

Programowanie Współbieżne w C++ 4 fun && \$ warsztaty

Bartek 'BaSz' Szurgot

`bartosz.1.szurgot@nokia.com`

21 kwietnia 2016

Co potrzebne?

Programowanie
Współbieżne
w C++ 4fun&&\$

BaSz

Intro

Wybrane wzorce
projektowe



[https://github.com/nokia-wroclaw/
wroclaw_does_it](https://github.com/nokia-wroclaw/wroclaw_does_it)

Pora na...

Programowanie
Współbieżne
w C++ 4fun&&\$

BaSz

Intro

Wybrane wzorce
projektowe

Intro

Wybrane wzorce projektowe

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

- ▶ 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

► Mutex:

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

¹Resource Acquisition Is Initialization

- ▶ Mutex:
 - ▶ `std::mutex::lock()`
 - ▶ `std::mutex::unlock()`
 - ▶ <http://en.cppreference.com/w/cpp/thread/mutex>
- ▶ Blokada RAII¹:
 - ▶ `std::unique_lock`
 - ▶ http://en.cppreference.com/w/cpp/thread/unique_lock

¹Resource Acquisition Is Initialization

- ▶ Mutex:
 - ▶ `std::mutex::lock()`
 - ▶ `std::mutex::unlock()`
 - ▶ <http://en.cppreference.com/w/cpp/thread/mutex>
- ▶ Blokada RAII¹:
 - ▶ `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

¹Resource Acquisition Is Initialization

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

- ▶ Zmienna warunkowa:
 - ▶ `std::condition_variable::wait()`
 - ▶ `std::condition_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 przepłotu)

- ▶ Zmienna warunkowa:
 - ▶ `std::condition_variable::wait()`
 - ▶ `std::condition_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 przepłotu)
 - ▶ 1xCV...? 2xCV...?

Pora na...

Programowanie
Współbieżne
w C++ 4fun&&\$

BaSz

Intro

Wybrane wzorce
projektowe

Intro

Wybrane wzorce projektowe

► Szablon klasy:

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

► Szablon klasy:

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

► Zadanie:

- Zaimplementować szablon kolejki FIFO²
- 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

²First-In First-Out

- Poszukiwanie liczb pierwszych:

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


► Poszukiwanie liczb pierwszych:

```
1  bool isPrime(const unsigned 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 liczby jest wyliczane czy jest liczbą pierwszą czy nie
- Wynik jest wypisywany na ekranie

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

- ▶ 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:
 - ▶ `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 :-)

Programowanie
Współbieżne
w C++ 4fun&&\$

BaSz

Intro

Wybrane wzorce
projektowe

