Template Shenanigans

Developing template code

Agenda

- Compiling
- Debugging
- Testing
- Benchmarking

Who am I?

- Jonathan O'Connor he/him
- I develop code for The Cluster Company GmbH
- C++20 in-memory database
- Metaprogramming in Basic, Lisp, C++, Java and Ruby since 1983
- Chess, making ice cream
- github/ninkibah, @ninkibah

Agenda

- Compiling
- Debugging
- Testing
- Benchmarking

Error messages

- Famous for being long
- Too many messages

Error messages

```
1 #include <set>
2 void main() {
3   std::set<int> uniqueInts;
4   uniqueInts.insert(42);
5 }
```

Error messages

```
#include <set>
void main() {
  std::set<int, std::allocator<int>> uniqueInts;
  uniqueInts.insert(42);
}
```

Error messages

```
/opt/compiler-explorer/gcc-10.2.0/include/c++/10.2.0/bits/stl_tree.h: In instantiation of 'std::pair<std::_Rb_tree_node_bas
/opt/compiler-explorer/gcc-10.2.0/include/c++/10.2.0/bits/stl_tree.h:2154:4:   required from 'std::pair<std::_Rb_tree_itera
/opt/compiler-explorer/gcc-10.2.0/include/c++/10.2.0/bits/stl_set.h:521:25:   required from 'std::pair<typename std::_Rb_tr
<source>:4:24:   required from here
/opt/compiler-explorer/gcc-10.2.0/include/c++/10.2.0/bits/stl_tree.h:2101:35: error: no match for call to '(std::allocator<
 2101 |       __comp = _M_impl._M_key_compare(__k, _S_key(__x));
      |                ~~~~~~~~~~~~~~~~~~~~~^~~~~~~~~~~~~~~~~~
/opt/compiler-explorer/gcc-10.2.0/include/c++/10.2.0/bits/stl_tree.h:2112:33: error: no match for call to '(std::allocator<
 2112 |       if (_M_impl._M_key_compare(_S_key(__j._M_node), __k))
      |           ~~~~~~~~~~~~~~~~~~~~~^~~~~~~~~~~~~~~~~~~~~~~~~~
/opt/compiler-explorer/gcc-10.2.0/include/c++/10.2.0/bits/stl_tree.h: In instantiation of 'std::_Rb_tree<_Key, _Val, _KeyOf
/opt/compiler-explorer/gcc-10.2.0/include/c++/10.2.0/bits/stl_tree.h:2159:26:   required from 'std::pair<std::_Rb_tree_iter
/opt/compiler-explorer/gcc-10.2.0/include/c++/10.2.0/bits/stl_set.h:521:25:   required from 'std::pair<typename std::_Rb_tr
<source>:4:24:   required from here
/opt/compiler-explorer/gcc-10.2.0/include/c++/10.2.0/bits/stl_tree.h:1813:35: error: no match for call to '(std::allocator<
 1813 |           || _M_impl._M_key_compare(_KeyOfValue()(__v),
      |              ~~~~~~~~~~~~~~~~~~~~~^~~~~~~~~~~~~~~~~~~~
```

Read to the end

```
std::pair<std::_Rb_tree_node_base*, std::_Rb_tree_node_base*>
std::_Rb_tree<_Key, _Val, _KeyOfValue, _Compare, _Alloc>::
  _M_get_insert_unique_pos(const key_type&)
[with _Key = int;
 _Val = int;
 _KeyOfValue = std::_Identity<int>;
 _Compare = std::allocator<int>;
 _Alloc = std::allocator<int>;
 std::_Rb_tree<_Key, _Val, _KeyOfValue, _Compare,
_Alloc>::key_type = int]
```

Format the error message

- Suggested by Ben Deane
- He uses ClangFormat
- Copy declaration to cpp file
- Format the copied text

Format the error message

```
1 std::pair<typename std::_Rb_tree<_Key, _Key,
  std::_Identity<_Tp>, _Compare,
2                          typename
  __gnu_cxx::__alloc_traits<_Alloc>::
3                       rebind<_Key>::other>::const_iterator,
4           bool>
5 std::set<_Key, _Compare, _Alloc>::insert(
6     std::set<_Key, _Compare, _Alloc>::value_type &&);
```

Simplify the error message

- https://github.com/SuperV1234/camomilla
- Python3 script from Vittorio Romeo
- Replaces namespaces and generic templates with shorter versions.
- Can't replace std::basic_string<char> with string

Simplify the error message

```
$ echo "metavector<metatype<metawhatever<int>>>::x()" \
  | camomilla -d0
# outputs
metavector<?>::x()
```

Simplify the error message

```
$ echo "std::vector<std::pair<std::int16_t, std::int32_t>>" \
  | camomilla --depth=100
# outputs
vector<pair<int16_t, int32_t>>
```

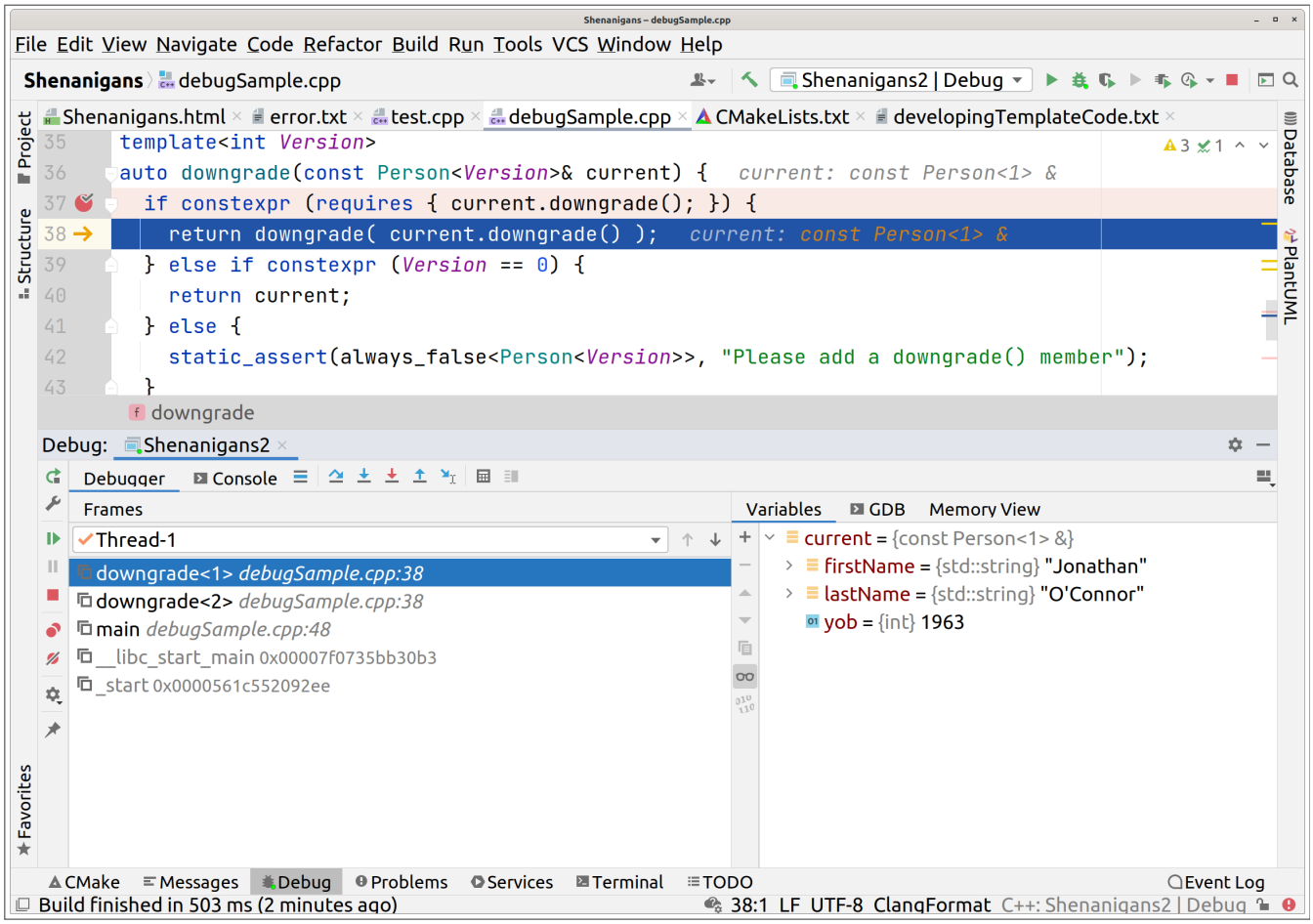
Agenda

- Compiling
- Debugging
- Testing
- Benchmarking

Template debugging

- Using the normal debugger
- Printing a type name
- Template Instantiation tools

Constexpr function debugging



What type do I have?

