Новости ISO C++ WG21

и планы

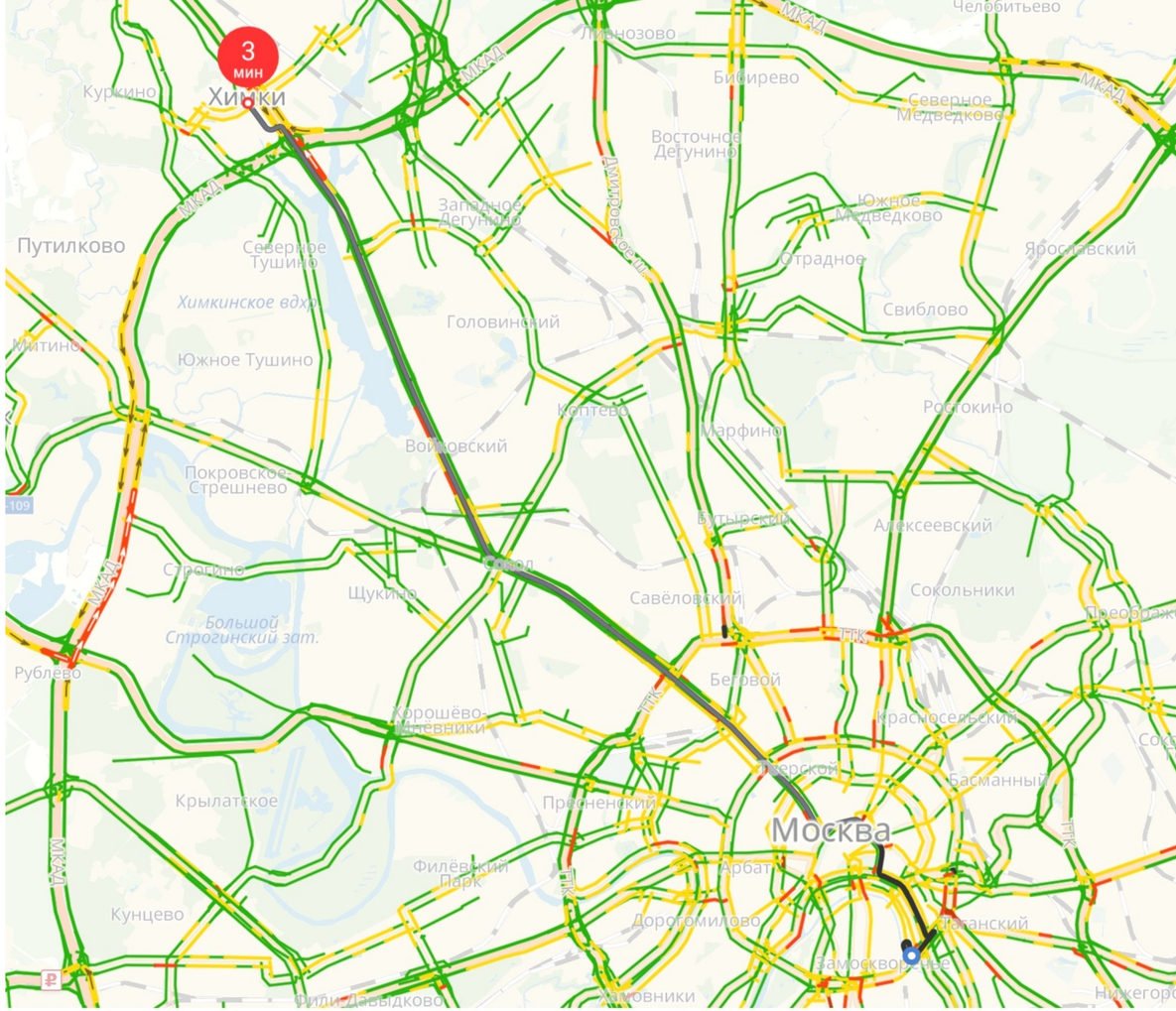
Полухин Антон

Antony Polukhin

Яндекс Go

Содержание

- std::generator
- std::stacktrace
- [[assume(x>0)]]
- import std;
- std::expected
- std::format



std::generator P2502

Февральская встреча WG21 C++

std::generator

```
std::generator<std::string> greeter() {
    std::size_t i = 0;
    while (true) {
        co_yield "hello " + std::to_string(++i);
    }
}
```

