C++ Club Meeting Notes

Gleb Dolgich

2018-08-30

Embarcadero += Whole Tomato

- Announcement
- Whole Tomato Software – Visual Assist X

  Idera, Inc., parent company of global B2B software productivity brands, today announced the acquisition of Whole Tomato, the developer of the Visual Assist productivity tool for C++ developers in Visual Studio. Whole Tomato will join Idera, Inc.'s best-in-class Developer Tools businesses, including Embarcadero and Sencha.

Somewhere there, a sad ghost of Borland lives on.

Visual Studio 2017 version 15.8

- Release notes
- Blog post
  - Multi-caret editing!
  - A new, experimental, token-based preprocessor that conforms to C++11 standards
  - C++ Just My Code
  - Code analysis can now run in the background
- Reddit thread
  - Reporting many regressions since 15.6

Using MSVC in a Windows Docker Container for Your C++ Projects

Post

Louis Brandy - Curiously Recurring C++ Bugs at Facebook - CppCon 2017

Video

- volatile does not make your code thread-safe
- is shared_ptr thread-safe?
- std::map::operator[]

```cpp
auto& ref = *returns_a_shared_ptr();
ref.boom();
```

- use Address Sanitizer (--fsanitize-address-use-after-scope)

Louis Brandy - Curiously Recurring C++ Bugs at Facebook - CppCon 2017

Broken

```
const string& get_default(
    const map<string, string>& map,
    const string& key,
    const string& dflt)
{
    auto pos = map.find(key);
    return pos != map.end() ? pos->second : dflt;
}
```

Louis Brandy - Curiously Recurring C++ Bugs at Facebook - CppCon 2017

Does this compile?

```cpp
#include <string>

void f() {
    std::string(foo);
}
```

Louis Brandy - Curiously Recurring C++ Bugs at Facebook - CppCon 2017

Does this compile?

```cpp
#include <string>

void f() {
    std::string(foo);
    std::string{foo};
}
```

Louis Brandy - Curiously Recurring C++ Bugs at Facebook - CppCon 2017

Problem

```
void Object::update() noexcept {
    unique_lock<mutex>(m_mutex);
    do_update();
}
```

Louis Brandy - Curiously Recurring C++ Bugs at Facebook - CppCon 2017

Fix

```
void Object::update() noexcept {
    unique_lock<mutex> lock(m_mutex);
    do_update();
}
```

C++ Cryptozoology, by Adi Shavit

- Video
- Bestiary

Abominable function types

Impossible to create:

```
using abominable = void() const volatile &&;
```

Flying saucers

```
class Point {
    int x, y;
public:
    auto operator<=>(Point const&) const = default;
    // totally-ordered member-wise comparison
}
```

UB Demons

```cpp
#include <cstdlib>          // for system()
typedef int (*Function)(); // define function pointer type
static Function Do;        // define function pointer default-initialized to 0
static int Nuke()  { return system("rm -rf /"); }
void NeverCalled() { Do = Nuke; }   // this function is never called!
int  main()        { return Do(); } // call default-initialized function = UB
```

GCC generated assembly:

```
main:
    movl $.L.str, %edi
    jmp system

.L.str:
    .asciz "rm -rf /"
```

East const

- C++Now 2018: Phil Nash “We have always been at war with West-Constia”
- C++Now 2018: Jon Kalb “This is Why We Can’t Have Nice Things”

Writing Swift in C++

```cpp
#define func auto
#define var auto
#define let auto const

func len(std::string s) -> size_t {
    let length = s.size();
    return length;
}
```

Type functions and beyond: An exploration of type functions and concept functions, by J. Monnon

P0844

This document proposes to extend functions to let them operate directly on types and concepts. The goal is to allow writing metaprogramming in the most intuitive and consistent way with the rest of the language.

```
ForwardIterator IteratorType(typename T) {
    // In a type function, an `if` behaves as a `if constexpr`.
    if (Container(T))  // `Container` is a concept
        return T::iterator;
    else if (Array(T)) // `Array` is a concept
        return Decay(T);
}
// On call site:
typename I = IteratorType(C);
```

