Яндекс Такси

C++23 и C++26

Планы

Полухин Антон

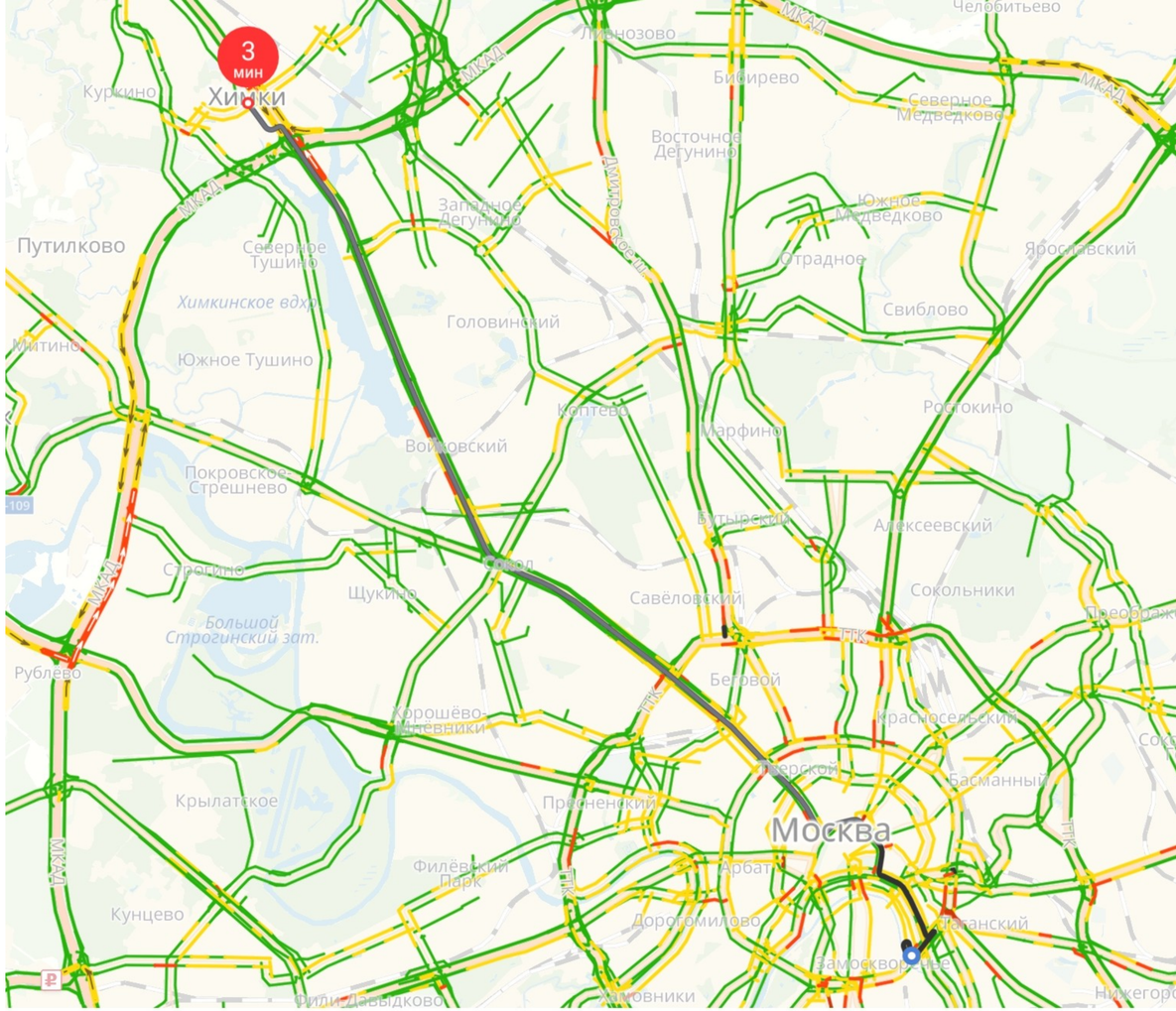
Antony Polukhin

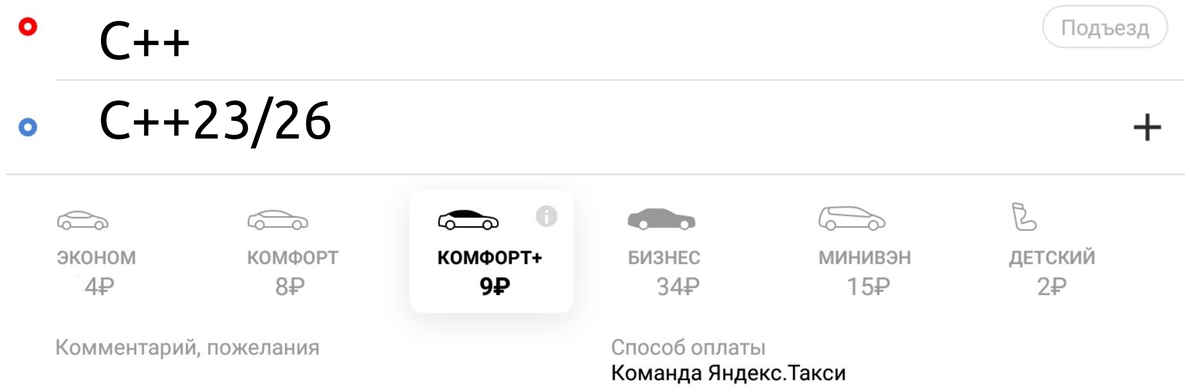
Яндекс Go

Содержание

Новинки, которые помогут вам когда что-то...

