Computer Science

Templator: Demo of a nice tool for Visualizing Template Instantiations

ACCU 2016

slides: http://wiki.hsr.ch/PeterSommerlad/

INSTITUTE FOR SOFTWARE

Prof. Peter Sommerlad
Director IFS Institute for Software
Bristol April 2016

HSR
HOCHSCHULE FÜR TECHNIK RAPPERSWIL
FHO Fachhochschule Ostschweiz

Cevelop
Your C++ code deserves it

Download IDE at:
www.cevelop.com

Thesis project of two of my students

- “Dear professor can you give us the hardest task you have in mind for our thesis project?”

Jonas
Biedermann

Marco
Syfrig

Problems with Templates

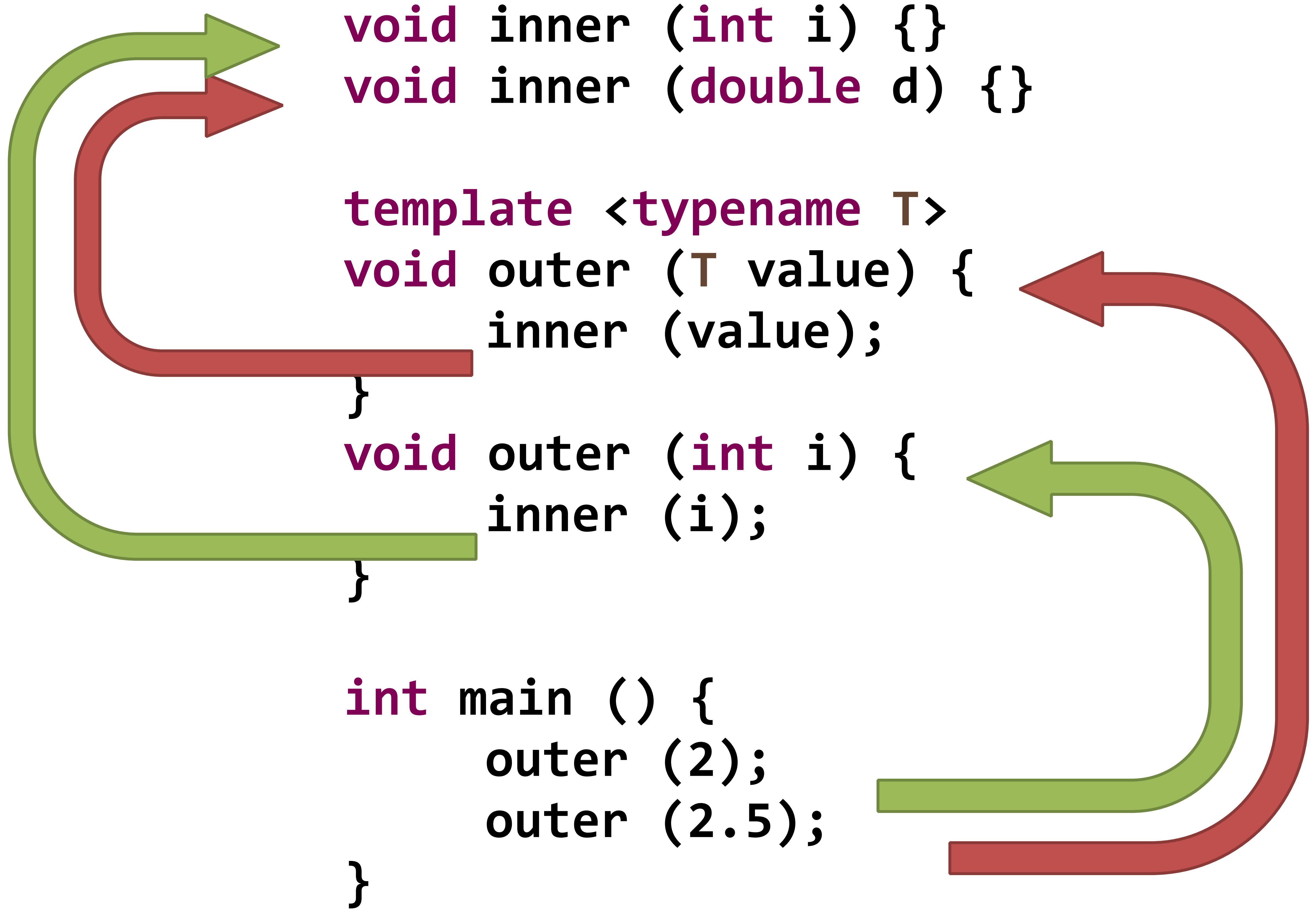
- compiler instantiates a template in the background
  - programmer cannot see the generated code
  - hard to understand
- nested template instantiations
  - compilers may behave unexpected
  - chosen overloads or specializations are not always obvious

Problems with Templates

```
std::vector<bool> v{};
```

- compiler instantiates a template in the background
  - programmer cannot see the generated code
  - hard to understand
- nested template instantiations
  - compilers may behave unexpected
  - chosen overloads or specializations are not always obvious

