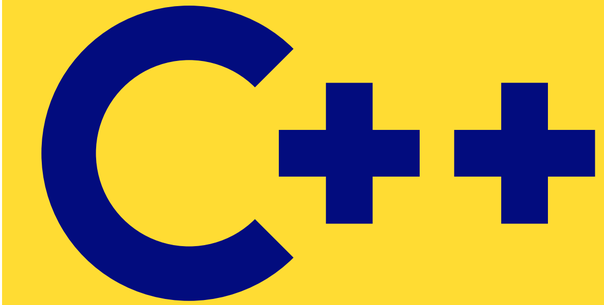
C++26

Новости последних встреч ISO



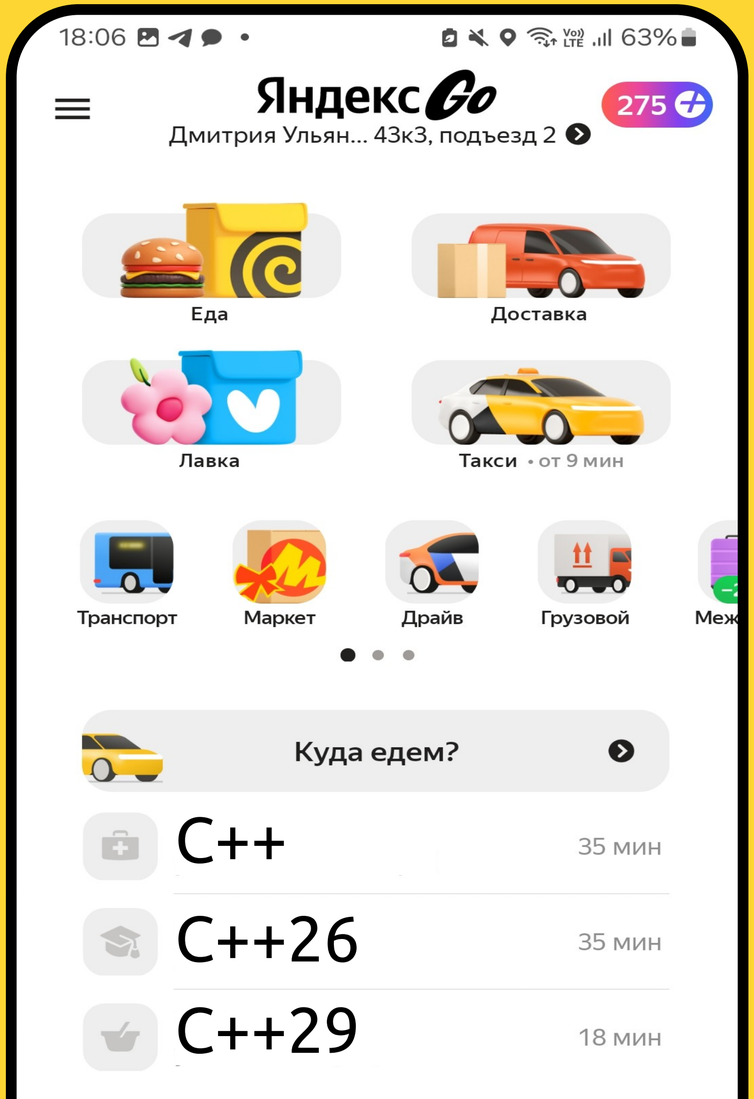
РГ21 C++ РОССИЯ

Полухин Антон

Antony Polukhin

Содержание

- Contracts
- Reflection
- #embed
- Сверхспособности



Почувствуем себя компилятором...