std::generator

```
std::generator<std::string> greeter() {
    std::size_t i = 0;
    while (true) {
        co_yield "hello " + std::to_string(++i);
    }
}
```

std::generator

```
std::generator<std::string> greeter() {
    std::size_t i = 0;
    while (true) {
        co_yield "hello " + std::to_string(++i);
    }
}
```

std::generator

```
std::generator<std::string> greeter() {
    std::size_t i = 0;
    while (true) {
        co_yield "hello " + std::to_string(++i);
    }
}
```

std::generator

```
std::generator<std::string> greeter() {
    std::size_t i = 0;
    while (true) {
        co_yield "hello " + std::to_string(++i);
    }
}
```

std::generator

```
std::generator<std::string> greeter() {
    std::size_t i = 0;
    while (true) {
        co_yield "hello " + std::to_string(++i);
    }
}

auto f = greeter();
```

std::generator

```
std::generator<std::string> greeter() {
    std::size_t i = 0;
    while (true) {
        co_yield "hello " + std::to_string(++i);
    }
}

auto f = greeter();
for (auto str : f) {
    std::cout << str;
}
```

std::generator

```
std::generator<std::string> greeter() {
    std::size_t i = 0;
    while (true) {
        co_yield "hello " + std::to_string(++i);
    }
}

auto f = greeter();
for (auto str : f) { // 0 copy
    std::cout << str;
}
```

std::generator

```
std::generator<std::string> greeter() {
    std::size_t i = 0;
    while (true) {
        co_yield "hello " + std::to_string(++i);
    }
}

auto f = greeter();
auto str = f(); // 0 copy
```

std::generator - особенности

std::generator

```
std::generator<std::string> greeter() {
    std::size_t i = 0;
    while (true) {
        co_yield "hello " + std::to_string(++i);
    }
}
```

std::generator

```
std::generator<std::string> greeter() {
    std::size_t i = 0;
    while (true) {
        co_yield "hello " + std::to_string(++i);
    }
}
```

std::generator

```
std::generator<std::string> greeter() {
    std::size_t i = 0;
    while (true) {
        co_yield "hello " + std::to_string(++i);
    }
}
```

std::generator

```
std::generator<const std::string&> greeter() {
    std::size_t i = 0;
    while (true) {
        co_yield "hello " + std::to_string(++i);
    }
}
```

std::generator

```
std::generator<const std::string&> greeter() {
    std::size_t i = 0;
    while (true) {
        co_yield "hello " + std::to_string(++i); // Dangling???
    }
}
```

И ничего такого не случилось

std::generator

```
std::generator<const std::string&> greeter() {
    std::size_t i = 0;
    while (true) {
        co_yield "hello " + std::to_string(++i); // Dangling???
    }
}
```

std::generator

```
std::generator<const std::string&> greeter() {
    std::size_t i = 0;
    while (true) {
        co_yield "hello " + std::to_string(++i);
    }
}
```

std::generator

```
std::generator<const std::string&> greeter() {
    std::size_t i = 0;
    while (true) {
        co_await promise::yield_value("hello " + std::to_string(++i));
    }
}
```

std::generator

```
std::generator<const std::string&> greeter() {
    std::size_t i = 0;
    while (true) {
        {
        auto&& __tmp = "hello " + std::to_string(++i);
        co_await promise::yield_value(std::forward<???>(__tmp));
        }
    }
}
```

std::generator

```
std::generator<const std::string&> greeter() {
    std::size_t i = 0;
    while (true) {
        {
        auto&& __tmp = "hello " + std::to_string(++i);
        co_await promise::yield_value(std::forward<???>(__tmp));
        }
    }
}

suspend_always promise::yield_value(const std::string& x) noexcept;
//    Effects: Equivalent to: value_ = addressof(x).
//    Returns: {}.
```

std::generator

```
std::generator<std::string> greeter() {
    std::size_t i = 0;
    while (true) {
        co_yield "hello " + std::to_string(++i);
    }
}
```

std::generator

```
std::generator<std::string> greeter() {
    std::size_t i = 0;
    while (true) {
        co_yield "hello " + std::to_string(++i); // coro.value_ = &__tmp
    }
}
```

std::generator

```
std::generator<std::string> greeter() {
    std::size_t i = 0;
    while (true) {
        co_yield "hello " + std::to_string(++i); // coro.value_ = &__tmp
    }
}

auto f = greeter();
for (auto str : f) {
    std::cout << str;
}
```

