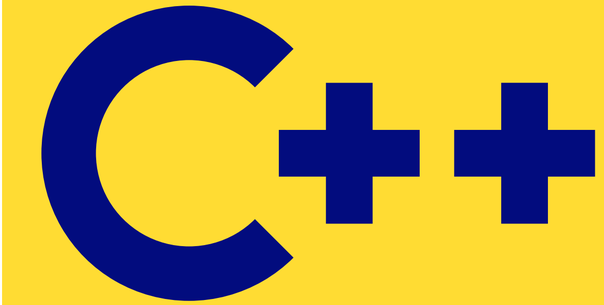
C++26

Новости последних встреч ISO



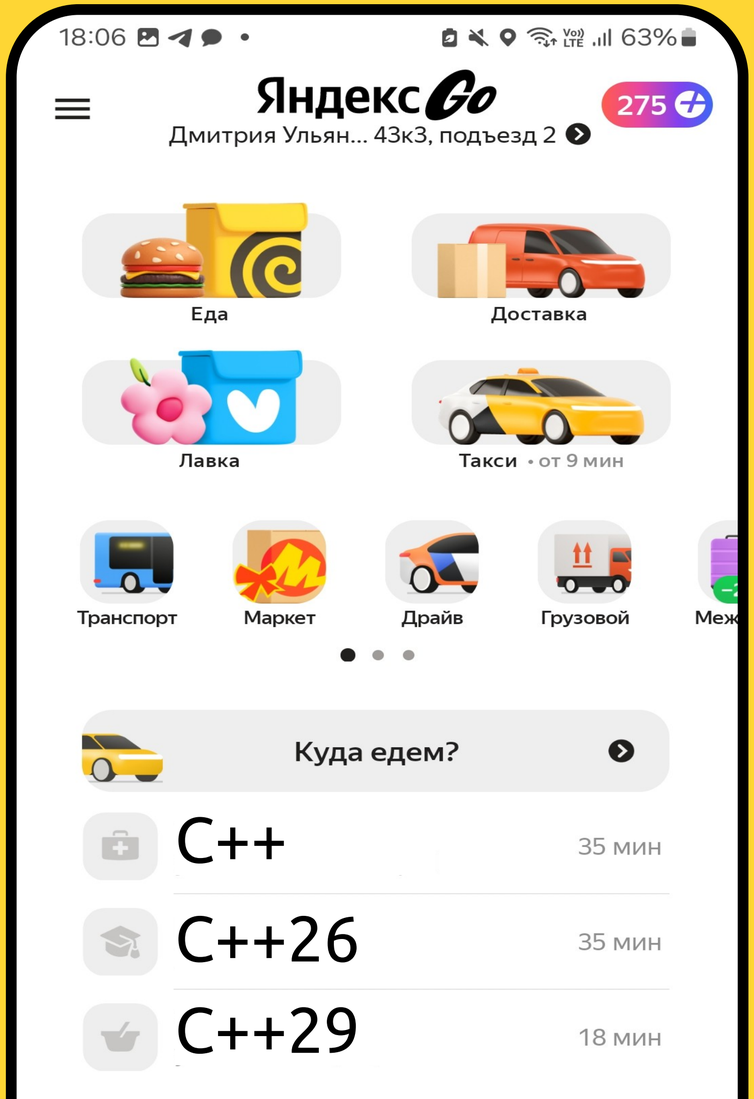
РГ21 C++ РОССИЯ

Полухин Антон

Antony Polukhin

Содержание

- Tuple protocol
- std::format
- Hardening
- consteval



Tuple protocol

Tuple protocol

Tuple protocol

```
template <std::size_t N>
struct sequence{};
```


Tuple protocol

```
template <std::size_t N>
struct std::tuple_size<sequence<N>>
         : std::integral_constant<std::size_t, N>
{};
```


Tuple protocol

```
template <std::size_t N>
struct std::tuple_size<sequence<N>>
        : std::integral_constant<std::size_t, N>
{};
template <std::size_t I, std::size_t N>
struct std::tuple_element<I, sequence<N>> {
    using type = std::size_t;
};
```