Почувствуем себя компилятором...

```
template<class T>
constexpr T foobarbuz(T n) noexcept {

  if (n <= T{1})
    return n;

  T i_current{0}, i_next{T(T{1} << ((std::bit_width(
     std::make_unsigned_t<T>(T(n - 1))) + 1) >> 1))};
  do {
    i_current = i_next;
    i_next = T((i_current + n / i_current) >> 1);
  } while (i_next < i_current);

  return i_current;
}
```


Почувствуем себя компилятором...

```
int main() {
    return foobarbuz(-4);
}
```


Почувствуем себя компилятором...

```
template<class T>
constexpr T foobarbuz(T n) noexcept {

  if (n <= T{1})
    return n;
  T i_current{0}, i_next{T(T{1} << ((std::bit_width(
     std::make_unsigned_t<T>(T(n - 1))) + 1) >> 1))};

  do {
    i_current = i_next;
    i_next = T((i_current + n / i_current) >> 1);
  } while (i_next < i_current);

  return i_current;
}
```

Почувствуем себя человеком...

```
template<class T>
constexpr T isqrt(T n) noexcept {

  if (n <= T{1})
    return n;
  T i_current{0}, i_next{T(T{1} << ((std::bit_width(
     std::make_unsigned_t<T>(T(n - 1))) + 1) >> 1))};

  do {
    i_current = i_next;
    i_next = T((i_current + n / i_current) >> 1);
  } while (i_next < i_current);

  return i_current;
}
```


Почувствуем себя человеком...

```
template<class T>
constexpr T isqrt(T n) noexcept {
  assert(n >= 0);
  if (n <= T{1})
    return n;
  T i_current{0}, i_next{T(T{1} << ((std::bit_width(
     std::make_unsigned_t<T>(T(n - 1))) + 1) >> 1))};
  assert(i_next >= 0);
  do {
    i_current = i_next;
    i_next = T((i_current + n / i_current) >> 1);
  } while (i_next < i_current);
  assert(i_current >= 0);
  return i_current;
}
```


Почувствуем себя человеком...

```
template<class T>
constexpr T isqrt(T n) noexcept {
  assert(n >= 0);
  if (n <= T{1})
    return n;
  T i_current{0}, i_next{T(T{1} << ((std::bit_width(
     std::make_unsigned_t<T>(T(n - 1))) + 1) >> 1))};
  assert(i_next >= 0);
  do {
    i_current = i_next;
    i_next = T((i_current + n / i_current) >> 1);
  } while (i_next < i_current);
  assert(i_current >= 0);
  return i_current;
}
```


Да тут же бага!..

```
int main() {
    return isqrt(-4);
}
```


Почувствуем себя человеком...

```
template<class T>
constexpr T isqrt(T n) noexcept {
  assert(n >= 0);
  if (n <= T{1})
    return n;
  T i_current{0}, i_next{T(T{1} << ((std::bit_width(
     std::make_unsigned_t<T>(T(n - 1))) + 1) >> 1))};
  assert(i_next >= 0);
  do {
    i_current = i_next;
    i_next = T((i_current + n / i_current) >> 1);
  } while (i_next < i_current);
  assert(i_current >= 0);
  return i_current;
}
```


Почувствуем себя человеком...

```
template<class T>
constexpr T isqrt(T n) noexcept {
  assert(n >= 0);
  if (n <= T{1})
    return n;
  T i_current{0}, i_next{T(T{1} << ((std::bit_width(
     std::make_unsigned_t<T>(T(n - 1))) + 1) >> 1))};
  assert(i_next >= 0);
  do {
    i_current = i_next;
    i_next = T((i_current + n / i_current) >> 1);
  } while (i_next < i_current);
  assert(i_current >= 0);
  return i_current;
}
```


Почувствуем себя человеком...

```
template<class T>
constexpr T isqrt(T n) noexcept {
  contract_assert(n >= 0);
  if (n <= T{1})
    return n;
  T i_current{0}, i_next{T(T{1} << ((std::bit_width(
     std::make_unsigned_t<T>(T(n - 1))) + 1) >> 1))};
  contract_assert(i_next >= 0);
  do {
    i_current = i_next;
    i_next = T((i_current + n / i_current) >> 1);
  } while (i_next < i_current);
  contract_assert(i_current >= 0);
  return i_current;
}
```


Да тут же бага!..

```
int main() {
    return isqrt(-4);
}
```


Да тут же бага!..

```
int main() {
    return isqrt(-4);  // Compiler warning/error
}
```


Да тут же бага!..

```
int main() {
    return isqrt(-4);
}
```


Итак, проблема...

```
int isqrt(int) noexcept;
```


Итак, проблема...

```
int isqrt(int) noexcept;

int main() {
    return isqrt(-4);
}
```


Итак, проблема...

```
int isqrt(int) noexcept;

int main() {
    return isqrt(-4);  // ?????
}
```


Итак, проблема...

