Яндекс Такси

Итоги встречи на Коне

Полухин Антон

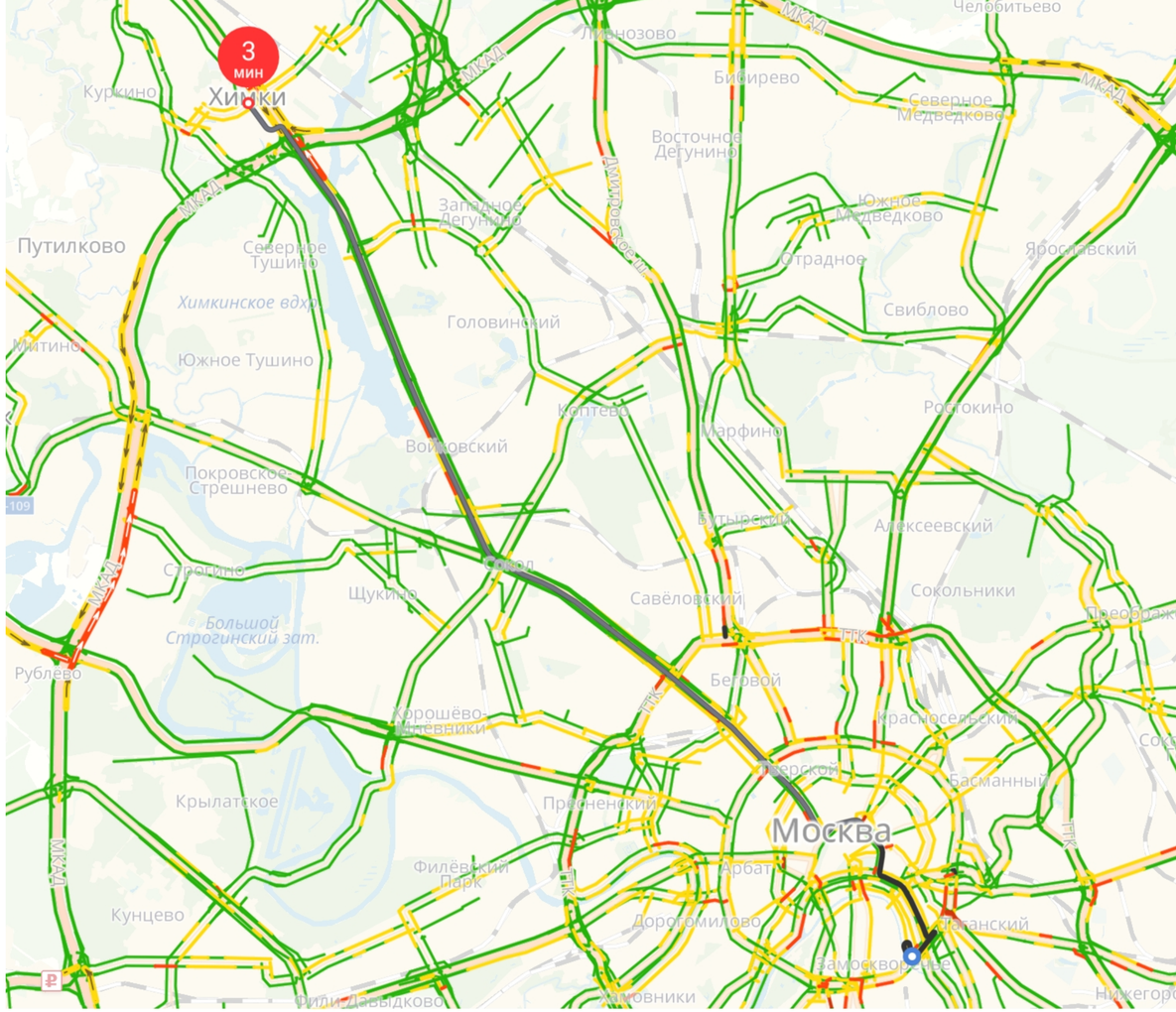
Antony Polukhin

Яндекс Такси


ACCU
ALLSTAR
DEPT.

Содержание

- Модули
- Сопрограммы
- std::format
- std::stacktrace
- Что дальше?



Модули?

Модули!

Модулям быть в C++20!

Итоги встречи на Коне

Modules Inro

```
// экспортирует макросы, экспортирует все символы, чувствительно к макросам
#include <iostream>
```

Modules Inro

```
// экспортирует макросы, экспортирует все символы, чувствительно к макросам
#include <iostream>

// экспортирует макросы, экспортирует все символы, к макросам НЕ чувствительно
import <iostream>;
```

Modules Inro

```
// экспортирует макросы, экспортирует все символы, чувствительно к макросам
#include <iostream>

// экспортирует макросы, экспортирует все символы, к макросам НЕ чувствительно
import <iostream>;

// НЕ экспортирует макросы, к макросам НЕ чувствительно
import std.iostream; // TODO: не в C++20
```

Modules Impl base

```
import <iostream>; // Что под капотом?
```

Modules Impl base

```
// <iostream>
#ifndef _GLIBCXX_IOSTREAM
#define _GLIBCXX_IOSTREAM 1
#pragma GCC system_header

#include <bits/c++config.h>
#include <ostream>
#include <istream>
<...>

#endif /* _GLIBCXX_IOSTREAM */
```

Modules Impl supreme

```
import std.iostream; // Что под капотом?
```

Modules Impl supreme

```
// std.iostream


// Всё будет не так, но для примера сгодится:
module std.iostream;
export import std.istream;
export import std.ostream; // Что под капотом?
```

Modules Impl supreme

```
// std.ostream


// Наверное как-то вот так:
export module std.ostream;
import <ostream>;
export std::basic_ostream;
export std::ostream;
export std::wostream;
...
```

Корутины?

