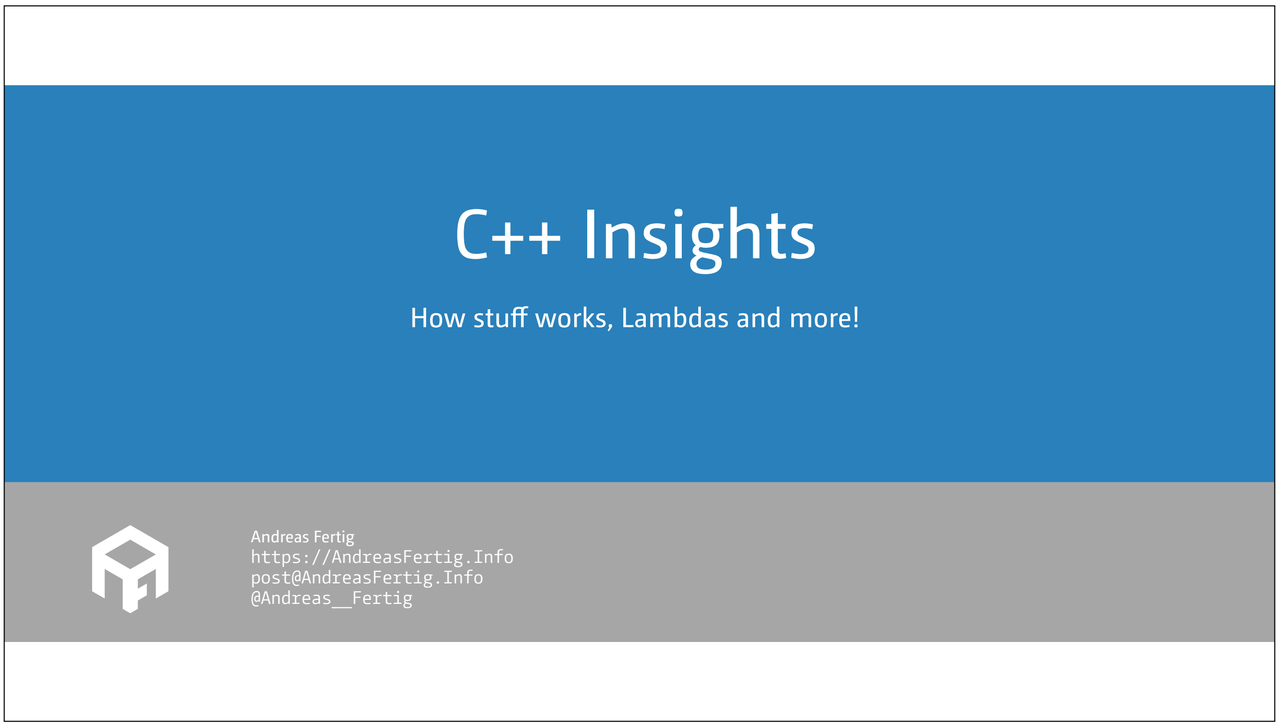


C++ Insights

How stuff works, Lambdas and more!

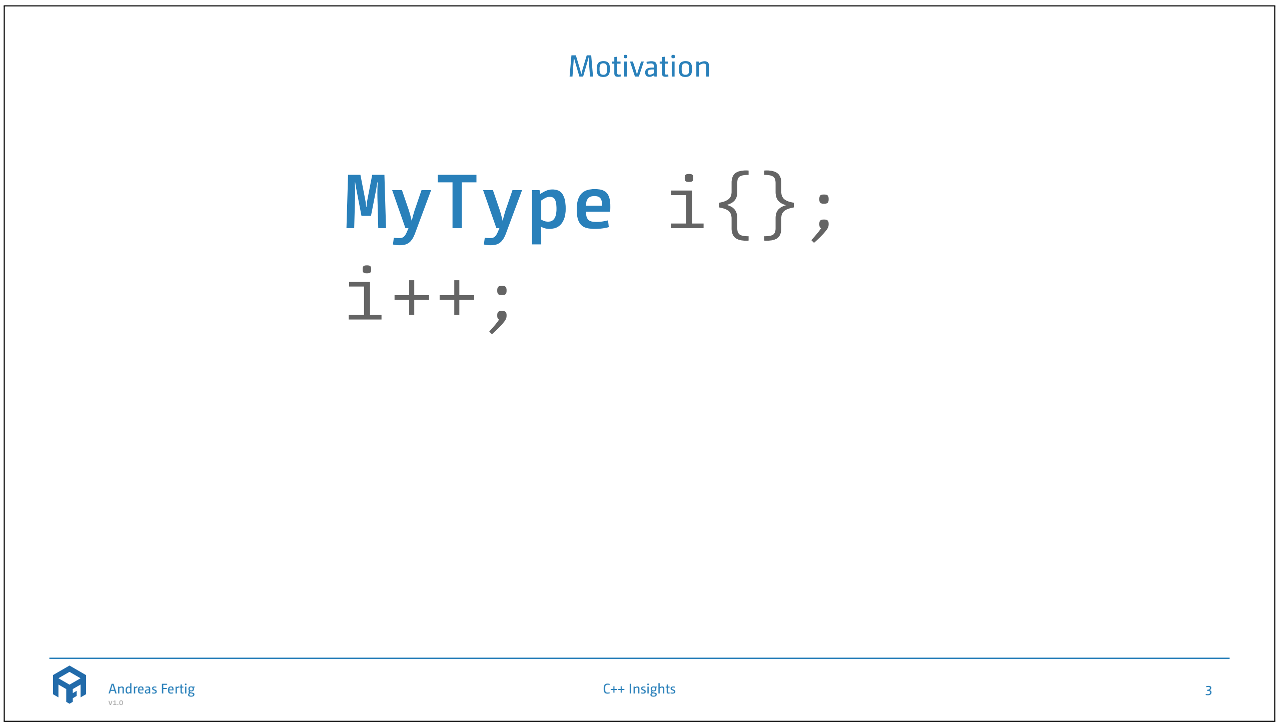
Andreas Fertig
https://AndreasFertig.Info
post@AndreasFertig.Info
@Andreas__Fertig

