Debugging and Profiling C++ Template Metaprograms

Zoltán Porkoláb
gsd@elte.hu

Ericsson Hungary
Eötvös Loránd University, Budapest

Agenda

- C++ Template Metaprogramming
- Possible debugging and profiling techiques
- Templight back-end tool
- Front-end tools
- 3rd party applications – please, contribute!
- Vision

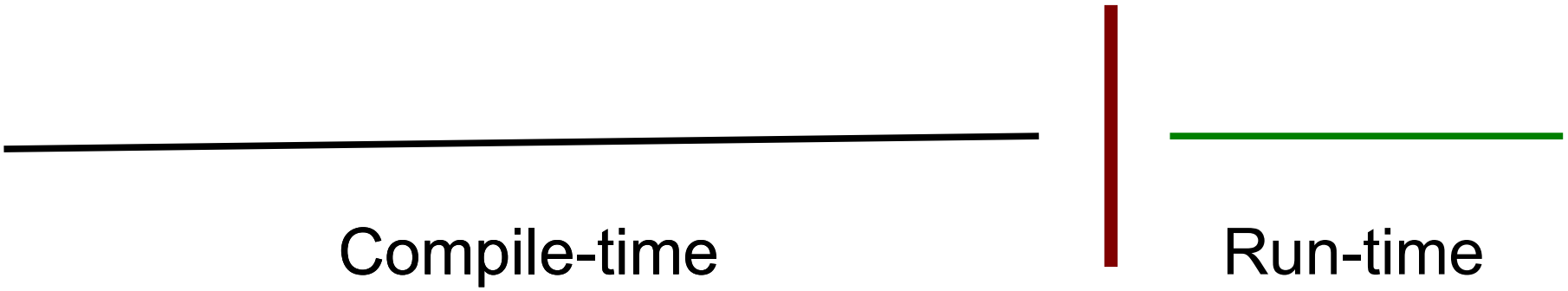
C++ Template Metaprograms

- Expression templates (since 1995!)
- Active libraries, compile-time adaption
- Static interface checking
- Simulating language extensions
- DSL embedding
- Many other areas ...

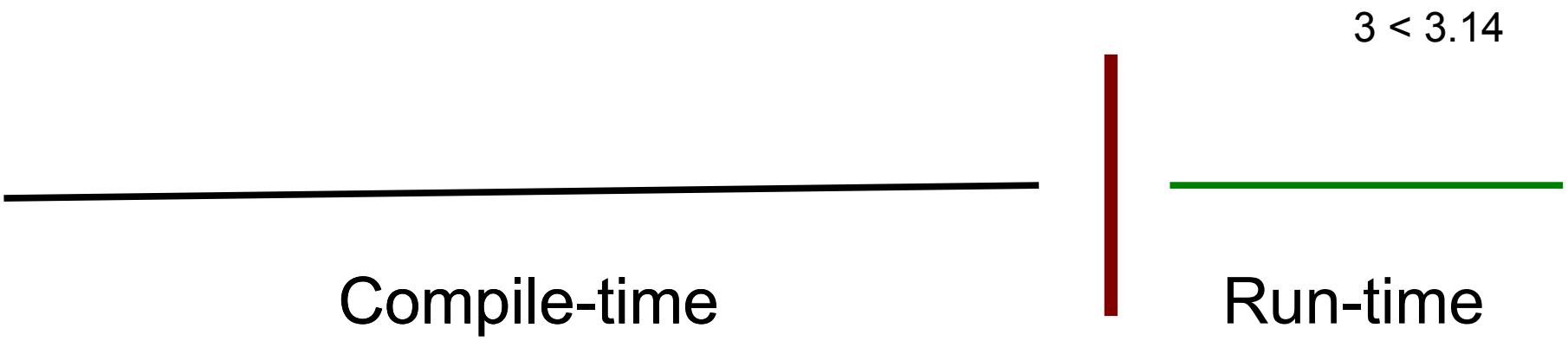
Motivation – a personal view

```
template <class T, class S>
? max( T a, S b)  // How to define the return value?
{
    if ( a > b )
        return a;
    else
        return b;
}

int main()
{
    short is = 3; long il = 2; double d = 3.14;
    cout << max( il, is);
    cout << max( is, d);
}
```