std::generator

```
std::generator<std::string> greeter() {
    std::size_t i = 0;
    while (true) {
        co_yield "hello " + std::to_string(++i); // coro.value_ = &__tmp
    }
}

auto f = greeter();
for (auto str : f) { // static_cast<std::string&&>(*coro.value_)
    std::cout << str;
}
```

std::generator

```
std::generator<std::string&> greeter() {
    std::size_t i = 0;
    while (true) {
        co_yield "hello " + std::to_string(++i); // coro.value_ = &__tmp
    }
}

auto f = greeter();
for (auto str : f) { // static_cast<std::string&>(*coro.value_)
    std::cout << str;
}
```

std::stacktrace

assert(lock)

```
Program returned: 139


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

Как понять, что к этому привело?

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
```

… 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
```

Когда случается исключительное

```
...

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

Внутри стандартной библиотеки

```
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];
}
```

Внутри стандартной библиотеки

```
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];
}
```

Февральская встреча WG21 C++

Февральская встреча WG21 C++

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();
  }
}
```

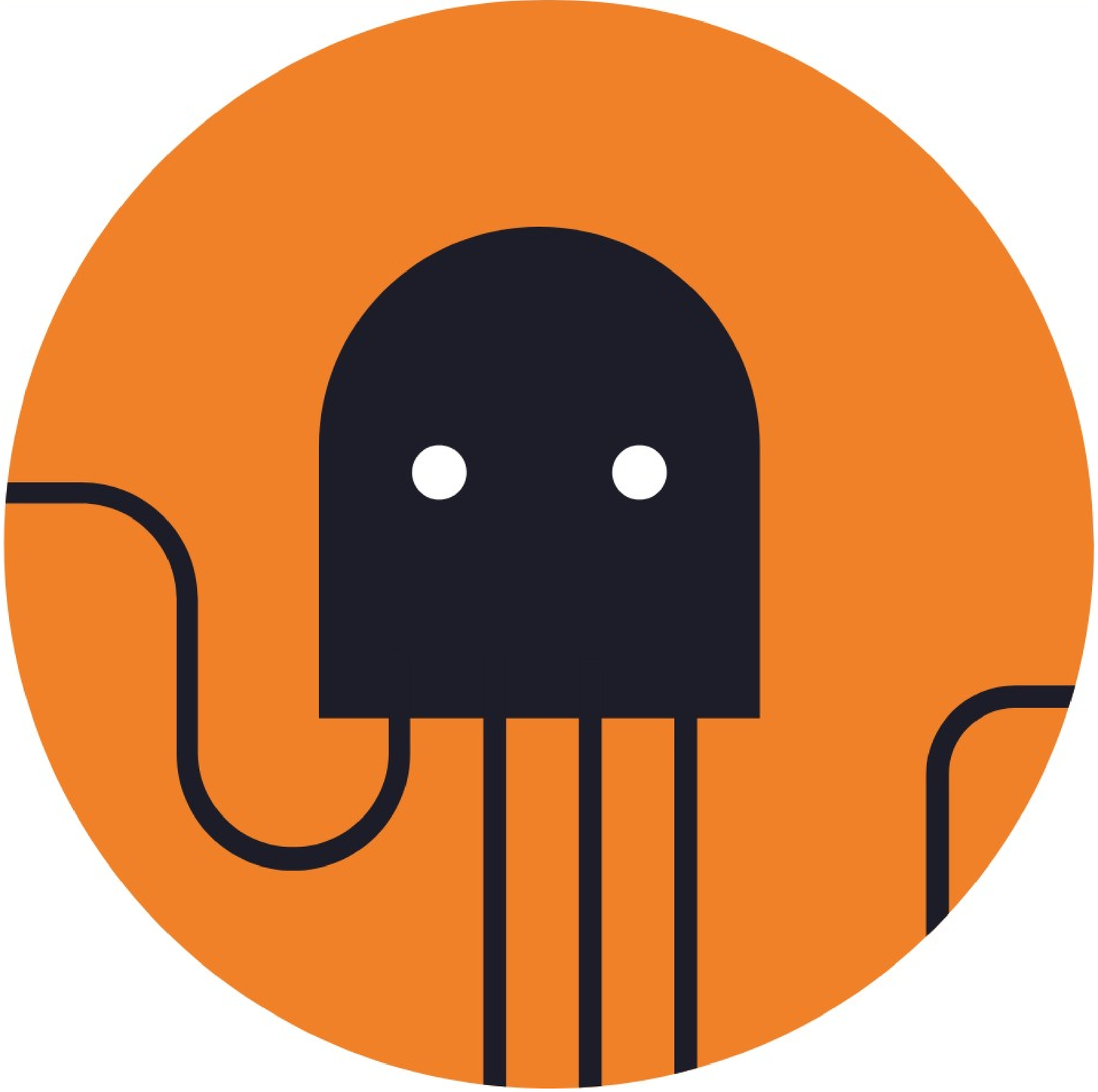
Небольшой рецепт для счастья

