

C++ Club Meeting Notes

Gleb Dolgich

2018-04-26

ACCU Conf 2018 (2018-04-10–2018-04-14) Trip Reports

- ▶ [JetBrains](#), by Anastasia Kazakova (PMM for C++ Tools), Timur Doumler (Software Developer in the CLion team) and Phil Nash (JetBrains C++ tools Developer Advocate)
- ▶ [Felix Petriconi](#)
- ▶ [Mathieu Ropert](#)
- ▶ [Arne Mertz](#)

C++17 - The Best Features - Nicolai Josuttis (ACCU 2018)

- ▶ YouTube
- ▶ Reddit

conference.accu.org

C++17 A Header-Only ::new Tracker

```
class TrackNew
{
private:
    inline static int numMalloc = 0; // number of ::new or ::new[] calls
    inline static long sumSize = 0; // bytes allocated so far
    inline static bool doTrace = false; // tracing enabled
public:
    static void trace(bool b) { // enable/disable tracing
        doTrace = b;
    }
    static void status() { // print current state
        std::cerr << numMalloc << " mallocs for " << sumSize << " Bytes" << '\n';
    }
    static void* allocate(std::size_t size, const char* call) { // implementation of
        trackedAllocation
        ++numMalloc;
        sumSize += size;
        if (doTrace) ...
        return std::malloc(size);
    }
};

inline void* operator new (std::size_t size) {
    return TrackNew::allocate(size, "::new");
}
...
```

C++
©2018 by IT communication.com

josuttis | eckstein
2 IT communication

@ACCUconf

ACCU
2018
April 11 - 14

C++17 - The Best Features - Nicolai Josuttis (ACCU 2018) (cont.)

conference.accu.org

C++17: Compile-Time if

```
template <typename T>
std::string asString(T x)
{
    if constexpr(std::is_arithmetic_v<T>) {
        return std::to_string(x);
    }
    else if constexpr(std::is_same_v<T, std::string>) {
        return x;
    }
    else {
        return std::string(x);
    }
}
std::cout << asString(42) << '\n';
std::cout << asString(std::string("hello")) << '\n';
std::cout << asString("hello") << '\n';
```

All example calls
would be invalid
when using run-time if



C++
©2018 by IT communication.com

3 | josuttis | eckstein
IT communication

@ACCUconF

ACCU
2018
April 11 – 14

C++17 - The Best Features - Nicolai Josuttis (ACCU 2018) (cont.)

conference.accu.org

Polymorphism Example with std::variant<>

• no pointers
• no new/delete in appl. code
• objects located together

• no common base class
• no virtual functions

~~Circle~~ Line

```
using GeoObjVar = std::variant<Circle, Line>;  
  
std::vector<GeoObjVar> createFig()  
{  
    std::vector<GeoObjVar> f;  
    f.push_back(Line(Coord(1, 2), Coord(3, 4)));  
    f.push_back(Circle(Coord(5, 5), 2));  
    return f;  
}  
  
void drawElems (const std::vector<GeoObjVar>& v)  
{  
    for (const auto& geoobj : v) {  
        std::visit([](auto& obj) {  
            obj.draw();  
        }, geoobj);  
    }  
}  
  
std::vector<GeoObjVar> fig = createFig();  
drawElems(fig);  
for (auto& geoobj : fig) {  
    std::visit([](auto& obj) {  
        obj.move(Coord(2, 2));  
    }, geoobj);  
}  
drawElems(fig);  
fig.clear(); // remove all elements in the vector
```

visitor uses a local viable

C++
©2018 by IT communication.com

7 IT communication

josuttis | eckstein

ACCU 2018
April 11 – 14



C++17 - The Best Features - Nicolai Josuttis (ACCU 2018) (cont.)

conference.accu.org

C++17: std::variant<> Visitors

```
std::variant<Circle, Line> var;
...
template<typename... Functors>
struct Ovld : Functors... {
    using Functors::operator()...; // derive op() from all functors
};
// deduce template arguments from initializers:
template<typename... Functors> Ovld(Functors...) -> Ovld<Functors...>;
std::visit(Ovld{[] (Circle c) {
    std::cout << "Circle " << c << '\n';
}, [] (Line l) {
    std::cout << "Line " << l << '\n';
}}, var);
```



The image shows Nicolai Josuttis, a man with glasses and a blue jacket, speaking at a conference. He is standing in front of a wooden wall. The slide behind him displays C++ code demonstrating the use of std::variant and std::visit.