- Can I use typeid(T).name()?
- ??? NSt7__cxx1112basic_stringIcSt11char_traitsIcESaIcEEE
- __FUNCTION__
- doesn't show templated arguments
- __PRETTY_FUNCTION__
- works on gcc and clang, but not MSVC

__PRETTY_FUNCTION__

```
1 void print_me(int) {
2   cout << __PRETTY_FUNCTION__ '\n';
3 }
```

void print_me(int)

__PRETTY_FUNCTION__

```
1 template<typename T>
2 void print_me(T* = nullptr) {
3   cout << __PRETTY_FUNCTION__ '\n';
4 }
5 print_me<int>();
```

void print_me(T *) [with T = int]

function to print the type

```
template<typename T>
auto name_of_T(T* = nullptr) {
  std::string pretty_func = __PRETTY_FUNCTION__;
  auto pos = pretty_func.find('=');
  if (pos == std::string::npos) {
      return pretty_func;
  }
  return pretty_func.substr(pos + 2, pretty_func.length() - pos - 3);
}
```

function to print the type

```
cout << name_of_T<std::string>() << '\n';
// gcc prints
//   std::__cxx11::basic_string<char>
// clang prints
//   std::__cxx11::basic_string<char,
//     std::char_traits<char>, std::allocator<char>>
```

Print types at compile time

- Ivan Čukić mentions a nice trick to print a deduced type at compile time
- Template declaration, no definition
- Instantiation causes an error
- Downside: Generates an error and stops compilation

Print types at compile time

```
1 template<typename T>
2 class print_me_as_error;
3
4 print_me_as_error<std::vector<int>::element_type>();
```

```
1 error: invalid use of incomplete type
2 'class print_me_as_error<int>'
```

Print types at compile time

- Daniel Frey (of TAOCPP fame) uses another version of this technique
- constexpr value marked as deprecated!
- To instantiate it, use static_assert
- Generates warning and continues compiling

Print types at compile time

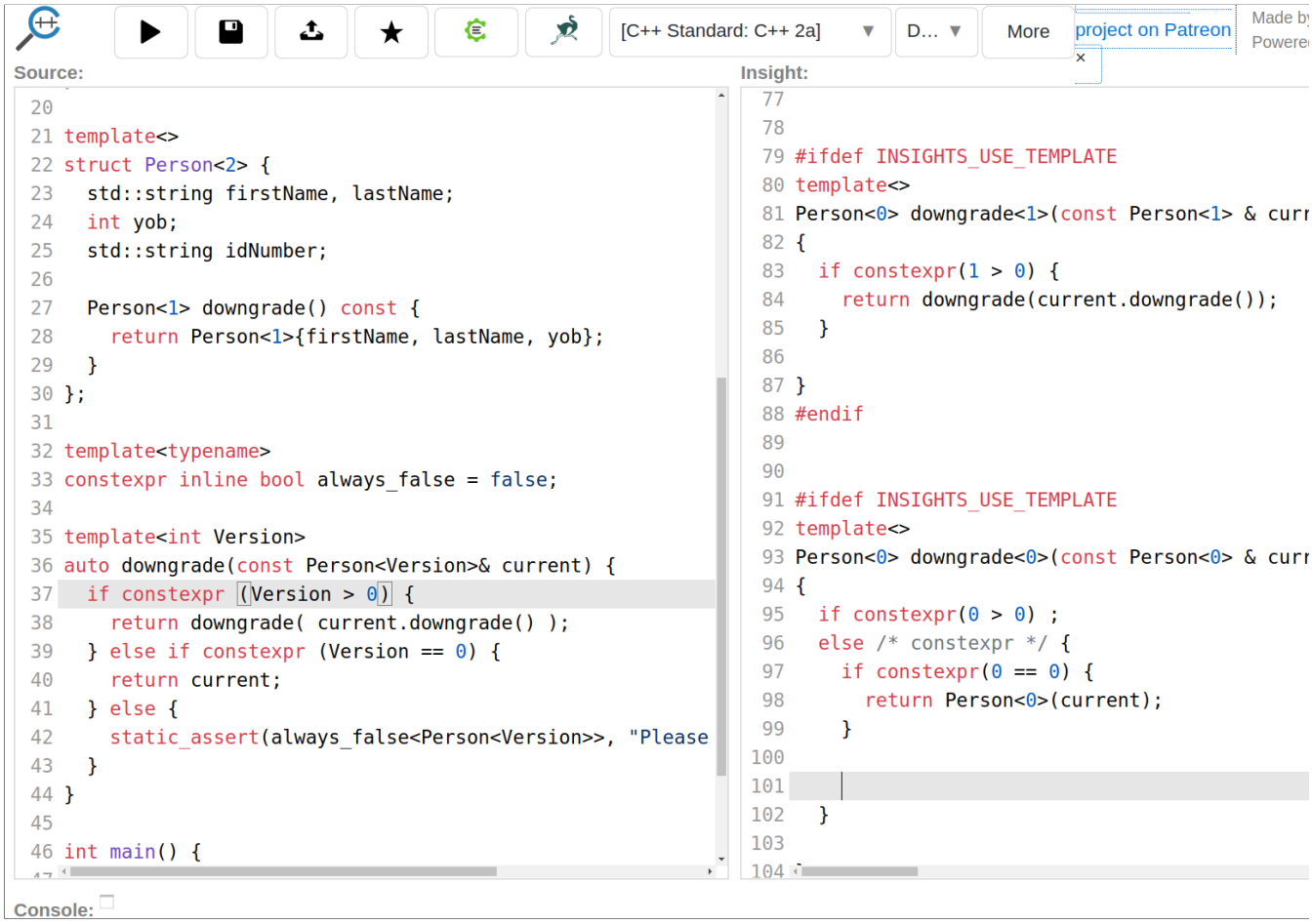
```
1 template<typename T>
2 [[deprecated]] constexpr bool print_type = true;
3
4 static_assert(print_type<std::vector<int>::element_type>);
```

```
1 warning: 'print_type<int>' is deprecated
2          [-Wdeprecated-declarations]
3 4 | static_assert(printType<std::vector<int>::value_type>);
```

Code Insights

- https://cppinsights.io/
- Brainchild of Andreas Fertig
- Like godbolt.org
- Generates simple C++ instead of assembler
- Shows range-based for loops
- Shows template instantiations