- … падает
- … вылетает
- … медленно собирается
- … медленно работает
- … не то делается





C++23 и C++26

Когда случается невозможное

Простая функция

```cpp
#include <mutex>
#include <boost/assert.hpp>

namespace impl {

template <class T>
void IncrementUnderMutex(std::unique_lock<std::mutex>& lock, T& value) {
    BOOST_ASSERT(lock);
    ++value;
}

}  // namespace impl
```

Простая функция

```cpp
#include <mutex>
#include <boost/assert.hpp>

namespace impl {

template <class T>
void IncrementUnderMutex(std::unique_lock<std::mutex>& lock, T& value) {
    BOOST_ASSERT(lock);
    ++value;
}

}  // namespace impl
```

... и она падает

```
Program returned: 139


output.s: /app/example.cpp:9
void impl::IncrementUnderMutex(std::unique_lock<std::mutex>&, T&) [with T = int]:
 Assertion `lock' failed.
```

Простая функция

```cpp
#include <mutex>
#include <boost/assert.hpp>

namespace impl {

template <class T>
void IncrementUnderMutex(std::unique_lock<std::mutex>& lock, T& value) {
    BOOST_ASSERT_MSG(lock, "Lock was not acquired");
    ++value;
}

}  // namespace impl
```

Простая функция

```cpp
#include <mutex>
#include <boost/assert.hpp>

namespace impl {

template <class T>
void IncrementUnderMutex(std::unique_lock<std::mutex>& lock, T& value) {
    BOOST_ASSERT_MSG(lock, "Lock was not acquired");
    ++value;
}

}  // namespace impl
```

… всё ещё падает

```
Program returned: 139


output.s: /app/example.cpp:9
void impl::IncrementUnderMutex(std::unique_lock<std::mutex>&, T&) [with T = int]:
Assertion `(lock)&&("Lock was not acquired")' failed.
```

Как понять, что к этому привело?

Правильная функция обработки assert

```
#define BOOST_ENABLE_ASSERT_HANDLER
```

Правильная функция обработки assert

```
#include <boost/stacktrace/stacktrace.hpp>
#include <iostream>

namespace boost {

void assertion_failed_msg(char const* expr, char const* msg,
                          char const* function, char const* file, long line) {
  std::cerr << boost::stacktrace::stacktrace();
  std::abort();
}

void assertion_failed(char const* expr, char const* function, char const* file,
                      long line) {
  boost::assertion_failed_msg(expr, nullptr, function, file, line);
}

}  // namespace boost
```

Правильная функция обработки assert

```cpp
#include <boost/stacktrace/stacktrace.hpp>
#include <iostream>

namespace boost {

void assertion_failed_msg(char const* expr, char const* msg,
                          char const* function, char const* file, long line) {
  std::cerr << boost::stacktrace::stacktrace();
  std::abort();
}

void assertion_failed(char const* expr, char const* function, char const* file,
                      long line) {
  boost::assertion_failed_msg(expr, nullptr, function, file, line);
}

}  // namespace boost
```

… и всё стало понятнее

```
Program returned: 139


output.s: /app/example.cpp:9
void impl::IncrementUnderMutex(std::unique_lock<std::mutex>&, T&) [with T = int]:
 Assertion `lock' failed:


 0# impl::IncrementUnderMutex at /home/ap/basic.cpp:600
 1# bar(std::string_view) at /home/ap/some_file.cpp:6
 2# main at /home/ap/main.cpp:17
```

Правильная функция обработки assert

```cpp
#include <boost/stacktrace/stacktrace.hpp>
#include <iostream>

namespace boost {

void assertion_failed_msg(char const* expr, char const* msg,
                          char const* function, char const* file, long line) {
  std::cerr << boost::stacktrace::stacktrace();
  std::abort();
}

void assertion_failed(char const* expr, char const* function, char const* file,
                      long line) {
  boost::assertion_failed_msg(expr, nullptr, function, file, line);
}

}  // namespace boost
```

Правильная функция обработки assert

```
#include <boost/stacktrace/stacktrace.hpp>
#include <iostream>

namespace boost {

void assertion_failed_msg(char const* expr, char const* msg,
                          char const* function, char const* file, long line) {
  std::cerr << std::stacktrace::current();
  std::abort();
}

void assertion_failed(char const* expr, char const* function, char const* file,
                      long line) {
  boost::assertion_failed_msg(expr, nullptr, function, file, line);
}

}  // namespace boost
```

Правильная функция обработки assert

```
#include <boost/stacktrace/stacktrace.hpp>
#include <iostream>

namespace boost {

void assertion_failed_msg(char const* expr, char const* msg,
                          char const* function, char const* file, long line) {
  std::cerr << std::stacktrace::current();
  std::abort();
}

void assertion_failed(char const* expr, char const* function, char const* file,
                      long line) {
  boost::assertion_failed_msg(expr, nullptr, function, file, line);
}

}  // namespace boost
```

Если решите писать свой assert_handler

```cpp
namespace userver::utils::impl {
[[noreturn]] void UASSERT_failed(std::string_view expr, const char* file,
                                 unsigned int line, const char* function,
                                 std::string_view msg) noexcept;
}  // namespace userver::utils::impl


// NOLINTNEXTLINE (cppcoreguidelines-macro-usage)
#define UASSERT_MSG(expr, msg)                                          \
  do {                                                                  \
    if (!(expr)) {                                                      \
      userver::utils::impl::UASSERT_failed(#expr, __FILE__,             \
                                           __LINE__, __func__, msg); \
    }                                                                   \
  } while (0)
```

Если решите писать свой assert_handler

```cpp
namespace userver::utils::impl {
[[noreturn]] void UASSERT_failed(std::string_view expr, const char* file,
                                 unsigned int line, const char* function,
                                 std::string_view msg) noexcept;
}  // namespace userver::utils::impl


// NOLINTNEXTLINE (cppcoreguidelines-macro-usage)
#define UASSERT_MSG(expr, msg)                                          \
  do {                                                                  \
    if (!(expr)) {                                                      \
      userver::utils::impl::UASSERT_failed(#expr, __FILE__,             \
                                           __LINE__, __func__, msg); \
    }                                                                   \
  } while (0)
```