fertig

adjective /ˈfɛrtɪç/

finished
ready
complete
completed

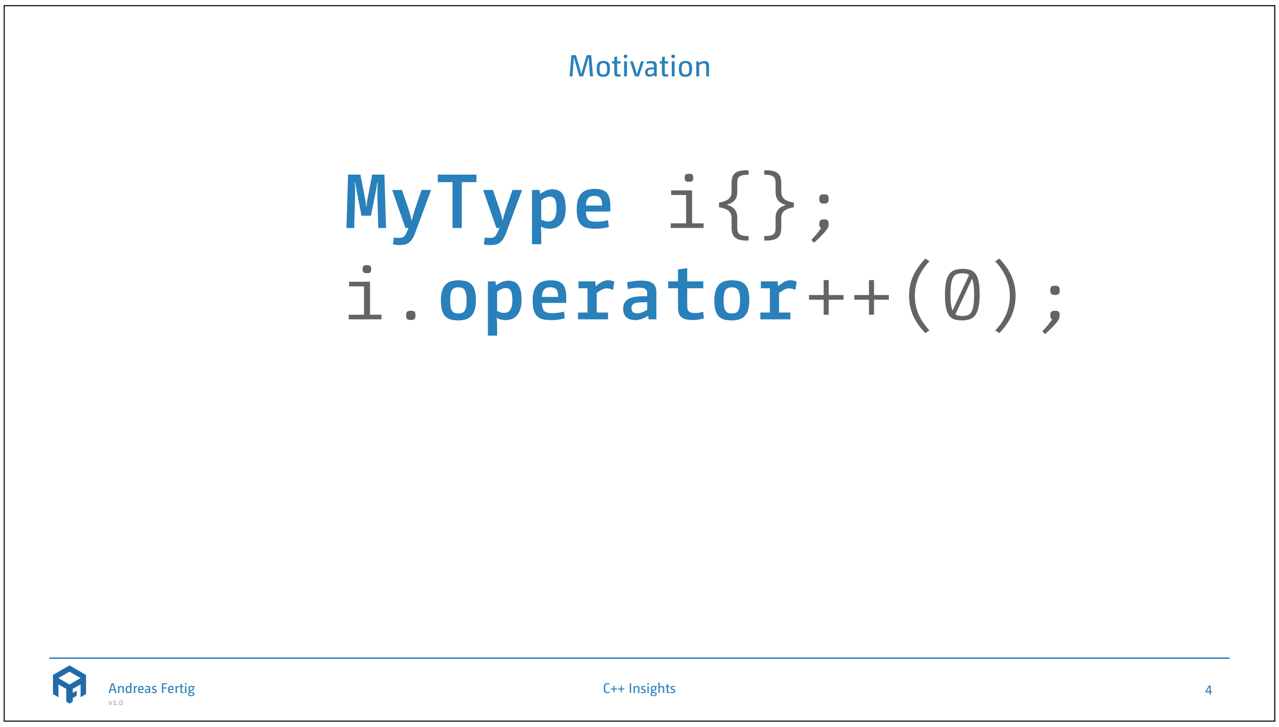
Andreas Fertig
v1.0
C++ Insights
2



Motivation

```
MyType i{};
i++;
```

Andreas Fertig v1.0 C++ Insights 3



Motivation

```
MyType i{};
i.operator++(0);
```

Andreas Fertig v1.0 C++ Insights 4

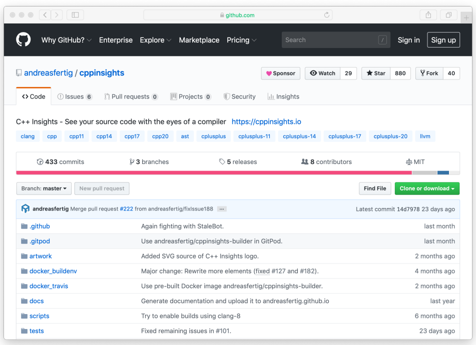
Implicit Conversions

```cpp

short int max(short int a, short int b)
{
  return (a > b) ? a : b;
}

void Main()
{
  short int          a = 1;
  unsigned short int b = 65’530;

  printf(”max: %d\n”, max(a, b));
}
```

C++ Insights

- Show what is going on.
- Make invisible things visible to assist in teaching.
- Create valid code.
- Create code that compiles.
- *Of course, it is open-source.*

https://github.com/andreasfertig/cppinsights/



andreasfertig / cppinsights

C++ Insights - See your source code with the eyes of a compiler https://cppinsights.io

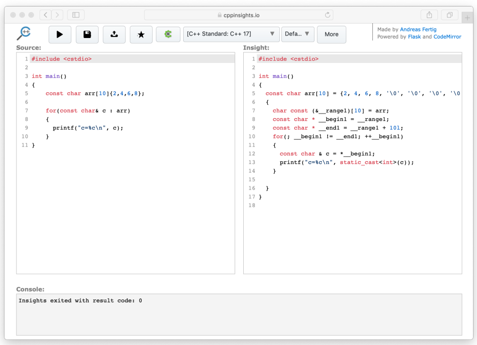
clang cpp cpp11 cpp14 cpp17 cpp20 ast cplusplus cplusplus-11 cplusplus-14 cplusplus-17 cplusplus-20 llvm

433 commits 3 branches 5 releases 8 contributors MIT

andreasfertig Merge pull request #222 from andreasfertig/fixIssue188 … Latest commit 14d7978 23 days ago

.github	Again fighting with StaleBot.	last month
.gitpod	Use andreasfertig/cppinsights-builder in GitPod.	last month
artwork	Added SVG source of C++ Insights logo.	2 months ago
docker_buildenv	Major change: Rewrite more elements (fixed #127 and #182).	4 months ago
docker_travis	Use pre-built Docker image andreasfertig/cppinsights-builder.	2 months ago
docs	Generate documentation and upload it to andreasfertig.github.io	last year
scripts	Try to enable builds using clang-8	6 months ago
tests	Fixed remaining issues in #101.	23 days ago

https://cppinsights.io



Source:

```cpp
#include <cstdio>

int main()
{
    const char arr[10]{2,4,6,8};

    for(const char& c : arr)
    {
      printf("c=%c\n", c);
    }
}
```

Insight:

```cpp
#include <cstdio>

int main()
{
  const char arr[10] = {2, 4, 6, 8, '\0', '\0', '\0', '\0
  {
    char const (&__range1)[10] = arr;
    const char * __begin1 = __range1;
    const char * __end1 = __range1 + 10l;
    for(; __begin1 != __end1; ++__begin1)
    {
      const char & c = *__begin1;
      printf("c=%c\n", static_cast<int>(c));
    }

  }
}

```

Console:

```
Insights exited with result code: 0
```

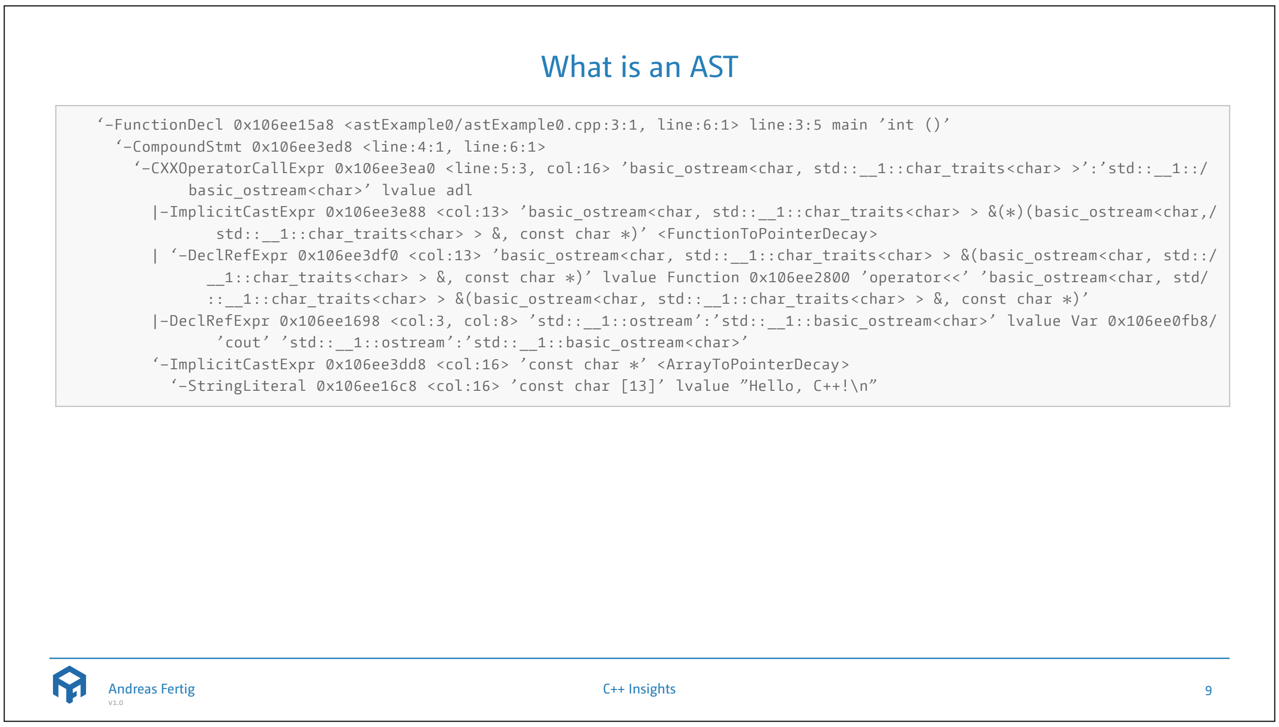
A word about limitations

- C++ Insights is a Clang based tool.
- The official builds use the latest release version of Clang.
  - Hence, not all the newest interesting features are available.
- It uses the Clang AST which shows no optimizations.
  - Hence, tuning with `-O n` does not change anything in C++ Insights.
- Not *all* statements are currently matched.