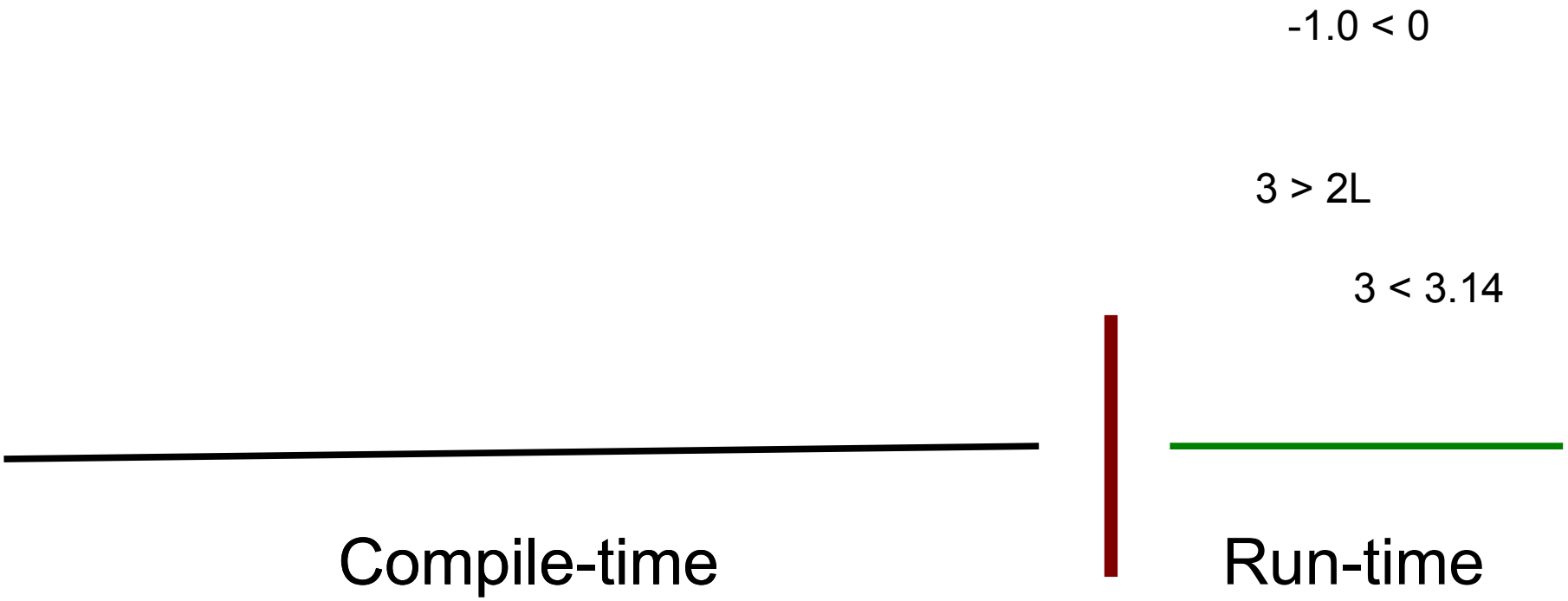
Compile-time vs. Run-time



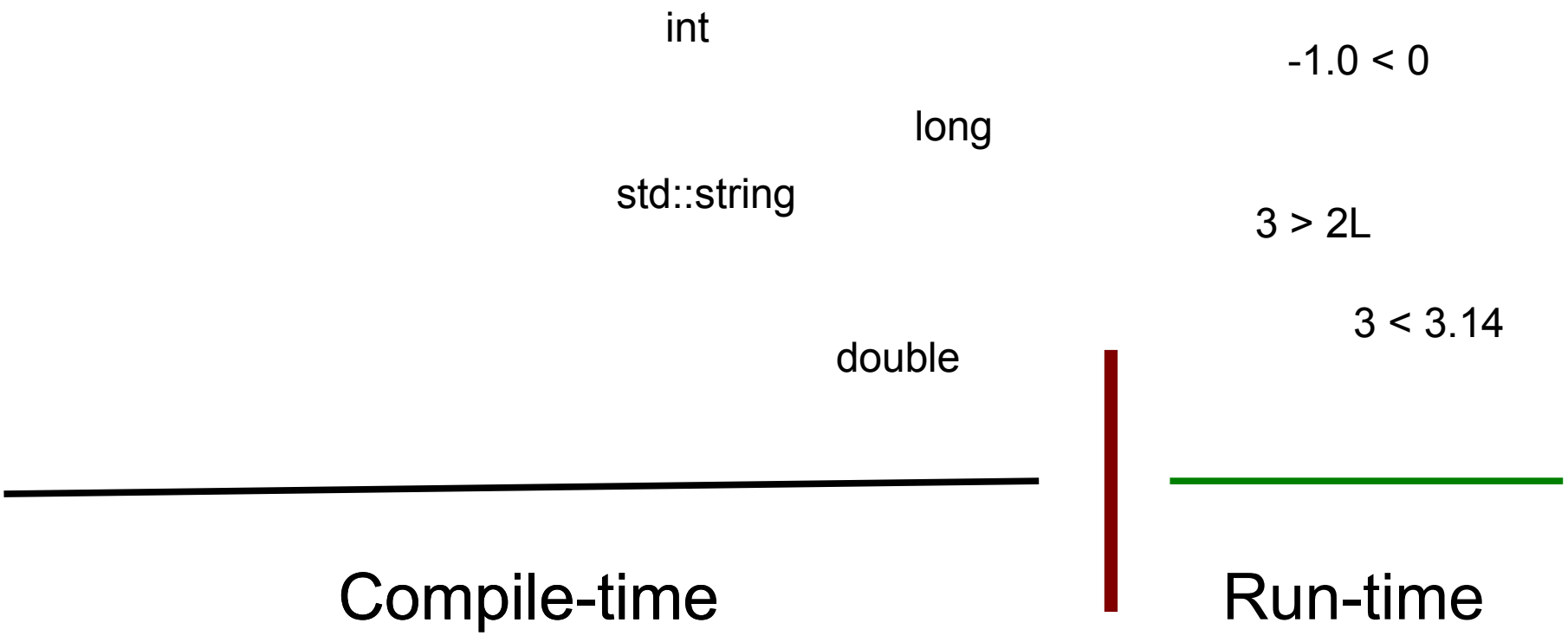
Compile-time vs. Run-time



Compile-time vs. Run-time



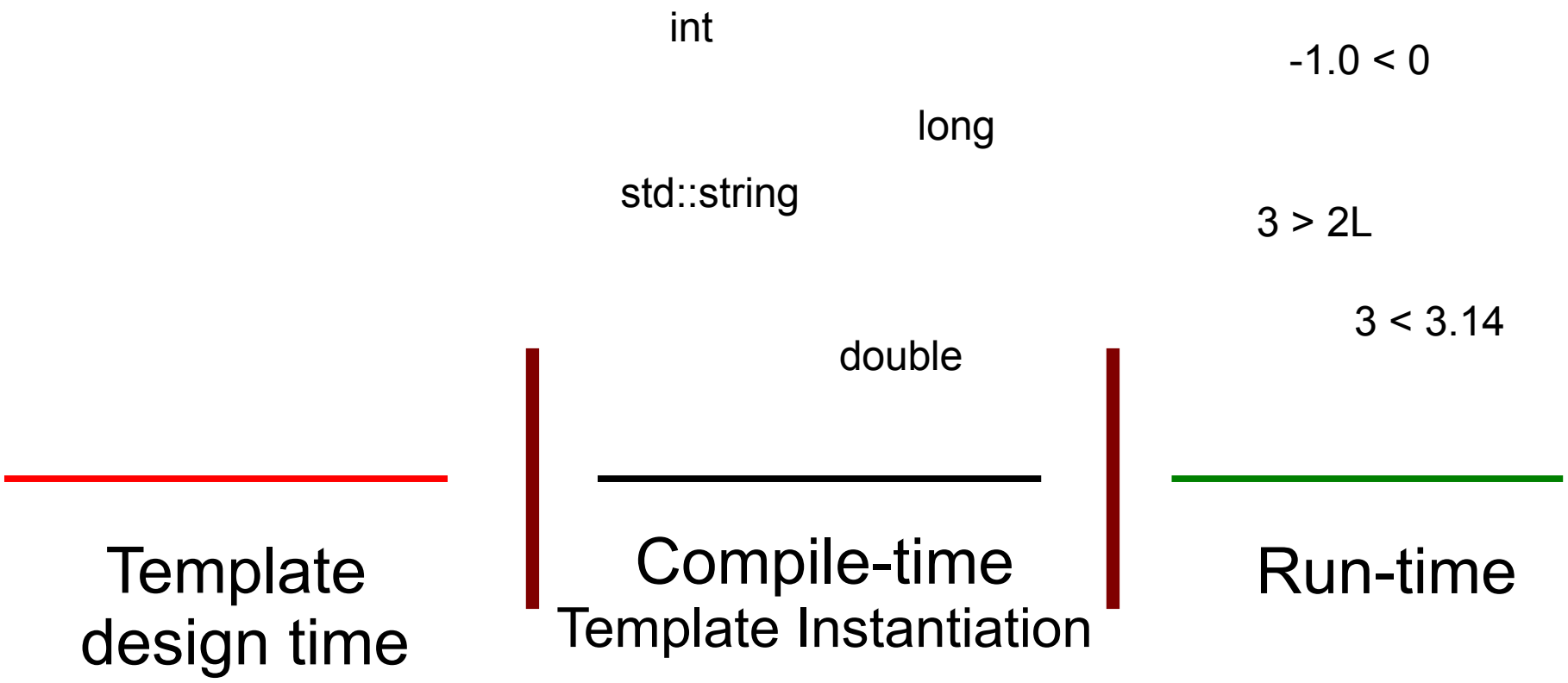
Compile-time vs. Run-time



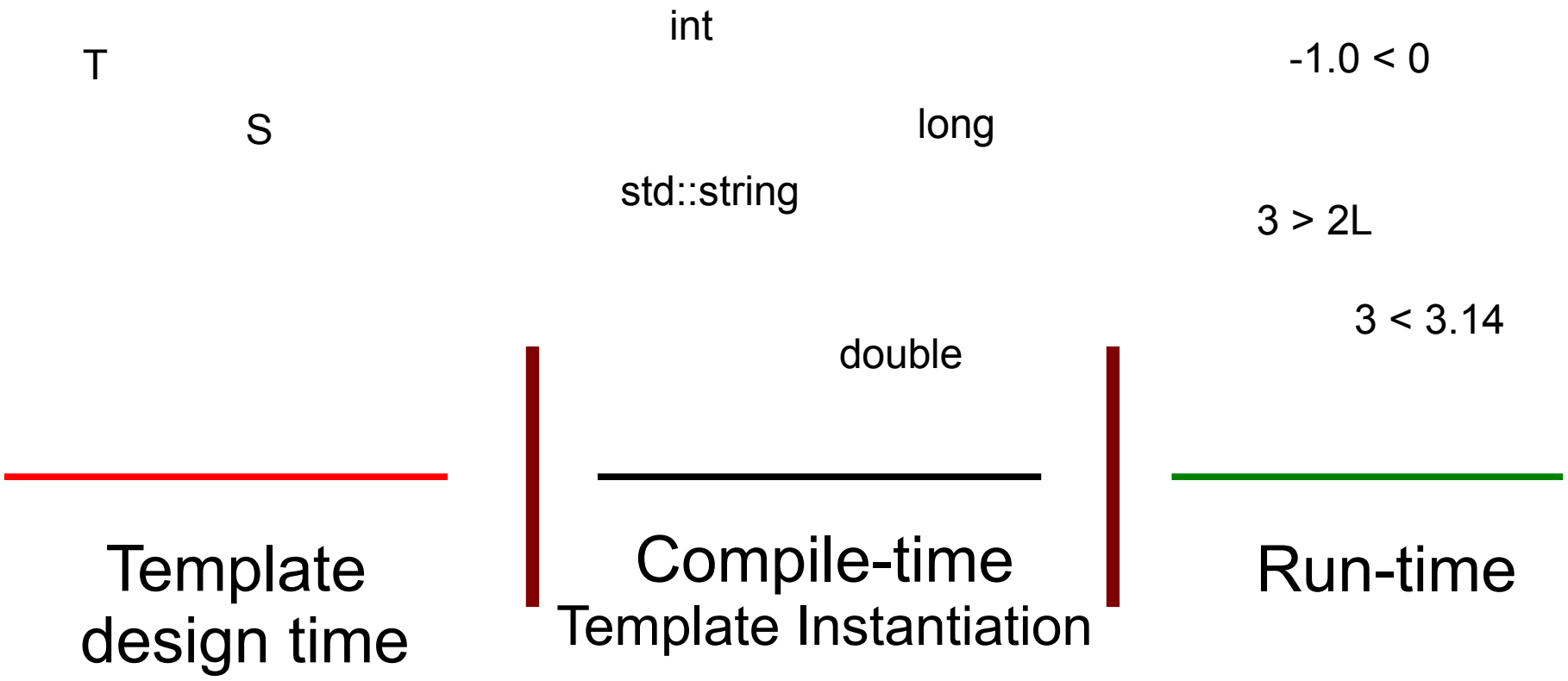
Motivation

```
template <class T, class S>
? max( T a, S b)  // How to define the return value?
{
    if ( a > b )
        return a;
    else
        return b;
}

int main()
{
    short is = 3; long il = 2; double d = 3.14;
    cout << max( il, is); // long is ''better'' than short
    cout << max( is, d);  // double is ''better'' than short
}
```