Code Insights



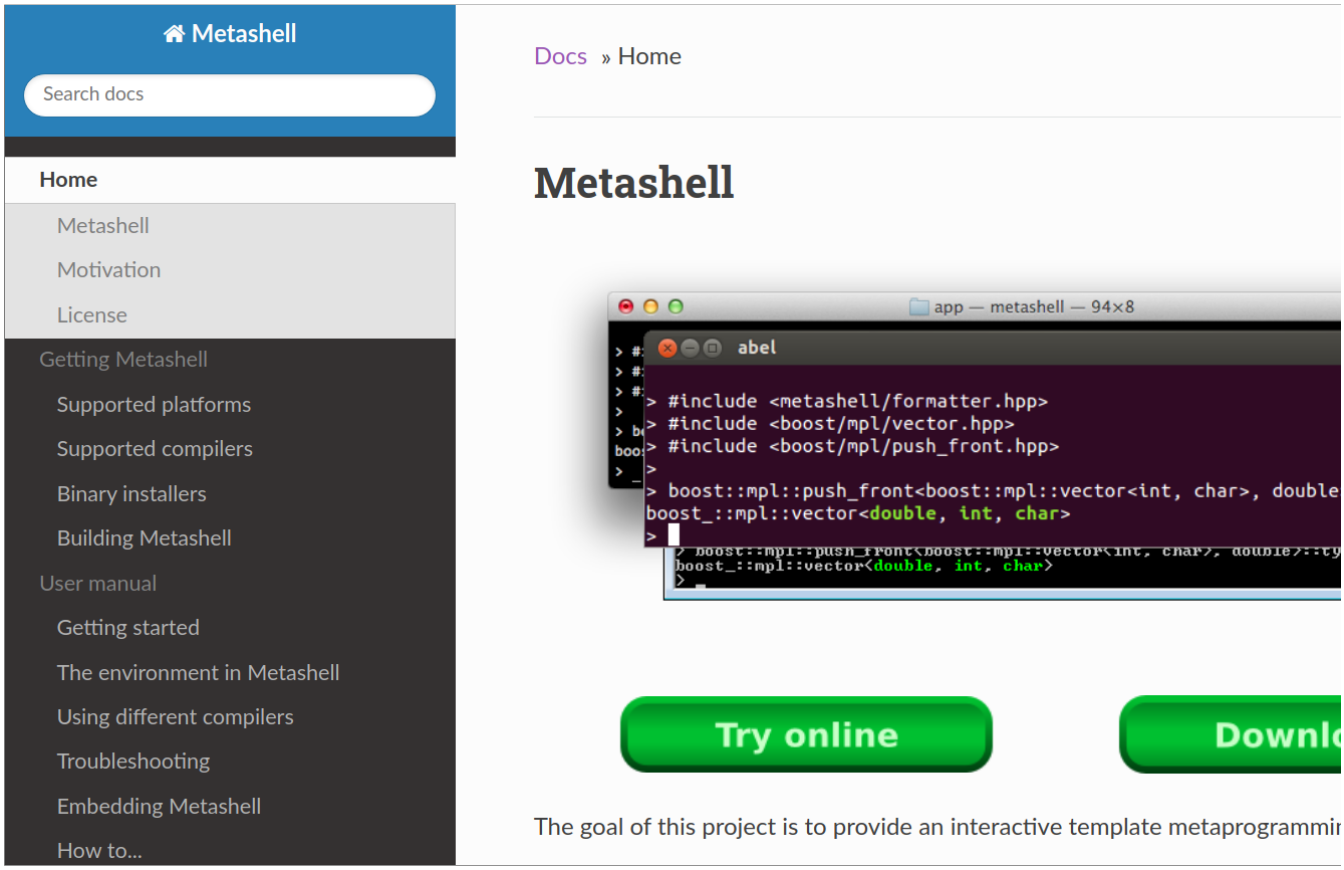

```
20
21 template<>
22 struct Person<2> {
23   std::string firstName, lastName;
24   int yob;
25   std::string idNumber;
26
27   Person<1> downgrade() const {
28     return Person<1>{firstName, lastName, yob};
29   }
30 };
31
32 template<typename>
33 constexpr inline bool always_false = false;
34
35 template<int Version>
36 auto downgrade(const Person<Version>& current) {
37   if constexpr (Version > 0) {
38     return downgrade( current.downgrade() );
39   } else if constexpr (Version == 0) {
40     return current;
41   } else {
42     static_assert(always_false<Person<Version>>, "Please
43   }
44 }
45
46 int main() {
```

```
77
78
79 #ifdef INSIGHTS_USE_TEMPLATE
80 template<>
81 Person<0> downgrade<1>(const Person<1> & curr
82 {
83   if constexpr(1 > 0) {
84     return downgrade(current.downgrade());
85   }
86
87 }
88 #endif
89
90
91 #ifdef INSIGHTS_USE_TEMPLATE
92 template<>
93 Person<0> downgrade<0>(const Person<0> & curr
94 {
95   if constexpr(0 > 0) ;
96   else /* constexpr */ {
97     if constexpr(0 == 0) {
98       return Person<0>(current);
99     }
100
101
102   }
103
104
```

Metashell

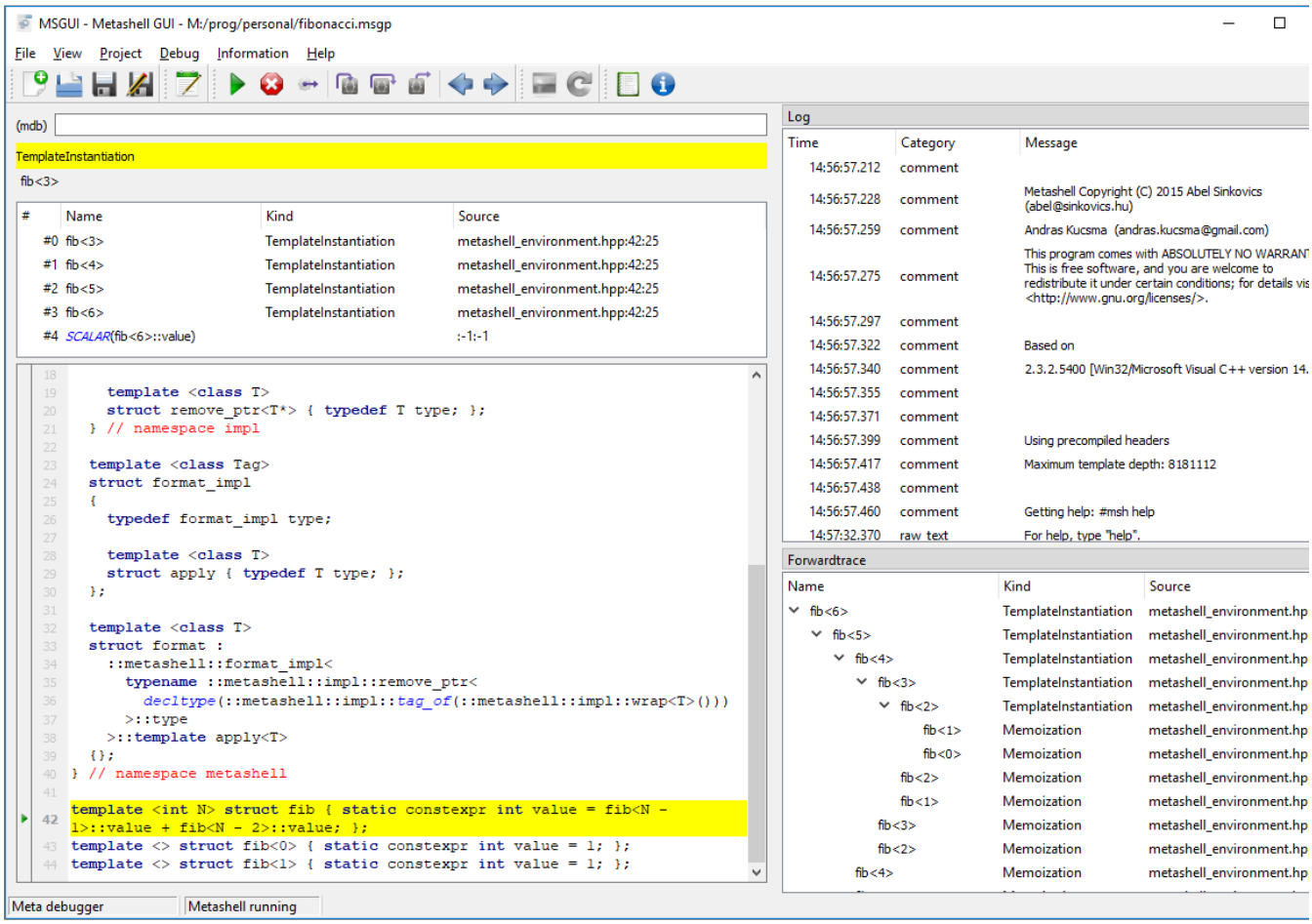
- http://metashell.org
- A Meta-debugger!
- Complicated to set up
- Please someone make a web front end