A word about limitations: Templates

- Creating code that compiles from templates is hard.
- To make it a bit easier for me there is a `#ifdef INSIGHTS_USE_TEMPLATE` to have the code, but inactive.

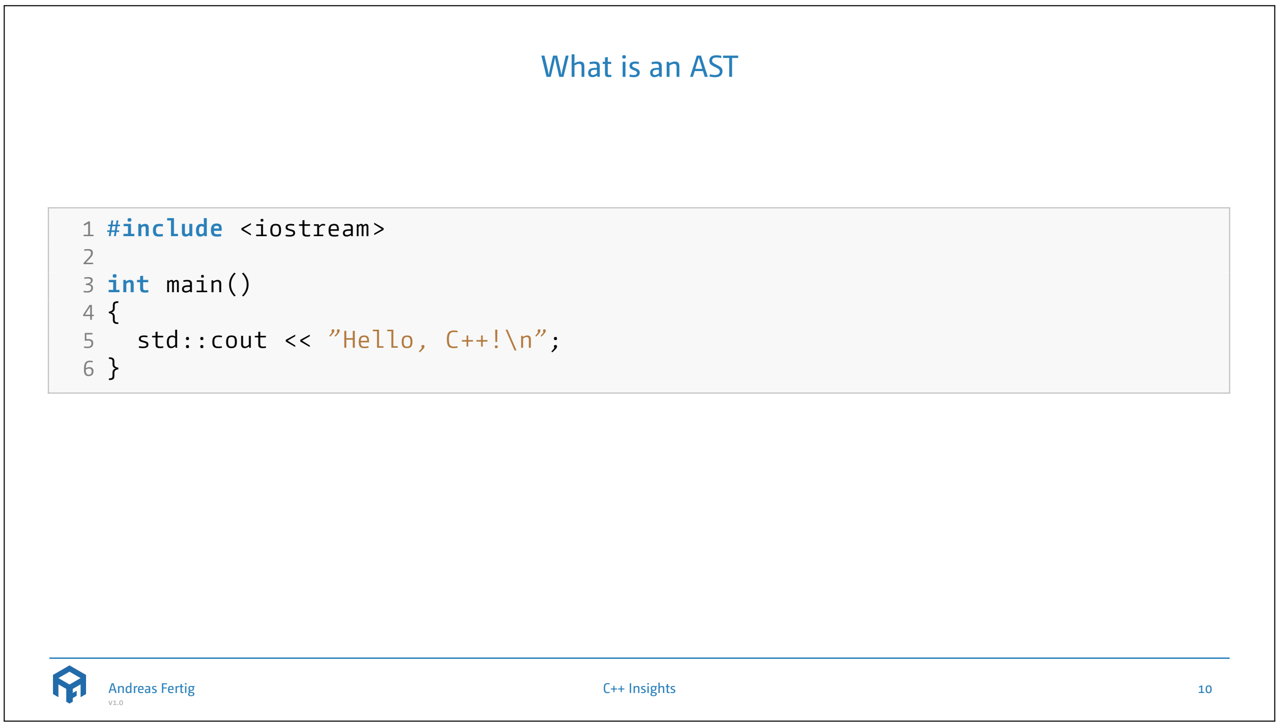
```
template<typename T>
void Func()
{}

class Demo
{
};

int main()
{
  Func<Demo>();
}
```



What is an AST

```
'-FunctionDecl 0x106ee15a8 <astExample0/astExample0.cpp:3:1, line:6:1> line:3:5 main 'int ()'
  '-CompoundStmt 0x106ee3ed8 <line:4:1, line:6:1>
    '-CXXOperatorCallExpr 0x106ee3ea0 <line:5:3, col:16> 'basic_ostream<char, std::__1::char_traits<char> >':'std::__1::/
          basic_ostream<char>' lvalue adl
      |-ImplicitCastExpr 0x106ee3e88 <col:13> 'basic_ostream<char, std::__1::char_traits<char> > &(*)(basic_ostream<char,/
             std::__1::char_traits<char> > &, const char *)' <FunctionToPointerDecay>
      | '-DeclRefExpr 0x106ee3df0 <col:13> 'basic_ostream<char, std::__1::char_traits<char> > &(basic_ostream<char, std::/
            __1::char_traits<char> > &, const char *)' lvalue Function 0x106ee2800 'operator<<' 'basic_ostream<char, std/
            ::__1::char_traits<char> > &(basic_ostream<char, std::__1::char_traits<char> > &, const char *)'
      |-DeclRefExpr 0x106ee1698 <col:3, col:8> 'std::__1::ostream':'std::__1::basic_ostream<char>' lvalue Var 0x106ee0fb8/
             'cout' 'std::__1::ostream':'std::__1::basic_ostream<char>'
      '-ImplicitCastExpr 0x106ee3dd8 <col:16> 'const char *' <ArrayToPointerDecay>
        '-StringLiteral 0x106ee16c8 <col:16> 'const char [13]' lvalue "Hello, C++!\n"
```

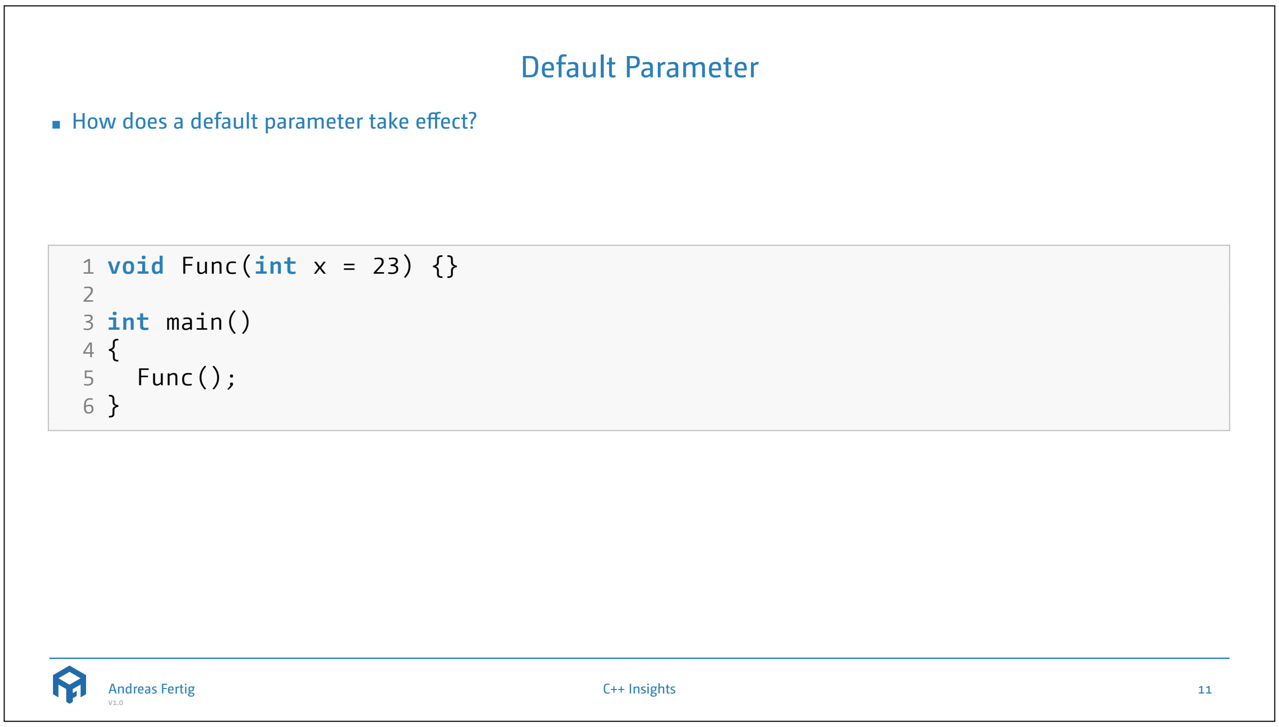
Andreas Fertig v1.0 C++ Insights 9



What is an AST

```cpp
#include <iostream>

int main()
{
  std::cout << "Hello, C++!\n";
}
```