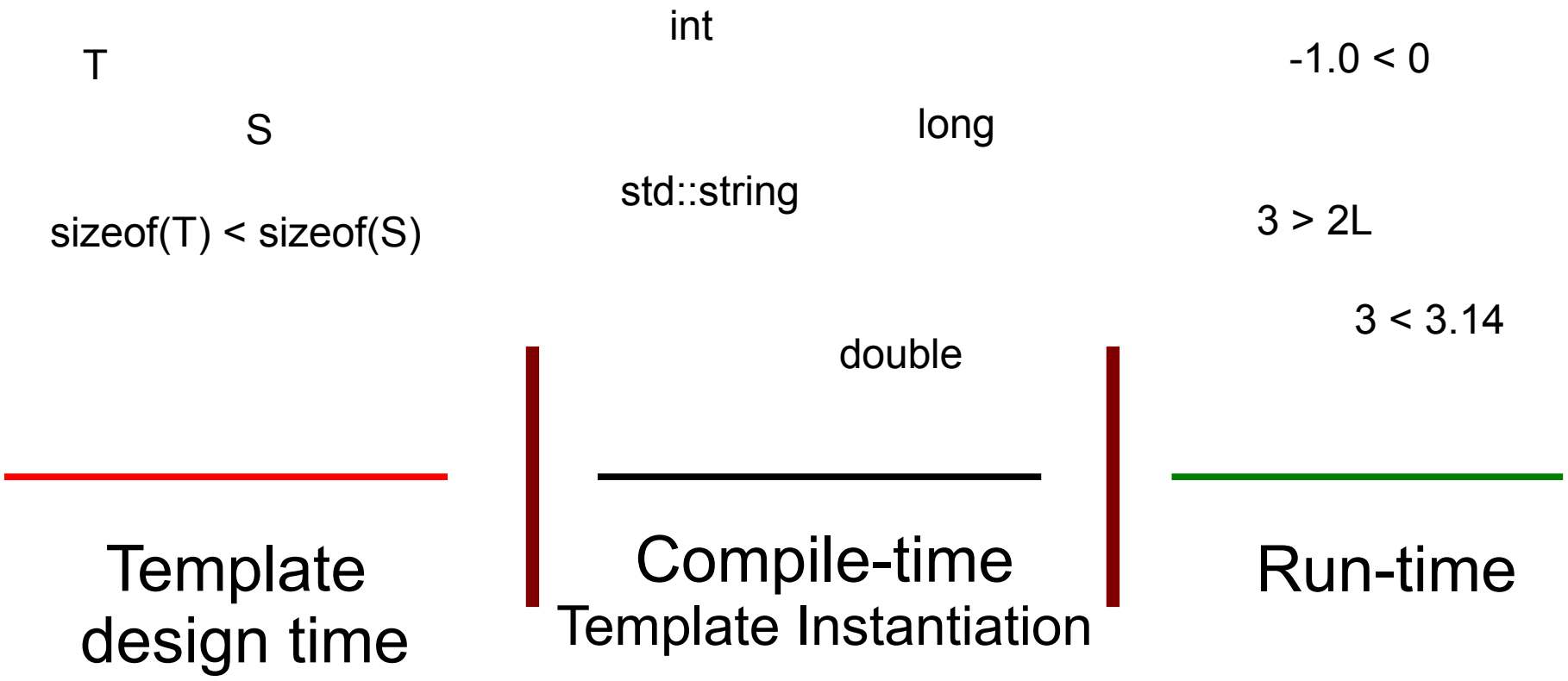
Compile-time vs. Run-time



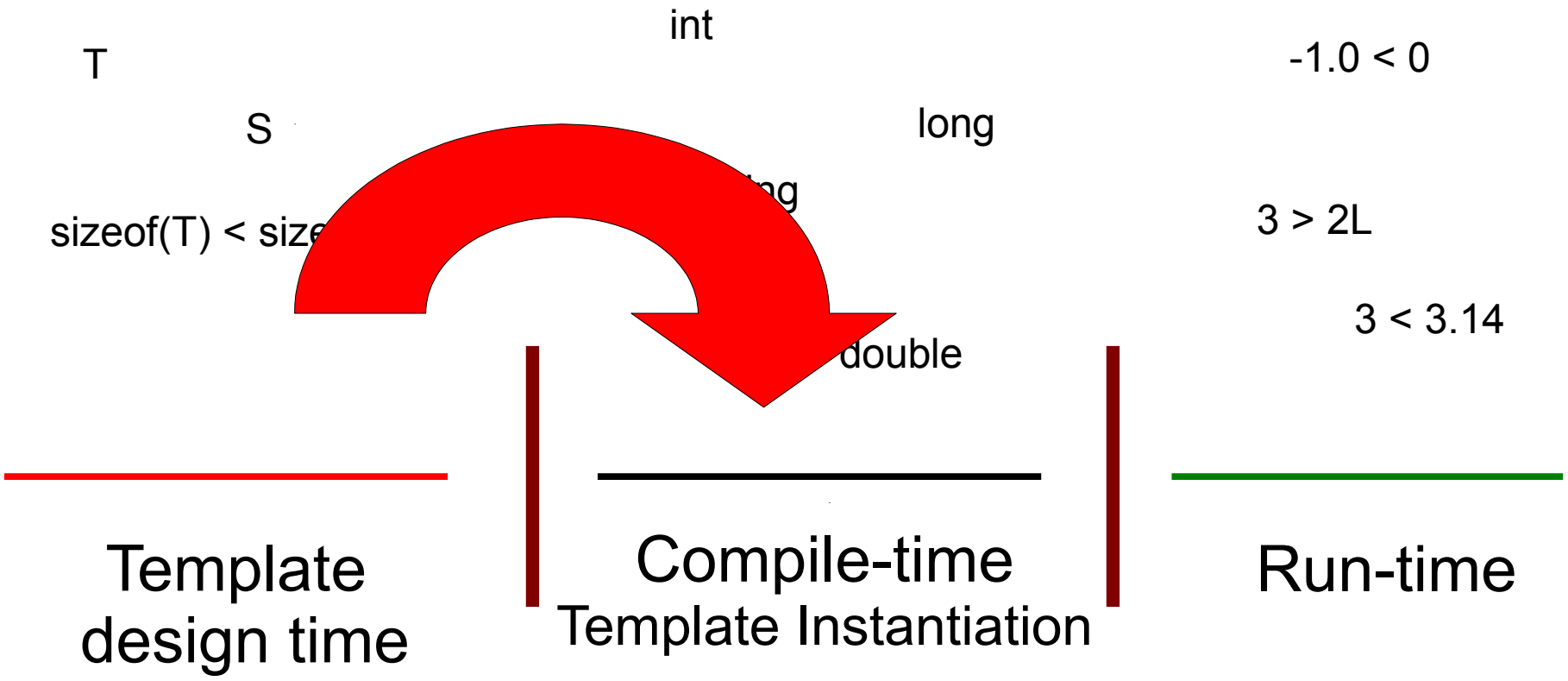
Compile-time vs. Run-time



Compile-time vs. Run-time



Compile-time vs. Run-time



Motivation

```
template <class T, class S>
? max( T a, S b)  // How to define the return value?
{
    if ( a > b )
        return a;
    else
        return b;
}

int main()
{
    short is = 3; long il = 2; double d = 3.14;
    cout << max( il, is); // long is ''better'' than short
    cout << max( is, d);  // double is ''better'' than short
}
```

(de)Motivation

```
template <class T, class S>
auto max( T a, S b) -> decltype(a+b) // C++11
{
    if ( a > b )
        return a;
    else
        return b;
}

int main()
{
    short is = 3; long il = 2; double d = 3.14;
    cout << max( il, is);  // -> long
    cout << max( is, d);   // -> double
}
```

(de)Motivation

```
template <class T, class S>
typename std::common_type<T,S>::value max( T a, S b) // C++11
{
    if ( a > b )
        return a;
    else
        return b;
}

int main()
{
    short is = 3; long il = 2; double d = 3.14;
    cout << max( il, is);  // -> long
    cout << max( is, d);   // -> double
}
```

- **Run-time**
- **Compile-time**

- **Run-time**

- Functions
- Values, literals
- Data structures
- If/else
- Loop
- Assignment
- May depend on input

- **Compile-time**

- **Run-time**

- Functions
- Values, literals
- Data structures
- If/else
- Loop
- Assignment
- May depend on input

- **Compile-time**

- Metafunctions (type)
- Const, enum,constexpr
- Typelist (type)
- Pattern matching
- Recursion
- Ref. Transparency
- Deterministic

- **Run-time**

- Imperative
- Object-oriented
- Generative
- (some) Functional

- **Compile-time**

- **Run-time**

- Imperative
- Object-oriented
- Generative
- (some) Functional

- **Compile-time**

- Pure Functional

The usual factorial program ...

```
template <int N>
struct Factorial
{
  enum { value = Factorial<N-1>::value * N };
};
template <>
struct Factorial<0>
{
  enum { value = 1 };
};
int main()
{
  const int fact5 = Factorial<5>::value;
}
```

Bugs!!! ...

The java programmer ...

```
template <int N>
struct Factorial
{
  enum { value = Factorial<N-1>::value * N };
};
template <>
struct Factorial<0>
{
  enum { value = 1 };
} //;
int main()
{
  const int fact5 = Factorial<5>::value;
}
```

The java programmer ...

```
template <int N>
struct Factorial
{
  enum { value = Fact
};
template <>
struct Factorial<0>
{
  enum { value = 1 };
} //;
int main()
{
  const int fact5 = Factorial<5>::value;
}
```

```
$ clang++ fact.cpp
fact.cpp:14:2: error: expected ';' after class
}
 ^
 ;
1 error generated.
```

The vim user ...

```
template <int N>
struct Factorial
{
  enum { value = Factorial<N-1>::value * N };
};
template <>
struct Factorial<0>
{
  enum { ivalue = 1 };
};
int main()
{
  const int fact5 = Factorial<5>::value;
}
```

The vim user ...

```
template <int N>
struct Factorial
{
  enum { value = Fact
};
template <>
struct Factorial<0>
{
  enum { ivalue = 1 }
};
int main()
{
  const int fact5 = F
}
```

```
$ clang++ fact.cpp
fact.cpp:5:34: error: no member named 'value' in 'Factorial<0>'
  enum { value = Factorial<N-1>::value * N };
                 ~~~~~~~~~~~~~~~~^
fact.cpp:5:18: note: in instantiation of template class 'Factorial<1>'
      requested here
  enum { value = Factorial<N-1>::value * N };
                 ^
fact.cpp:5:18: note: in instantiation of template class 'Factorial<2>'
      requested here
  enum { value = Factorial<N-1>::value * N };
                 ^
fact.cpp:5:18: note: in instantiation of template class 'Factorial<3>'
      requested here
  enum { value = Factorial<N-1>::value * N };
                 ^
fact.cpp:5:18: note: in instantiation of template class 'Factorial<4>'
      requested here
  enum { value = Factorial<N-1>::value * N };
                 ^
fact.cpp:16:21: note: in instantiation of template class 'Factorial<5>'
      requested here
  const int fact5 = Factorial<5>::value;
                    ^
1 error generated.
```