C++
©2018 by IT communication.com

josuttis | eckstein
8 IT communication

ACCU
2018
April 11 - 14

@ACCUconf

C++17 - The Best Features - Nicolai Josuttis (ACCU 2018) (cont.)



conference.accu.org

C++17: std::variant<> Visitors C++17 Features:

- std::variant<>
- Aggregates can have base classes
- Deduction guides for class template arguments

```
std::variant<Circle, Line> var;
...
template<typename... Functors>
struct Ovld : Functors... {
    using Functors::operator()...; // derive op() from all functors
};
// deduce template arguments from initializers:
template<typename... Functors> Ovld(Functors...) -> Ovld<Functors...>;
std::visit(Ovld{[] (Circle c) {
    std::cout << "Circle " << c << '\n';
}, [] (Line l) {
    std::cout << "Line " << l << '\n';
}}, var);
```

C++
©2018 by IT communication.com

josuttis | eckstein
8 IT communication

@ACCUconf

ACCU 2018
April 11 - 14

C++17 - The Best Features - Nicolai Josuttis (ACCU 2018) (cont.)

conference.accu.org

Using transform_reduce()

```
#include <numeric> // for transform_reduce()
#include <execution> // for the execution policy
...
void printDirSize(const std::string& dir)
{
    // init paths with all paths from directory dir:
    std::vector<std::filesystem::path> paths;
    std::filesystem::recursive_directory_iterator dirpos(dir);
    std::copy(begin(dirpos), end(dirpos),
              std::back_inserter(paths));

    // sum size of regular files:
    auto sz = std::transform_reduce(
        std::execution::par, // parallel execution policy (opt)
        paths.cbegin(), paths.cend(), // range
        std::uintmax_t{0}, // initial value (and return type)
        std::plus<>(),
        [](const std::filesystem::path& p) { // ... file size if regular file
            return is_regular_file(p) ? file_size(p)
                                      : std::uintmax_t{0};
        });
    std::cout << "sum of size of regular files: " << sz << '\n';
}
```

Can't pass directory iterator to parallel algorithms, because they require forward iterators

Call lambda for each element and accumulate results

C++
©2018 by IT communication.com

9 | josuttis | eckstein
IT communication

@ACCUconf



ACCU
2018
April 11 – 14

C++17 - The Best Features - Nicolai Josuttis (ACCU 2018) (cont.)



conference.accu.org

Uniform Initialization now Almost Done

```
int i2 = 0;           // "copy initialization" (from C)
int i3(0);           // "direct initialization" (since C++98)
int i4(0);           // "direct initialization" from Modern C++ (since C++11)
int i5 = {0};         // "copy initialization" from Modern C++
```

```
unsigned long l1(-17);      // ERROR (good!): "narrowing" detected
unsigned long l2(4.8);       // ERROR (good!): "narrowing" detected
std::vector<int> v1(8, 15); // self explanatory (unlike v1(8,15))
```

```
auto x[42];            // x is int since C++17 (was std::initializer_list<int> before C++17)
auto y[0,8,15];          // ERROR since C++17 (good!) (was std::initializer_list<int> before C++17)
auto z = {0,8,15};       // OOPS: still std::initializer_list<int>
```

```
enum class MyIntegralEnum;
MyIntegralEnum e(17);     // OK since C++17 (all other initializations are still ill-formed)
```

```
struct CData {
    int amount;
    double value;
};
struct DData : CData {
    std::string name;
    void print() const;
};
DData a({42, 6.7}, "book"); // C++17: initialize all elems
```

```
AnyType t();           // "value initialization" (default value or 0/false/nullptr for FDT's)
int i6();              // initialized with 0 (since C++11)
DData b();              // same as: {0,0.0,""}
MyIntegralEnum f();     // still OK
```

josuttis | eckstein

C++
©2018 by IT communication.com

10 IT communication

@ACCUconF

ACCU
2018
April 11 – 14

VCPkg, now for Linux and macOS

- ▶ [Post](#)
- ▶ [Reddit](#), [Reddit](#)

{fmt}

- ▶ Website
- ▶ GitHub

code::dive 2017 – John Lakos – Value semantics: It ain't about the syntax!