Tuple protocol

```
template <std::size_t N>
struct std::tuple_size<sequence<N>>
        : std::integral_constant<std::size_t, N>
{};
template <std::size_t I, std::size_t N>
struct std::tuple_element<I, sequence<N>> {
    using type = std::size_t;
};
template<std::size_t I, std::size_t N>
constexpr std::size_t get(sequence<N>) noexcept {
    return I;
}
```


Tuple protocol

```
template <std::size_t N>
struct std::tuple_size<sequence<N>>
        : std::integral_constant<std::size_t, N>
{};
template <std::size_t I, std::size_t N>
struct std::tuple_element<I, sequence<N>> {
    using type = std::size_t;
};
template<std::size_t I, std::size_t N>
constexpr std::size_t get(sequence<N>) noexcept {
    return I;
}
```


Tuple protocol

```
auto sample() {
    auto [x0, x1, x2] = sequence<3>{};
    assert(x0 = 0);
    assert(x1 = 1);
    assert(x2 = 2);
}
```


Tuple protocol

```
auto sample2(auto tuple) {
    constexpr auto [...I] = sequence<3>{};
    return (std::get<I>(tuple) + ...);
}
```


Tuple protocol

```
auto sample2(auto tuple) {
    constexpr auto [...I] = sequence<3>{};
    return (std::get<I>(tuple) + ...);
}

auto sample3(auto tuple) {
    int sum = 0;
    template for(constexpr std::size_t I : sequence<3>()) {
        sum += std::get<I>(tuple);
    }
    return sum;
}
```


Tuple protocol for std::integer_sequence

```
template<class T, T... Values>
struct tuple_size<integer_sequence<T, Values...>>;


template<size_t I, class T, T... Values>
struct tuple_element<I, integer_sequence<T, Values...>>;


template<size_t I, class T, T... Values>
constexpr T get(integer_sequence<T, Values...>) noexcept;
```


Tuple protocol for std::integer_sequence

```
auto sample2(auto tuple) {
    constexpr auto [...I] = std::make_index_sequence<3>{};
    return (std::get<I>(tuple) + ...);
}

auto sample3(auto tuple) {
    int sum = 0;
    template for(constexpr std::size_t I: std::make_index_sequence<3>()) {
        sum += std::get<I>(tuple);
    }
    return sum;
}
```


Диагностика

Диагностические сообщения

```
template <class T>
void CallMe(T x) {
    return sqrt(x);
}
```


Диагностические сообщения

```
#include <boost/type_index/ctti_type_index.hpp>
#include <format>
template <class T>
void CallMe(T x) {
    static_assert (
        requires { sqrt(x); }
    );
    return sqrt(x);
}
```


Диагностические сообщения

```
#include <boost/type_index/ctti_type_index.hpp>
#include <format>
template <class T>
void CallMe(T x) {
    static_assert (
        requires { sqrt(x); }, std::format(
            "Define sqrt() function for {0} in its namespace",
            boost::typeindex::ctti_type_index::type_id<T>().name()
        ).c_str()
    );
    return sqrt(x);
}
```

Пример из userver

```
template <class T, std::size_t Size /* ... */>
class FastPimpl {
private:
    template <std::size_t ActualSize /* ... */>
    static void Validate() noexcept {
        static_assert(
            Size >= ActualSize,
            "invalid Size: Size >= sizeof(T) failed"

        );
        // ...
    }
```


Пример из userver

```
template <class T, std::size_t Size /* ... */>
class FastPimpl {
private:
    template <std::size_t ActualSize /* ... */>
    static void Validate() noexcept {
        static_assert(
            Size >= ActualSize, std::format(
                "'Size' should be set to at least {}.", Size
            ).c_str()
        );
        // ...
    }
```


Пример из userver

```
template <class T, std::size_t Size /* ... */>
class FastPimpl {
private:
    ~FastPimpl() {
        static_assert(
            Size >= sizeof(T), std::format(
                "’Size’ should be set to at least {}.", Size
            ).c_str()

        );
        // ...
    }
```

Дальнейшие улучшения

Дальнейшие улучшения

- P3652 Constexpr floating-point <charconv> functions

Баги

Что случится?

```
std::optional<int> opt;
return *opt;
```


Что случится?

```
std::optional<int> opt;
return *opt;  // C Hardening: SIGABORT, trap, std::terminate() ...
```


Hardening & Contracts

```
std::optional<int> opt;
return *opt;


int isqrt(int n) noexcept
  pre(n >= 0)
  post(r: r >= 0)
;
```


Дальнейшие улучшения контрактов

Дальнейшие улучшения контрактов

1 Виртуальные функции

Дальнейшие улучшения контрактов

1. Виртуальные функции
2. Side effects & UB

Дальнейшие улучшения контрактов

1. Виртуальные функции
2. Side effects & UB
3. Отключение части контрактов

Дальнейшие улучшения контрактов

1. Виртуальные функции
2. Side effects & UB
3. Отключение части контрактов
4. Теги и более гранулярная настройка

Reflection

Доступность приватных членов

Доступность приватных членов

!