The negative approach ...

```
template <int N>
struct Factorial
{
  enum { value = Factorial<N-1>::value * N };
};
template <>
struct Factorial<0>
{
  enum { value = 1 };
};
int main()
{
  const int fact5 = Factorial<-5>::value;
}
```

The negative approach ...

```
template <int N>
struct Factorial
{
  enum { value = Fact
};
template <>
struct Factorial<0>
{
  enum { value = 1 };
};
int main()
{
  const int fact5 = F
}
```

```
$ clang++ fact4.cpp
fact4.cpp:6:18: fatal error: recursive template instantiation exceeded
maximum
    depth of 512
 enum { value = Factorial<N-1>::value * N };
                ^
fact4.cpp:6:18: note: in instantiation of template class 'Factorial<-517>'
    requested here
 enum { value = Factorial<N-1>::value * N };

Fact4.cpp:6:18: note: (skipping 503 contexts in backtrace; use
    -ftemplate-backtrace-limit=0 to see all)

fact4.cpp:18:21: note: in instantiation of template class 'Factorial<-5>'
    requested here
 const int fact5 = Factorial<-5>::value;
                   ^
fact4.cpp:6:18: note: use -ftemplate-depth=N to increase recursive
template
    instantiation depth
 enum { value = Factorial<N-1>::value * N };
                ^
1 error generated.
```

The greedy ...

```
template <int N>
struct Factorial
{
  enum { value = Fact
};
template <>
struct Factorial<0>
{
  enum { value = 1 };
};
int main()
{
  const int fact5 = F
}
```

```
$ clang++ -ftemplate-depth=10000 fact4.cpp
```

The greedy ...

```
template <int N>
struct Factorial
{
  enum { value = Fact
};
template <>
struct Factorial<0>
{
  enum { value = 1 };
};
int main()
{
  const int fact5 = F
}
```

```
$ clang++ -ftemplate-depth=10000 fact4.cpp
clang: error: unable to execute command: Segmentation fault
clang: error: clang frontend command failed due to signal (use -v to
see invocation)
clang version 3.2 (branches/release_32 180710)
Target: x86_64-unknown-linux-gnu
Thread model: posix
clang: note: diagnostic msg: PLEASE submit a bug report to
http://llvm.org/bugs/ and include the crash backtrace, preprocessed
source, and associated run script.
clang: note: diagnostic msg:
********************

PLEASE ATTACH THE FOLLOWING FILES TO THE BUG REPORT:
Preprocessed source(s) and associated run script(s) are located at:
clang: note: diagnostic msg: /tmp/fact4-iy6zKp.cpp
clang: note: diagnostic msg: /tmp/fact4-iy6zKp.sh
clang: note: diagnostic msg:

********************
```

We need tools

- C++ syntax is not designed for metaprogramming
- Compilers are not optimised for detecting and reporting template metaprogram errors
- Compilers are not optimised for template metaprogram execution
- Compiler internals are black box for most programmers
- Programmers have less experience with template metaprograms

Tool support

- Pretty good support for run-time C++

Tool support

- Pretty good support for run-time C++
  - Static analyzers, lint-like tools
  - Debuggers
  - Profilers
  - Code comprehension tools
  - Style checkers

Tool support

- Pretty good support for run-time C++
  - Static analyzers, lint-like tools
  - Debuggers
  - Profilers
  - Code comprehension tools
  - Style checkers
- Tools for template metaprogramming

Tool support

- Pretty good support for run-time C++
  - Static analyzers, lint-like tools
  - Debuggers
  - Profilers
  - Code comprehension tools
  - Style checkers
- Tools for template metaprogramming
  - ?

Tool support

Run-time

Compile-time

Tool support

Run-time

Compile-time

Tool support

Run-time

Compile-time

Tool support

Run-time

Compile-time

Tool support

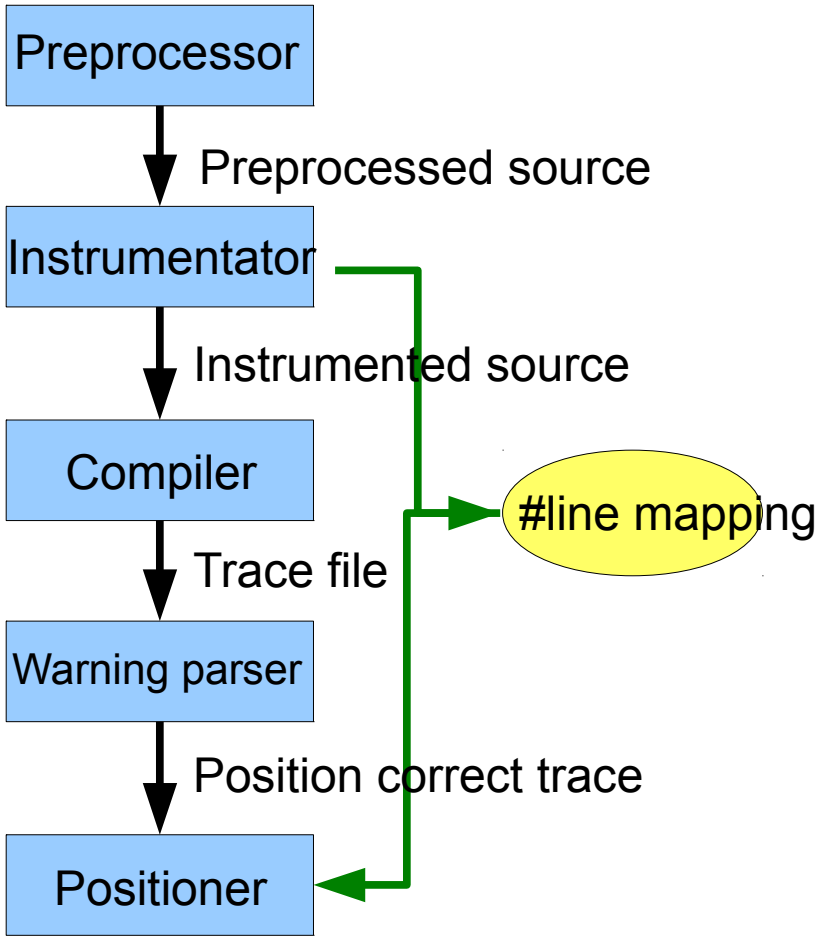
Run-time

Compile-time

Related work

- Debugging
  - Static assert/Concept check (Siek-Lumsdaine, McNamara-Smaragdakis, Alexandrescu, others...)
  - Warning generation (many attempt)
  - Instrumentation
- Profiling
  - Measuring full compilation (Gurtovoy-Abrahams)
  - Measuring warning apperance (Watanabe)
- Visualize
  - Source execution
  - Instantiation graph

GPCE 2006: Porkoláb, Mihalicza, Sipos:
Debugging C++ template metaprograms




```
template<int i>
struct Factorial
{
/* ---------------- begin inserted ----------- */
struct _TEMPLIGHT_0s { int a; };
enum { _TEMPLIGHT_0 =
Templight::ReportTemplateBegin<_TEMPLIGHT_0s,
&_TEMPLIGHT_0s::a>::Value
};
/* ----------------- end inserted ------------ */
enum { value = Factorial<i-1>::value };
/* ---------------- begin inserted ----------- */
struct _TEMPLIGHT_1s { int a; };
enum { _TEMPLIGHT_1 =
Templight::ReportTemplateEnd<_TEMPLIGHT_1s,
&_TEMPLIGHT_1s::a>::Value
};
/* ----------------- end inserted ------------ */
};
template<>
struct Factorial<1>
{
/* ---------------- begin inserted ----------- */
struct _TEMPLIGHT_2s { int a; };
enum { _TEMPLIGHT_2 =
Templight::ReportTemplateBegin<_TEMPLIGHT_2s,
&_TEMPLIGHT_2s::a>::Value
};
/* ----------------- end inserted ------------ */
enum { value = 1 };
/* ---------------- begin inserted ----------- */
struct _TEMPLIGHT_3s { int a; };
enum { _TEMPLIGHT_3 =
Templight::ReportTemplateEnd<
_TEMPLIGHT_3s, &_TEMPLIGHT_3s::a>::Value
};
/* ----------------- end inserted ------------ */
};
```

Instrumentation

- Advantages
  - Light-way approach (compared to compiler hack)
  - Grammar support (we used wave)
  - Easier to port: just change the warning generator

