C++ Club

24 August 2017

C++ Core Guideline Checker in VS2017

Post

- How to enable the checker in VS2017
- How to choose rules

LLVM now supports PDB on Windows

Post

- CodeView format: Microsoft, mid-1980's, poorly documented
- Linux: DWARF
- PDB = CodeView + database (completely undocumented)

  I got some odd stares (to put it lightly) when I suggested that we just ask Microsoft if they would help us out. But ultimately we did, and... they agreed!

Microsoft: `cvdump`, LLVM: `llvm-pdbutil`

GCC 7.2 released

Announcement

- Bugfix release (>95 bugs fixed)
- Changes in GCC 7

Boost 1.65 released

Link

New libraries

- **PolyCollection**: Fast containers of polymorphic objects
- **StackTrace**: Gather, store, copy and print backtraces

Outcome v2 is stable

- Reddit post
- Ready for the 2nd Boost review
- Deployed in 2 libs
- Shots fired!

  I know those coming from Rust-land feel amazement how long it has taken C++ to replicate Rust's `Result<T, E>`, but I'm very sure ours is enormously superior to theirs already simply by doing so much less because it doesn't need to. Taking time to get design right is one of the big things which continues to separate C++ from the upstart systems programming languages. Long may it continue!
- You'll never guess what happened next.

Mutable Lambdas

Blog post

Object captured in lambda are immutable by default. This is because `operator()` of the generated functor is `const` by default.

```cpp
auto func = [a]() mutable -> int { ++a; std::cout << a; return a; };
```

Beware of C++ - Nicolai Josuttis - C++Now 2014 Keynote

YouTube

We need guidelines on how to use C++ features

- `noexcept`: HH did a test with a move ctor => 10 times faster with `noexcept`
- Should we have semantics for "strongly encouraged to be `noexcept`"?
- Issues with constructors, initializer lists and `explicit`
- Guidelines for `constexpr`
- The many iterations of `std::make_pair`
- Guidelines for template parameters

C++ Core Guidelines

The Speed Game: Automated Trading Systems in C++ - Carl Cook - Meeting C++ 2016

YouTube

- A few important lines of code
- Millions of market data events per second
- Jitter is a killer
- Latency, not throughput
- No mistakes (or very good recovery)
- Know your hardware
- Don’t try to be an optimizing compiler
- Cache warming
- Bypass the kernel (user-space code, incl. networking - OpenOnload)

The Speed Game: Automated Trading Systems in C++ - Carl Cook - Meeting C++ 2016 (cont.)

YouTube

- Compile-time dispatch
- `constexpr`
- Variadic templates
- Loop unrolling
- Expression short-circuiting
- Branch prediction/reduction
- Avoiding allocations; placement `new`
- Fast containers
- LTO

The Speed Game: Automated Trading Systems in C++ - Carl Cook - Meeting C++ 2016 (cont.)

YouTube

Downsides of C++:

- Zero-size vectors may have a cost
- `std::function` allocates memory
- x86 has a stronger memory model than C++11
  - you see fewer concurrency bugs, but at a performance cost
- Standard containers and allocators are non-deterministic in runtime cost

SG14 Google group

Type_safe by Jonathan Müller

- Zero overhead utilities for preventing bugs at compile time (MIT)
- GitHub :: Docs
- C++11
- Header-only
- Reddit thread on constraining parameter values

“At first I loved `auto`, now I’m just annoyed by it”

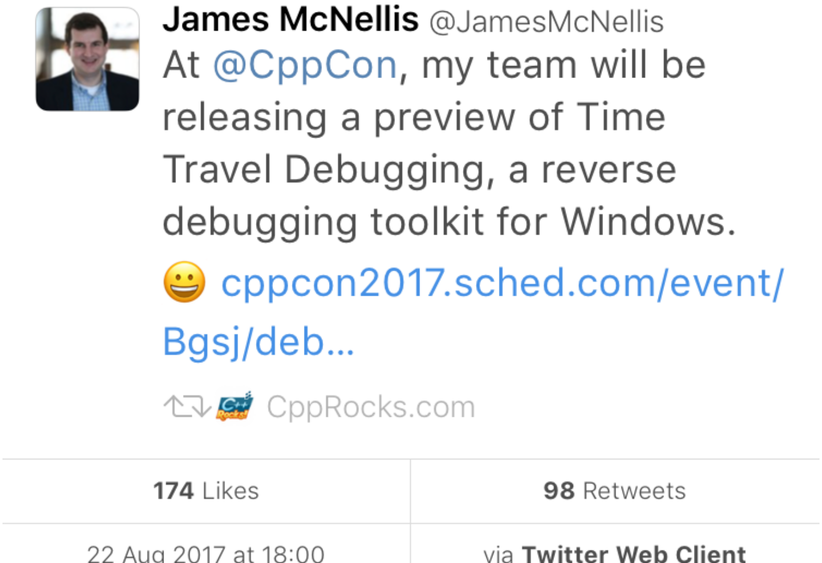
- Reddit thread
- Use everywhere or only when it can’t be avoided?
- Use only for iteratots, range for loops and lambdas?
- Video: “Autofobia”
- Article
  - AAA = Almost Always Auto (Herb Sutter)
  - AAAA = Almost Always Avoid Auto (James McNellis, Michael Caisse)
  - CRA = Contextually Relevant Auto

Handling std::bad_alloc

- CppCon 2016 talk by Sergey Zubkov

Time Travel Debugging is coming to Windows

James McNellis's announcement for the CppCon 2017 session



James McNellis @JamesMcNellis

At @CppCon, my team will be releasing a preview of Time Travel Debugging, a reverse debugging toolkit for Windows. 😀 cppcon2017.sched.com/event/Bgsj/deb...

CppRocks.com

174 Likes	**98** Retweets
22 Aug 2017 at 18:00	via **Twitter Web Client**

Ferret

Website

Ferret is a free software Clojure implementation, it compiles a restricted subset of the Clojure language to self contained ISO C++11 which allows for the use of Clojure in real time embedded control systems.

Generated code is self contained ISO C++11, it is not tied to any one compiler, generated code should be portable between any Operating System and/or Microcontroller that supports a C++11 compliant compiler. It has been verified to run on architectures ranging from embedded systems with as little as 2KB of RAM to general purpose computers running Linux/Mac OS X/Windows.

Ferret (cont.)

Here's a program that sums the first 5 positive numbers.

```clojure
(defn positive-numbers
  ([]
   (positive-numbers 1))
  ([n]
   (cons n (lazy-seq (positive-numbers (inc n))))))

(println (->> (positive-numbers)
              (take 5)
              (apply +)))
```

```
$ ./ferret -i lazy-sum.clj
$ g++ -std=c++11 -pthread lazy-sum.cpp
$ ./a.out
15
```

SQLite ORM

GitHub

- C++14
- Header-only
- The only dependency is `libsqlite3`
- Licence: BSD

Arthur O'Dwyer - The Rule of Seven (Plus or Minus Two) - Modern C++ Boilerplate - C++Now 2015

- YouTube
- Slides PDF

C++98 had the Rule of Three (or was it Four?). C++11 has the Rule of Five — or Six, if you count the default constructor — or Seven, if you count `swap()`. Should `swap()` be a member function? When is a default constructor absolutely mandatory? When is `noexcept` required for good performance? Should our classes support self-assignment and self-move? When is `=default` different from empty braces? We'll present reasonable answers to these questions and more.

Overload #140

PDF

- The C++ Core Guidelines
- A C++ Developer sees Rustlang for the first time
- Allocator for (Re)Actors