C++ Club Meeting Notes

Gleb Dolgich

2019-06-13

P0976r0: Bjarne Stroustrup - The Evils of Paradigms

http://wg21.link/p0976r0

> *In programming, some people deem imperative, object-oriented, and functional programming different paradigms. I think the very notion of a paradigm does harm to use and to design because people all too easily fall into the trap of considering only one paradigm "good" and then try to fit everything into it, discarding all aspects of alternative "paradigms" as wrong or inferior (aka "If your only tool is a hammer, everything looks like a nail").*

CppCast: Pattern Matching with Michael Park

http://cppcast.com/2019/06/michael-park/

P1371: Pattern Matching http://wg21.link/p1371r0

The Story: Bloomberg senior developer Elliot Goodrich

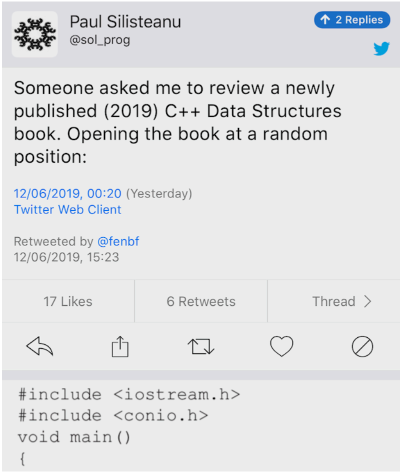
https://www.client-server.com/blog/2019/06/the-story-bloomberg-senior-developer-elliot-goodrich

> *I don't think C++ is dying by any stretch of the imagination.*

https://www.reddit.com/r/cpp/comments/bw8fyu/interesting_interview_on_client_servers_blog_with/

Modern C++ - authors and books

https://www.reddit.com/r/cpp/comments/by60wx/modern_c_authors_and_books/


Paul Silisteanu
@sol_prog

Someone asked me to review a newly published (2019) C++ Data Structures book. Opening the book at a random position:

12/06/2019, 00:20 (Yesterday)
Twitter Web Client

Retweeted by @fenbf
12/06/2019, 15:23

17 Likes | 6 Retweets | Thread

```
#include <iostream.h>
#include <conio.h>
void main()
{
```

Enum template parameters

```cpp
enum Foo {FooBar, FooBaz};

template<Foo foo>
struct Test {
    Kind kind{foo};
};

int main() {
    Test<FooBar> fooBar;
    assert(fooBar.kind==FooBar);
}
```

Roman numerals in C++

https://github.com/tcbrindle/numeris_romanis

https://www.reddit.com/r/cpp/comments/bxiqmm/numeris_romanis_roman_numeral_support_for_c17/

Unrelated: What are the rules about using an underscore in a C++ identifier?

> *Each name that begins with an underscore is reserved to the implementation for use as a name in the global namespace.*

Understanding when not to **std::move** in C++

https://developers.redhat.com/blog/2019/04/12/understanding-when-not-to-stdmove-in-c/

GCC 9:

```
-Wall -Wpessimizing-move
```

Example:

```cpp
T fn() {
  T t;
  return std::move(t); // Prevents NRVO: returned expression must be a name
}
```

Understanding when not to **std::move** in C++ (cont.)

GCC 9:

```
-Wextra -Wredundant-move
```

Example:

```cpp
struct T {
  T(const T&) = delete;
  T(T&&);
};

T fn(T t) {
  return std::move(t); // Redundant: move used implicitly
}
```

Understanding when not to **std::move** in C++ (cont.)

When `std::move` makes sense:

```cpp
struct U {};
struct T : U {};

U f() {
  T t;
  return std::move(t); // Necessary
}
```

Explanation:

> *When a function returns an object whose type is a class derived from the class type the function returns. In that case, overload resolution is performed a second time, this time treating the object as an* lvalue.

Building better software with better tools: sanitizers versus valgrind

https://lemire.me/blog/2019/05/16/building-better-software-with-better-tools-sanitizers-versus-valgrind/

Also: No more leaks with sanitize flags in gcc and clang

variadic_future

A variadic, completion-based future class for C++17

https://github.com/FrancoisChabot/variadic_future (Apache 2.0)

Re-implementing an old DOS game in C++17

https://lethalguitar.wordpress.com/2019/05/28/re-implementing-an-old-dos-game-in-c-17/

Code: https://github.com/lethal-guitar/RigelEngine

Duke Nukem II: https://en.wikipedia.org/wiki/Duke_Nukem_II

Reddit:

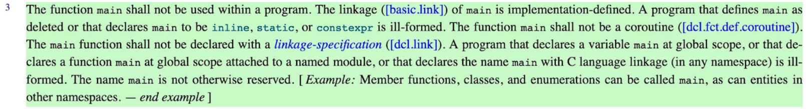
- https://www.reddit.com/r/cpp/comments/bubyrn/reimplementing_an_old_dos_game_in_c_17/
- https://www.reddit.com/r/programming/comments/buc3u4/reimplementing_an_old_dos_game_in_c_17/

Also: https://osgameclones.com/

Using **main** is undefined behaviour

Shafik Yaghmour:

```cpp
int main() {
    decltype(main()) x;
    return static_cast<bool>(&main);
}
```



3 The function `main` shall not be used within a program. The linkage ([basic.link]) of `main` is implementation-defined. A program that defines `main` as deleted or that declares `main` to be `inline`, `static`, or `constexpr` is ill-formed. The function `main` shall not be a coroutine ([dcl.fct.def.coroutine]). The `main` function shall not be declared with a *linkage-specification* ([dcl.link]). A program that declares a variable `main` at global scope, or that declares a function `main` at global scope attached to a named module, or that declares the name `main` with C language linkage (in any namespace) is ill-formed. The name `main` is not otherwise reserved. [*Example:* Member functions, classes, and enumerations can be called `main`, as can entities in other namespaces. — *end example*]

Attempting to modify a const object is undefined behaviour

Shafik Yaghmour:

```
int b() {
    const int x=1;
    int* p = const_cast<int*>(&x); // OK
    *p = 2;                        // UB
    return *p;
}
```

C++17 STL Parallel Algorithms - with GCC 9.1 and Intel TBB on Linux and macOS

https://solarianprogrammer.com/2019/05/09/cpp-17-stl-parallel-algorithms-gcc-intel-tbb-linux-macos/

Threading Building Blocks (TBB) https://www.threadingbuildingblocks.org/

GitHub: https://github.com/intel/tbb (Apache 2.0)

> *Since 2018 U5 TBB binary packages include Parallel STL as a high-level component.*

Parallel STL: https://github.com/intel/parallelstl (Apache 2.0)

Quote

Melinda Varian:

> *The best programs are the ones written when the programmer is supposed to be working on something else.*