Рефлексия — это замена для внешних утилит, которые работают с C++ заголовками

Доступность приватных членов

!

Рефлексия — это замена для внешних утилит, которые работают с C++ заголовками

Всё что видно в исходнике должно быть доступно рефлексии

consteval block

consteval block

```
#include <meta>

template<typename... Ts> struct Tuple {
  struct storage;




  storage data;
};
```


consteval block

```
#include <meta>

template<typename... Ts> struct Tuple {
  struct storage;
  consteval {



  }
  storage data;
};
```


consteval block

```
#include <meta>

template<typename... Ts> struct Tuple {
  struct storage;
  consteval {
    std::meta::define_class(^storage,
                {std::meta::data_member_spec(^Ts)...});
  }
  storage data;
};
```


Прочие баги

P3725

```cpp
#include <ranges>

std::vector<std::string> coll1{"Amsterdam", "Berlin", "Cologne", "LA"};
// Перемещаем длинные строки в обратном порядке в другой контейнер
auto large = [](const auto& s) { return s.size() > 5; };
auto sub = coll1 | std::views::filter(large)
                 | std::views::reverse
                 | std::views::as_rvalue
                 | std::ranges::to<std::vector>();
```


… ещё проблемы

… ещё проблемы

1. UB в <type_traits>

… ещё проблемы

1. UB в <type_traits>
2. basic_string::append/assign

… ещё проблемы

1. UB в <type_traits>
2. basic_string::append/assign
3. uniform_int_distribution<uint8_t>

… ещё проблемы

1. UB в <type_traits>
2. basic_string::append/assign
3. uniform_int_distribution<uint8_t>
4. …

… ещё проблемы

1. UB в <type_traits>
2. basic_string::append/assign
3. uniform_int_distribution<uint8_t>
4. …
5. … и ещё ~300 проблем

C++26

… а ещё

… а ещё

1. SIMD

… а ещё

1. SIMD
2. Executors

… а ещё

1. SIMD
2. Executors
3. Constexpr

… а ещё

1. SIMD
2. Executors
3. Constexpr
4. Linalg

… а ещё

1. SIMD
2. Executors
3. Constexpr
4. Linalg
5. Hazard Pointer

… а ещё

1. SIMD
2. Executors
3. Constexpr
4. Linalg
5. Hazard Pointer
6. Freestanding

… а ещё

1. SIMD
2. Executors
3. Constexpr
4. Linalg
5. Hazard Pointer
6. Freestanding
7. relocate

… а ещё

1. SIMD
2. Executors
3. Constexpr
4. Linalg
5. Hazard Pointer
6. Freestanding
7. ~~relocate~~

… а ещё

1. SIMD
2. Executors
3. Constexpr
4. Linalg
5. Hazard Pointer
6. Freestanding
7. ~~relocate~~
8. std::hive

… а ещё

1. SIMD
2. Executors
3. Constexpr
4. Linalg
5. Hazard Pointer
6. Freestanding
7. ~~relocate~~
8. std::hive
9. Ranges

… а ещё

1. SIMD
2. Executors
3. Constexpr
4. Linalg
5. Hazard Pointer
6. Freestanding
7. ~~relocate~~
8. std::hive
9. Ranges
A. -UB

… а ещё

1. SIMD
2. Executors
3. Constexpr
4. Linalg
5. Hazard Pointer
6. Freestanding
7. ~~relocate~~
8. std::hive
9. Ranges
A. -UB
B. auto [x…] = t; x...[42];

… а ещё

1. SIMD
2. Executors
3. Constexpr
4. Linalg
5. Hazard Pointer
6. Freestanding
7. ~~relocate~~
8. std::hive
9. Ranges
A. -UB
B. auto [x…] = t; x...[42];
C. ...