Если решите писать свой assert_handler

```cpp
namespace userver::utils::impl {
[[noreturn]] void UASSERT_failed(std::string_view expr, const char* file,
                                 unsigned int line, const char* function,
                                 std::string_view msg) noexcept;
}  // namespace userver::utils::impl


// NOLINTNEXTLINE (cppcoreguidelines-macro-usage)
#define UASSERT_MSG(expr, msg)                                          \
  do {                                                                  \
    if (!(expr)) {                                                      \
      userver::utils::impl::UASSERT_failed(#expr, __FILE__,             \
                                           __LINE__, __func__, msg); \
    }                                                                   \
  } while (0)
```

Если решите писать свой assert_handler

```cpp
namespace userver::utils::impl {
[[noreturn]] void UASSERT_failed(std::string_view expr, const char* file,
                                 unsigned int line, const char* function,
                                 std::string_view msg) noexcept;
}  // namespace userver::utils::impl


// NOLINTNEXTLINE (cppcoreguidelines-macro-usage)
#define UASSERT_MSG(expr, msg)                                          \
  do {                                                                  \
    if (!(expr)) {                                                      \
      userver::utils::impl::UASSERT_failed(#expr, __FILE__,             \
                                           __LINE__, __func__, msg); \
    }                                                                   \
  } while (0)
```

Если решите писать свой assert_handler

```cpp
namespace userver::utils::impl {
[[noreturn]] void UASSERT_failed(std::string_view expr, const char* file,
                                 unsigned int line, const char* function,
                                 std::string_view msg) noexcept;
}  // namespace userver::utils::impl


// NOLINTNEXTLINE (cppcoreguidelines-macro-usage)
#define UASSERT_MSG(expr, msg)                                          \
  do {                                                                  \
    if (!(expr)) {                                                      \
      userver::utils::impl::UASSERT_failed(#expr, __FILE__,             \
                                           __LINE__, __func__, msg); \
    }                                                                   \
  } while (0)
```

Если решите писать свой assert_handler

```cpp
namespace userver::utils::impl {
[[noreturn]] void UASSERT_failed(std::string_view expr, const char* file,
                                 unsigned int line, const char* function,
                                 std::string_view msg) noexcept;
}  // namespace userver::utils::impl


// NOLINTNEXTLINE (cppcoreguidelines-macro-usage)
#define UASSERT_MSG(expr, msg)                                          \
  do {                                                                  \
    if (!(expr)) {                                                      \
      userver::utils::impl::UASSERT_failed(#expr, __FILE__,             \
                                           __LINE__, __func__, msg); \
    }                                                                   \
  } while (0)
```

Если решите писать свой assert_handler

```cpp
namespace userver::utils::impl {
[[noreturn]] void UASSERT_failed(std::string_view expr, const char* file,
                                 unsigned int line, const char* function,
                                 std::string_view msg) noexcept;
}  // namespace userver::utils::impl


// NOLINTNEXTLINE (cppcoreguidelines-macro-usage)
#define UASSERT_MSG(expr, msg)                                          \
  do {                                                                  \
    if (!(expr)) {                                                      \
      userver::utils::impl::UASSERT_failed(#expr, __FILE__,             \
                                           __LINE__, __func__, msg); \
    }                                                                   \
  } while (0)
```

Когда случается исключительное

```
...

terminating with uncaught exception of type std::out_of_range: vector
```

Если решите писать свой assert_handler

```
template <class _Tp, class _Allocator>
typename vector<_Tp, _Allocator>::reference
vector<_Tp, _Allocator>::at(size_type __n)
{
    if (__n >= size())
        this->__throw_out_of_range();
    return this->__begin_[__n];
}
```

Если решите писать свой assert_handler

```
template <class _Tp, class _Allocator>
typename vector<_Tp, _Allocator>::reference
vector<_Tp, _Allocator>::at(size_type __n)
{
    if (__n >= size())
        this->__throw_out_of_range();
    return this->__begin_[__n];
}
```

Не надо отчаиваться!

Не надо отчаиваться: P2370

Небольшой рецепт для счастья

```
int main() {
  try {
    std::this_thread::capture_stacktraces_at_throw(true);
    process();
  } catch (const std::exception& e) {
    std::cerr << e.what() << " at " << std::stacktrace::from_current_exception();
  }
}
```

Небольшой рецепт для счастья

```
int main() {
  try {
    std::this_thread::capture_stacktraces_at_throw(true);
    process();
  } catch (const std::exception& e) {
    std::cerr << e.what() << " at " << std::stacktrace::from_current_exception();
  }
}
```

Небольшой рецепт для счастья

```
int main() {
  try {
    std::this_thread::capture_stacktraces_at_throw(true);
    process();
  } catch (const std::exception& e) {
    std::cerr << e.what() << " at " << std::stacktrace::from_current_exception();
  }
}
```

Небольшой рецепт для счастья

```
int main() {
  try {
    std::this_thread::capture_stacktraces_at_throw(true);
    process();
  } catch (const std::exception& e) {
    std::cerr << e.what() << " at " << std::stacktrace::from_current_exception();
  }
}
```

Небольшой рецепт для счастья

```
int main() {
  try {
    std::this_thread::capture_stacktraces_at_throw(true);
    process();
  } catch (const std::exception& e) {
    std::cerr << e.what() << " at " << std::stacktrace::from_current_exception();
  }
}
```