Metashell


Metashell
Search docs
Home
Metashell
Motivation
License
Getting Metashell
Supported platforms
Supported compilers
Binary installers
Building Metashell
User manual
Getting started
The environment in Metashell
Using different compilers
Troubleshooting
Embedding Metashell
How to...
Docs » Home
Metashell
app — metashell — 94×8
abel
> #include <metashell/formatter.hpp>
> #include <boost/mpl/vector.hpp>
> #include <boost/mpl/push_front.hpp>
> boost::mpl::push_front<boost::mpl::vector<int, char>, double
boost_::mpl::vector<double, int, char>
Try online
The goal of this project is to provide an interactive template metaprogrammi

MSGUI

- https://github.com/RangelReale/msgui
- Qt frontend for Metashell
- Looks cool


MSGUI - Metashell GUI - M:/prog/personal/fibonacci.msgp
File View Project Debug Information Help
(mdb)
TemplateInstantiation
fib<3>
Name Kind Source
#0 fib<3> TemplateInstantiation metashell_environment.hpp:42:25
#1 fib<4> TemplateInstantiation metashell_environment.hpp:42:25
#2 fib<5> TemplateInstantiation metashell_environment.hpp:42:25
#3 fib<6> TemplateInstantiation metashell_environment.hpp:42:25
#4 SCALAR(fib<6>::value) :-1:-1
template <class T>
struct remove_ptr<T*> { typedef T type; };
} // namespace impl
template <class Tag>
struct format_impl
{
typedef format_impl type;
template <class T>
struct apply { typedef T type; };
};
template <class T>
struct format :
::metashell::format_impl<
typename ::metashell::impl::remove_ptr<
decltype(::metashell::impl::tag_of(::metashell::impl::wrap<T>()))
>::type
>::template apply<T>
{};
} // namespace metashell
template <int N> struct fib { static constexpr int value = fib<N - 1>::value + fib<N - 2>::value; };
template <> struct fib<0> { static constexpr int value = 1; };
template <> struct fib<1> { static constexpr int value = 1; };
Meta debugger
Metashell running
Log
Time Category Message
14:56:57.212 comment
14:56:57.228 comment Metashell Copyright (C) 2015 Abel Sinkovics (abel@sinkovics.hu)
14:56:57.259 comment Andras Kucsma (andras.kucsma@gmail.com)
14:56:57.275 comment This program comes with ABSOLUTELY NO WARRANT
This is free software, and you are welcome to redistribute it under certain conditions; for details vis
<http://www.gnu.org/licenses/>.
14:56:57.297 comment
14:56:57.322 comment Based on
14:56:57.340 comment 2.3.2.5400 [Win32/Microsoft Visual C++ version 14.
14:56:57.355 comment
14:56:57.371 comment
14:56:57.399 comment Using precompiled headers
14:56:57.417 comment Maximum template depth: 8181112
14:56:57.438 comment
14:56:57.460 comment Getting help: #msh help
14:57:32.370 raw_text For help, type "help".
Forwardtrace
Name Kind Source
fib<6> TemplateInstantiation metashell_environment.hp
fib<5> TemplateInstantiation metashell_environment.hp
fib<4> TemplateInstantiation metashell_environment.hp
fib<3> TemplateInstantiation metashell_environment.hp
fib<2> TemplateInstantiation metashell_environment.hp
fib<1> Memoization metashell_environment.hp
fib<0> Memoization metashell_environment.hp
fib<2> Memoization metashell_environment.hp
fib<1> Memoization metashell_environment.hp
fib<3> Memoization metashell_environment.hp
fib<2> Memoization metashell_environment.hp
fib<4> Memoization metashell_environment.hp

Agenda

- Compiling
- Debugging
- Testing
- Benchmarking

Run-time tests

```
template<typename T>
requires std::integral<T>
constexpr T double_it(T x) {
  return 2 * x;
}

TEST_CASE( "my template test" ) {
  REQUIRE( double_it(3) == 6 );
}
```

Compile-time tests

- Pre-2000 The Boost Static Assertion Library
- 2002 Proposed by Robert Klarer and Dr. John Maddock
- C++11 introduced static_assert keyword
- static_assert(constexpr bool, "Error text")
- Generates an error if the bool expression evaluates to false
- C++17 made the error text optional

```
1 static_assert( false, "Woops!!!" );
```

static_assert in code

```
template<typename T>
requires std::integral<T>
constexpr T double_it(T x) {
  return 2 * x;
}

static_assert( double_it(3) == 6,
  "double_it should multiply by 2" );
```

When to test at compile-time?

- To ensure a type keeps certain properties
- Often in if constexpr statements
- Generates an error closer to point of call
- Can reduce size of error messages

Versioned data

```
template<int> struct Person;

template<>
struct Person<0> {
  std::string firstName, lastName;
};

template<>
struct Person<1> {
  std::string firstName, lastName;
  int yob;
  Person<0> migrateDown() const;
};
```

Versioned data

```
template<int N>
inline auto downgradeOnce (const Person<N>& newer) {
  if constexpr (N == 0) {
      return newer;
  } else if constexpr ( requires { newer.migrateDown(); }) {
      return newer.migrateDown();
  } else {
      static_assert(false,
        "newer needs a migrateDown member function");
  }
}
```

Versioned data

```
template<int N>
inline auto downgradeOnce (const Person<N>& newer) {
  if constexpr (N == 0) {
      return newer;
  } else if constexpr ( requires { newer.migrateDown(); }) {
      return newer.migrateDown();
  } else {
      static_assert(always_false<Person<N>>,
        "The Person version of newer needs a migrateDown
member function");
  }
}
```

always_false

https://en.cppreference.com/w/cpp/utility/variant/visit

```
1 template<typename T>
2 constexpr inline bool always_false = false;
```

Archetypes

- Test classes that contain minimal behaviour
- Described in Chapter 28 of C++ Templates book
- Mentioned by David Abrahams and Douglas Gregor
- http://www.generic-programming.org/languages/cpp/techniques.html#archetypes
- Not quite superceded by C++20 concepts

Archetypes

- Blog by Andrzej Krzemieński
- https://akrzemi1.wordpress.com/2020/09/02/concept-archetypes
- https://akrzemi1.wordpress.com/2020/09/10/concept-archetypes-update
- https://akrzemi1.wordpress.com/2020/10/26/semantic-requirements-in-concepts

Andrzej's C++ blog

Guidelines and thoughts about

Home About C++ Glossary Examples

← Ordering by constraints

Concept archetypes — update →

Concept archetypes

Posted on September 2, 2020

Concepts in the form added in C++20 used to be called *lite*. This is because they do not provide one quite important functionality: having the compiler check if the author of a constrained template is only using operations and types allowed by the constraining concept. In other words, we can say that our template only requires operations *A* and *B* to be valid, but we can still use some other operations inside and this is fine with the compiler. In this post we will show how this is problematic, even for programmers aware of the issue, and how to address it with *concept archetypes*.

Se

Subscribe

RSS - Posts

RSS - Comments

Follow Andrzej's C++ blog

Recent Posts

- Semantic requirements in co
- Reflection for aggregates
- Concept archetypes — upda
- Concept archetypes
- Ordering by constraints

Andrzej's C++ blog

Guidelines and thoughts about C++

Home About C++ Glossary Examples

← Concept archetypes

Reflection for aggregates →

Concept archetypes — update

Posted on September 10, 2020

An observant reader indicated that in the previous post where I was trying to implement a concept archetype — a type with minimal interface that models a given concept — I actually failed. This deserves a closer examination.