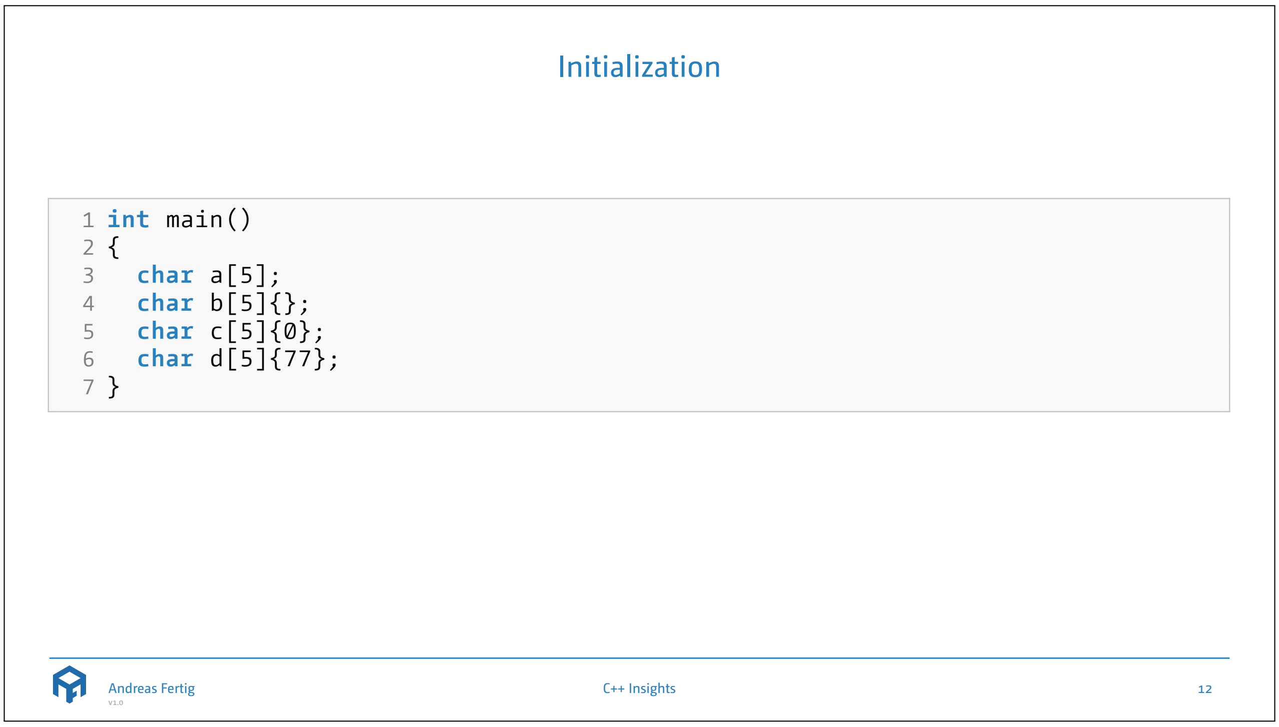
Andreas Fertig v1.0 C++ Insights 10



Default Parameter

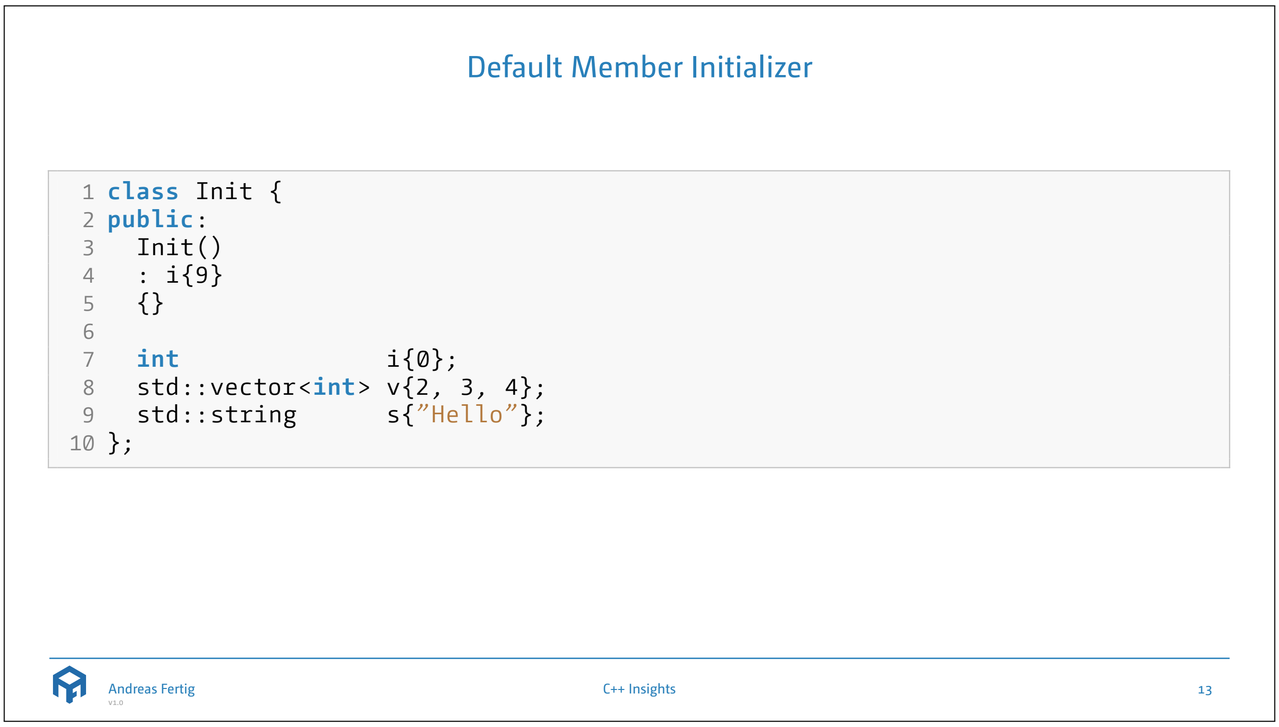
- How does a default parameter take effect?

```cpp
void Func(int x = 23) {}

int main()
{
  Func();
}
```