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

int main() {
    return isqrt(-4);  // ?????
}
```


Итак, проблема...

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

int main() {
    return isqrt(-4);  // ?????
}
```


Итак, проблема...

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

int main() {
    return isqrt(-4);  // Warning/Error
}
```


Итак, проблема...

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

int main() {
    const auto x = isqrt(4);
    return isqrt(x);  // ???
}
```


Итак, проблема...

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

int main() {
    const auto x = isqrt(4);
    return isqrt(x);  // ???
}
```


Итак, проблема...

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

int main() {
    const auto x = isqrt(4);
    return isqrt(x);  // No warning/Error
}
```


Итак, проблема...

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

int main() {
    const auto x = isqrt(4);
    return isqrt(x);  // No warning/Error
}
```


Как сделать себе больно с помощью assert?

Как сделать себе больно с помощью assert?

* Написать в нём важную логику

Как сделать себе больно с помощью assert?

* Написать в нём важную логику
* Side effects

Как сделать себе больно с помощью assert?

* Написать в нём важную логику
* Side effects
* ODR-violation

Как сделать себе больно с помощью assert?

* Написать в нём важную логику
* Side effects
* ODR-violation (для контрактов работает только вместе с предыдущим пунктом)

Кастомизация

Кастомизация

```
void handle_contract_violation(const std::contracts::contract_violation& violation) noexcept {
    std::print("Contract {} violation. Trace:\n{}", violation.comment(), std::stacktrace::current());
}
```


Открытые вопросы

Открытые вопросы

1. Виртуальные функции

Открытые вопросы

1. Виртуальные функции
2. Side effects & UB

Открытые вопросы

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

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

#embed

#embed

```
constexpr unsigned char whl[] = {
#embed "ches.glsl" \
  prefix(0xEF, 0xBB, 0xBF, ) /* префикс для вставки, если ресурс не пустой */ \
  suffix(,)                  /* суффикс для вставки, если ресурс не пустой */ \
  if_empty(0xBB, 0xBF, )     /* что вставить, если ресурс пустой */ \
  limit(1024*1024)           /* максимальный размер для вставки */ \
  0
};
```


#embed

```
constexpr unsigned char whl[] = {
#embed "ches.glsl" \
  prefix(0xEF, 0xBB, 0xBF, ) /* префикс для вставки, если ресурс не пустой */ \
  suffix(,)                  /* суффикс для вставки, если ресурс не пустой */ \
  if_empty(0xBB, 0xBF, )     /* что вставить, если ресурс пустой */ \
  limit(1024*1024)           /* максимальный размер для вставки */ \
  0
};
```


#embed

```
constexpr unsigned char whl[] = {
#embed "ches.glsl" \
  prefix(0xEF, 0xBB, 0xBF, ) /* префикс для вставки, если ресурс не пустой */ \
  suffix(,)                  /* суффикс для вставки, если ресурс не пустой */ \
  if_empty(0xBB, 0xBF, )     /* что вставить, если ресурс пустой */ \
  limit(1024*1024)           /* максимальный размер для вставки */ \
  0
};
```


#embed

```
constexpr unsigned char whl[] = {
#embed "ches.glsl" \
  prefix(0xEF, 0xBB, 0xBF, ) /* префикс для вставки, если ресурс не пустой */ \
  suffix(,)                  /* суффикс для вставки, если ресурс не пустой */ \
  if_empty(0xBB, 0xBF, )     /* что вставить, если ресурс пустой */ \
  limit(1024*1024)           /* максимальный размер для вставки */ \
  0
};
```


#embed