```
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();
  }
}
```

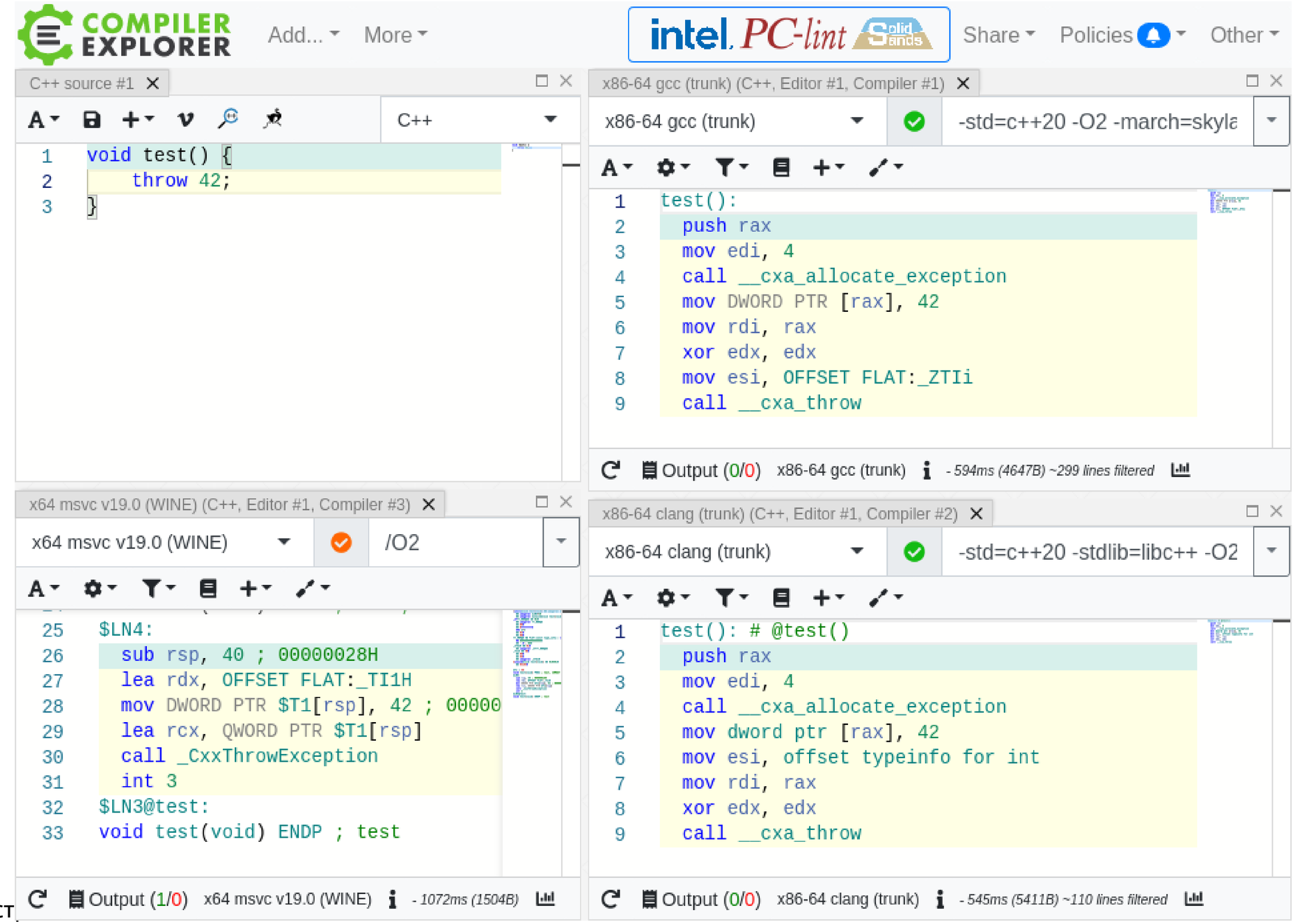
Небольшой рецепт для счастья