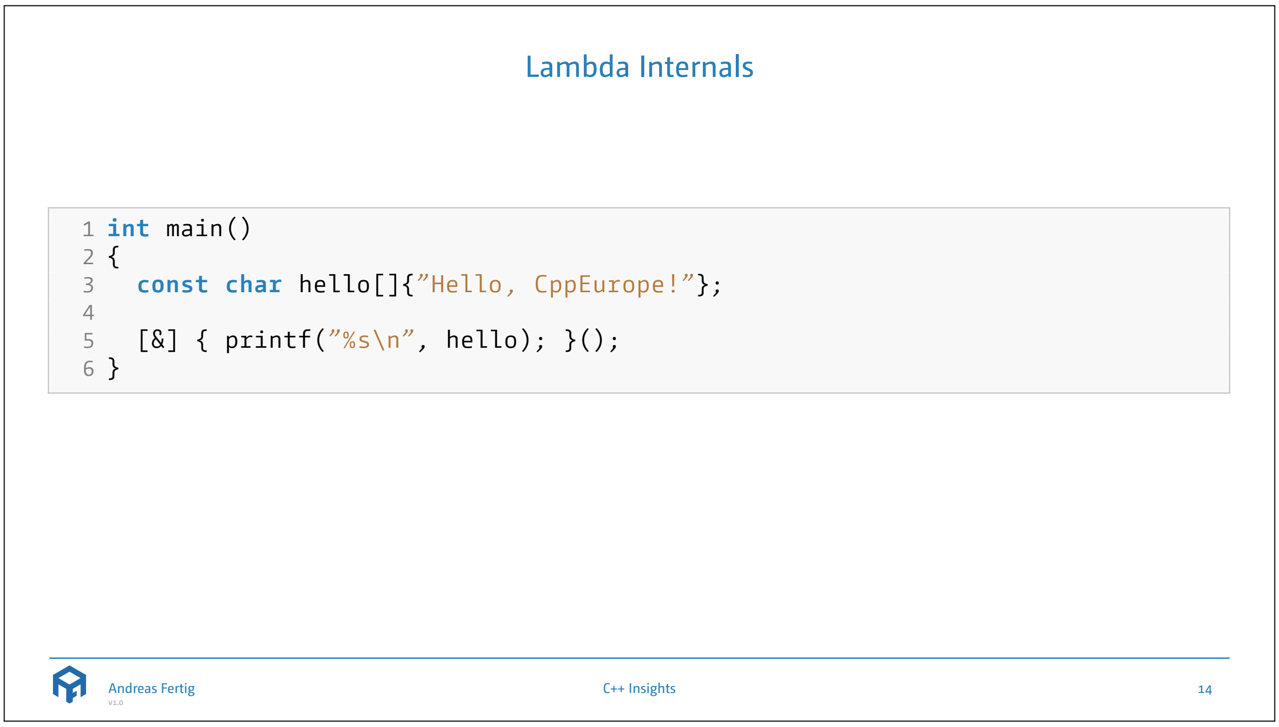
Initialization

```cpp
int main()
{
  char a[5];
  char b[5]{};
  char c[5]{0};
  char d[5]{77};
}
```



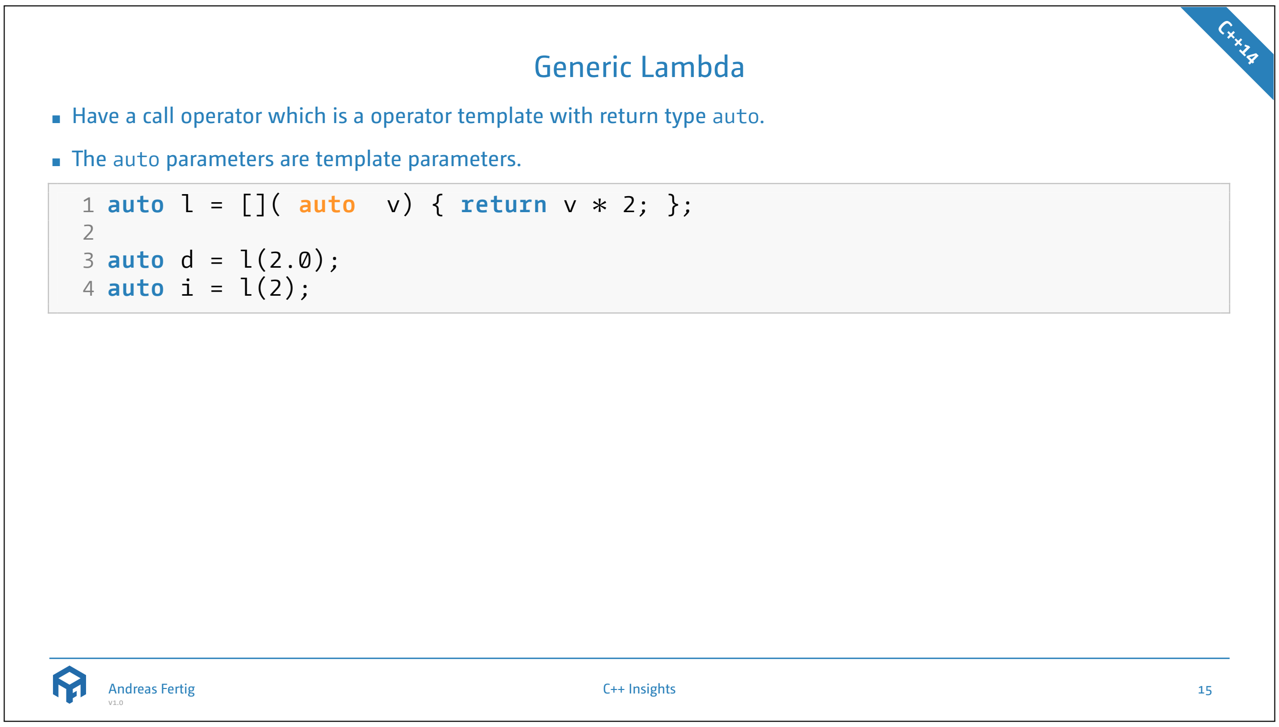
Default Member Initializer

```cpp
class Init {
public:
  Init()
  : i{9}
  {}

  int              i{0};
  std::vector<int> v{2, 3, 4};
  std::string      s{"Hello"};
};
```



Lambda Internals

```cpp
int main()
{
  const char hello[]{"Hello, CppEurope!"};

  [&] { printf("%s\n", hello); }();
}
```



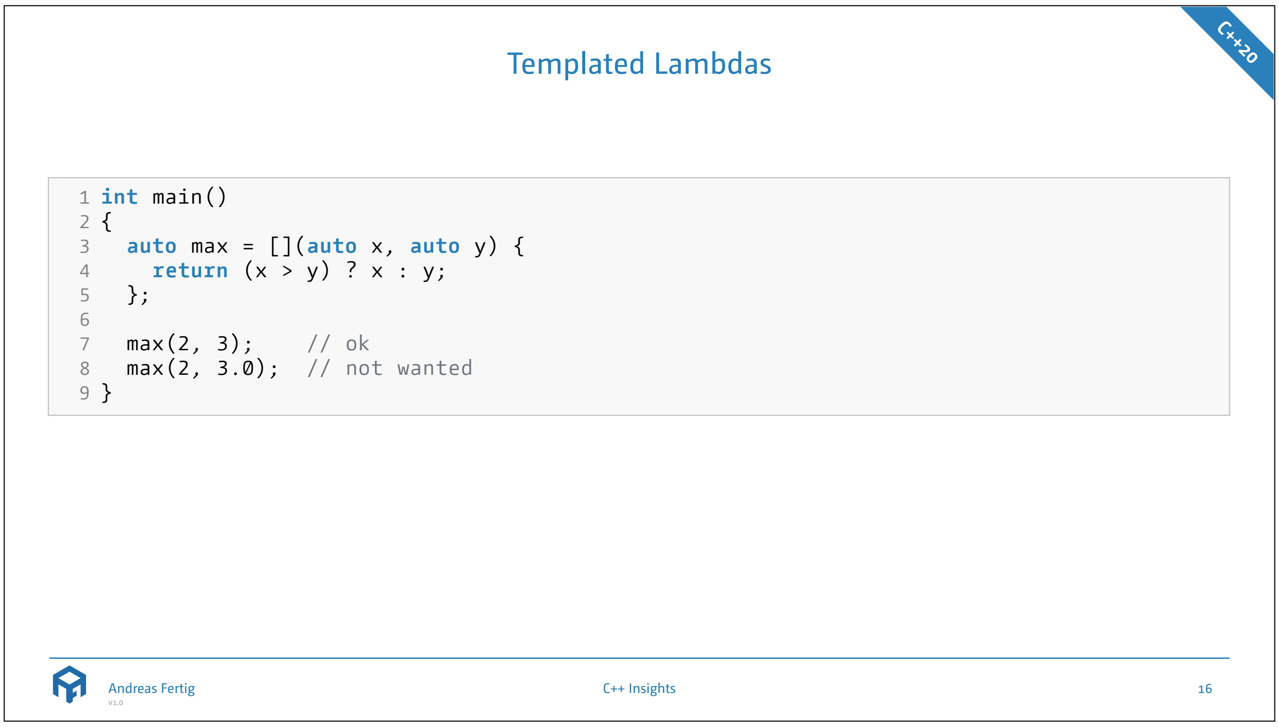
C++14

Generic Lambda

- Have a call operator which is a operator template with return type `auto`.
- The `auto` parameters are template parameters.

```cpp
auto l = []( auto  v) { return v * 2; };

auto d = l(2.0);
auto i = l(2);
```

