
Яндекс

Яндекс

Жаркая встреча на Гаваях

Antony Polukhin
Полухин Антон

Автор Boost библиотек TypeIndex, DLL, Stacktrace
Maintainer Boost Any, Conversion, LexicalCast, Variant
Представитель РГ21, national body

О чём поговорим

- Modules
- wide_int и планы группы Numerics
- constexpr-отладка, детектирование, контейнеры
- Contracts
- Классы для работы с динамическими библиотеками

Modules

В чём сложности?

```
// Module interface unit of A
module A;
export template<class T1, class T2> void a(T1 t1, T2 t2) {
    t1 + t2;  // #1
}
```

В чём сложности?

```
// Module interface unit of B
#include <string> // not in the purview of B
import A;
module B;
export template <class T> void b(T t) {
  a(std::string{}, t);
}
```

В чём сложности?

```
// Translation unit of main()
import B;
int main() {
  b('0'); // ill-formed: ?+? not found at #1
}
```

Другие проблемы

- Макросы делают модули больше и их работу медленнее

Другие проблемы

- Макросы делают модули больше и их работу медленнее
- Include guards важны!

Другие проблемы

- Макросы делают модули больше и их работу медленнее
- Include guards важны!
- Заголовочные файлы некоторых ОС — огромны и имеют множество макросов (хочется из них сделать модули)

Другие проблемы

- Макросы делают модули больше и их работу медленнее
- Include guards важны!
- Заголовочные файлы некоторых ОС — огромны и имеют множество макросов (хочется из них сделать модули)
- Выпусть модули как есть в виде TS, а потом сильно обновить их? Или обновить и тогда выпускать?

Итого

Очередная попытка разрешить все проблемы будет в Торонто

wide_int

wide_int

Наше предложение прошло дальше

Numbers

Наполеоновские планы:

Numbers

- Наполеоновские планы:
  - Добавить множество классов для работы с числами
    - wide float
    - wide integer
    - decimal
    - unbounded float
    - unbounded integer
    - rational
    - safe integer

Numbers

- Наполеоновские планы:
  - Добавить множество классов для работы с числами
  - Добавить вспомогательные метафункции
    - template<int bits> using exact_2int = ...
    - template<int bits> using fast_2int = ...
    - template<int bits> using least_2int = ...
    - template<int bits> using exact_2uint = ...
    - template<int bits> using fast_2uint = ...
    - template<int bits> using least_2uint = ...

Numbers

- Наполеоновские планы:
  - Добавить множество классов для работы с числами
  - Добавить вспомогательные метафункции
  - Подправить numeric_traits

Numbers

- Наполеоновские планы:
  - Добавить множество классов для работы с числами
  - Добавить вспомогательные метафункции
  - Подправить numeric_traits
  - ...

Numbers

- Наполеоновские планы:
  - Добавить множество классов для работы с числами
  - Добавить вспомогательные метафункции
  - Подправить numeric_traits
  - ...
  - Состыковать всё вместе и убедиться что всё хорошо взаимодействует

constexpr

Функция constexpr()

```
constexpr double power(double b, int x) {
  if (constexpr() && x >= 0) {
    double r = 1.0, p = b; unsigned u = (unsigned)x;
    while (u != 0) {
      if (u & 1) r *= p;
      u /= 2; p *= p;
    }
    return r;
  } else {
    return __asm__ "...";
  }
```

std::constexpr_assert()

```
constexpr int sqr(int n) {
  if (n > 100) {
    std::constexpr_report("Largish sqr operand", n);
  }
  return n*n;
}
```

std::constexpr_trace()

???

std::constexpr_vector

```
constexpr constexpr_vector<int> x;  // Okay.
constexpr constexpr_vector<int> y{ 1, 2, 3 };  // Okay.
using Ints = std::constexpr_vector<int>;
constexpr auto series(int n)->Ints {
  Ints r{};
  for (int k; k<n; ++k) r.push_back(k);
  return r;
}
```

std::constexpr_vector альтернатива

```
template <class T>
using constexpr_vector = std::vector<T, std::constexpr_allocator<T>>;

template <class T>
using constexpr_set = std::set<T, std::constexpr_allocator<T>>;

template <class CharT, class Traits = std::char_traits<CharT> >
using constexpr_basic_string =
        std::basic_string<CharT, Traits, std::constexpr_allocator<CharT>>;
```

Contracts

Contracts — ассерты на стероидах

```
void push(int x, queue & q)
  [[expects: !q.full()]]
  [[ensures: !q.empty()]]
{
  //...
  [[assert: q.is_valid()]];
  //...
}
```

Contracts — ассерты на стероидах

```
void push(int x, queue & q) [[expects: !q.full()]] [[ensures: !q.empty()]] ;

queue q;

// ...

if (!q.full()) {
    push(10, q); // Хм... можно не проверять
    if (q.empty()) { // Хм...
    }
} else {
    push(11, q); // Хм... подозрительно
}
```

Contracts — ассерты на стероидах

```
void(const std::contract_violation &);


namespace std {
  class contract_violation {
  public:
    int line_number() const noexcept;
    const char * file_name() const noexcept;
    const char * function_name() const noexcept;
    const char * comment() const noexcept;
  };
}
```

DLL

Что сделали в Коне с DLL

Что сделали в Коне с DLL

- Покритиковали

Что сделали в Коне с DLL

- Покритиковали
- Can of worms

Что сделали в Коне с DLL

- Покритиковали
- Can of worms:
  - Нужно прописать в стандарте достаточно много вещей

Что сделали в Коне с DLL

- Покритиковали
- Can of worms:
  - Нужно прописать в стандарте достаточно много вещей
  - Нужно пройтись по множеству подгрупп

Что сделали в Коне с DLL

- Покритиковали
- Can of worms:
  - Нужно прописать в стандарте достаточно много вещей
  - Нужно пройтись по множеству подгрупп
  - Очень много тёмных уголков
    - Исключения
    - RTTI
    - TLS

Что сделали в Коне с DLL

- Покритиковали
- Can of worms:
  - Нужно прописать в стандарте достаточно много вещей
  - Нужно пройтись по множеству подгрупп
  - Очень много тёмных уголков
    - Исключения
    - RTTI
    - TLS
- Насколько люди заинтересованы в идее?

Спасибо! Вопросы?

https://stdcpp.ru/