Let's recall. We have the following concept:

```
1  template <typename T>
2  concept Addable =
3    std::regular<T> &&
4    std::totally_ordered<T> &&
5    requires(T x, T y) {
6      { x += y } -> std::same_as<T&>;
7      { x +  y } -> std::convertible_to<T>;
8    };
```

Search

Subscribe

RSS - Posts
RSS - Comments

Follow Andrzej's C++ blog

Recent Posts

- Semantic requirements in concepts
- Reflection for aggregates
- Concept archetypes — update
- Concept archetypes
- Ordering by constraints

Archives

Select Month

Blogroll

- // info
- Alf on programming

Andrzej's C++ blog

Guidelines and thought

Home About C++ Glossary Examples

← Reflection for aggregates

Semantic requirements in concepts

Posted on October 26, 2020

The word ‘concept’ in the context of C++ generic programming has two meanings. The first is more abstract: it is the notion from the domain of Generic Programming (GP) in general. GP is not tied to any specific language: it is an approach to writing programs, and concepts are part of this approach. In this sense concepts have been with us since the inception of the STL. The second meaning is the keyword `concept` in C++20 with its associated semantics: its goal is to approximate the more abstract notion of a concept from GP, and this works only to some extent. One notable

Subscribe

RSS - Posts
RSS - Comments

Follow Andrzej's C

Recent Posts

- Semantic requireme
- Reflection for aggre
- Concept archetypes
- Concept archetypes

Archetypes for testing

- Tiny classes
- Delete functionality

Creating Archetypes

```
1 template<typename T>
2 int find(T const* array, int n, T const& value) {
3   int i = 0;
4   while ( i != n && array[i] != value)
5     ++i;
6   return i;
7 }
```

Creating Archetypes

```
class EqualityComparableArchetype {
};

bool
operator==(EqualityComparableArchetype const&,
           EqualityComparableArchetype const&);
```

Creating Archetypes

```
1  class EqualityComparableArchetype {
2  };
3
4  struct ConvertibleToBoolArchetype {
5    operator bool() const;
6  };
7
8  ConvertibleToBoolArchetype
9  operator==(EqualityComparableArchetype const&,
10            EqualityComparableArchetype const&);
```

Creating Archetypes

```
1 template int find(
2     EqualityComparableArchetype const* array, int n,
3     EqualityComparableArchetype const& value);
```

Creating Archetypes

```
In instantiation of 'int find(const T*, int, const T&) [with T
= EqualityComparableArchetype]':
21:45:   required from here
4:32: error: return type of 'ConvertibleToBoolArchetype
operator==(const EqualityComparableArchetype&, const
EqualityComparableArchetype&)' is not 'bool'
4 |     while ( i != n && array[i] != value)
|                       ~~~~~~~~~^~~~~~~~
4:32: note: used as rewritten candidate for comparison of
'const EqualityComparableArchetype' and 'const
EqualityComparableArchetype'
```

Creating Archetypes

```
1 template<typename T>
2 int find(T const* array, int n, T const& value) {
3   int i = 0;
4   while ( i != n && !(array[i] == value))
5     ++i;
6   return i;
7 }
```

Agenda

- Compiling
- Debugging
- Testing
- Benchmarking

Benchmarking

- Optimizing template code
- Comparing build times

Tracers

- Described in chapter 28 of C++ Templates book
- Can test quality of implementation
- Count calls to particular functions
- Found in libstdc++ test code

Tracers

```
1 struct TracedItem {
2   inline static int def_ctors = 0;
3   inline static int copy_ctors = 0;
4   inline static int move_ctors = 0;
5   TracedItem() { ++def_ctors; }
6   TracedItem(const TracedItem&) { ++copy_ctors; }
7   TracedItem(TracedItem&&) noexcept { ++move_ctors; }
8 };
```

Tracers

```
int main() {
  std::vector<TracedItem> items;
  items.emplace_back();             // 1 def
  items.push_back(TracedItem{});    // 1 def + 1 move
  items.emplace_back(TracedItem{}); // 1 def + 1 move

  std::cout << "default ctors " << TracedItem::def_ctors
            << "\ncopy ctors " << TracedItem::copy_ctors
            << "\nmove ctors " << TracedItem::move_ctors <<
'\n';
  return 0;
}
```

No copy ctors were harmed in making this example!

Comparing build times

- Compiler flags
- Microsoft Visual Studio Build Insights
- build-bench.com
- metaBench tool

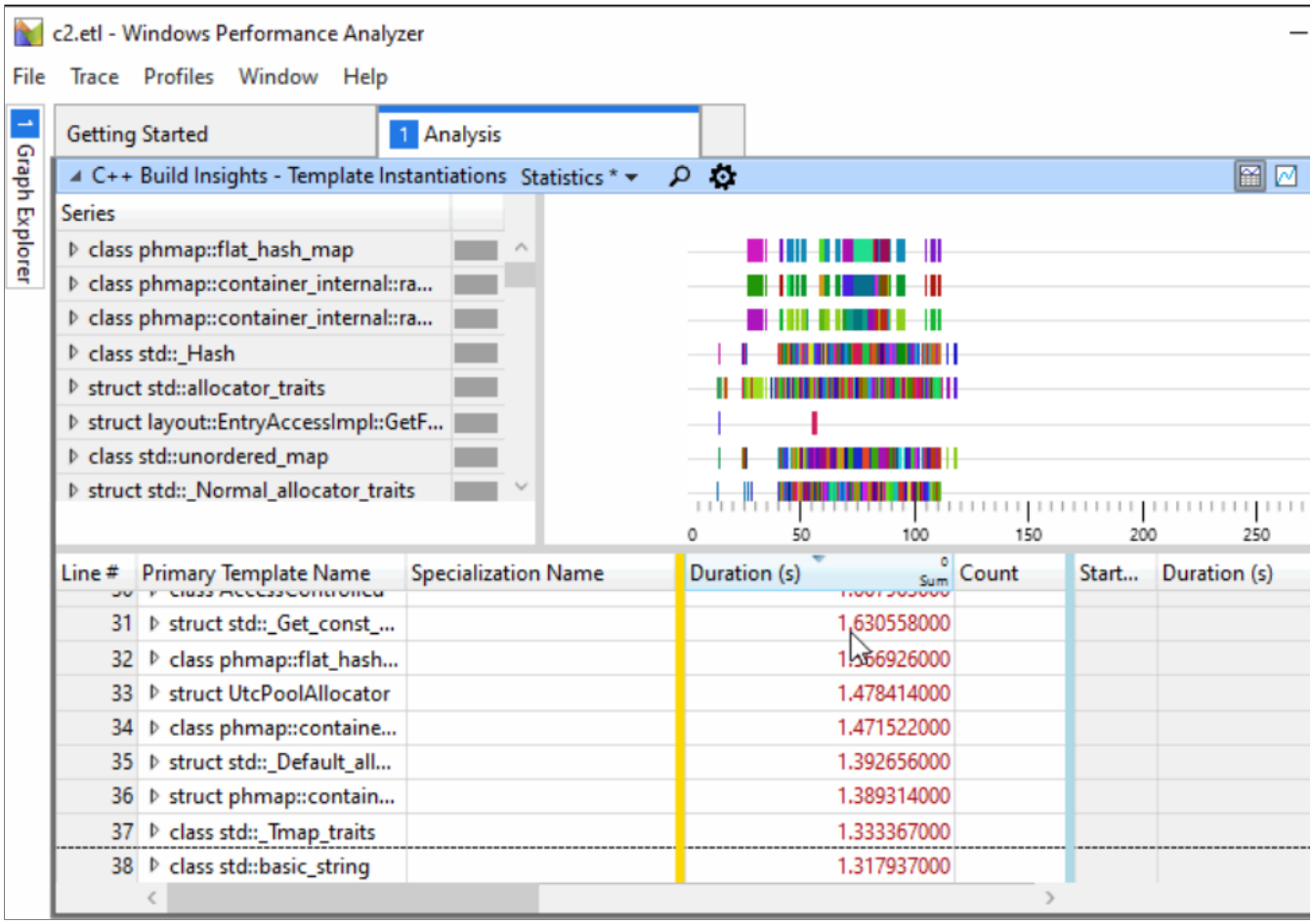
gcc & Clang -ftime-report

```
Time variable                   usr               GGC
phase setup                : 0.00 (  0%)     1505 kB (  2%)
phase parsing              : 0.59 ( 83%)    63979 kB ( 81%)
phase lang. deferred       : 0.06 (  8%)     7268 kB (  9%)
phase opt and generate     : 0.06 (  8%)     6136 kB (  8%)
|name lookup               : 0.10 ( 14%)     2262 kB (  3%)
|overload resolution       : 0.17 ( 24%)    19631 kB ( 25%)
callgraph construction     : 0.01 (  1%)     2209 kB (  3%)
callgraph optimization     : 0.01 (  1%)        0 kB (  0%)
callgraph ipa passes       : 0.01 (  1%)      328 kB (  0%)
preprocessing              : 0.11 ( 15%)     1504 kB (  2%)
template instantiation     : 0.25 ( 35%)    25293 kB ( 32%)
constant expr. evaluation  : 0.01 (  1%)      113 kB (  0%)
constraint satisfaction    : 0.02 (  3%)     2570 kB (  3%)
TOTAL                      : 0.71           79097 kB
```