Andreas Fertig v1.0 C++ Insights 15

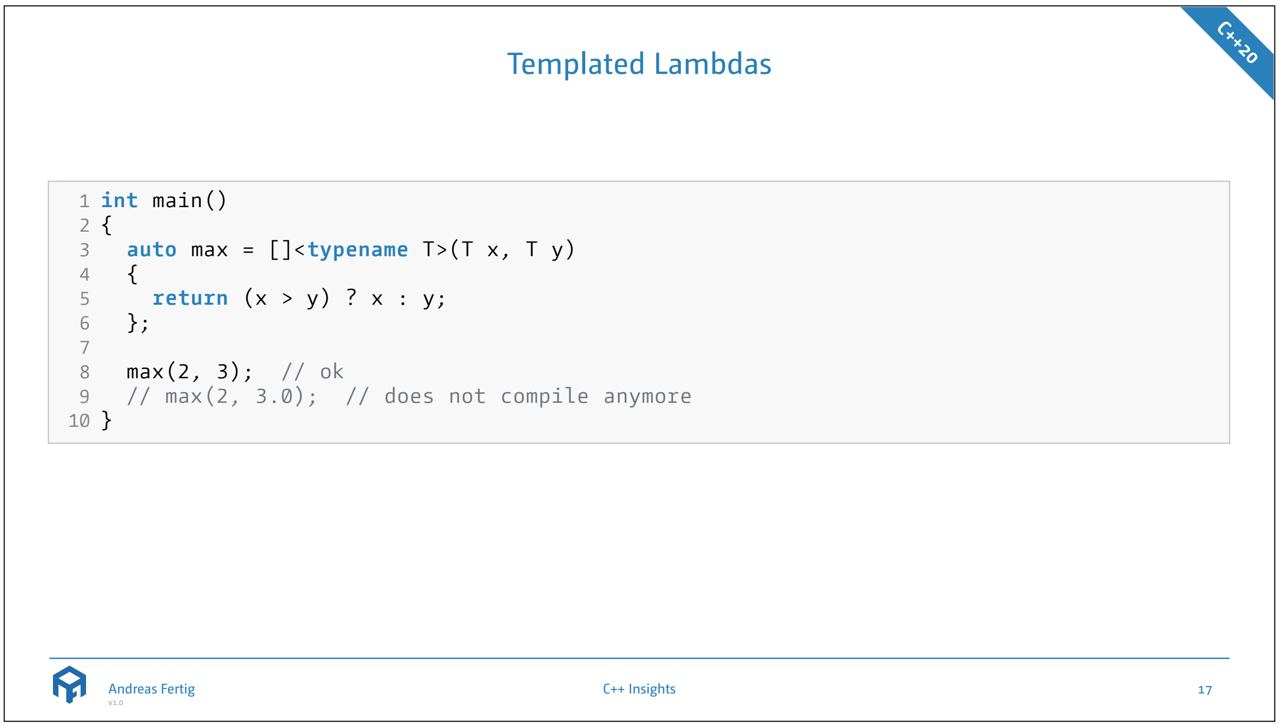


C++20

Templated Lambdas

```cpp
int main()
{
  auto max = [](auto x, auto y) {
    return (x > y) ? x : y;
  };

  max(2, 3);     // ok
  max(2, 3.0);   // not wanted
}
```