Спасибо

Полухин Антон

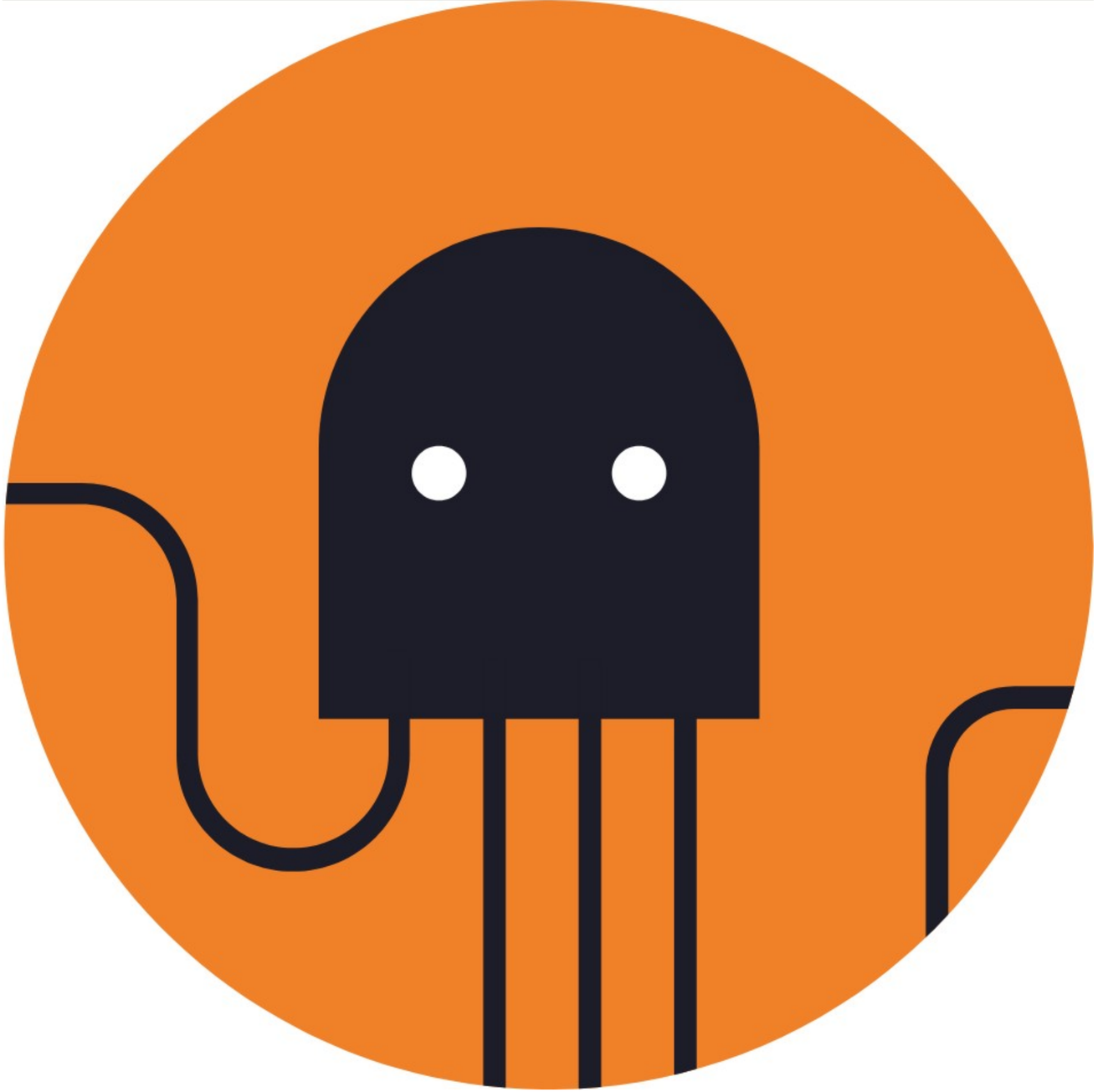
Эксперт-разработчик C++

antoshkka@gmail.com

antoshkka@yandex-team.ru

https://github.com/apolukhin

C++ РГ21 C++ РОССИЯ https://stdcpp.ru/



https://github.com/userver-framework