```
constexpr unsigned char whl[] = {
#embed "ches.glsl" \
  prefix(0xEF, 0xBB, 0xBF, ) /* префикс для вставки, если ресурс не пустой */ \
  suffix(,)                  /* суффикс для вставки, если ресурс не пустой */ \
  if_empty(0xBB, 0xBF, )     /* что вставить, если ресурс пустой */ \
  limit(1024*1024)           /* максимальный размер для вставки */ \
  0
};
```


#embed

```
constexpr unsigned char whl[] = {
#embed "ches.glsl" \
  prefix(0xEF, 0xBB, 0xBF, ) /* префикс для вставки, если ресурс не пустой */ \
  suffix(,)                  /* суффикс для вставки, если ресурс не пустой */ \
  if_empty(0xBB, 0xBF, )     /* что вставить, если ресурс пустой */ \
  limit(1024*1024)           /* максимальный размер для вставки */ \
  0
};
```


#embed

```
constexpr unsigned char whl[] = {
#embed "ches.glsl" \
  prefix(0xEF, 0xBB, 0xBF, ) /* префикс для вставки, если ресурс не пустой */ \
  suffix(,)                  /* суффикс для вставки, если ресурс не пустой */ \
  if_empty(0xBB, 0xBF, )     /* что вставить, если ресурс пустой */ \
  limit(1024*1024)           /* максимальный размер для вставки */ \
  0
};
```


C++26

C++26
Сверхспособности

Сверхспособности

Сверхспособности

```
import Protobuf.Reflection;
```


Сверхспособности

```
import Protobuf.Reflection;
constexpr auto _ = []() {




}();
```


Сверхспособности

```
import Protobuf.Reflection;
constexpr auto _ = []() {




}();
```


Сверхспособности

```
import Protobuf.Reflection;
constexpr auto _ = []() {




}();
```


Сверхспособности

```
import Protobuf.Reflection;
constexpr auto _ = []() {
    constexpr unsigned char proto[] = {
        #embed PROTO_FILE
    };

}();
```


Сверхспособности

```
import Protobuf.Reflection;
constexpr auto _ = []() {
    constexpr unsigned char proto[] = {
        #embed PROTO_FILE
    };
    protobuf::reflection_generate_client(proto, exporting = DO_EXPORT);
}();
```


Сверхспособности

```
// generate_protobuf_client.pp
import Protobuf.Reflection
constexpr auto _ = []() {
    constexpr unsigned char proto[] = {
        #embed PROTO_FILE
    };
    protobuf::reflection_generate_client(proto, exporting = DO_EXPORT);
}();
```


Сверхспособности

Сверхспособности

```
export module MyProject.Protobufs;
```


Сверхспособности

```
export module MyProject.Protobufs;
export import std;
#define DO_EXPORT true
```


Сверхспособности

```
export module MyProject.Protobufs;
export import std;
#define DO_EXPORT true

#define PROTO_FILE "schemas/sample/hello.pb"
```


Сверхспособности

```
export module MyProject.Protobufs;
export import std;
#define DO_EXPORT true

#define PROTO_FILE "schemas/sample/hello.pb"
#include "generate_protobuf_client_server.pp"
```


Сверхспособности

```
export module MyProject.Protobufs;
export import std;
#define DO_EXPORT true

#define PROTO_FILE "schemas/sample/hello.pb"
#include "generate_protobuf_client_server.pp"

#define PROTO_FILE "schemas/sample/hello2.pb"
#include "generate_protobuf_client_server.pp"