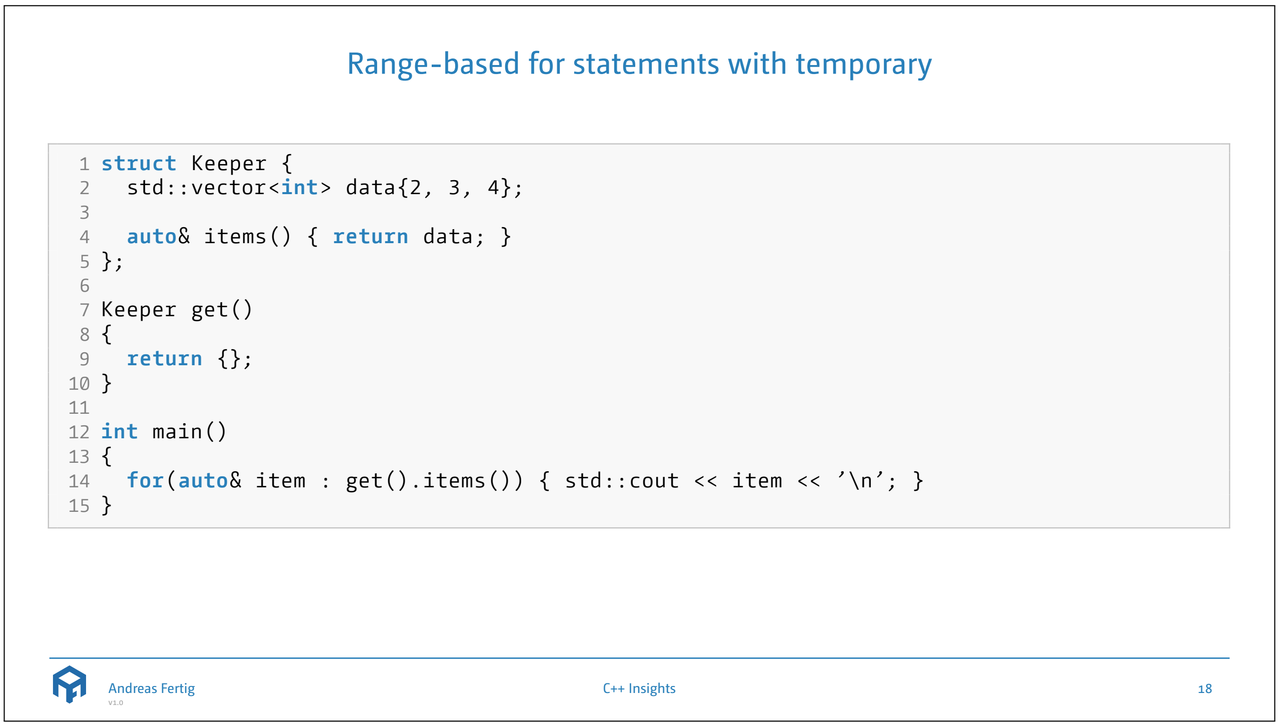
Andreas Fertig v1.0 C++ Insights 16



Templated Lambdas

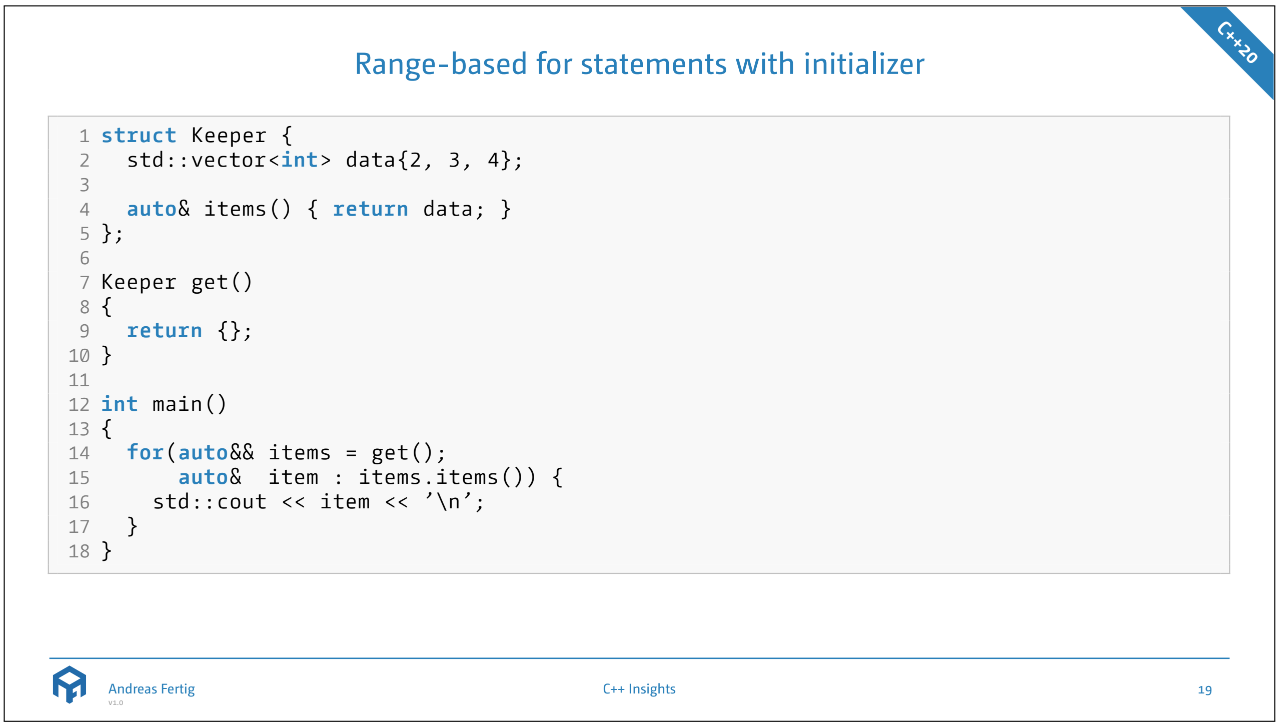
C++20

```cpp
int main()
{
  auto max = []<typename T>(T x, T y)
  {
    return (x > y) ? x : y;
  };

  max(2, 3);  // ok
  // max(2, 3.0);  // does not compile anymore
}
```



Range-based for statements with temporary

```cpp
struct Keeper {
  std::vector<int> data{2, 3, 4};

  auto& items() { return data; }
};

Keeper get()
{
  return {};
}

int main()
{
  for(auto& item : get().items()) { std::cout << item << '\n'; }
}
```

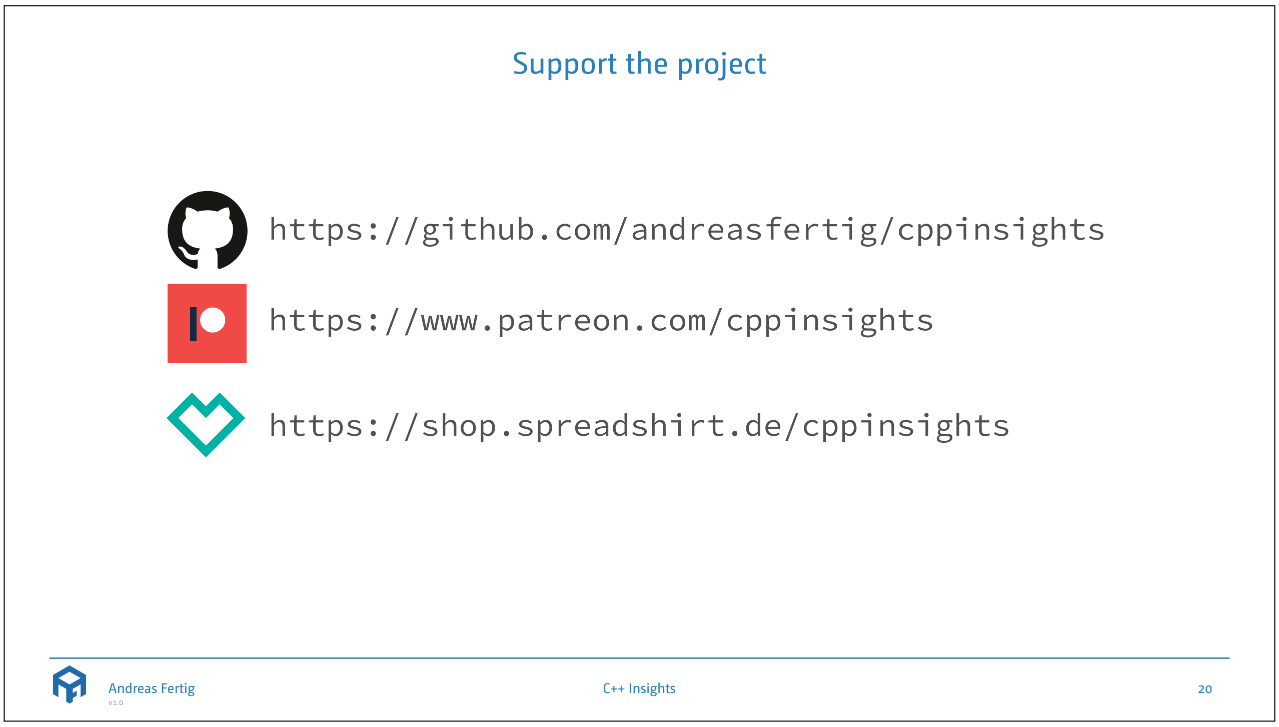


C++20

Range-based for statements with initializer

```cpp
struct Keeper {
  std::vector<int> data{2, 3, 4};

  auto& items() { return data; }
};

Keeper get()
{
  return {};
}

int main()
{
  for(auto&& items = get();
      auto&  item : items.items()) {
    std::cout << item << '\n';
  }
}
```

Andreas Fertig
v1.0 C++ Insights 19



Support the project

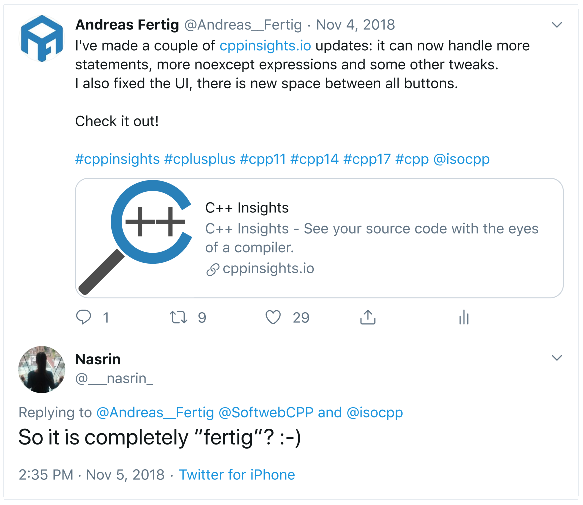
https://github.com/andreasfertig/cppinsights

https://www.patreon.com/cppinsights

https://shop.spreadshirt.de/cppinsights

Andreas Fertig
v1.0 C++ Insights 20

So it is completely fertig?


Andreas Fertig @Andreas__Fertig · Nov 4, 2018
I've made a couple of cppinsights.io updates: it can now handle more statements, more noexcept expressions and some other tweaks.
I also fixed the UI, there is new space between all buttons.

Check it out!