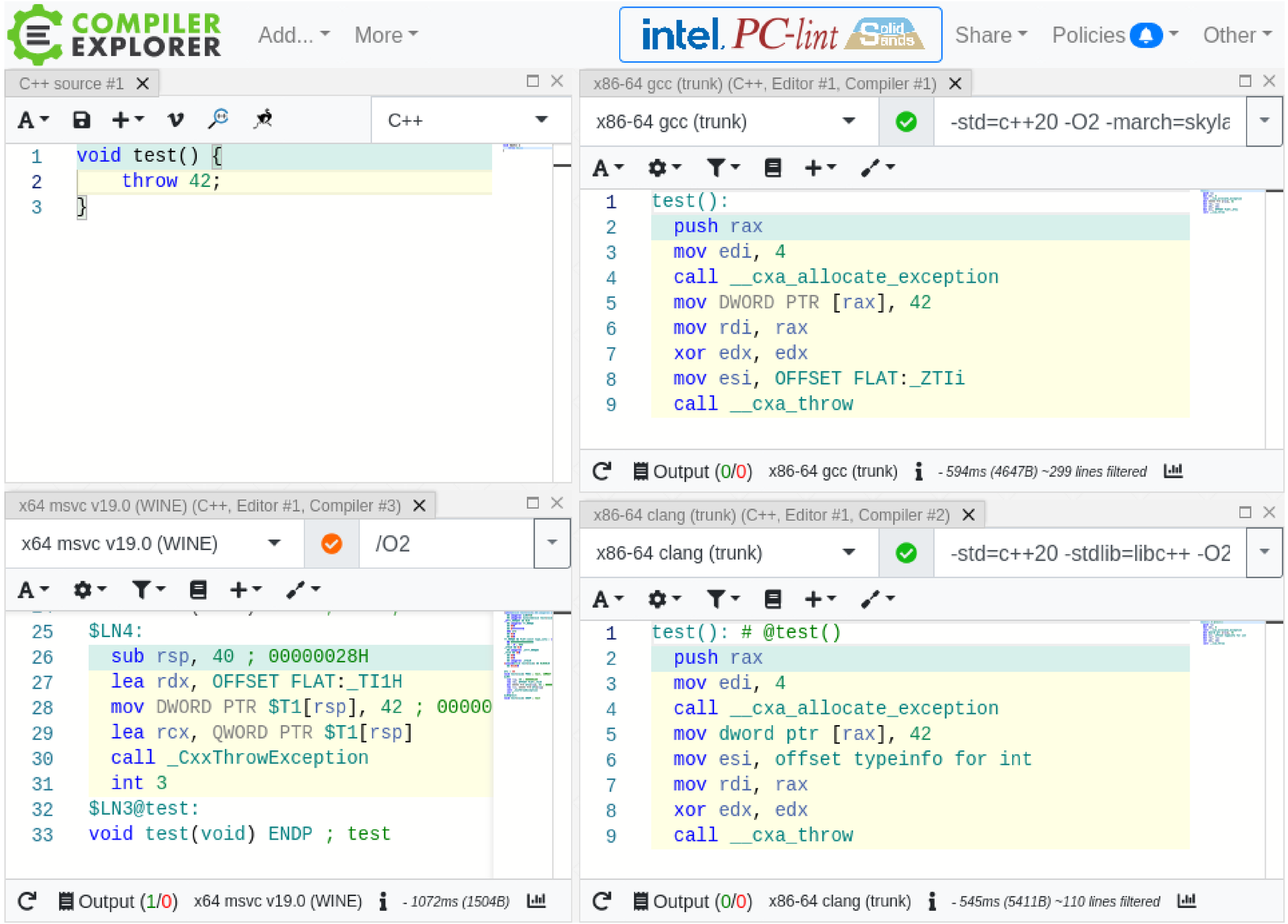
Небольшой рецепт для счастья

```
int main() {
  try {
    std::this_thread::capture_stacktraces_at_throw(true);
    process();
  } catch (const std::exception& e) {
    std::cerr << e.what() << " at " << std::stacktrace::from_current_exception();
  }
}
```

```
...

Exception trace:
 0# std::__throw_out_of_range(char const*) at libc++/src/c++11/functexcept.cc:82
 1# std::vector<int>::at(std::size_t) at libc++/include/vector:9000
 2# broken_function() at /home/ap/too_bad.cpp:8
 3# process() at /home/ap/sample.cpp:17
 4# main at /home/ap/sample.cpp:14
```

Работает с любыми исключениями


COMPILER EXPLORER
Add...
More
intel PC-lint
Share
Policies
Other
C++ source #1
C++
1 void test() {
2 throw 42;
3 }
x86-64 gcc (trunk) (C++, Editor #1, Compiler #1)
x86-64 gcc (trunk)
-std=c++20 -O2 -march=skyla
1 test():
2 push rax
3 mov edi, 4
4 call __cxa_allocate_exception
5 mov DWORD PTR [rax], 42
6 mov rdi, rax
7 xor edx, edx
8 mov esi, OFFSET FLAT:_ZTIi
9 call __cxa_throw
Output (0/0) x86-64 gcc (trunk) - 594ms (4647B) ~299 lines filtered
x64 msvc v19.0 (WINE) (C++, Editor #1, Compiler #3)
x64 msvc v19.0 (WINE)
/O2
25 $LN4:
26 sub rsp, 40 ; 00000028H
27 lea rdx, OFFSET FLAT:_TI1H
28 mov DWORD PTR $T1[rsp], 42 ; 00000
29 lea rcx, QWORD PTR $T1[rsp]
30 call _CxxThrowException
31 int 3
32 $LN3@test:
33 void test(void) ENDP ; test
Output (1/0) x64 msvc v19.0 (WINE) - 1072ms (1504B)
x86-64 clang (trunk) (C++, Editor #1, Compiler #2)
x86-64 clang (trunk)
-std=c++20 -stdlib=libc++ -O2
1 test(): # @test()
2 push rax
3 mov edi, 4
4 call __cxa_allocate_exception
5 mov dword ptr [rax], 42
6 mov esi, offset typeinfo for int
7 mov rdi, rax
8 xor edx, edx
9 call __cxa_throw
Output (0/0) x86-64 clang (trunk) - 545ms (5411B) ~110 lines filtered

Собирается медленно?