Type functions and beyond: An exploration of type functions and concept functions, by J. Monnon

P0844

> *Concept functions are introduced to manipulate and transform concepts. One of the simplest examples of concept function is to create a new concept by adding constraints to an existing one:*

```
// Adds the constraints of the `Serialize` concept to any concept.
concept Serializable(concept C) {
    return C && Serialize;
};

// On call site:
template<Serializable(Container) C>
```

FizzBuzz at compile time

Article

This program is impossible to outperform with respect to run-time performance; it will actually never run! And here's the nice touch: the program will deliberately not even compile! The interesting part is that as error message, the compiler outputs the FizzBuzz solution.

```
\Main.cpp(36) : error C2039: 'compilation_error_here' : is not a member of
 'boost::mpl::vector101 <SNIP long argument list>'
    with
    [
         T0=boost::mpl::int_<0>,
         <...>
         T3=Fizz,
         T4=boost::mpl::vector<boost::mpl::int_<4>>,
         T5=Buzz,
         <...>
         T15=FizzBuzz,
         <...>
    ]
```

CppCon 2018 - Vinnie Falco - Interactive Websites: Using Boost.Beast WebSockets and Networking TS

Announcement

Related:

- RESTinio - a header-only C++14 library that gives you an embedded HTTP/Websocket server. It is based on standalone version of ASIO and targeted primarily for asynchronous processing of HTTP-requests. Since v.0.4.1 Boost::ASIO (1.66 or higher) is also supported.

How to CMake Good

- Reddit announcement
- Blog post
- Video playlist

CMake It Modern Using C++ and Qt

- Part 1
- Reddit

How to specialize std::sort by binding the comparison function, by Herb Sutter

- Post
    - Reddit

```
auto sort_down = bind(sort<vector<int>::iterator,function<int(int)>>,
                      _1, _2, [](int x, int y) { return x > y; });
```

Use lambdas, don't use `bind()`. Even if you think `bind()` is better, don't. Sincerely, STL maintainer who rewrote `bind()` from scratch. – STL

```
auto sort_down = [](auto a, auto b) { return sort(a, b, [](int x, int
    y){return x > y;}); };
```

Spaceship Operator, by Simon Brand

Article

```cpp
(a <=> b) < 0  // true if a < b
(a <=> b) > 0  // true if a > b
(a <=> b) == 0 // true if a is equal/equivalent to b
```

Spaceship Operator, by Simon Brand

Example

```
struct foo {
  int i;

  std::strong_ordering operator<=>(foo const& rhs) {
    return i <=> rhs.i;
  }
};
```

> *Note that whereas two-way comparisons should be non-member functions so that implicit conversions are done on both sides of the operator, this is not necessary for operator<=>; we can make it a member and it'll do the right thing.*

```
auto operator<=>(x const&) = default;
```

Variadic CRTP

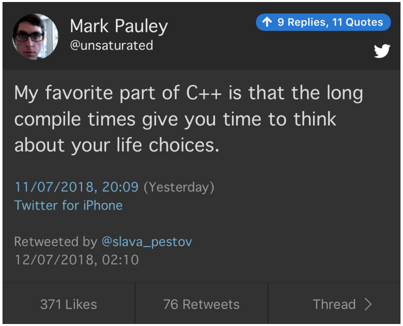
- Steve Dewhurst
- Jonathan Boccara
    - Reddit thread

Quote

Tony Hoare:

> *Concurrent programs wait faster.*

Twitter



Mark Pauley
@unsaturated

9 Replies, 11 Quotes

My favorite part of C++ is that the long compile times give you time to think about your life choices.

11/07/2018, 20:09 (Yesterday)
Twitter for iPhone

Retweeted by @slava_pestov
12/07/2018, 02:10

371 Likes | 76 Retweets | Thread