► Video

2. Understanding Value Semantics

Where is “Value” Defined? `operator==`

The associated, homogeneous (free) `operator==` for a type T

Implementation

1. Provides an *operational definition* of what it means for two objects of type T to have “the same” *value*.
2. Defines the *salient attributes* of T as those attributes whose respective values must *compare equal* in order for two instances of T to *compare equal*.

Interface/Contract

235



code::dive 2017 – John Lakos – Value semantics: It ain't about the syntax! (cont.)

2. Understanding Value Semantics

“Value Types” having Value Semantics

A C++ type that “properly” represents (a subset of) the values of an abstract “mathematical” type is said to have value semantics.



code::dive 2017 – John Lakos – Value semantics: It ain't about the syntax! (cont.)

The screenshot shows a mobile device displaying a video player interface. At the top, there's a navigation bar with icons for back, forward, search, and other controls. The main content area has a white background. In the top left corner of the content area, there's a small yellow icon followed by the text "cs - It ain't about the syntax! code::dive conference - 20171126 - code ::dive 2017 – J H/W ...". Below this, the title "What to Remember about VSTs" is displayed in blue. Underneath the title, the text "The key take-away:" is written in black. A bulleted list follows, with each item preceded by a yellow icon:

- ⌚ What makes a value-type *proper*
- ⌚ has essentially nothing to do with *syntax*; it has everything to do with *semantics*: A class that respects the
- ⌚ **Essential Property of Value** is
- ⌚ value-semantic...

At the bottom of the slide, there's a progress bar showing the time "1:35:18" on the left and "-0:25" on the right, indicating the video is slightly behind schedule. The bottom right corner of the slide features the "dts HEADPHONE" logo. The overall interface includes standard video player controls like play/pause, volume, and brightness adjustment.

Reddit: Why is modern C++ seemingly not being taught?

- ▶ Reddit

Reddit: Why is C++ so hated?

- ▶ Reddit

C++ in Zircon (Fuchsia OS kernel)

- ▶ [Readme](#)

Brigand – Instant compile time C++11 metaprogramming library

- ▶ [GitHub](#) (header-only, Boost Software License 1.0)
- ▶ [Docs](#)
- ▶ [Meeting C++ 2015 video](#)
- ▶ [CppCon 2016 video](#)
- ▶ [Ebook](#)

Everything you were doing with Boost.MPL can be done with Brigand.

Brigand – Instant compile time C++ 11 metaprogramming library (cont.)

- ▶ Example tasks:
 - ▶ Create a tuple from a list of types and then transform it into a variant
 - ▶ Look for the presence of a type in a tuple and get its index
 - ▶ Sort a list of types
 - ▶ Advanced static assertion with arithmetics and complex functions
 - ▶ Go through a list of types and perform a runtime action for each type

Raspberry Pi - Install Clang 6 and compile C++17 programs

► Post

FoundationDB and Flow by Apple

► Post

Flow, a new programming language that brings actor-based concurrency to C++11. Flow is implemented as a compiler which analyzes an asynchronous function (actor) and rewrites it as an object with many different sub-functions that use callbacks to avoid blocking. The Flow compiler's output is normal C++11 code, which is then compiled to a binary using traditional tools. Flow also provides input to our simulation tool, which conducts deterministic simulations of the entire system, including its physical interfaces and failure modes. In short, Flow allows efficient concurrency within C++ in a maintainable and extensible manner.

C++ Patterns

[Website](#)

Twitter

 T045TBR0T
@t045tbr0t

[↑ 1 Reply, 99 Quotes](#) 

finally, a monitor that will fit the entire name of my Java classes

10/03/2018, 00:57 (Saturday)
Twitter for Android

Retweeted by @lenoir_aaron
10/03/2018, 20:44

19024 Likes

6941 Retweets

Thread >