#cppinsights #cplusplus #cpp11 #cpp14 #cpp17 #cpp @isocpp

C++ Insights
C++ Insights - See your source code with the eyes of a compiler.
cppinsights.io

1 9 29

Nasrin
@__nasrin_

Replying to @Andreas__Fertig @SoftwebCPP and @isocpp
So it is completely “fertig”? :-)

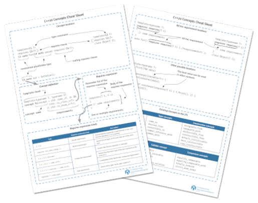
2:35 PM · Nov 5, 2018 · Twitter for iPhone

Source: [1]

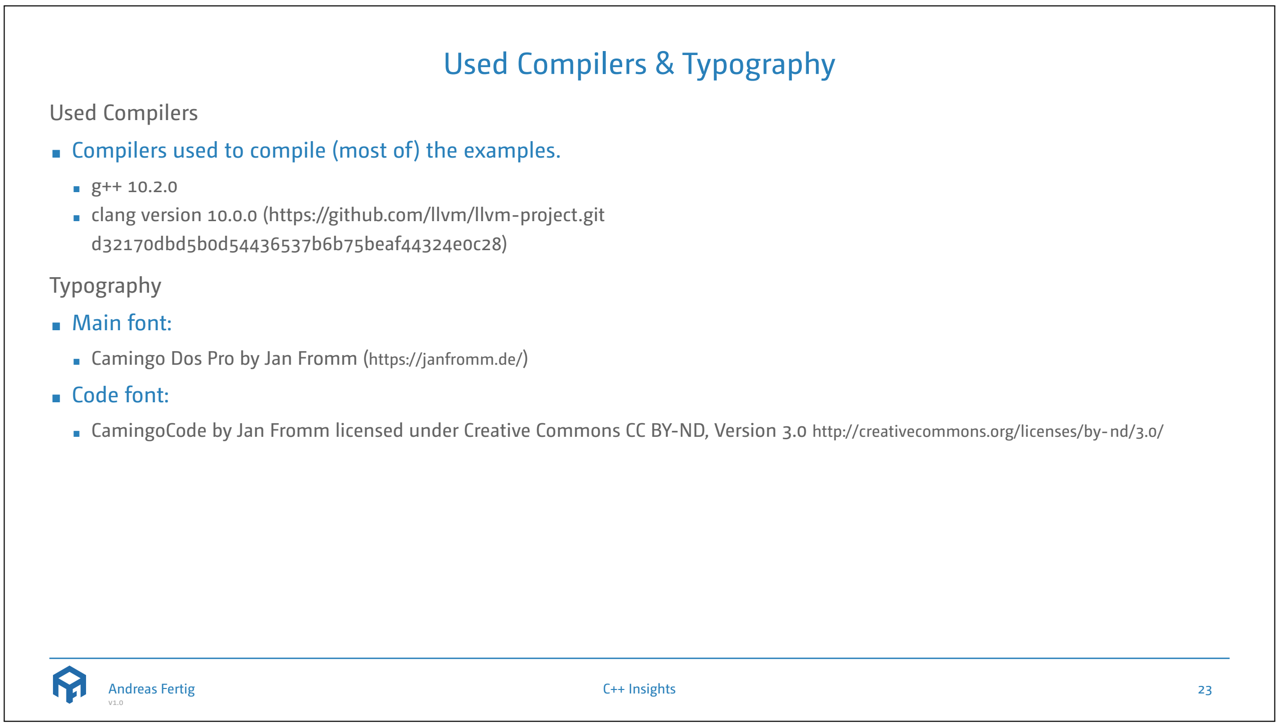
}

I am Fertig.

C++20 Concepts Cheat Sheet



andreasfertig.info/newsletter/



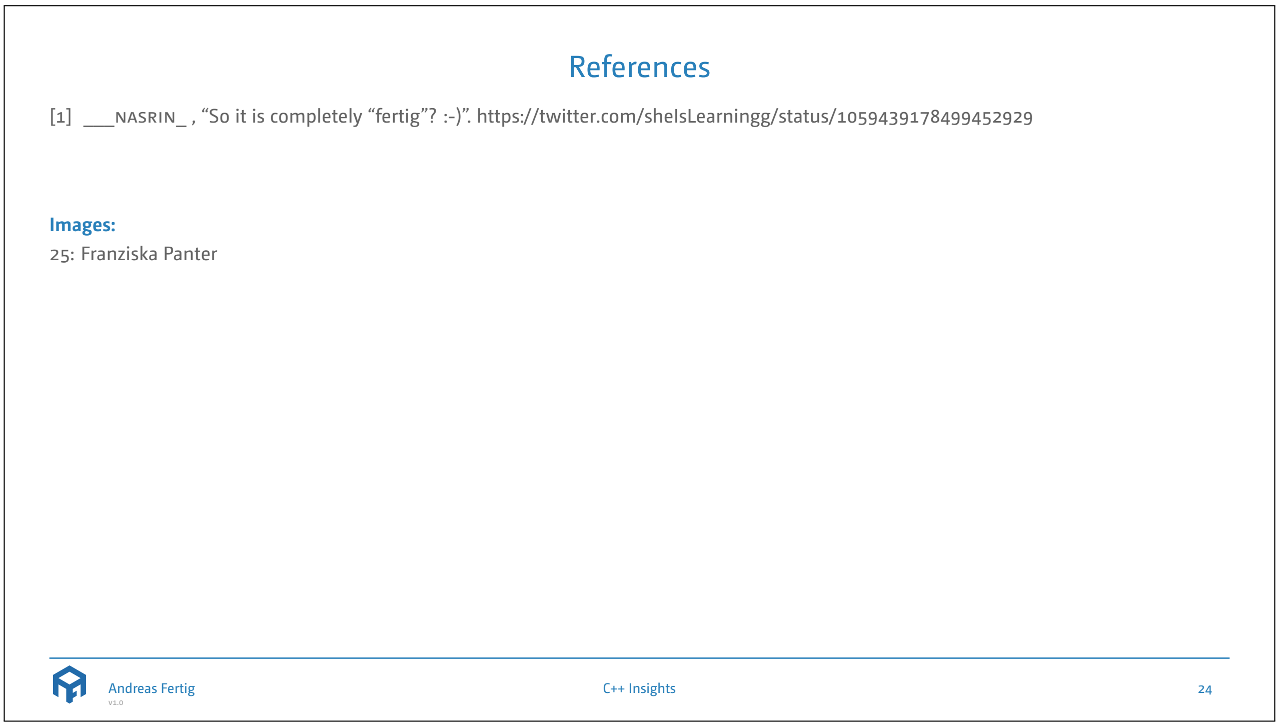
Used Compilers & Typography

Used Compilers

- Compilers used to compile (most of) the examples.
  - g++ 10.2.0
  - clang version 10.0.0 (https://github.com/llvm/llvm-project.git d32170dbd5b0d54436537b6b75beaf44324e0c28)

Typography

- Main font:
  - Camingo Dos Pro by Jan Fromm (https://janfromm.de/)
- Code font:
  - CamingoCode by Jan Fromm licensed under Creative Commons CC BY-ND, Version 3.0 http://creativecommons.org/licenses/by-nd/3.0/



References

[1] ___NASRIN_ , “So it is completely “fertig”? :-)”. https://twitter.com/sheIsLearningg/status/1059439178499452929

Images:
25: Franziska Panter

Upcoming Events

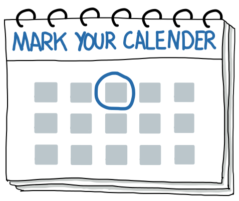
Talks

- *C++20 Templates - The next level: Concepts and more*, ACCU, March 13

Training Classes

- *C++ Clean Code – Best Practices für Programmierer*, golem Akademie, March 08 - 12
- *C++20: Five Features in Five Weeks*, Andreas Fertig, March 30 - April 27
- *Programming with C++11 to C++17*, Andreas Fertig, April 12 - 16
- *C++1x für eingebettete Systeme*, QA Systems, October 14 - 15

For my upcoming talks you can check https://andreasfertig.info/talks/.
For my courses you can check https://andreasfertig.info/courses/.
Like to always be informed? Subscribe to my newsletter: https://andreasfertig.info/newsletter/.



About Andreas Fertig

Photo: Kristijan Matic www.kristijanmatic.de

Andreas Fertig is the CEO of Unique Code GmbH, which offers training and consulting for C++ specialized in embedded systems. He worked for Philips Medizin Systeme GmbH for ten years as a C++ software developer and architect focusing on embedded systems.
Andreas is involved in the C++ standardization committee. He is a regular speaker at conferences internationally. Textbooks and articles by Andreas are available in German and English.
His passion for teaching people how C++ works is why he created C++ Insights (https://cppinsights.io).