```
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

[[assume(x != 0)]] P1774

Небольшой рецепт для скорости

```
void clamp_range(float* data, std::size_t size) {
    [[ assume(size > 0) ]]
    [[ assume(size % 32 == 0) ]]

    for (size_t i = 0; i < size; ++i) {
        [[ assume(std::isfinite(data[i])) ]];
        data[i] = std::clamp(data[i], -1.0f, 1.0f);
    }
}
```

Небольшой рецепт для скорости

```
void clamp_range(float* data, std::size_t size) {
    [[ assume(size > 0) ]]
    [[ assume(size % 32 == 0) ]]

    for (size_t i = 0; i < size; ++i) {
        [[ assume(std::isfinite(data[i])) ]];
        data[i] = std::clamp(data[i], -1.0f, 1.0f);
    }
}
```

Небольшой рецепт для скорости

```
void clamp_range(float* data, std::size_t size) {
    [[ assume(size > 0) ]]
    [[ assume(size % 32 == 0) ]]

    for (size_t i = 0; i < size; ++i) {
        [[ assume(std::isfinite(data[i])) ]];
        data[i] = std::clamp(data[i], -1.0f, 1.0f);
    }
}
```

Небольшой рецепт для скорости

```
void clamp_range(float* data, std::size_t size) {
    [[ assume(size > 0) ]]
    [[ assume(size % 32 == 0) ]]

    for (size_t i = 0; i < size; ++i) {
        [[ assume(std::isfinite(data[i])) ]];
        data[i] = std::clamp(data[i], -1.0f, 1.0f);
    }
}
```

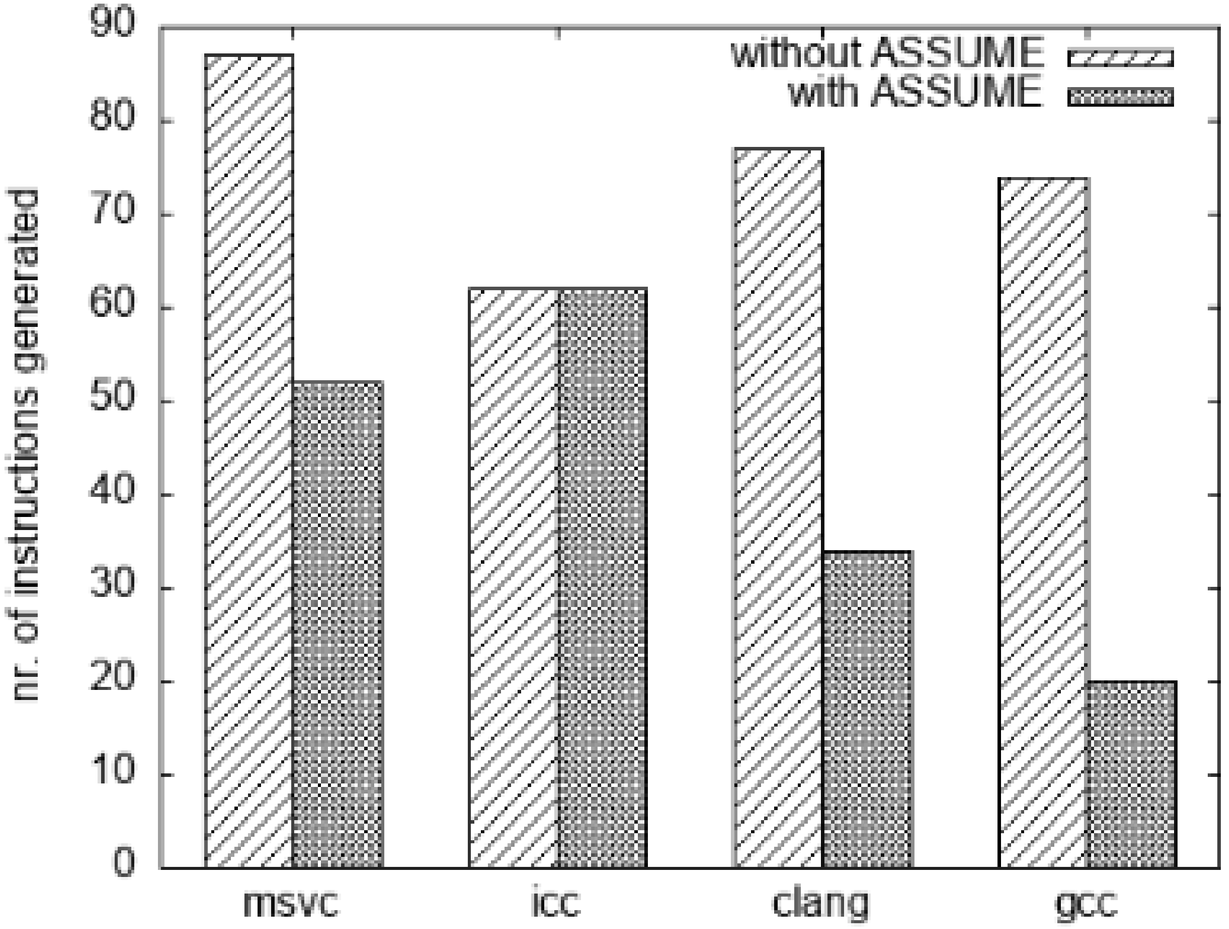
Небольшой рецепт для скорости

```
void clamp_range(float* data, std::size_t size) {
    [[ assume(size > 0) ]]
    [[ assume(size % 32 == 0) ]]

    for (size_t i = 0; i < size; ++i) {
        [[ assume(std::isfinite(data[i])) ]];
        data[i] = std::clamp(data[i], -1.0f, 1.0f);
    }
}
```