Модуль std: P2412r0

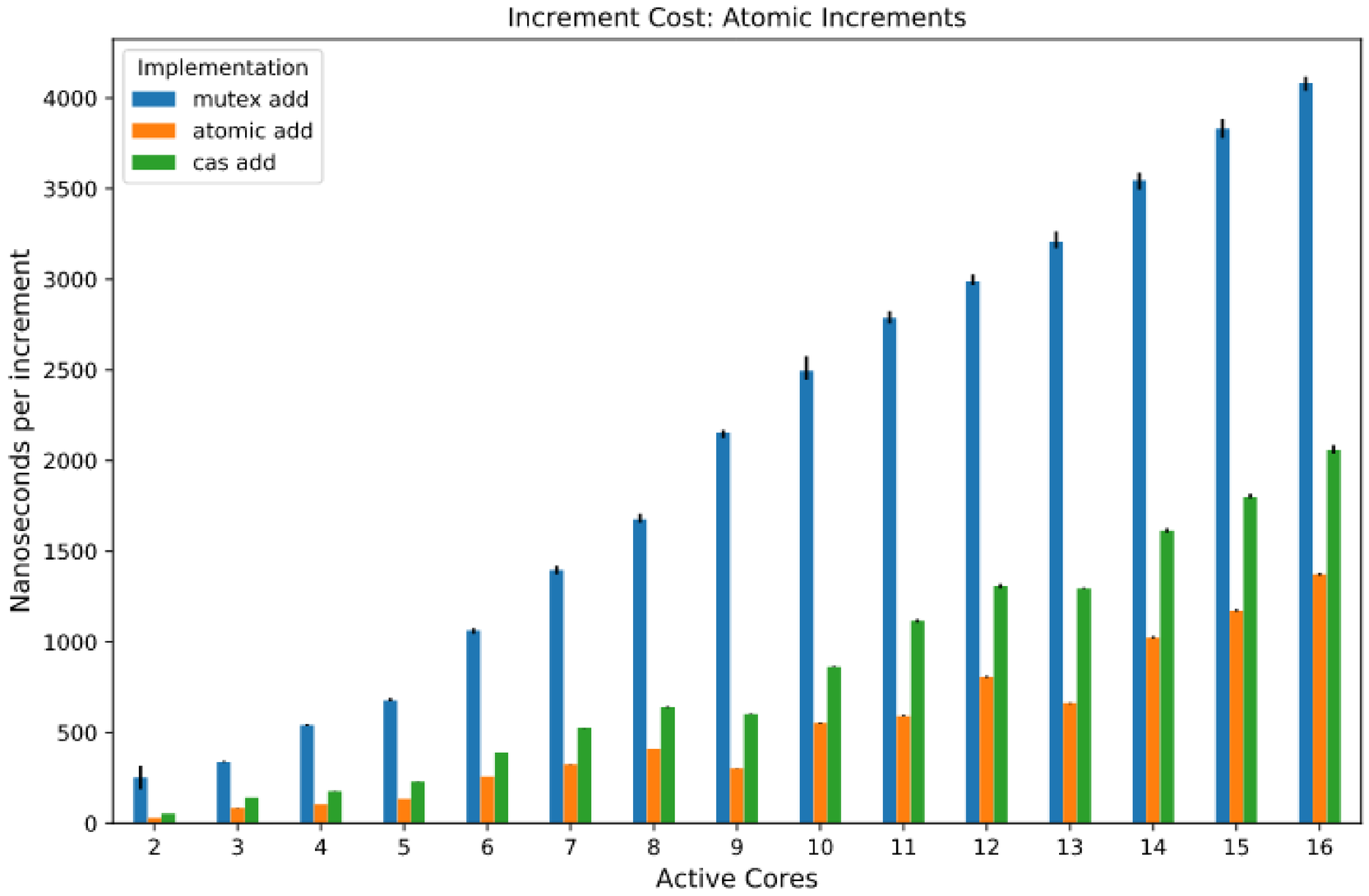
	#include needed headers	**Import** needed headers	**import std**	**#include** all headers	**Import** all headers
“Hello world” (<iostream>)	0.87s	0.32s	0.08s	3.43s	0.62s

Медленно работает?

Кеши

```
std::mutex my_cache_mutex;
std::shared_ptr<const Data> my_cache;
```

Особенности кешей



https://travisdowns.github.io/blog/2020/07/06/concurrency-costs.html

Кеши

Кеши

10kRPS

Кеши

10kRPS:

- mutex: 4us * 10 000 * 2 == 80ms

Как сделать получше?

Кеши

```
std::atomic<std::shared_ptr<const Data>> my_cache;
```

Кеши

10kRPS:

- mutex: 4us * 10 000 * 2 == 80ms

Кеши

10kRPS:

- mutex: 4us * 10 000 * 2 == 80ms
- atomic<shared_ptr>: 1us * 10 000 * 2 == 20ms

Как сделать идеально?

Получше

Получше

Основные тормоза — rmw атомарная операция

Получше

Основные тормоза — rmw атомарная операция

Надо просто её убрать с горячего пути!