gcc -fmem-report

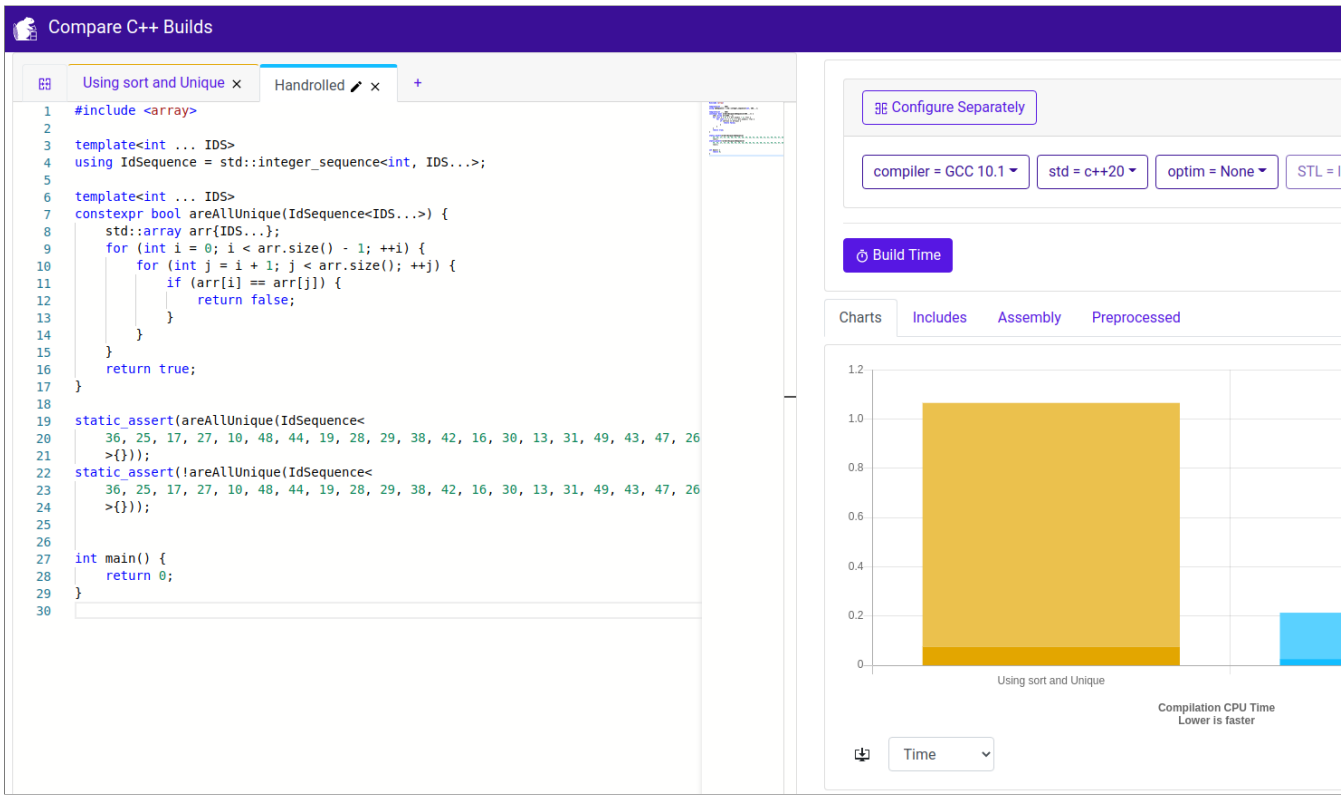
```
Number of expanded macros:                       6059
Average number of tokens per macro expansion:       3

Line Table allocations during the compilation process
Number of ordinary maps used:          500
Ordinary map used size:                 11k
Number of ordinary maps allocated:     682
Ordinary maps allocated size:           15k
Number of macro maps used:            5965
Macro maps used size:                  186k
Macro maps locations size:             183k
Macro maps size:                       369k
Duplicated maps locations size:         75k
Total allocated maps size:             455k
Total used maps size:                  381k
Ad-hoc table size:                     768k
Ad-hoc table entries used:              21k
optimized_ranges:                      276k
unoptimized_ranges:                     23k
```

MS Build Insights



build-bench.com

- Created by Fred Tingaud
- Like godbolt.org
- Add 2 or more versions of an application
- Graphs compilation time and process size