Корутины!

Корутинам быть в C++20!

Итоги встречи на Коне

Coroutines Inro

```
awaitable<void> echo(tcp::socket socket) {
  try {
    char data[1024];
    for (;;) {
      std::size_t n = co_await socket.async_read_some(boost::asio::buffer(data),
                                                      use_awaitable);
      co_await async_write(socket, boost::asio::buffer(data, n), use_awaitable);
    }
  } catch (std::exception& e) {
    std::printf("echo Exception: %s\n", e.what());
  }
}
```

Coroutines Inro

```
awaitable<void> echo(tcp::socket socket) {
  try {
    char data[1024];
    for (;;) {
      std::size_t n = co_await socket.async_read_some(boost::asio::buffer(data),
                                                      use_awaitable);
      co_await async_write(socket, boost::asio::buffer(data, n), use_awaitable);
    }
  } catch (std::exception& e) {
    std::printf("echo Exception: %s\n", e.what());
  }
}
```

Coroutines Inro

```
awaitable<void> echo(tcp::socket socket) {
  try {
    char data[1024];
    for (;;) {
      std::size_t n = co_await socket.async_read_some(boost::asio::buffer(data),
                                                      use_awaitable);
      co_await async_write(socket, boost::asio::buffer(data, n), use_awaitable);
    }
  } catch (std::exception& e) {
    std::printf("echo Exception: %s\n", e.what());
  }
}
```

Coroutines Inro

```
awaitable<void> echo(tcp::socket socket) {
  try {
    char data[1024];
    for (;;) {
      std::size_t n = co_await socket.async_read_some(boost::asio::buffer(data),
                                                      use_awaitable);
      co_await async_write(socket, boost::asio::buffer(data, n), use_awaitable);
    }
  } catch (std::exception& e) {
    std::printf("echo Exception: %s\n", e.what());
  }
}
```

Coroutines Inro

```
awaitable<void> echo(tcp::socket socket) {
  try {
    char data[1024];
    for (;;) {
      std::size_t n = co_await socket.async_read_some(boost::asio::buffer(data),
                                                      use_awaitable);
      co_await async_write(socket, boost::asio::buffer(data, n), use_awaitable);
    }
  } catch (std::exception& e) {
    std::printf("echo Exception: %s\n", e.what());
  }
}
```

Coroutines Inro

```
awaitable<void> echo(tcp::socket socket) {
  try {
    char data[1024];
    for (;;) {
      std::size_t n = co_await socket.async_read_some(boost::asio::buffer(data),
                                                      use_awaitable);
      co_await async_write(socket, boost::asio::buffer(data, n), use_awaitable);
    }
  } catch (std::exception& e) {
    std::printf("echo Exception: %s\n", e.what());
  }
}
```

Coroutines Inro

```
awaitable<void> echo(tcp::socket socket) {
  try {
    char data[1024];
    for (;;) {
      std::size_t n = co_await socket.async_read_some(boost::asio::buffer(data),
                                                      use_awaitable);
      co_await async_write(socket, boost::asio::buffer(data, n), use_awaitable);
    }
  } catch (std::exception& e) {
    std::printf("echo Exception: %s\n", e.what());
  }
}
```

Coroutines Inro

```
awaitable<void> listener() {
  auto executor = co_await this_coro::executor;
  tcp::acceptor acceptor(executor, {tcp::v4(), 55555});
  for (;;) {
    tcp::socket socket = co_await acceptor.async_accept(use_awaitable);
    co_spawn(executor,
             [socket = std::move(socket)]() mutable {
               return echo(std::move(socket));
             },
             detached);
  }
}
```

Coroutines Inro

```
awaitable<void> listener() {
  auto executor = co_await this_coro::executor;
  tcp::acceptor acceptor(executor, {tcp::v4(), 55555});
  for (;;) {
    tcp::socket socket = co_await acceptor.async_accept(use_awaitable);
    co_spawn(executor,
             [socket = std::move(socket)]() mutable {
               return echo(std::move(socket));
             },
             detached);
  }
}
```

Coroutines Inro

```
awaitable<void> listener() {
  auto executor = co_await this_coro::executor;
  tcp::acceptor acceptor(executor, {tcp::v4(), 55555});
  for (;;) {
    tcp::socket socket = co_await acceptor.async_accept(use_awaitable);
    co_spawn(executor,
             [socket = std::move(socket)]() mutable {
               return echo(std::move(socket));
             },
             detached);
  }
}
```