Overload resolution

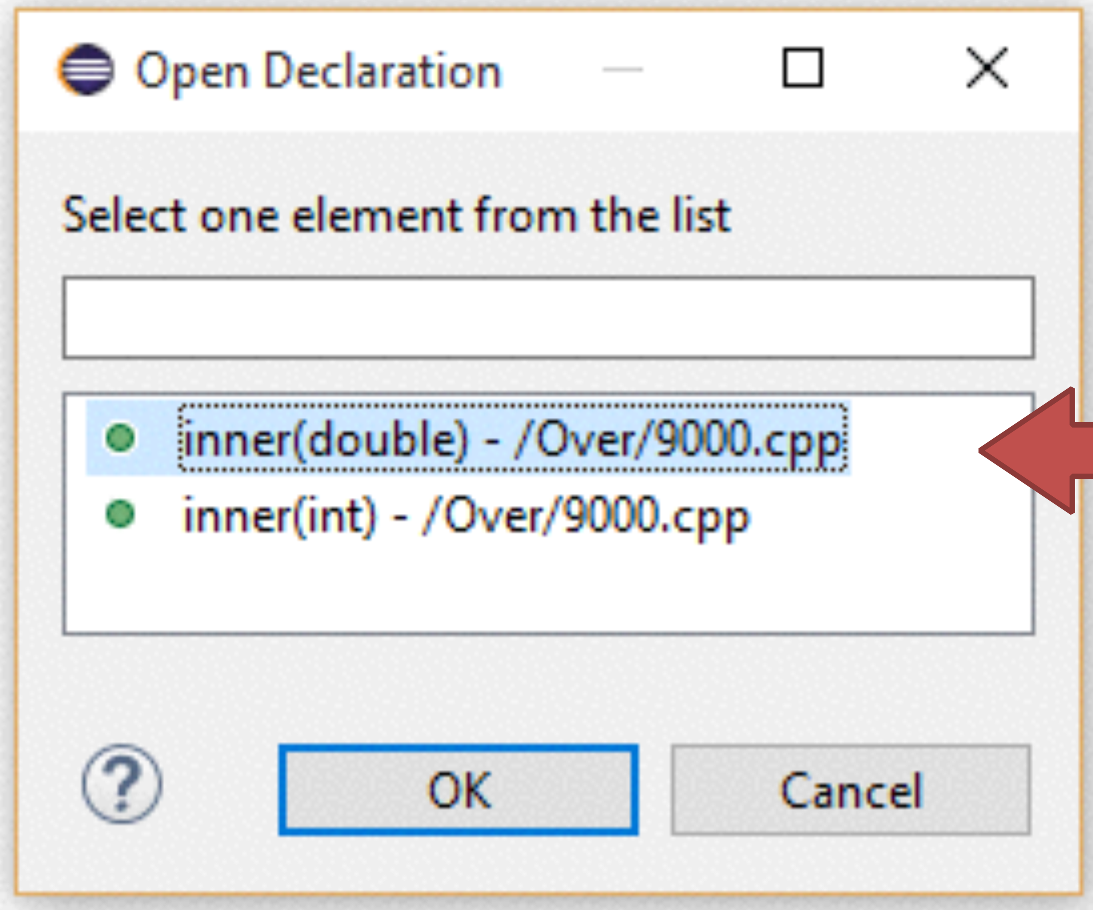


```
void inner (int i) {}
void inner (double d) {}

template <typename T>
void outer (T value) {
     inner (value);
}
void outer (int i) {
     inner (i);
}

int main () {
     outer (2);
     outer (2.5);
}
```

Not completely supported by CDT


Open Declaration

Select one element from the list

- inner(double) - /Over/9000.cpp
- inner(int) - /Over/9000.cpp

```cpp
void inner (int i) {}
void inner (double d) {}

template <typename T>
void outer (T value) {
    inner (value);
}
void outer (int i) {
    inner (i);
}

int main () {
    outer (2);
    outer (2.5);
}
```

Olve Maudal's pub quiz 2014 (Q3)

- Overload resolution can be tricky for programmers to grasp!
- Guess the output without running it!

Olve Maudal’s pub quiz 2014 (Q3)

```cpp
#include <iostream>
#include <utility>
void y(int&) { std::cout << '1'; }
void y(int&&) { std::cout << '2'; }

template <typename T>
void f(T && x) { y(x); }
template <typename T>
void g(T && x) { y(std::move(x)); }
template <typename T>
void h(T && x) { y(std::forward<T>(x)); }

int main() {
   int i{42}; // changed original  i=42 to i{42}
   y(i), y(42);
   f(i), f(42);
   g(i), g(42);
   h(i), h(42);
}
```

- Overload resolution can be tricky for programmers to grasp!
- Guess the output without running it!

Let’s Demo

and see if Templator helps

Wrap up

- Templator visualizes template instantiation and call chains
- Templator shows code, only the compiler would generate internally, but it is not perfect yet, due to parser deficiencies
  - we are working on SFINAE, C++14 constexpr, and Concepts
- is available as part of free Cevelop-neon 1.5 IDE Preview
  - financial and other support highly appreciated for Cevelop
    - (unfortunately no micro-payments acceptable yet)

Questions?

- contact: peter.sommerlad@hsr.ch
- Looking for a better free IDE:

Cevelop

Your C++ code deserves it

Preview 1.5
includes Templator

Download IDE at:
www.cevelop.com

- examples available at: https://github.com/PeterSommerlad/Publications