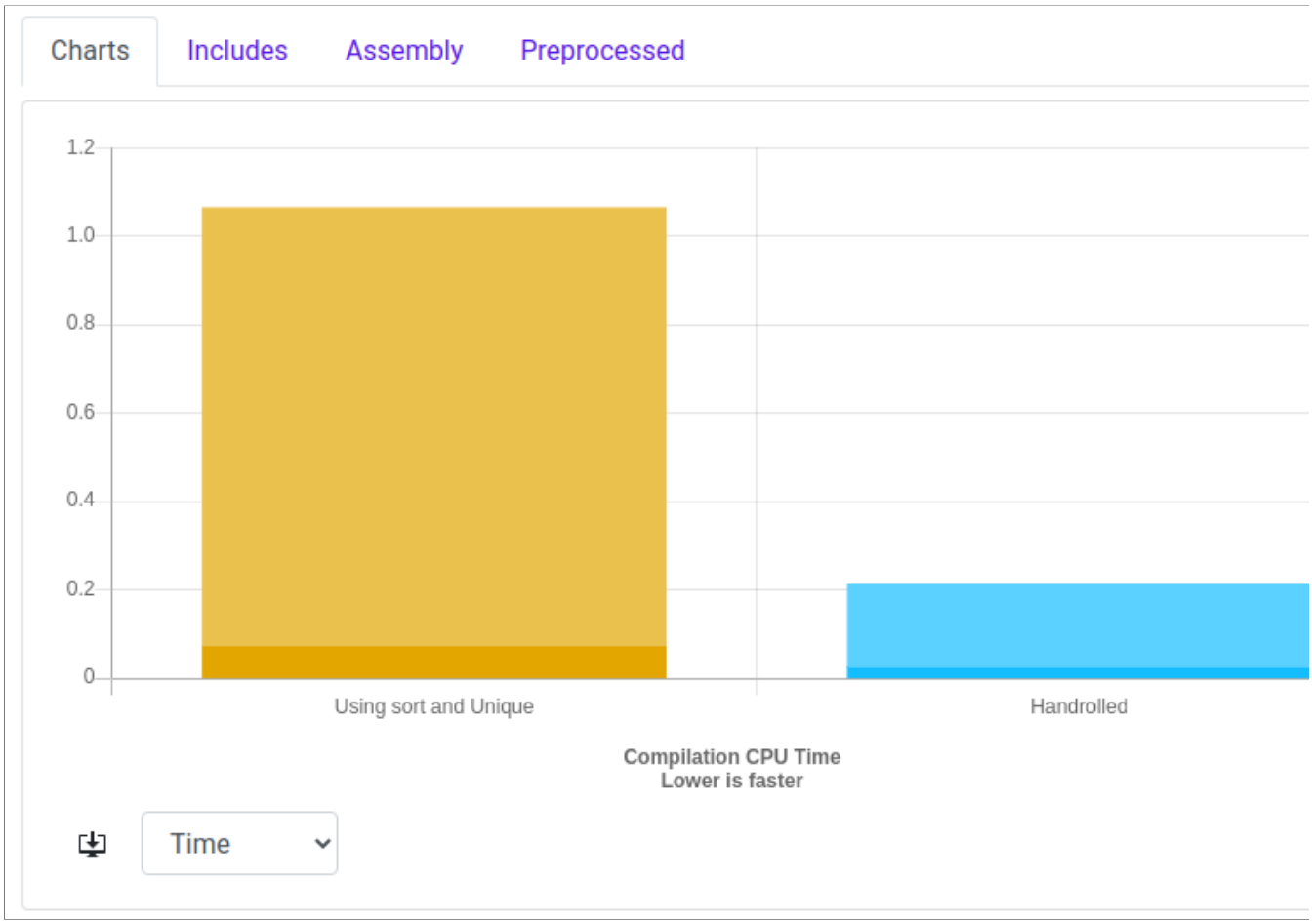

Compare C++ Builds
Using sort and Unique
Handrolled
#include <array>
template<int ... IDS>
using IdSequence = std::integer_sequence<int, IDS...>;
template<int ... IDS>
constexpr bool areAllUnique(IdSequence<IDS...>) {
std::array arr{IDS...};
for (int i = 0; i < arr.size() - 1; ++i) {
for (int j = i + 1; j < arr.size(); ++j) {
if (arr[i] == arr[j]) {
return false;
}
}
}
return true;
}
static_assert(areAllUnique(IdSequence<
36, 25, 17, 27, 10, 48, 44, 19, 28, 29, 38, 42, 16, 30, 13, 31, 49, 43, 47, 26
>{}));
static_assert(!areAllUnique(IdSequence<
36, 25, 17, 27, 10, 48, 44, 19, 28, 29, 38, 42, 16, 30, 13, 31, 49, 43, 47, 26
>{}));
int main() {
return 0;
}
Configure Separately
compiler = GCC 10.1
std = c++20
optim = None
Build Time
Charts
Includes
Assembly
Preprocessed
1.2
1.0
0.8
0.6
0.4
0.2
0
Using sort and Unique
Compilation CPU Time
Lower is faster
Time

Using sort and Unique | Handrolled

```cpp
#include <array>
#include <algorithm>

template<int ... IDS>
using IdSequence = std::integer_sequence<int, IDS...>;

template<int ... IDS>
constexpr bool areAllUnique(IdSequence<IDS...>) {
    std::array arr{IDS...};
    std::sort(arr.begin(), arr.end());
    return std::unique(arr.begin(), arr.end()) == arr.end();
}

static_assert(areAllUnique(IdSequence<
    36, 25, 17, 27, 10, 48, 44, 19, 28, 29, 38, 42, 16, 30, 13, 31, 49, 43, 4
    >{}));
static_assert(!areAllUnique(IdSequence<
    36, 25, 17, 27, 10, 48, 44, 19, 28, 29, 38, 42, 16, 30, 13, 31, 49, 43, 4
    >{}));

int main() {
    return 0;
}
```

Using sort and Unique × Handrolled × +

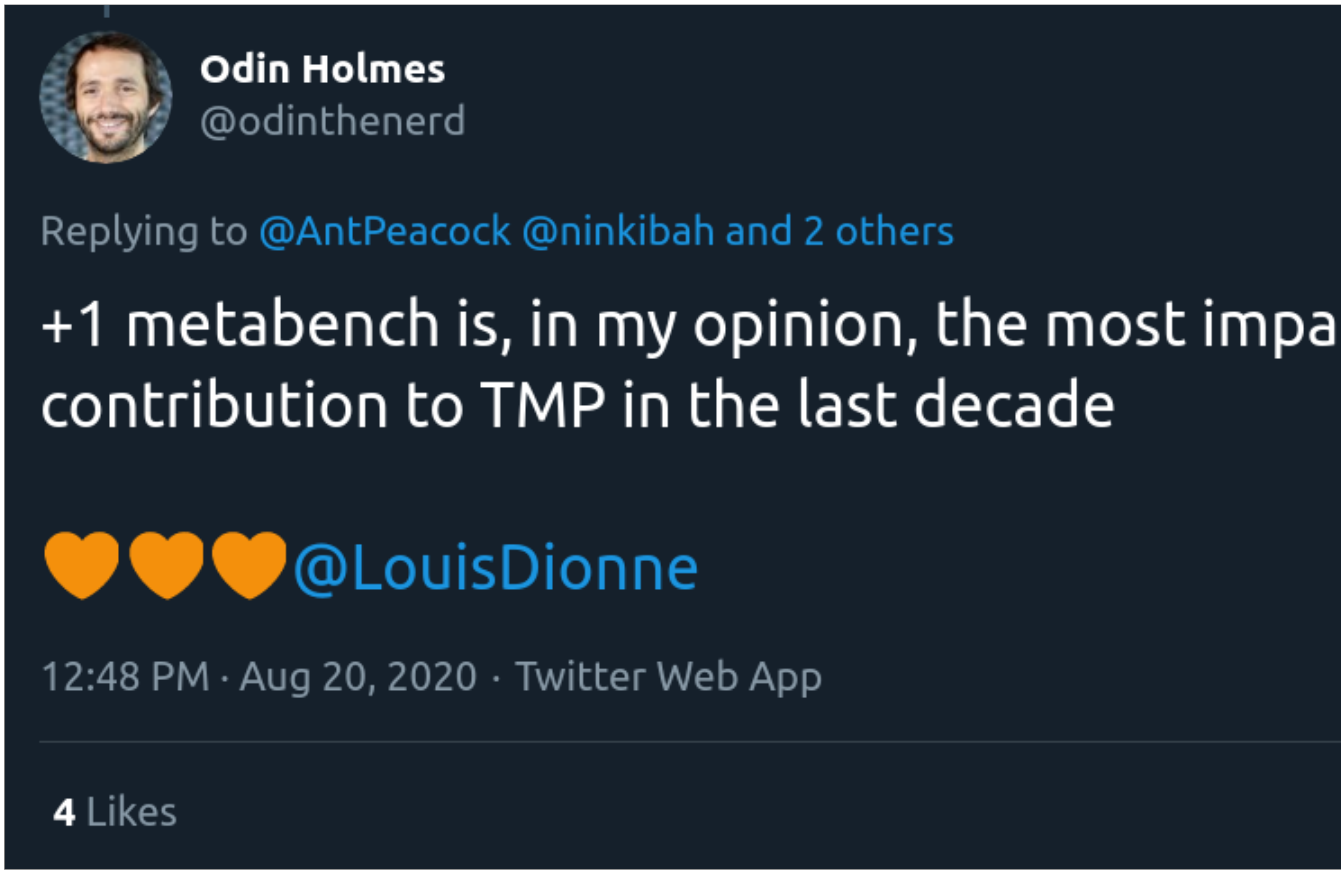
```cpp
#include <array>

template<int ... IDS>
using IdSequence = std::integer_sequence<int, IDS...>;

template<int ... IDS>
constexpr bool areAllUnique(IdSequence<IDS...>) {
    std::array arr{IDS...};
    for (int i = 0; i < arr.size() - 1; ++i) {
        for (int j = i + 1; j < arr.size(); ++j) {
            if (arr[i] == arr[j]) {
                return false;
            }
        }
    }
    return true;
}

static_assert(areAllUnique(IdSequence<
    36, 25, 17, 27, 10, 48, 44, 19, 28, 29, 38, 42, 16, 30, 13, 31, 49, 43, 47,
    >{}));
static_assert(!areAllUnique(IdSequence<
    36, 25, 17, 27, 10, 48, 44, 19, 28, 29, 38, 42, 16, 30, 13, 31, 49, 43, 47,
    >{}));


int main() {
    return 0;
}

