C++ Club Meeting Notes

Gleb Dolgich

2018-02-15

C++ in 2018, by Jens Weller, Meeting C++

Post

Overload #143

ACCU, PDF

- Class template argument deduction, by Roger Orr
- C++ with metaclasses, by Francis Glassborow
- Functional Error-Handling with Optional and Expected, by Simon Brand
- A Multiple-Writers-Single-Reader (MWSR) queue with minimalist locking

Book: Better Code: Goals for Software Developers, by Sean Parent

Preorder on Amazon

Released 19 May 2018.

StackOverflow: `int` in C++

- Why aren't the C-supplied integer types good enough for basically any project?
- Is there still a reason to use `int` in C++ code?

StackOverflow: Scope of a variable initialized in the parameter list of a function

Question

emBO++: Embedded C++ Conference in Bochum, Germany

Home page

Insomniac Games C++ Coding Standard (2011)

Gist – Reddit

- 2-space indents, no tabs
- No exceptions
- Constructors shouldn't do any real work, the real initialization is done in `Create()` or `TryCreate()`
- Destructors are empty, all deinitialization is done in `Destroy()`
- Integer scalars are signed
- No `enum` data members (size is implementation-specific)
- Output parameters are pointers, not references ("References can be confusing")
- Function parameter order: output, update, input
- Use C-style casts. Do not use C++-style casts ("The brevity of the C-style cast outweighs the semantic benefits of explicitness of C++-style casts")
- Under no circumstances should the keyword `mutable` be used

JUCE C++ Coding Standard

Link

- 4 spaces, no tabs, Allman-style braces
- Space before opening parens `func (foo, bar)`
- Avoid underscores in names (except macros)
- Don't use macros
- Avoid C-style casts
- Pass small types by value

The 15 C++11 features you must really use in your C++ projects

Article

CppCon 2016: David Sankel “Variants: Past, Present, and Future”

YouTube

- Variant: ‘OR’ type, sum type, discriminated union, ADT, “one-of” type
- P0088R3
- Implementations: Anthony Williams, Eric Fiselier
- `std::overload` by Vicente Botet Escriba: P0051R2

```cpp
void output(const std::variant<std::string, int>& v) {
    return std::visit(std::overload(
        [](const std::string& s) {std::cout << "String: " << s << "\n";},
        [](const int i) {std::cout << "Integer: " << i << "\n";}), v);
}
```

Mach7: A pattern-matching library for C++, by Yuriy Solodkyy, Gabriel Dos Reis, Bjarne Stroustrup

- GitHub (BSD)
- Article: Another Polymorphism
- Generated code is faster than visitors

```
void print(const boost::variant<double,float,int>& v)
{
    var<double> d; var<float> f; var<int> n;
    Match(v)
    {
        Case(C<double>(d)) cout << "double " << d << endl; break;
        Case(C<float> (f)) cout << "float  " << f << endl; break;
        Case(C<int>   (n)) cout << "int    " << n << endl; break;
    }
    EndMatch
}
```

C++Now 2017: Vittorio Romeo “Implementing `variant` visitation using lambdas”

- YouTube
- Scelta on GitHub: C++17 zero-overhead syntactic sugar for `variant` and `optional`

This is trivial. *Is this clear to everyone?* ***Silence***

C++Now 2017: Vittorio Romeo “Implementing `variant` visitation using lambdas” (cont.)

```
enum MyVariant {
    IntTag(i32),
    FloatTag(f32),
    DoubleTag(f64)
}

let v0 = FloatTag(2.0);
match v0 {
    IntTag(x)    => println!("{}i", x),
    FloatTag(x)  => println!("{}f", x),
    DoubleTag(x) => println!("{}d", x)
}
```

C++Now 2017: Vittorio Romeo “Implementing `variant` visitation using lambdas” (cont.)

```cpp
using shape = std::variant<circle, box>;
shape s0{circle{/*...*/}};
shape s1{box{/*...*/}};

// In place `match` visitation.
scelta::match([](circle, circle){ /* ... */ },
              [](circle, box)   { /* ... */ },
              [](box,    circle){ /* ... */ },
              [](box,    box)   { /* ... */ })(s0, s1);
```

CppCon 2015: John R. Bandela “Simple, Extensible Pattern Matching in C++14”

- YouTube
- simple_match (C++14), no macros, focus on clarity and simplicity (not speed)
- GitHub

YouTube

Computers	Biology
Byte: 8 bits	Byte: 3 ‘bits’
Bit: 0 or 1	‘Bit’: (G or C) or (T or A)
C++	Assembler + punch tape

Programming life? New cures? Growing things? A new Clang target?

Verdigris: Qt without MOC, v1.0

- GitHub
- Docs
- Reddit post
- HackerNews post from 2 years ago

Selene: A C++14 image representation, processing and I/O library

- GitHub (MIT)
- Offers flexible classes for image and multi-channel pixel representations, and functions for image data access.
- Provides easy-to-use APIs to read and write images in JPEG and PNG formats (leveraging libjpeg and libpng).
- Offers basic image processing algorithms such as color conversions, pixel-wise operations, rotation, flipping, etc.
- Lightweight and easy to build using CMake on Linux, MacOS, Windows.

Vc: SIMD Vector Classes for C++ by Matthias Kretz

- GitHub (C++14) (BSD-3-Clause)
- P0214R7: Data-Parallel Vector Types & Operations

libbson: A BSON utility library by MongoDB

- GitHub (Apache-2.0)

TscanCode: Static analyser for C++, C#, Lua

GitHub

Transwarp: A header-only C++ library for task concurrency

GitHub