Небольшой рецепт для скорости

```
void clamp_range(float* data, std::size_t size) {
    [[ assume(size > 0) ]]
    [[ assume(size % 32 == 0) ]]

    for (size_t i = 0; i < size; ++i) {
        [[ assume(std::isfinite(data[i])) ]];
        data[i] = std::clamp(data[i], -1.0f, 1.0f);
    }
}
```

Небольшой рецепт для скорости



Aliasing

```
[[ assume(data_ != &capacity_) ]]
[[ assume(data_ != &size_) ]]
```

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

Не std::expected

std::expected ?

```
// throwing
std::filesystem::path read_symlink(const std::filesystem::path& p);

// почти non-throwing
std::filesystem::path read_symlink(const std::filesystem::path& p,
                                   std::error_code& ec);
```

std::expected ?

```
std::expected<std::filesystem::path, std::filesystem_error>
    read_symlink(const std::filesystem::path& p);
```

std::expected ?

```
std::expected<std::filesystem::path, std::filesystem_error>
    read_symlink(const std::filesystem::path& p);
```

std::expected ?

```
std::expected<std::filesystem::path, std::filesystem_error>
    read_symlink(const std::filesystem::path& p);
```

std::expected ?

```
std::expected<std::filesystem::path, std::filesystem_error>
    read_symlink(const std::filesystem::path& p);
```

std::expected ?

```
std::expected<std::filesystem::path, std::filesystem_error>
    read_symlink(const std::filesystem::path& p);

// локальная обработка ошибок
auto res = fs::read_symlink("/data");
if (res) do_something(*res);
```

std::expected ?

```
std::expected<std::filesystem::path, std::filesystem_error>
    read_symlink(const std::filesystem::path& p);

// локальная обработка ошибок
auto res = fs::read_symlink("/data");
if (res) do_something(*res);

// централизованная обработка ошибок
try {
    do_something(fs::read_symlink("/data").value());
    do_something(network::receive(socket).value());
    do_something(http::handle("/ping").value());
    // много кода
} catch (const std::exception& e) {
   std::cout << e.what();
}
```

std::expected ?

```
std::expected<std::filesystem::path, std::filesystem_error>
    read_symlink(const std::filesystem::path& p);

// локальная обработка ошибок
auto res = fs::read_symlink("/data");
if (res) do_something(*res);

// централизованная обработка ошибок
try {
    do_something(fs::read_symlink("/data").value());
    do_something(network::receive(socket).value());
    do_something(http::handle("/ping").value());
    // много кода
} catch (const std::exception& e) {
   std::cout << e.what();
}
```