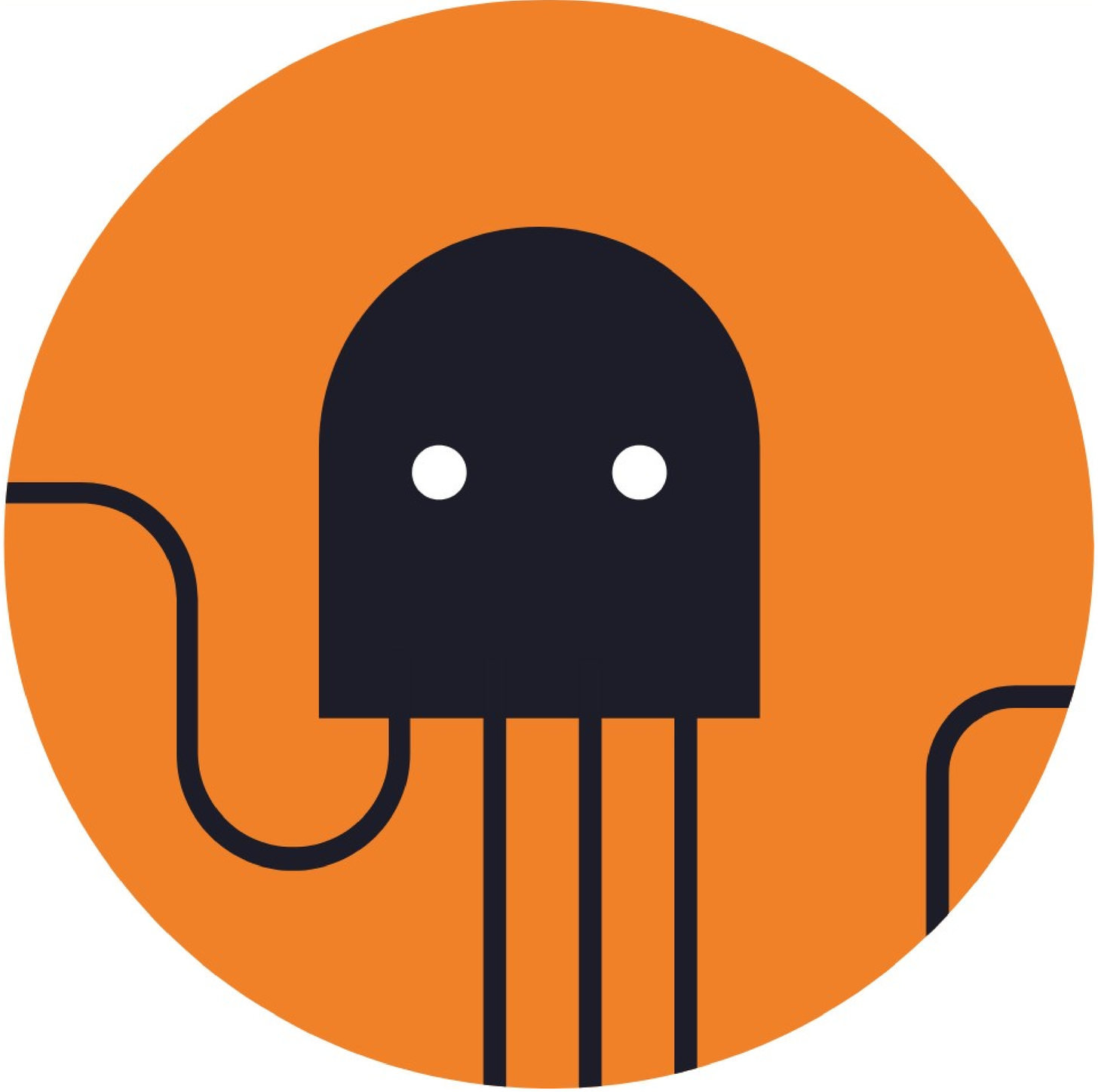
Получше

Основные тормоза — rmw атомарная операция

Надо просто её убрать с горячего пути!

А горячий путь — чтение данных

Hazard Pointer, Concurrency TS 2



Hazard Pointer

```
class MyCache {
 public:
  struct Data : std::hazard_pointer_obj_base<Data> {};

  ProtectedData Get() const {
    std::hazard_pointer h = std::make_hazard_pointer();
    const Data* d = h.protect(data_);
    return ProtectedData{std::move(h), d};
  }

  void Set(std::unique_ptr<Data> new_data) {
    const Data* old = data_.exchange(new_data.release());
    old->retire();
  }

 private:
  std::atomic<const Data*> data_;
};
```

Hazard Pointer

```
class MyCache {
 public:
  struct Data : std::hazard_pointer_obj_base<Data> {};

  ProtectedData Get() const {
    std::hazard_pointer h = std::make_hazard_pointer();
    const Data* d = h.protect(data_);
    return ProtectedData{std::move(h), d};
  }

  void Set(std::unique_ptr<Data> new_data) {
    const Data* old = data_.exchange(new_data.release());
    old->retire();
  }

 private:
  std::atomic<const Data*> data_;
};
```

Hazard Pointer

```
class MyCache {
 public:
  struct Data : std::hazard_pointer_obj_base<Data> {};

  ProtectedData Get() const {
    std::hazard_pointer h = std::make_hazard_pointer();
    const Data* d = h.protect(data_);
    return ProtectedData{std::move(h), d};
  }

  void Set(std::unique_ptr<Data> new_data) {
    const Data* old = data_.exchange(new_data.release());
    old->retire();
  }

 private:
  std::atomic<const Data*> data_;
};
```

Hazard Pointer

```
class MyCache {
 public:
  struct Data : std::hazard_pointer_obj_base<Data> {};

  ProtectedData Get() const {
    std::hazard_pointer h = std::make_hazard_pointer();
    const Data* d = h.protect(data_);
    return ProtectedData{std::move(h), d};
  }

  void Set(std::unique_ptr<Data> new_data) {
    const Data* old = data_.exchange(new_data.release());
    old->retire();
  }

 private:
  std::atomic<const Data*> data_;
};
```

Hazard Pointer

```
class MyCache {
 public:
  struct Data : std::hazard_pointer_obj_base<Data> {};

  ProtectedData Get() const {
    std::hazard_pointer h = std::make_hazard_pointer();
    const Data* d = h.protect(data_);
    return ProtectedData{std::move(h), d};
  }

  void Set(std::unique_ptr<Data> new_data) {
    const Data* old = data_.exchange(new_data.release());
    old->retire();
  }

 private:
  std::atomic<const Data*> data_;
};
```

