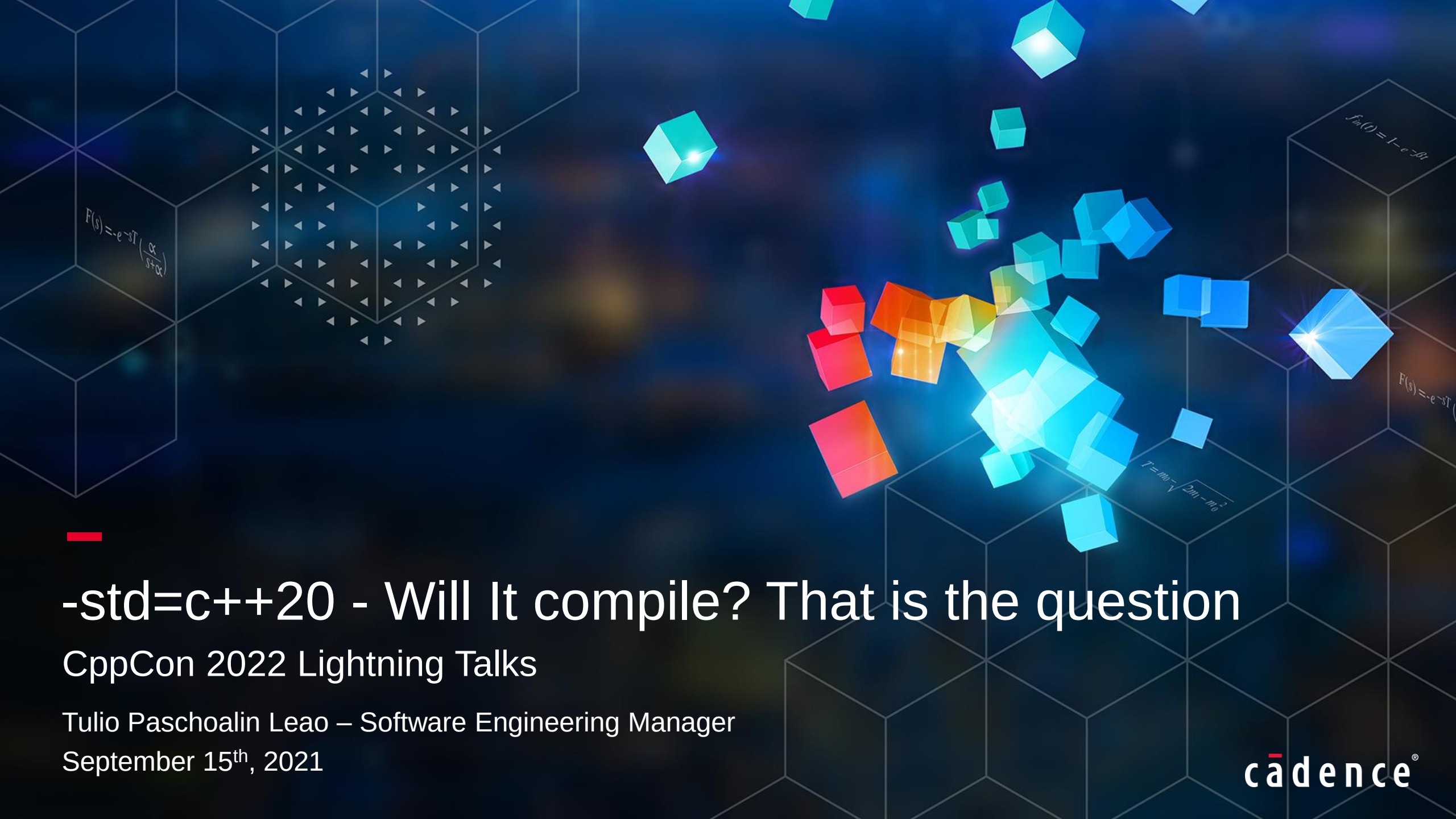


-std=c++20 - Will It compile? That is the question

CppCon 2022 Lightning Talks

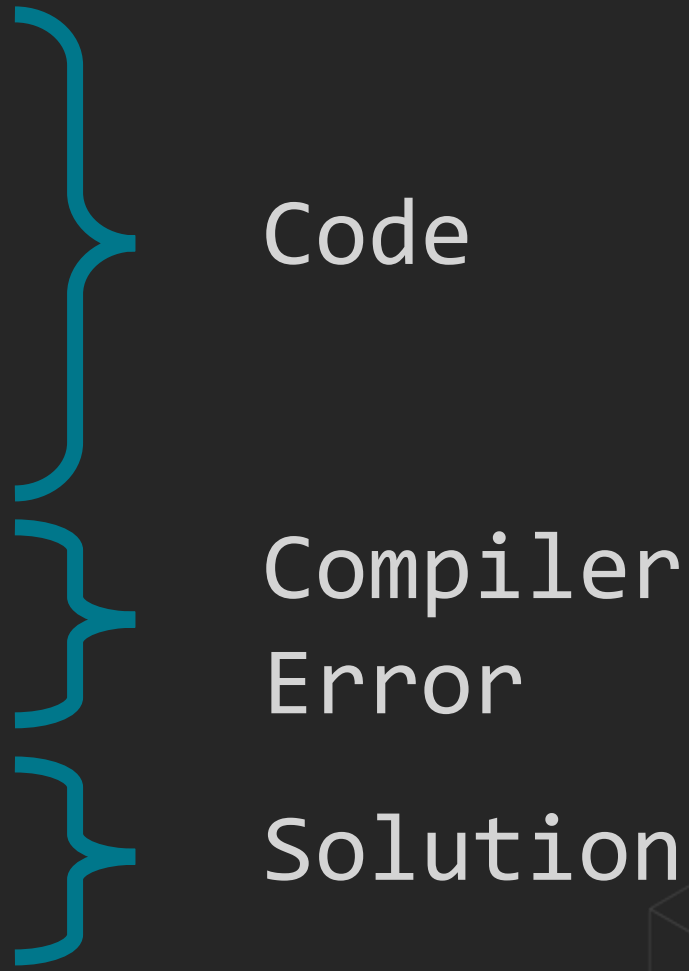
Tulio Paschoalin Leao – Software Engineering Manager

September 15th, 2021

cadence®

Issue Summary

https://godbolt.org/z/runnable_link



Using C++20 reserved keywords as names (vars, params)

https://godbolt.org/z/Pbjxq4611

```
int main()
{
    int requires = 3;
    int concept = 5;
}
```

```
<source>: In function 'int main()':
<source>:3:9: error: expected unqualified-id before 'requires'
    3 |     int requires = 3;
      |         ^~~~~~~~
<source>:4:9: warning: C++20 concept definition syntax is 'concept <name> = <expr>'
    4 |     int concept = 5;
      |         ^~~~~~~
<source>:4:17: error: expected unqualified-id before '=' token
    4 |     int concept = 5;
      |                 ^
```

Solution(s): Rename those variables ASAP!

Using identifiers reserved for the standard library

https://godbolt.org/z/95vaM7rYq

```
#define _TR 3

#include <algorithm>
```

```
/opt/compiler-explorer/gcc-12.1.0/include/c++/12.1.0/bits/ranges_algo.h: In lambda function:
<source>:1:13: error: expected nested-name-specifier before numeric constant
    1 | #define _TR 3
      |             ^
```

Solution(s): Rename those identifiers ASAP!

