)
4     if( (n%i)==0 )
5       return false;
6   return true;
7 }
```

- Zadanie:
  - Uruchomić N wątków
  - Każdy wątek pobiera liczbę z kolejki
  - Dla każdej liczy jest wyliczane czy jest liczbą pierwszą czy nie
  - Wynik jest wypisywany na ekranie

Future/promise

- Future:
  - `std::future::get()`
  - http://en.cppreference.com/w/cpp/thread/future

Future/promise

- Future:
  - `std::future::get()`
  - http://en.cppreference.com/w/cpp/thread/future
- Promise:
  - `std::promise::get_future()`
  - `std::promise::set_value()`
  - `http://en.cppreference.com/w/cpp/thread/promise`

Future/promise

- Future:
  - `std::future::get()`
  - http://en.cppreference.com/w/cpp/thread/future
- Promise:
  - `std::promise::get_future()`
  - `std::promise::set_value()`
  - `http://en.cppreference.com/w/cpp/thread/promise`
- Zadanie:
  - Przerobić program z „puli wątków”
  - Przekazywać wartości `promise`
  - Wypisywać wynik z `future` w `main()`

Pytania i (mam nadzieję) odpowiedzi :-)