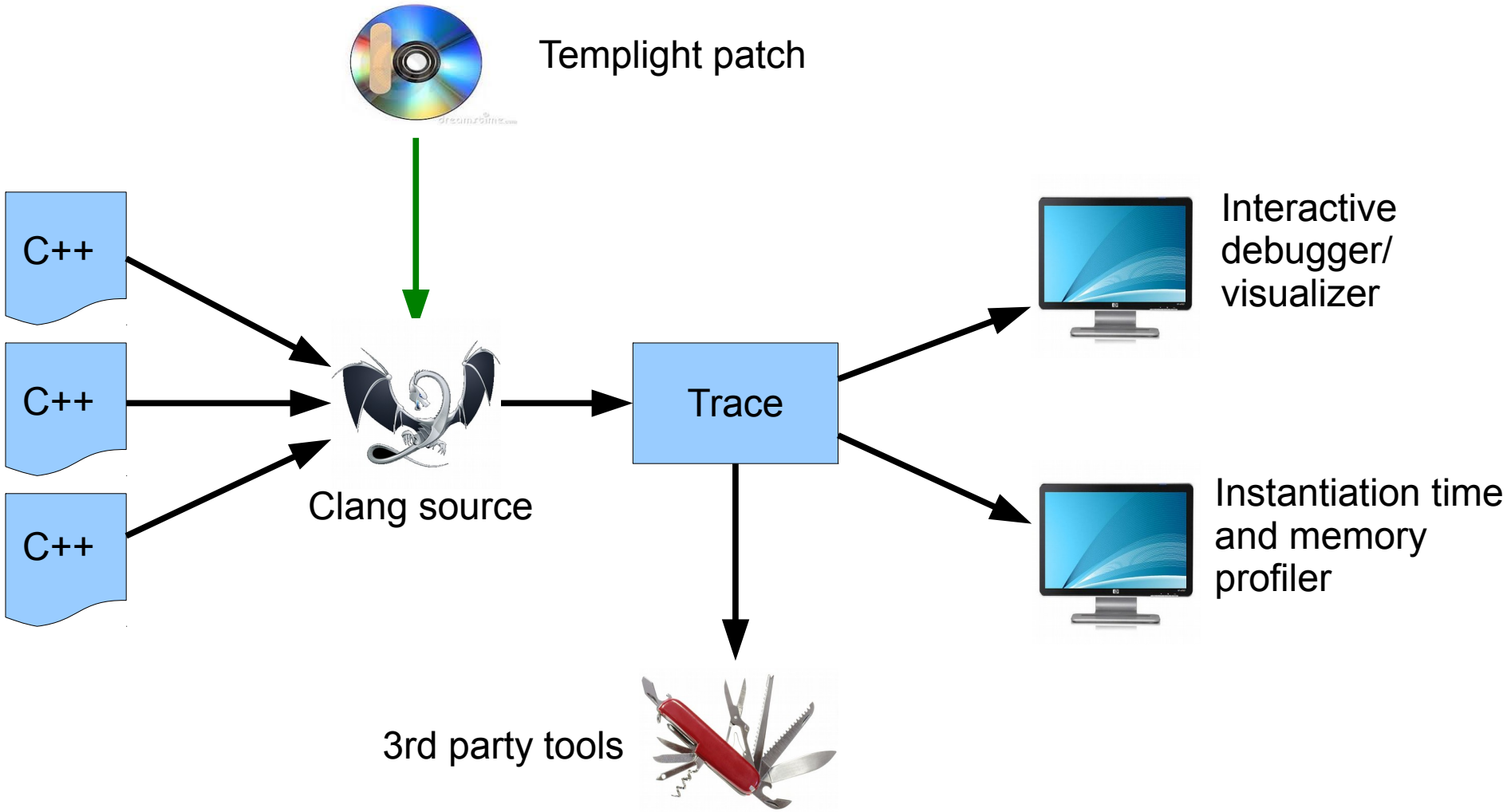
Instrumentation

- Advantages
  - Light-way approach (compared to compiler hack)
  - Grammar support (we used wave)
  - Easier to port: just change the warning generator
- Disadvantages
  - Complex constructs are hard (e.g. inheritance)
  - Serious distortion in profiling information
  - Memoization not detected

Templight 2.0

- Based on LLVM/Clang compiler infrastructure
- Patch to
  - Detect/measure instantiation
  - Detect memoization
  - Put timestamp on events
  - Measure memory consumption (optional)
- Emit trace in various formats (txt, YAML, XML)
- Front-end tools
  - Visual debugger
  - Profiler data viewer

Templight 2.0



Installation

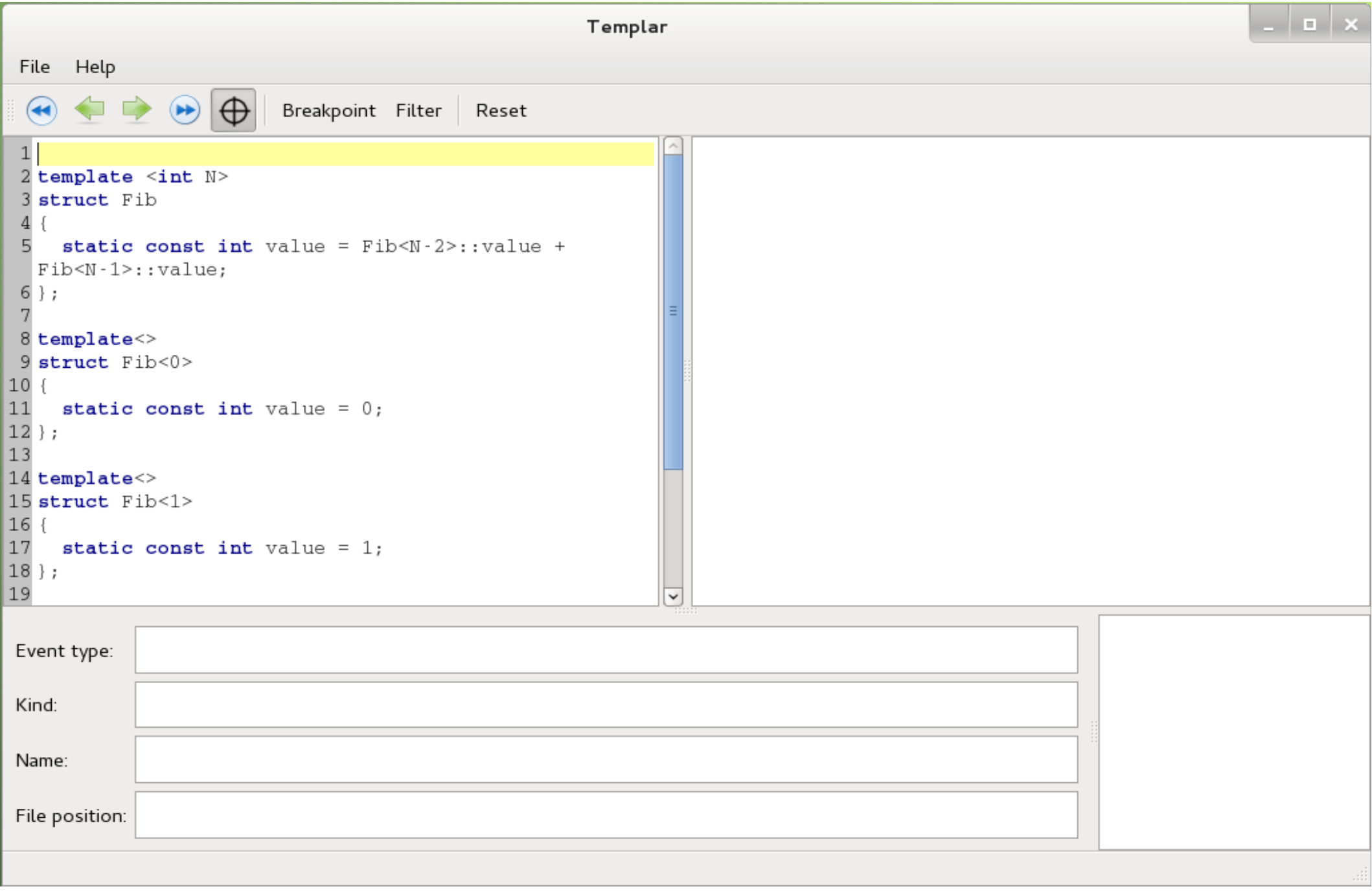
- Visit http://plc.inf.elte.hu/templight
- Download templight-<timestamp>.tar.gz
  - Contains clang patch and the two frontends
- Download Clang source
- Patch and build clang
- Build front-end tools (optional)
  - >=Qt 4.6 and >=Graphviz 2.28.0 required
  - $ qmake; make

How to use

```
template<int N>
struct Fib
{
  static const int value = Fib<N-2>::value + Fib<N-1>::value;
};
template<>
struct Fib<0>
{
  static const int value = 0;
};
template<>
struct Fib<1>
{
  static const int value = 1;
};
int main()
{
  static const int fib5 = Fib<5>::value;
}
```

How to use

```
$ clang++ -templight fib.cpp

$ ls
fib.cpp.trace.xml

$ wc fib.cpp.trace.xml
 123  275 3838 fib.cpp.trace.xml

$ head fib.cpp.trace.xml
<?xml version="1.0" standalone="yes"?>
<Trace>
<TemplateBegin>
    <Kind>TemplateInstantiation</Kind>
    <Context context = "Fib&lt;5&gt;"/>
    <PointOfInstantiation>fib.cpp|22|
14</PointOfInstantiation>
    <TimeStamp time = "421998401.188854"/>
    <MemoryUsage bytes = "0"/>
</TemplateBegin>
<TemplateBegin>
```

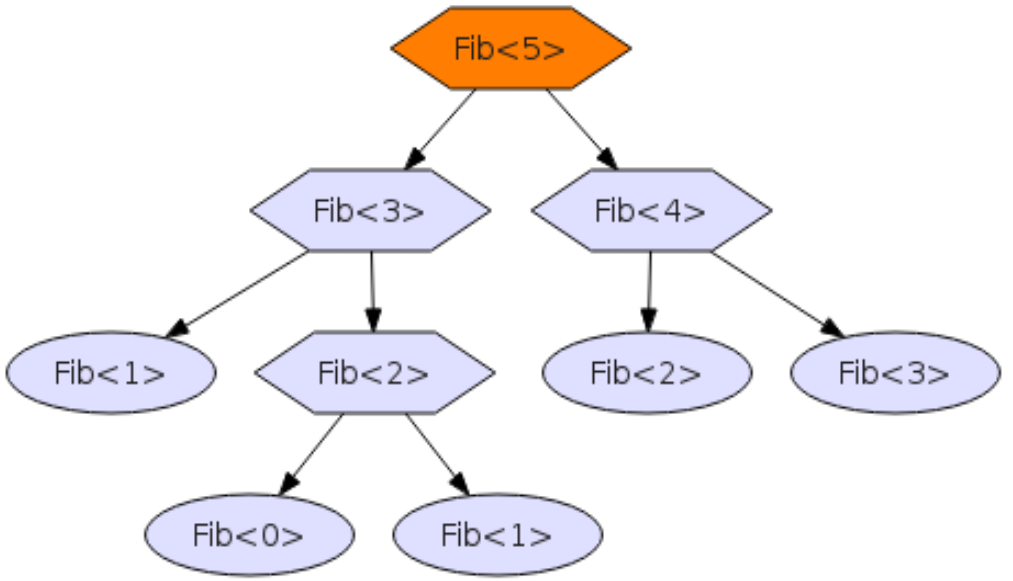

Templar
File
Help
Breakpoint
Filter
Reset
1
2 template <int N>
3 struct Fib
4 {
5 static const int value = Fib<N-2>::value +
Fib<N-1>::value;
6 };
7
8 template<>
9 struct Fib<0>
10 {
11 static const int value = 0;
12 };
13
14 template<>
15 struct Fib<1>
16 {
17 static const int value = 1;
18 };
19
Event type:
Kind:
Name:
File position:

Templar

File Help

Breakpoint Filter Reset

```
10 {
11   static const int value = 0;
12 };
13
14 template<>
15 struct Fib<1>
16 {
17   static const int value = 1;
18 };
19
20 int main()
21 {
22   int fib5 = Fib<5>::value;
23 }
24
25
```



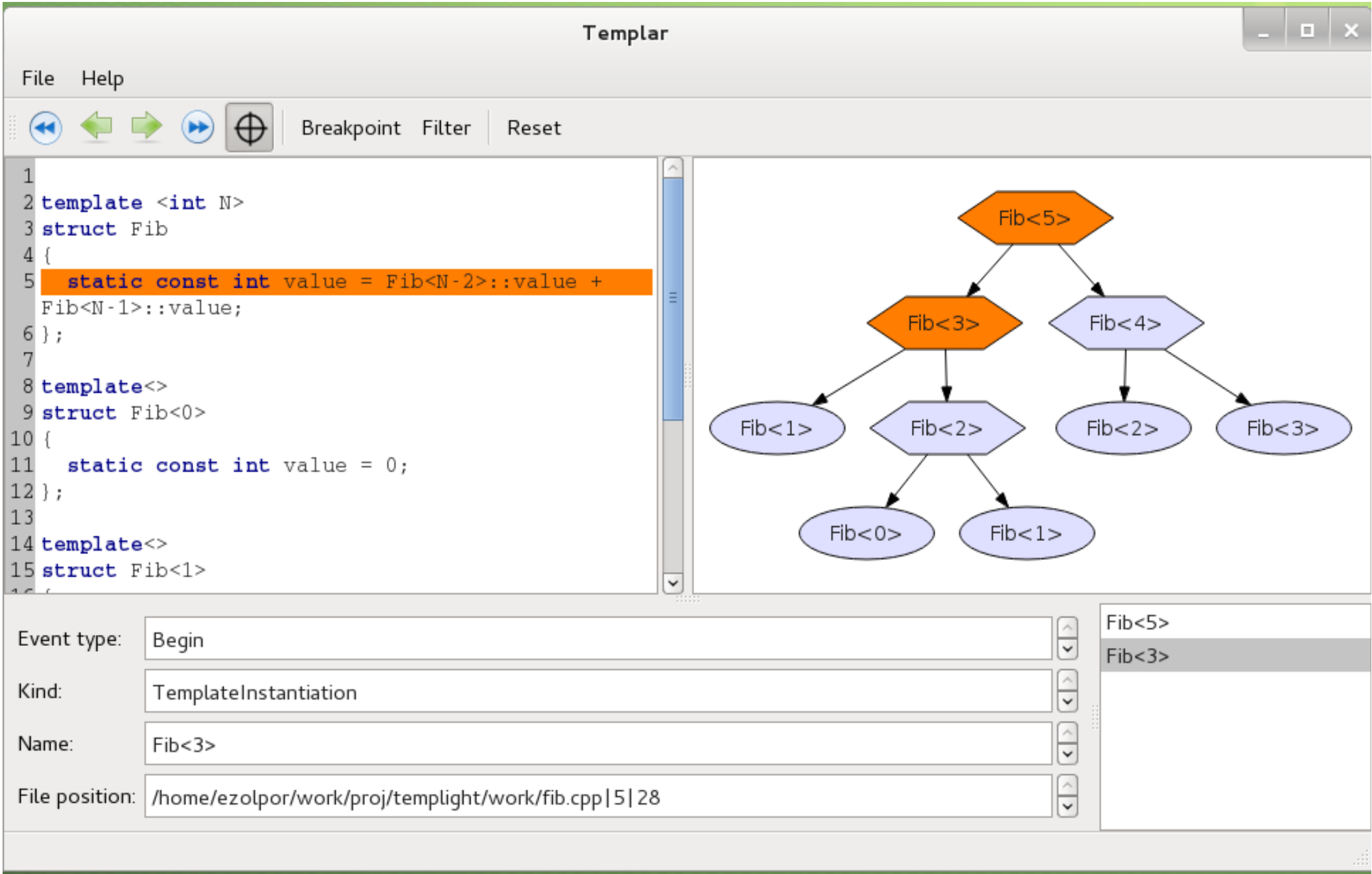
Event type: Begin

Kind: TemplateInstantiation

Name: Fib<5>

File position: /home/ezolpor/work/proj/templight/work/fib.cpp|22|14

Fib<5>

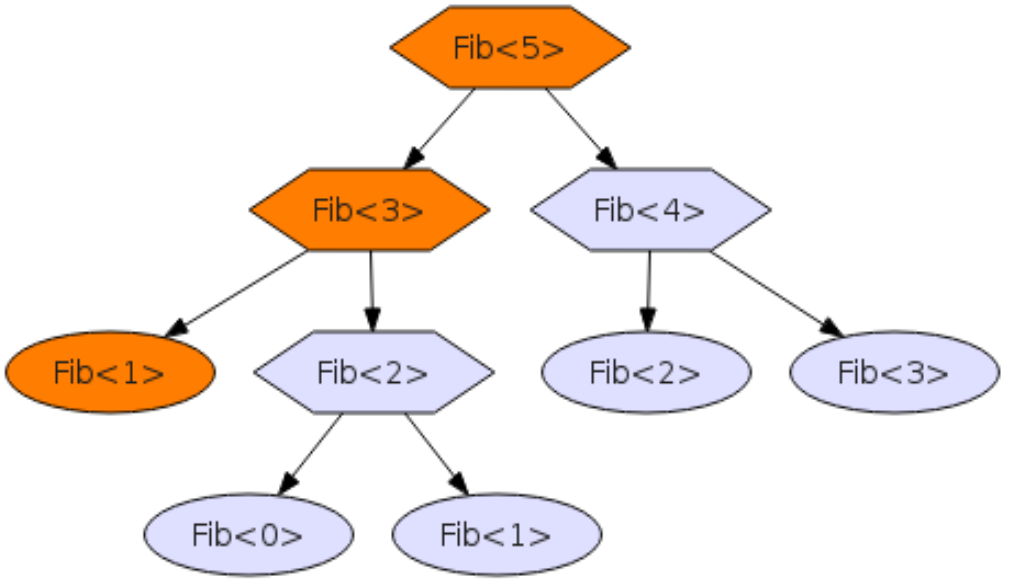

Templar
File
Help
Breakpoint
Filter
Reset
1
2 template <int N>
3 struct Fib
4 {
5 static const int value = Fib<N-2>::value +
Fib<N-1>::value;
6 };
7
8 template<>
9 struct Fib<0>
10 {
11 static const int value = 0;
12 };
13
14 template<>
15 struct Fib<1>
Fib<5>
Fib<3>
Fib<4>
Fib<1>
Fib<2>
Fib<2>
Fib<3>
Fib<0>
Fib<1>
Event type:
Begin
Kind:
TemplateInstantiation
Name:
Fib<3>
File position:
/home/ezolpor/work/proj/templight/work/fib.cpp|5|28
Fib<5>
Fib<3>

Templar

File Help

Breakpoint Filter Reset

```
1
2 template <int N>
3 struct Fib
4 {
5   static const int value = Fib<N-2>::value +
  Fib<N-1>::value;
6 };
7
8 template<>
9 struct Fib<0>
10 {
11   static const int value = 0;
12 };
13
14 template<>
15 struct Fib<1>
```



Event type: Begin

Kind: Memoization

Name: Fib<1>

File position: /home/ezolpor/work/proj/templight/work/fib.cpp|5|28

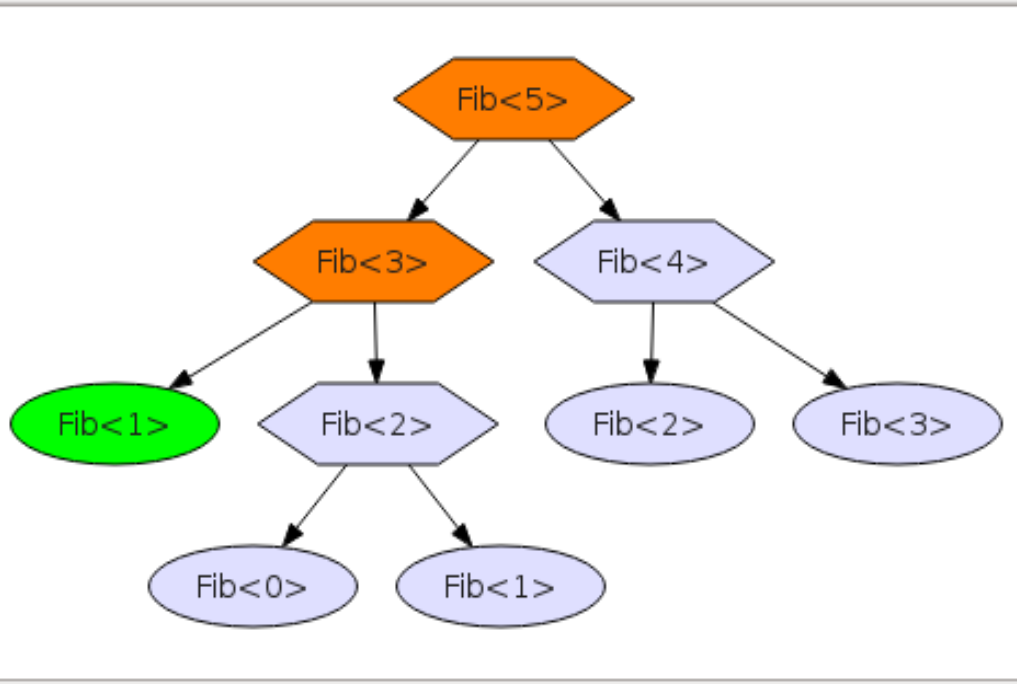
Fib<5>
Fib<3>
Fib<1>

Templar

File Help

Breakpoint Filter Reset

```
1
2 template <int N>
3 struct Fib
4 {
5   static const int value = Fib<N-2>::value +
  Fib<N-1>::value;
6 };
7
8 template<>
9 struct Fib<0>
10 {
11   static const int value = 0;
12 };
13
14 template<>
15 struct Fib<1>
```