Incompatibility between string and u8string on std::fs::path API

https://godbolt.org/z/1564PqG8T

```
#include <filesystem>
#include <string>

int main()
{
    std::string a = std::filesystem::path("").u8string();
}
```

```
<source>: In function 'int main()':
<source>:6:55: error: conversion from 'basic_string<char8_t>' to non-scalar type 'basic_string<char>' requested
    6 |     std::string a = std::filesystem::path("").u8string();
      |                     ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~^~
```

Solution(s): Either change use of u8string() or string variable to u8string

Redundant/unallowed template-id on constructors and destructors

https://godbolt.org/z/oKzG144Yr

```
template<typename T>
struct Foo
{
    Foo<T>();
    ~Foo<T>();
};
```

```
<source>:3:12: error: expected unqualified-id before ')' token
   3 |     Foo<T>();
     |            ^
<source>:4:5: error: template-id not allowed for destructor
   4 |     ~Foo<T>();
     |     ^
```

Solution(s): Remove redundant <T> on the constructor and destructor definition

Aggregate Initialization of structs with deleted default constructors

https://godbolt.org/z/Wa4Kb3sjG

```
struct Foo
{
    Foo() = delete;
    int value;
};
int main()
{
    Foo f = {3};
}
```

```
<source>: In function 'int main()':
<source>:8:15: error: could not convert '{3}' from '<brace-enclosed initializer list>' to 'Foo'
    8 |     Foo f = {3};
      |             ^
      |             |
      |             <brace-enclosed initializer list>
```

Solution(s): Implement constructor for your class

Removed std::allocator members and functions

https://godbolt.org/z/bs6GbKPG4

```
#include <memory>

int main()
{
    std::allocator<int> allocator;
    int* a;
    allocator.construct(a, 3);
}
```

```
<source>: In function 'int main()':
<source>:7:15: error: 'class std::allocator<int>' has no member named 'construct'
    7 |     allocator.construct(a, 3);
      |               ^~~~~~~~~
```

Solution(s): Use *std::allocator_traits* alternatives

Bonus C++17 incompatible solution(s): Use *std::construct_at* and *std::destroy_at*

Change in std::accumulate handling of BinaryOperation (1/2)

https://godbolt.org/z/KrY9vvsYr

```
#include <numeric>
#include <vector>

int main()
{
    std::vector<int> a {1,2,3};
    return std::accumulate(a.begin(), a.end(), 0, [](auto& b, auto& c)
        { return b + 2*c; });
}
```

Change in std::accumulate handling of BinaryOperation (2/2)

https://godbolt.org/z/KrY9vvsYr

```
In file included from /opt/compiler-explorer/gcc-12.1.0/include/c++/12.1.0/numeric:62,
                 from <source>:1:
/opt/compiler-explorer/gcc-12.1.0/include/c++/12.1.0/bits/stl_numeric.h: In instantiation of 'constexpr _Tp
std::accumulate(_InputIterator, _InputIterator, _Tp, _BinaryOperation) [with _InputIterator = __gnu_cxx::__normal_iterator<int*,
vector<int> >; _Tp = int; _BinaryOperation = main()::<lambda(auto:3&, auto:4&)>]':
<source>:7:27:   required from here
/opt/compiler-explorer/gcc-12.1.0/include/c++/12.1.0/bits/stl_numeric.h:169:29: error: no match for call to '(main()::<lambda(auto:3&,
auto:4&)>) (std::remove_reference<int&>::type, int&)'
  169 |         __init = __binary_op(_GLIBCXX_MOVE_IF_20(__init), *__first);
      |                  ~~~~~~~~~~~^~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
<source>:7:51: note: candidate: 'main()::<lambda(auto:3&, auto:4&)> [with auto:3 = int; auto:4 = int]' (near match)
    7 |     return std::accumulate(a.begin(), a.end(), 0, [](auto& b, auto& c) { return b + 2*c; });
      |                                                   ^
<source>:7:51: note:   conversion of argument 1 would be ill-formed:
/opt/compiler-explorer/gcc-12.1.0/include/c++/12.1.0/bits/stl_numeric.h:169:29: error: cannot bind non-const lvalue reference of type
'int&' to an rvalue of type 'std::remove_reference<int&>::type' {aka 'int'}
  169 |         __init = __binary_op(_GLIBCXX_MOVE_IF_20(__init), *__first);
      |                  ~~~~~~~~~~~^~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

Solution(s): Use *const&*, remove reference, stop using *std::accumulate*?

Tulio Paschoalin Leao | @tupaschoal