std::expected ?

```
std::expected<std::filesystem::path, std::filesystem_error>
    read_symlink(const std::filesystem::path& p);

// локальная обработка ошибок
auto res = fs::read_symlink("/data");
if (res) do_something(*res);

// централизованная обработка ошибок
try {
    do_something(fs::read_symlink("/data").value());
    do_something(network::receive(socket).value());
    do_something(http::handle("/ping").value());
    // много кода
} catch (const std::exception& e) {
   std::cout << e.what();
}
```

std::expected ?

```
std::expected<std::filesystem::path, std::filesystem_error>
    read_symlink(const std::filesystem::path& p);

// локальная обработка ошибок
auto res = fs::read_symlink("/data");
if (res) do_something(*res);

// централизованная обработка ошибок
try {
    do_something(fs::read_symlink("/data").value());
    do_something(network::receive(socket).value());
    do_something(http::handle("/ping").value());
    // много кода
} catch (const std::exception& e) {
   std::cout << e.what();
}
```

Unexpected std::expected

Февральская встреча WG21 C++

std::expected ?

```
} catch (const std::bad_excepected_access<std::errc>& e) {
   std::cout << "Error at filesystem or networking: " <<
std::make_error_code(e.error()).message();
} catch (const std::exception& e) {
   std::cout << e.what();
}
```

std::expected ?

```
} catch (const std::bad_excepected_access<std::errc>& e) {
   std::cout << "Error at filesystem or networking: " <<
std::make_error_code(e.error()).message();
} catch (const std::exception& e) {
   std::cout << e.what();
}
```

std::expected ?

```
} catch (const std::bad_excepected_access<std::errc>& e) {
   std::cout << "Error at filesystem or networking: " <<
std::make_error_code(e.error()).message();
} catch (const std::exception& e) {
   std::cout << e.what();
}
```

std::expected ?

```
} catch (const std::bad_excepected_access<std::errc>& e) {
   std::cout << "Error at filesystem or networking: " <<
std::make_error_code(e.error()).message();
} catch (const std::bad_excepected_access<lib1>& e) {
   std::cout << ???;
} catch (const std::bad_excepected_access<lib2>& e) {
   std::cout << ???;
} catch (const std::bad_excepected_access<lib3>& e) {
   std::cout << ???;
} catch (const std::exception& e) {
   std::cout << e.what();
}
```

std::expected ?

```
} catch (const std::bad_excepected_access<std::errc>& e) {
   std::cout << "Error at filesystem or networking: " <<
std::make_error_code(e.error()).message();
} catch (const std::bad_excepected_access<lib1>& e) {
   std::cout << ???;
} catch (const std::bad_excepected_access<lib2>& e) {
   std::cout << ???;
} catch (const std::bad_excepected_access<lib3>& e) {
   std::cout << ???;
} catch (const std::exception& e) {
   std::cout << e.what();
}
```

std::expected ?

```
} catch (const std::bad_excepected_access<std::errc>& e) {
   std::cout << "Error at filesystem or networking: " <<
std::make_error_code(e.error()).message();
} catch (const std::bad_excepected_access<lib1>& e) {
   std::cout << ???;
} catch (const std::bad_excepected_access<lib2>& e) {
   std::cout << ???;
} catch (const std::bad_excepected_access<lib3>& e) {
   std::cout << ???;
} catch (const std::exception& e) {
   std::cout << e.what();
}
```

std::expected ?

```
} catch (const std::bad_excepected_access<std::errc>& e) {
   std::cout << "Error at filesystem or networking: " <<
std::make_error_code(e.error()).message();
} catch (const std::bad_excepected_access<lib1>& e) {
   std::cout << ???;
} catch (const std::bad_excepected_access<lib2>& e) {
   std::cout << ???;
} catch (const std::bad_excepected_access<lib3>& e) {
   std::cout << ???;
} catch (const std::exception& e) {
   std::cout << e.what();
}
```

std::expected ?

```
} catch (const std::bad_excepected_access<std::errc>& e) {
   std::cout << "Error at filesystem or networking: " <<
std::make_error_code(e.error()).message();
} catch (const std::bad_excepected_access<lib1>& e) {
   std::cout << ???;
} catch (const std::bad_excepected_access<lib2>& e) {
   std::cout << ???;
} catch (const std::bad_excepected_access<lib3>& e) {
   std::cout << ???;
} catch (const std::exception& e) {
   std::cout << e.what();
}
```