Event type: End

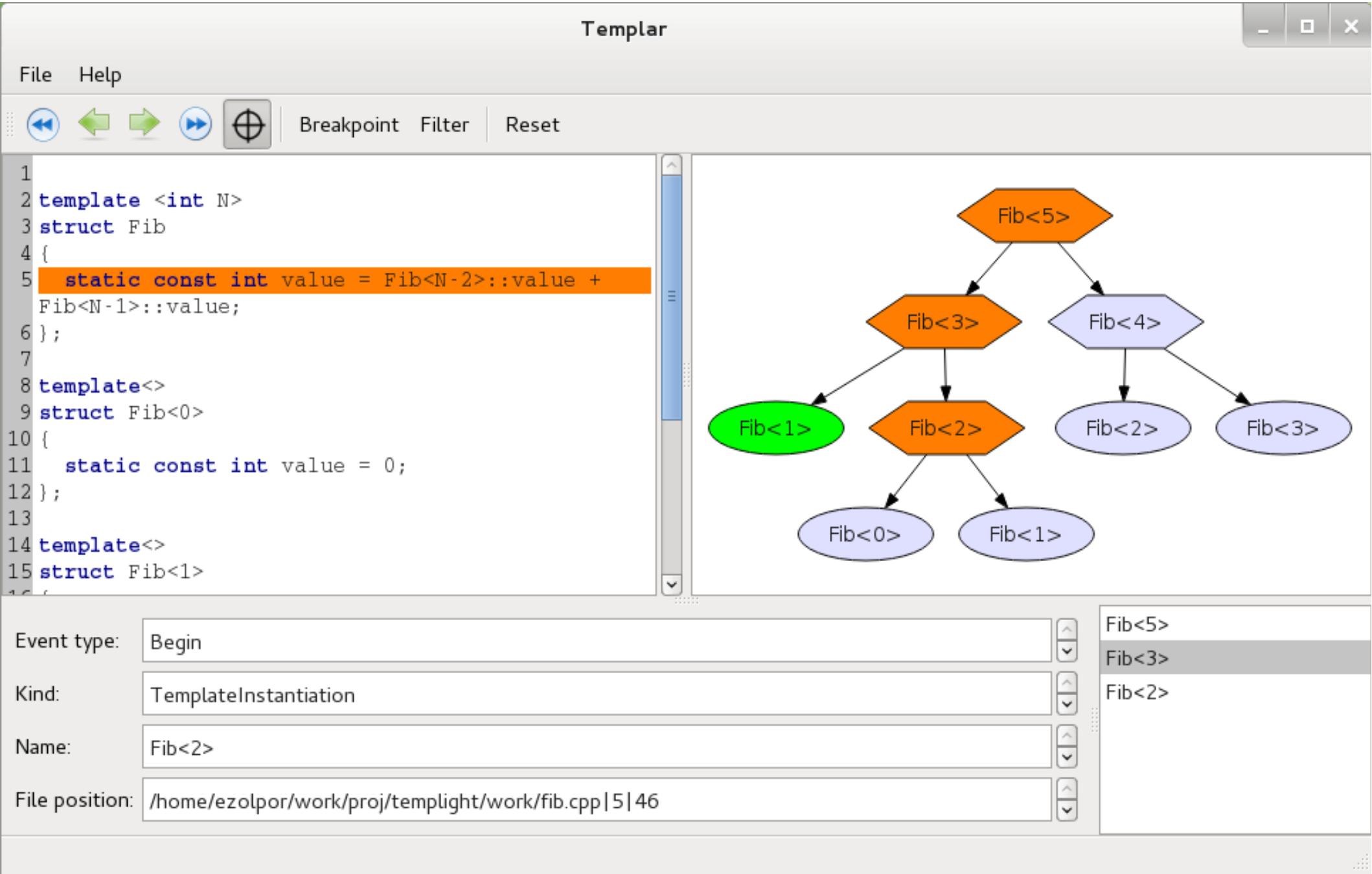
Kind: Memoization

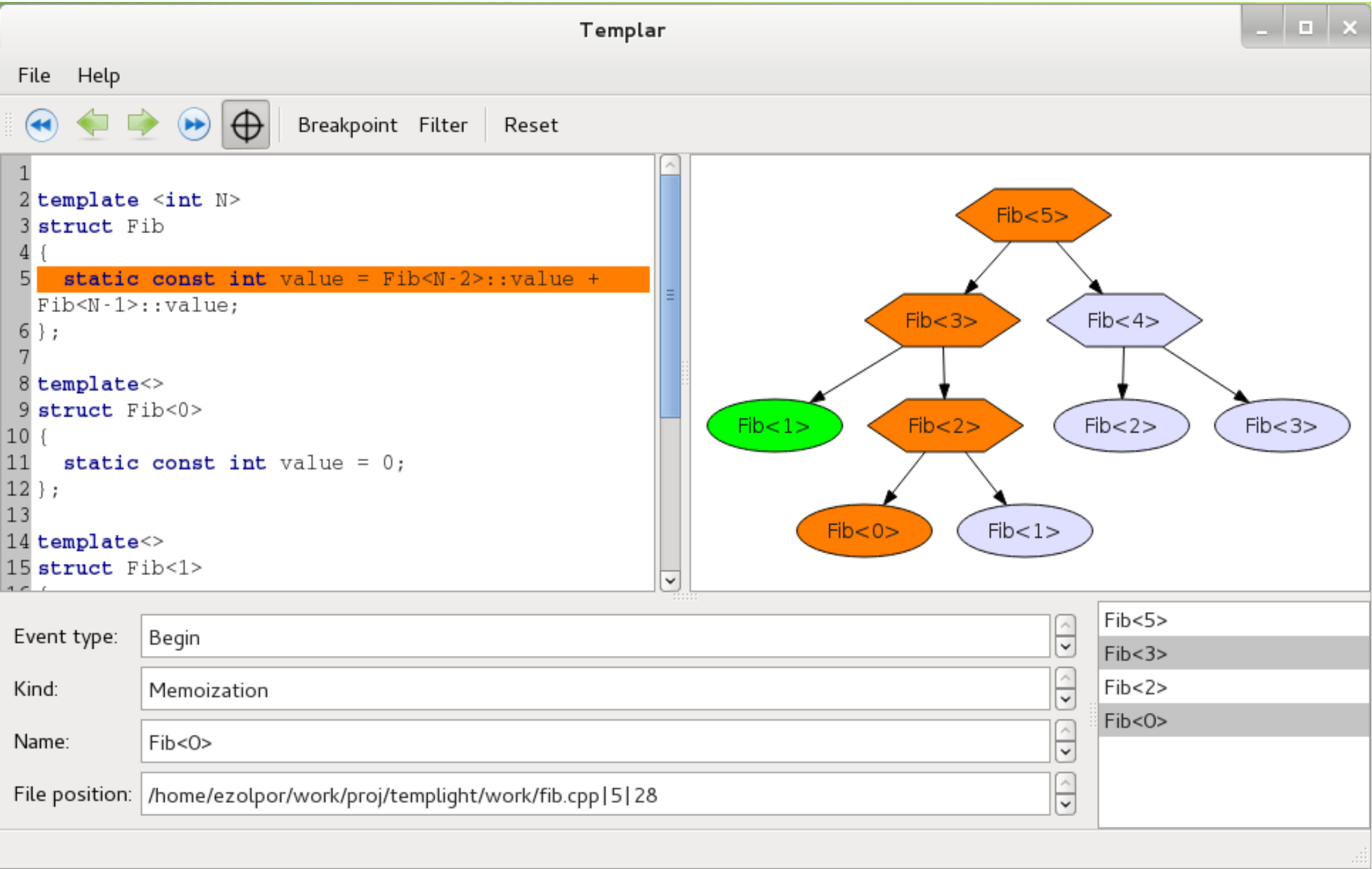
Name: Fib<1>

File position: /home/ezolpor/work/proj/templight/work/fib.cpp|5|28

Fib<5>

Fib<3>


Templar
File
Help
Breakpoint
Filter
Reset
1
2 template <int N>
3 struct Fib
4 {
5 static const int value = Fib<N-2>::value +
Fib<N-1>::value;
6 };
7
8 template<>
9 struct Fib<0>
10 {
11 static const int value = 0;
12 };
13
14 template<>
15 struct Fib<1>
Fib<5>
Fib<3>
Fib<4>
Fib<1>
Fib<2>
Fib<2>
Fib<3>
Fib<0>
Fib<1>
Event type:
Begin
Kind:
TemplateInstantiation
Name:
Fib<2>
File position:
/home/ezolpor/work/proj/templight/work/fib.cpp|5|46
Fib<5>
Fib<3>
Fib<2>