Coroutines Inro

```
awaitable<void> listener() {
  auto executor = co_await this_coro::executor;
  tcp::acceptor acceptor(executor, {tcp::v4(), 55555});
  for (;;) {
    tcp::socket socket = co_await acceptor.async_accept(use_awaitable);
    co_spawn(executor,
             [socket = std::move(socket)]() mutable {
               return echo(std::move(socket));
             },
             detached);
  }
}
```

Coroutines Inro

```
awaitable<void> listener() {
  auto executor = co_await this_coro::executor;
  tcp::acceptor acceptor(executor, {tcp::v4(), 55555});
  for (;;) {
    tcp::socket socket = co_await acceptor.async_accept(use_awaitable);
    co_spawn(executor,
             [socket = std::move(socket)]() mutable {
               return echo(std::move(socket));
             },
             detached);
  }
}
```

Coroutines Inro

```
int main() {
  try {
    boost::asio::io_context io_context(1);
    boost::asio::signal_set signals(io_context, SIGINT, SIGTERM);
    signals.async_wait([&](auto, auto) { io_context.stop(); });
    co_spawn(io_context, listener, detached);
    io_context.run();
  } catch (std::exception& e) {
    std::printf("Exception: %s\n", e.what());
  }
}
```

Корутины могут стать красивее!

Coroutines Inro

```
awaitable<void> echo(tcp::socket socket) {
  try {
    char data[1024];
    for (;;) {
      std::size_t n = co_await socket.async_read_some(boost::asio::buffer(data),
                                                      use_awaitable);
      co_await async_write(socket, boost::asio::buffer(data, n), use_awaitable);
    }
  } catch (std::exception& e) {
    std::printf("echo Exception: %s\n", e.what());
  }
}
```

Coroutines Inro

```
awaitable<void> echo(tcp::socket socket) async {
  try {
    char data[1024];
    for (;;) {
      std::size_t n = await socket.async_read_some(boost::asio::buffer(data),
                                                   use_awaitable);
      await async_write(socket, boost::asio::buffer(data, n), use_awaitable);
    }
  } catch (std::exception& e) {
    std::printf("echo Exception: %s\n", e.what());
  }
}
```

std::format

Format Inro

Format Inro

```
std::string res0 = std::format("{} from {}", "Hello", "Russia");
```

Format Inro

```
std::string res0 = std::format("{} from {}", "Hello", "Russia");


std::string res1 = std::format("{1} from {0}", "Russia", "Hello");
```

Format Inro

```
std::string res0 = std::format("{} from {}", "Hello", "Russia");


std::string res1 = std::format("{1} from {0}", "Russia", "Hello");


int width = 10;
int precision = 3;
std::string s = std::format("{0:{1}.{2}f}", 12.345678, width, precision); // "    12.346"
```

Format Inro

```
std::string res0 = std::format("{} from {}", "Hello", "Russia");


std::string res1 = std::format("{1} from {0}", "Russia", "Hello");


int width = 10;
int precision = 3;
std::string s = std::format("{0:{1}.{2}f}", 12.345678, width, precision); // "    12.346"


std::array<char, 200> buffer;
std::format_to_n(buffer.data(), buffer().size(), "{0:b} {0:d} {0:o} {0:x}", 42);
assert(buffer.data() == "101010 42 52 2a"sv);
```

std::stacktrace

std::stacktrace

std::stacktrace

- std::stack_frame → std::stacktrace_entry

std::stacktrace

- std::stack_frame → std::stacktrace_entry
- std::stacktrace → std::backtrace

std::stacktrace

- std::stack_frame → std::stacktrace_entry
- std::stacktrace → std::backtrace
- std::stacktrace_entry → std::backtrace_entry

Есть замечания к C++20?

https://stdcpp.ru

C++
РГ21 C++ РОССИЯ

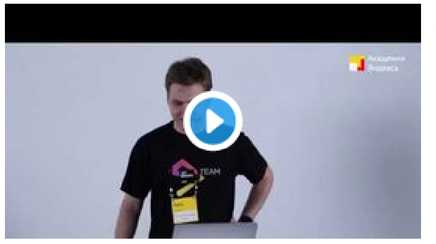
RSS Ru En yndx-antoshkka, выход

- О проекте
- Новости
- Предложения
- Вопросы и ответы
- Инструкция по подготовке proposal

stdcppru
@stdcppru

Для тех, кто пропустил встречу в декабре:
– Обзор встречи Комитета по стандартизации C++ в Сан-Диего, США – Антон Полухин
youtube.com/watch?v=QaDO9L…
– Модули в C++ – Дмитрий Кожевников
youtube.com/watch?v=p8MkTJ…

YouTube @YouTube



9 янв. 2019 г.

Помогаем готовить предложения для защиты перед рабочей группой Комитета ISO C++

Новости

27 ноября 2018 29 мая 2018 14 марта 2018

Спасибо

Полухин Антон

Старший разработчик Yandex.Taxi

antoshkka@gmail.com

antoshkka@yandex-team.ru

https://github.com/apolukhin

https://stdcpp.ru/

РГ21 C++ РОССИЯ