std::expected ?

```
} catch (const std::bad_excepected_access<std::errc>& e) {
   std::cout << "Error at filesystem or networking: " <<
std::make_error_code(e.error()).message();
} catch (const std::bad_excepected_access<lib1>& e) {
   std::cout << ???;
} catch (const std::bad_excepected_access<lib2>& e) {
   std::cout << ???;
} catch (const std::bad_excepected_access<lib3>& e) {
   std::cout << ???;
} catch (const std::exception& e) {
   std::cout << e.what();
}
```

std::expected ?

```
auto v1 = foo(1).or_else(throw_fs_error);
```

std::expected ?

```
auto v1 = foo(1).or_else(throw_fs_error);
auto v2 = foo(2).or_else(throw_fs_error);
auto v3 = foo(3).or_else(throw_fs_error);
```

std::expected ?

```
auto v1 = foo(1).or_else(throw_fs_error);
auto v2 = foo(2).or_else(throw_fs_error);
auto v3 = foo(3).or_else(throw_fs_error);

auto v1 = foo(1).value();
auto v2 = foo(2).value();
auto v3 = foo(3).value();
```

std::format и крутой трюк

Февральская встреча WG21 C++

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::generator неожиданно прост и надёжен (C++23 ?)

Итого

- std::generator неожиданно прост и надёжен (C++23 ?)
- std::stacktrace поможет вам в assert (C++23)

Итого

- std::generator неожиданно прост и надёжен (C++23 ?)
- std::stacktrace поможет вам в assert (C++23)
- std::stacktrace в исключениях (C++26 ?)

Итого

- std::generator неожиданно прост и надёжен (C++23 ?)
- std::stacktrace поможет вам в assert (C++23)
- std::stacktrace в исключениях (C++26 ?)
- [[assume(много-интересных-открытий)]]

Итого

- std::generator неожиданно прост и надёжен (C++23 ?)
- std::stacktrace поможет вам в assert (C++23)
- std::stacktrace в исключениях (C++26 ?)
- [[assume(много-интересных-открытий)]]
- Модули жгут, даже если импортировать всё (C++23)

Итого

- std::generator неожиданно прост и надёжен (C++23 ?)
- std::stacktrace поможет вам в assert (C++23)
- std::stacktrace в исключениях (C++26 ?)
- [[assume(много-интересных-открытий)]]
- Модули жгут, даже если импортировать всё (C++23)
- std::expected немного неожиданный

Итого

- std::generator неожиданно прост и надёжен (C++23 ?)
- std::stacktrace поможет вам в assert (C++23)
- std::stacktrace в исключениях (C++26 ?)
- [[assume(много-интересных-открытий)]]
- Модули жгут, даже если импортировать всё (C++23)
- std::expected немного неожиданный
- std::format станет надёжнее (C++23)

Спасибо

Полухин Антон

Эксперт-разработчик C++

antoshkka@gmail.com

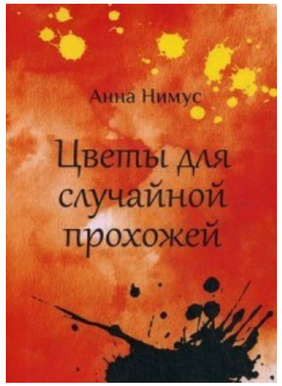
antoshkka@yandex-team.ru

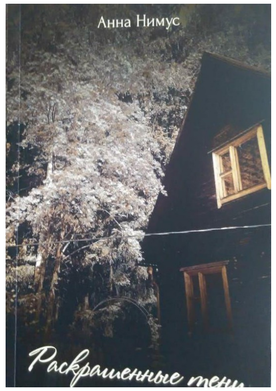
https://github.com/apolukhin

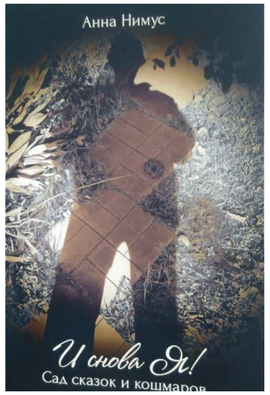
https://stdcpp.ru/

РГ21 C++ РОССИЯ

Спасибо


Анна Нимус
Цветы для случайной прохожей


Анна Нимус


Анна Нимус
И снова Я!
Сад сказок и кошмаров