#define PROTO_FILE "schemas/sample/hi.pb"
#include "generate_protobuf_client.pp"
```


… 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
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

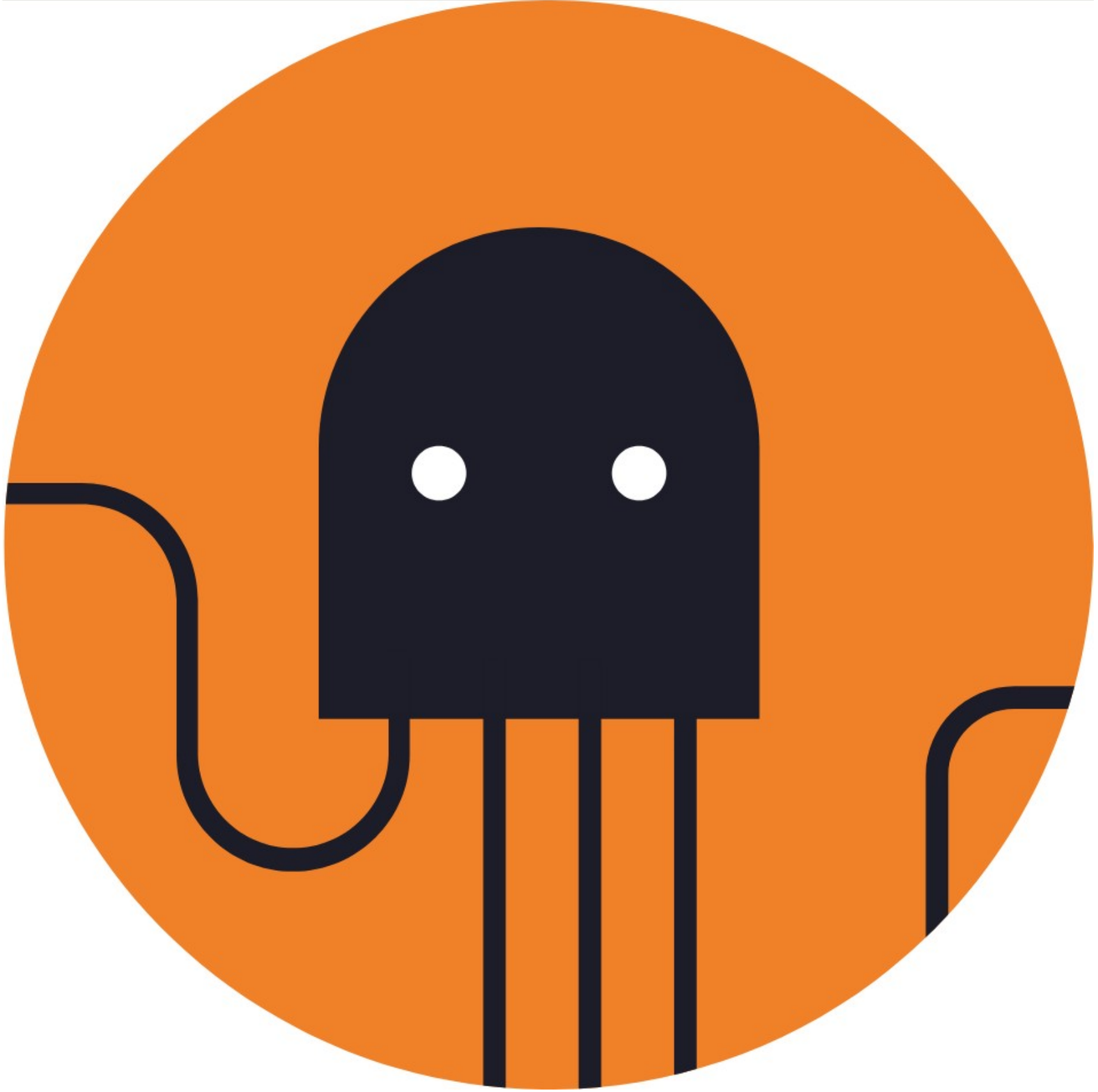
Эксперт-разработчик C++

antoshkka@gmail.com

antoshkka@yandex-team.ru

https://github.com/apolukhin

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



https://github.com/userver-framework