```


Charts
Includes
Assembly
Preprocessed
1.2
1.0
0.8
0.6
0.4
0.2
0
Using sort and Unique
Handrolled
Compilation CPU Time
Lower is faster
Time

Metabench

- https://github.com/ldionne/metabench
- Times versions of your program
- Uses ruby

Metabench


Odin Holmes
@odinthenerd
Replying to @AntPeacock @ninkibah and 2 others
+1 metabench is, in my opinion, the most impa
contribution to TMP in the last decade
@LouisDionne
12:48 PM · Aug 20, 2020 · Twitter Web App
4 Likes

Metabench CMakeLists.txt

```
# Make sure Metabench can be found when writing include(metabench)
list(APPEND CMAKE_MODULE_PATH "../metabench")

# Actually include the module
include(../metabench/metabench.cmake)

# Add new datasets
metabench_add_dataset(sortUnique "sortUnique.cpp.erb" "(1..40).to_a")
metabench_add_dataset(handRolled "handRolled.cpp.erb" "(1..40).to_a")

# Add a new chart
metabench_add_chart(chart DATASETS sortUnique handRolled)
```

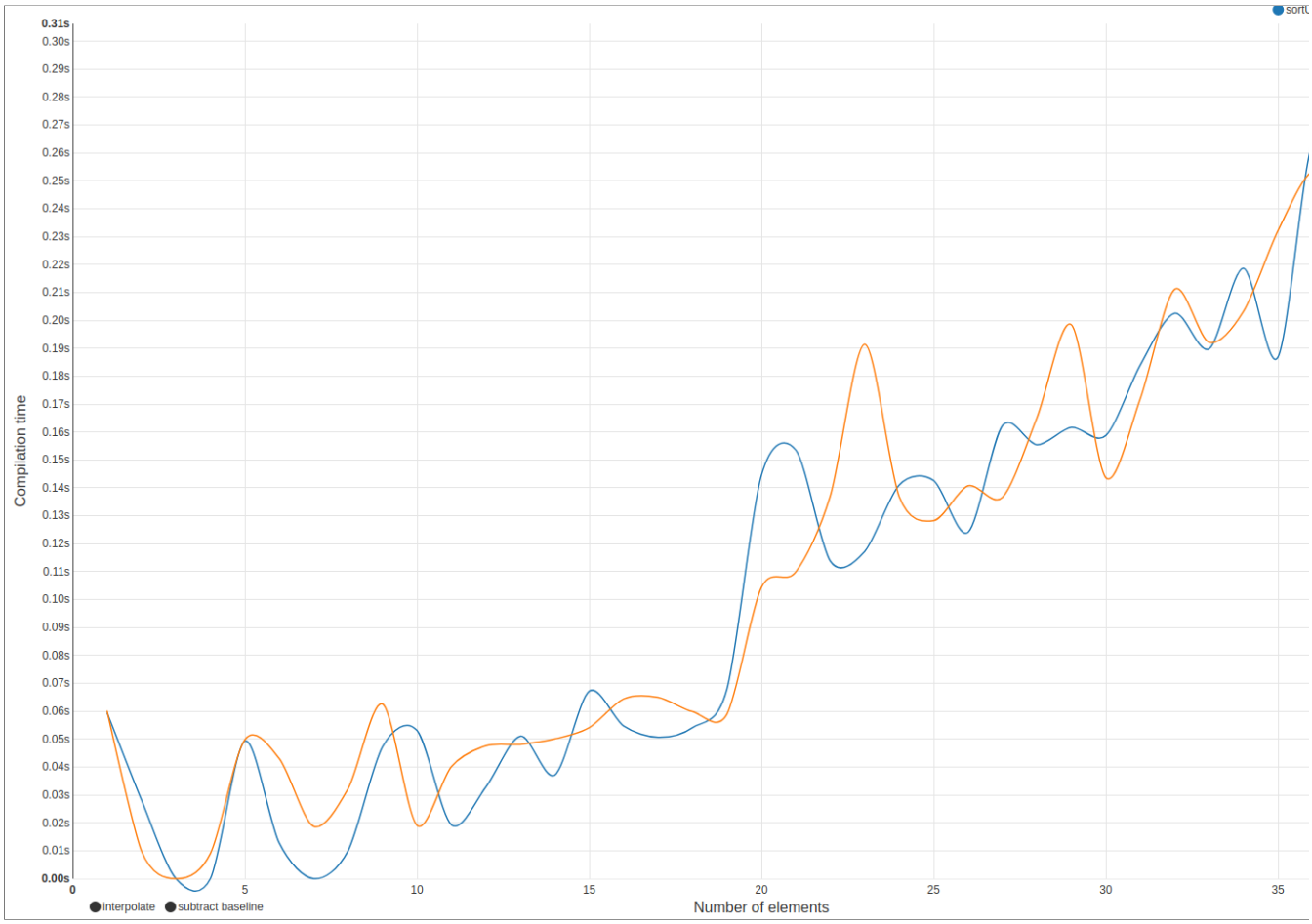
eRuby files

```
#if defined(METABENCH)
static auto tup1 = std::make_tuple(
  <%= (1..n).to_a.reverse.join(', ') %>
);
#endif
```

eRuby files

```
#if defined(METABENCH)
static auto tup1 = std::make_tuple(
  10, 9, 8, 7, 6, 5, 4, 3, 2, 1
);
#endif
```

```
/home/joconnor/bin/clion-203.4818.55/bin/cmake/linux/bin/cmake --build /
[ 33%] Generating handRolled.json
[ 66%] Generating sortUnique.json
==> 2% (0.0s) dataset1.cpp.erb (n = 1)/home/joconnor/.rvm/rubies/ruby-2.
==> 2% (0.0s) handrolled.cpp.erb (n = 1)/home/joconnor/.rvm/rubies/ruby-
==> 100% (30.1s) handrolled.cpp.erb (n = 40)
==> 100% (30.16s) dataset1.cpp.erb (n = 40)
Generating chart.html
Built target chart
```


sort
0.31s
0.30s
0.29s
0.28s
0.27s
0.26s
0.25s
0.24s
0.23s
0.22s
0.21s
0.20s
0.19s
0.18s
0.17s
0.16s
0.15s
0.14s
0.13s
0.12s
0.11s
0.10s
0.09s
0.08s
0.07s
0.06s
0.05s
0.04s
0.03s
0.02s
0.01s
0.00s
Compilation time
0
5
10
15
20
25
30
35
Number of elements
interpolate
subtract baseline

Conclusions: Compiling

- Read the error message
- Get your editor to format the declaration
- Use Camomilla to simplify the error messages

Conclusions: Debugging

- The normal debugger can handle constexpr functions
- Print a deduced type using deprecated trick
- Code Insights is a great tool
- Meta-debugging with Metashell and MSGUI

Conclusions: Testing

- Unit tests still work
- static_assert
- Remember the always_false template trick
- Archtypes can test templates and concepts

Conclusions: Benchmarking

- Tracer types to count function calls
- Compile-time Benchmarking
- build-bench.com
- metabench CMake module

PS

- Stack Overflow
- #include discord server
- godbolt.org

The End