Hazard Pointer

```
class MyCache {
 public:
  struct Data : std::hazard_pointer_obj_base<Data> {};

  ProtectedData Get() const {
    std::hazard_pointer h = std::make_hazard_pointer();
    const Data* d = h.protect(data_);
    return ProtectedData{std::move(h), d};
  }

  void Set(std::unique_ptr<Data> new_data) {
    const Data* old = data_.exchange(new_data.release());
    old->retire();
  }

 private:
  std::atomic<const Data*> data_;
};
```

Hazard Pointer

```
class MyCache {
 public:
  struct Data : std::hazard_pointer_obj_base<Data> {};

  ProtectedData Get() const {
    std::hazard_pointer h = std::make_hazard_pointer();
    const Data* d = h.protect(data_);
    return ProtectedData{std::move(h), d};
  }

  void Set(std::unique_ptr<Data> new_data) {
    const Data* old = data_.exchange(new_data.release());
    old->retire();
  }

 private:
  std::atomic<const Data*> data_;
};
```

Hazard Pointer

```
class MyCache {
 public:
  struct Data : std::hazard_pointer_obj_base<Data> {};

  ProtectedData Get() const {
    std::hazard_pointer h = std::make_hazard_pointer();
    const Data* d = h.protect(data_);
    return ProtectedData{std::move(h), d};
  }

  void Set(std::unique_ptr<Data> new_data) {
    const Data* old = data_.exchange(new_data.release());
    old->retire();
  }

 private:
  std::atomic<const Data*> data_;
};
```

Hazard Pointer

```
class MyCache {
 public:
  struct Data : std::hazard_pointer_obj_base<Data> {};

  ProtectedData Get() const {
    std::hazard_pointer h = std::make_hazard_pointer();
    const Data* d = h.protect(data_);
    return ProtectedData{std::move(h), d};
  }

  void Set(std::unique_ptr<Data> new_data) {
    const Data* old = data_.exchange(new_data.release());
    old->retire();
  }

 private:
  std::atomic<const Data*> data_;
};
```

Hazard Pointer

```
class MyCache {
 public:
  struct Data : std::hazard_pointer_obj_base<Data> {};

  ProtectedData Get() const {
    std::hazard_pointer h = std::make_hazard_pointer();
    const Data* d = h.protect(data_);
    return ProtectedData{std::move(h), d};
  }

  void Set(std::unique_ptr<Data> new_data) {
    const Data* old = data_.exchange(new_data.release());
    old->retire();
  }

 private:
  std::atomic<const Data*> data_;
};
```

Hazard Pointer

```
class MyCache {
 public:
  struct Data : std::hazard_pointer_obj_base<Data> {};

  ProtectedData Get() const {
    std::hazard_pointer h = std::make_hazard_pointer();
    const Data* d = h.protect(data_);
    return ProtectedData{std::move(h), d};
  }

  void Set(std::unique_ptr<Data> new_data) {
    const Data* old = data_.exchange(new_data.release());
    old->retire();
  }

 private:
  std::atomic<const Data*> data_;
};
```

Hazard Pointer

```
class MyCache {
 public:
  struct Data : std::hazard_pointer_obj_base<Data> {};

  ProtectedData Get() const {
    std::hazard_pointer h = std::make_hazard_pointer();
    const Data* d = h.protect(data_);
    return ProtectedData{std::move(h), d};
  }

  void Set(std::unique_ptr<Data> new_data) {
    const Data* old = data_.exchange(new_data.release());
    old->retire();
  }

 private:
  std::atomic<const Data*> data_;
};
```

Hazard Pointer

```
class MyCache {
 public:
  struct Data : std::hazard_pointer_obj_base<Data> {};

  ProtectedData Get() const {
    std::hazard_pointer h = std::make_hazard_pointer();
    const Data* d = h.protect(data_);
    return ProtectedData{std::move(h), d};
  }

  void Set(std::unique_ptr<Data> new_data) {
    const Data* old = data_.exchange(new_data.release());
    old->retire();
  }

 private:
  std::atomic<const Data*> data_;
};
```

Hazard Pointer

```
class MyCache {
 public:
  struct Data : std::hazard_pointer_obj_base<Data> {};

  ProtectedData Get() const {
    std::hazard_pointer h = std::make_hazard_pointer();
    const Data* d = h.protect(data_);
    return ProtectedData{std::move(h), d};
  }

  void Set(std::unique_ptr<Data> new_data) {
    const Data* old = data_.exchange(new_data.release());
    old->retire();
  }

 private:
  std::atomic<const Data*> data_;
};
```

Кеши

10kRPS:

- mutex: 4us * 10 000 * 2 == 80ms
- atomic<shared_ptr>: 1us * 10 000 * 2 == 20ms

Кеши

10kRPS:

- mutex: 4us * 10 000 * 2 == 80ms
- atomic<shared_ptr>: 1us * 10 000 * 2 == 20ms
- rcu: 20ns * 10 000 * 2 == <1ms

Hazard Pointer

Hazard Pointer

Плюсы:

Hazard Pointer

Плюсы:

- Очень быстрое чтение

Hazard Pointer

Плюсы:

- Очень быстрое чтение

Минусы:

Hazard Pointer

Плюсы:

- Очень быстрое чтение

Минусы:

- Дорогая запись или обновление данных

Hazard Pointer

Плюсы:

- Очень быстрое чтение

Минусы:

- Дорогая запись или обновление данных
- Приличные затраты на внутренние нужды

Hazard Pointer

Плюсы:

- Очень быстрое чтение

Минусы:

- Дорогая запись или обновление данных
- Приличные затраты на внутренние нужды
- В памяти может находиться сразу несколько разных поколений данных

Hazard Pointer

Плюсы:

- Очень быстрое чтение

Минусы:

- Дорогая запись или обновление данных
- Приличные затраты на внутренние нужды
- В памяти может находиться сразу несколько разных поколений данных

Итог: хорошее решение для задач с количеством чтения данных ≫ записи

std::format и крутой трюк

Ошибочка

```
std::format("At {} expected type {} but found ", path, expected,
actual);
```

Ошибочка

```
template <typename... Args>
struct format_string_impl {
  std::string_view str;

  template <class S>
  consteval format_string_impl(const S& s) : str(s) {
    if (sizeof...(Args) != (str[0] - '0')) {
      throw 42;
    }
  }
};
```

Ошибочка

```
template <typename... Args>
struct format_string_impl {
  std::string_view str;

  template <class S>
  consteval format_string_impl(const S& s) : str(s) {
    if (sizeof...(Args) != (str[0] - '0')) {
      throw 42;
    }
  }
};
```

Ошибочка

```
template <typename... Args>
struct format_string_impl {
  std::string_view str;

  template <class S>
  consteval format_string_impl(const S& s) : str(s) {
    if (sizeof...(Args) != (str[0] - '0')) {
      throw 42;
    }
  }
};
```

Ошибочка

```
template <typename... Args>
struct format_string_impl {
  std::string_view str;

  template <class S>
  consteval format_string_impl(const S& s) : str(s) {
    if (sizeof...(Args) != (str[0] - '0')) {
      throw 42;
    }
  }
};
```

Ошибочка

```
template <typename... Args>
using format_string = format_string_impl<std::add_const_t<Args>...>;

template <typename... Args>
int format(format_string<Args...> str, Args&&... args);

auto v1 = format("0 Hello");
auto v2 = format("3 World", 1, 2);
```

Ошибочка

```
template <typename... Args>
using format_string = format_string_impl<std::add_const_t<Args>...>;

template <typename... Args>
int format(format_string<Args...> str, Args&&... args);

auto v1 = format("0 Hello");
auto v2 = format("3 World", 1, 2);
```

Ошибочка

```
template <typename... Args>
using format_string = format_string_impl<std::add_const_t<Args>...>;

template <typename... Args>
int format(format_string<Args...> str, Args&&... args);

auto v1 = format("0 Hello");
auto v2 = format("3 World", 1, 2);
```

Ошибочка

```
template <typename... Args>
using format_string = format_string_impl<std::add_const_t<Args>...>;

template <typename... Args>
int format(format_string<Args...> str, Args&&... args);

auto v1 = format("0 Hello");
auto v2 = format("3 World", 1, 2);
```

Ошибочка

```
template <typename... Args>
using format_string = format_string_impl<std::add_const_t<Args>...>;

template <typename... Args>
int format(format_string<Args...> str, Args&&... args);

auto v1 = format("0 Hello");
auto v2 = format("3 World", 1, 2);
```

Ошибочка

```
template <typename... Args>
using format_string = format_string_impl<std::add_const_t<Args>...>;

template <typename... Args>
int format(format_string<Args...> str, Args&&... args);

auto v1 = format("0 Hello");
auto v2 = format("3 World", 1, 2);
```

Ошибочка

```
error: call to consteval function 'format_string_impl<const int, const
int>::format_string_impl<char [8]>' is not a constant expression
auto v2 = format("3 World", 1, 2);
                 ^
<source>:11:7: note: subexpression not valid in a constant expression
      throw 42;
      ^
```

Итог

Итого

Итого

- std::stacktrace поможет вам в assert (C++23)

Итого

- std::stacktrace поможет вам в assert (C++23)
- std::stacktrace в исключениях (C++26)

Итого

- std::stacktrace поможет вам в assert (C++23)
- std::stacktrace в исключениях (C++26)
- Модули жгут, даже если импортировать всё (C++23)

Итого

- std::stacktrace поможет вам в assert (C++23)
- std::stacktrace в исключениях (C++26)
- Модули жгут, даже если импортировать всё (C++23)
- Кеши + Hazard Pointer = ❤

Итого

- std::stacktrace поможет вам в assert (C++23)
- std::stacktrace в исключениях (C++26)
- Модули жгут, даже если импортировать всё (C++23)
- Кеши + Hazard Pointer = ❤
- В C++20 можно писать compile time проверки DSL и собственно DSL похожие на обычные строки

Спасибо

Полухин Антон

Эксперт-разработчик C++

antoshkka@gmail.com

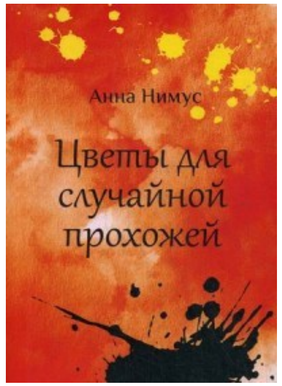
antoshkka@yandex-team.ru

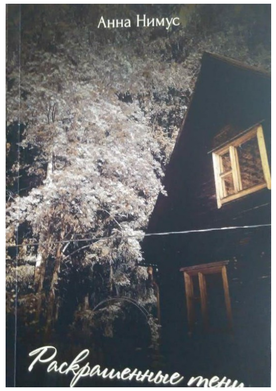
https://github.com/apolukhin

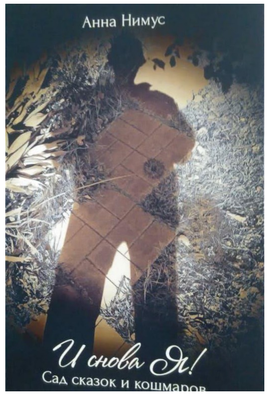
https://stdcpp.ru/

РГ21 C++ РОССИЯ

Спасибо


Анна Нимус
Цветы для случайной прохожей


Анна Нимус


Анна Нимус
И снова Я!
Сад сказок и кошмаров