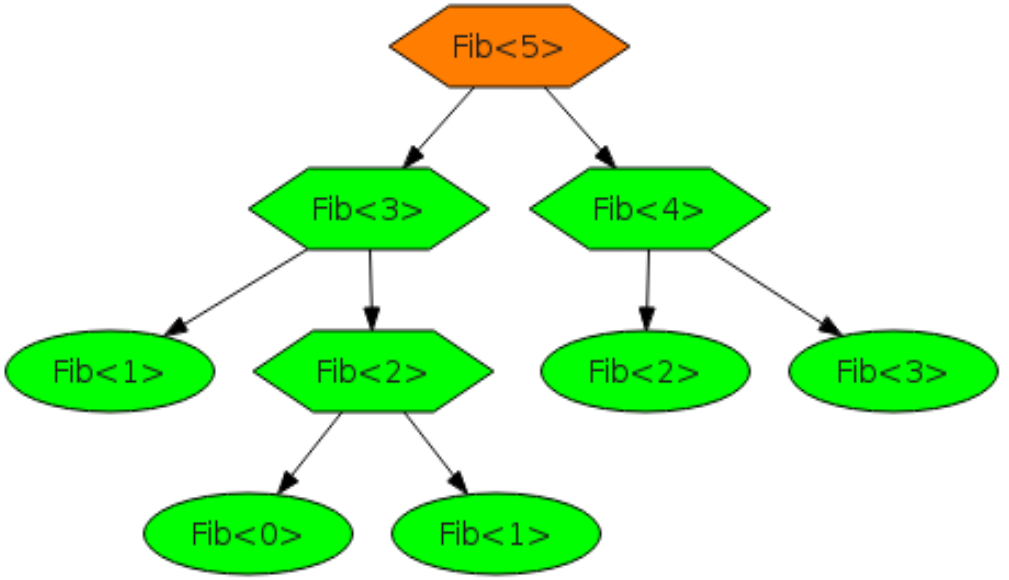

Templar
File
Help
Breakpoint
Filter
Reset
1
2 template <int N>
3 struct Fib
4 {
5 static const int value = Fib<N-2>::value +
Fib<N-1>::value;
6 };
7
8 template<>
9 struct Fib<0>
10 {
11 static const int value = 0;
12 };
13
14 template<>
15 struct Fib<1>
Fib<5>
Fib<3>
Fib<4>
Fib<1>
Fib<2>
Fib<2>
Fib<3>
Fib<0>
Fib<1>
Event type:
Begin
Kind:
Memoization
Name:
Fib<0>
File position:
/home/ezolpor/work/proj/templight/work/fib.cpp|5|28
Fib<5>
Fib<3>
Fib<2>
Fib<0>

Templar

File Help

Breakpoint Filter Reset

```
1
2 template <int N>
3 struct Fib
4 {
5   static const int value = Fib<N-2>::value +
  Fib<N-1>::value;
6 };
7
8 template<>
9 struct Fib<0>
10 {
11   static const int value = 0;
12 };
13
14 template<>
15 struct Fib<1>
```



Event type: End

Kind: TemplateInstantiation

Name: Fib<4>

File position: /home/ezolpor/work/proj/templight/work/fib.cpp|5|46

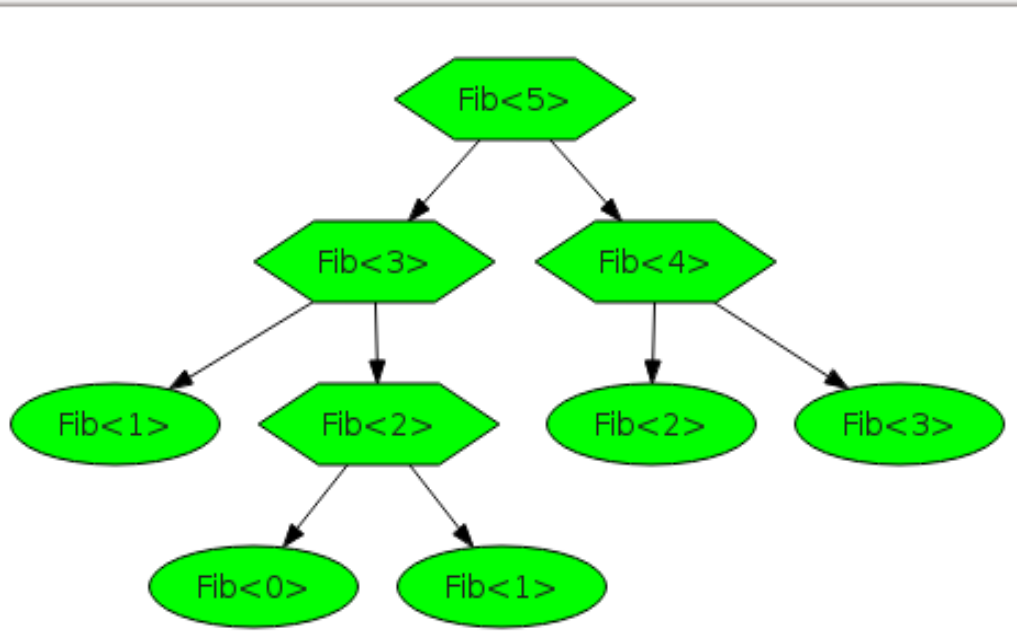
Fib<5>

Templar

File Help

Breakpoint Filter Reset

```
10 {
11   static const int value = 0;
12 };
13
14 template<>
15 struct Fib<1>
16 {
17   static const int value = 1;
18 };
19
20 int main()
21 {
22   int fib5 = Fib<5>::value;
23 }
24
25
```

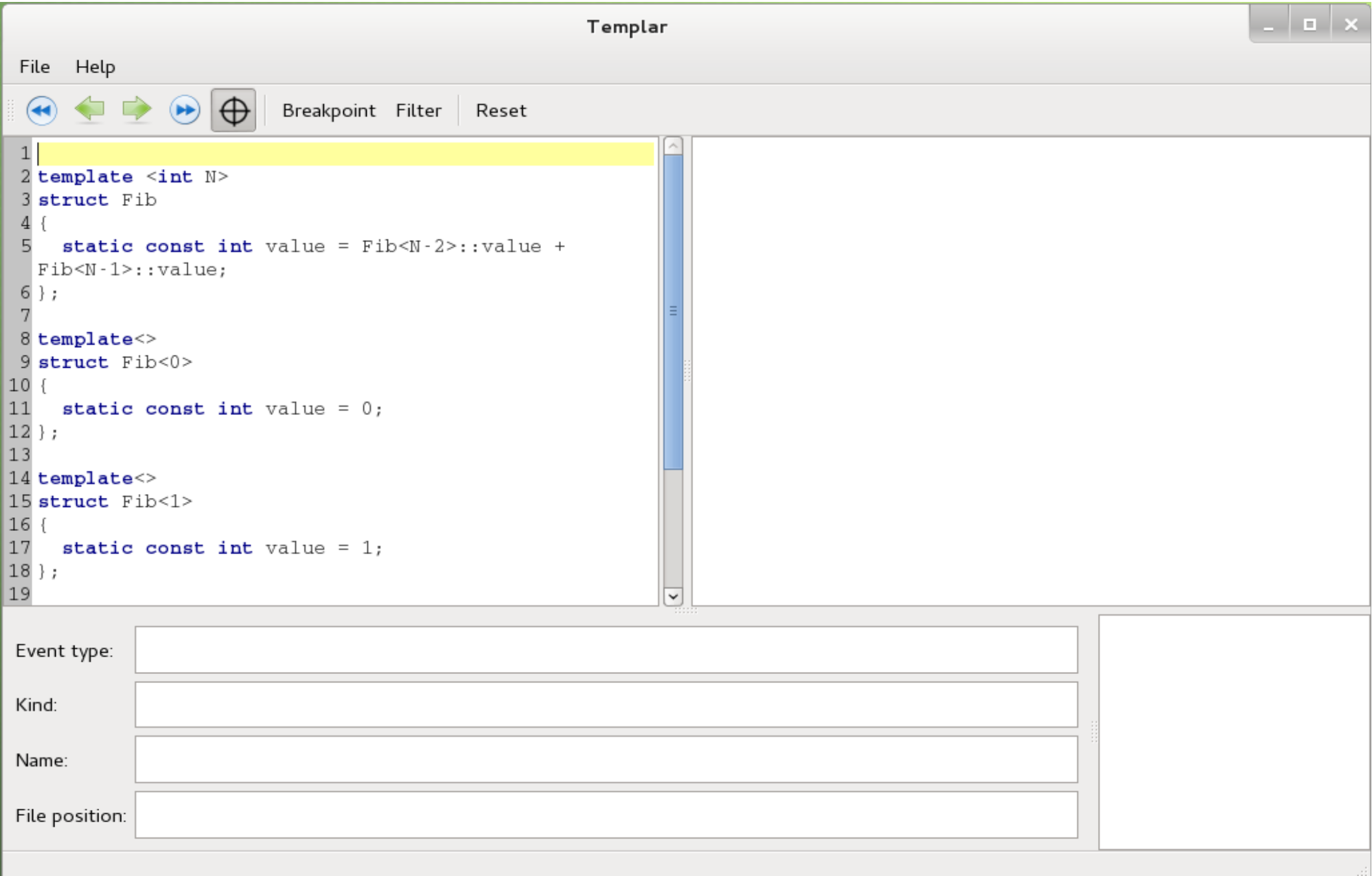


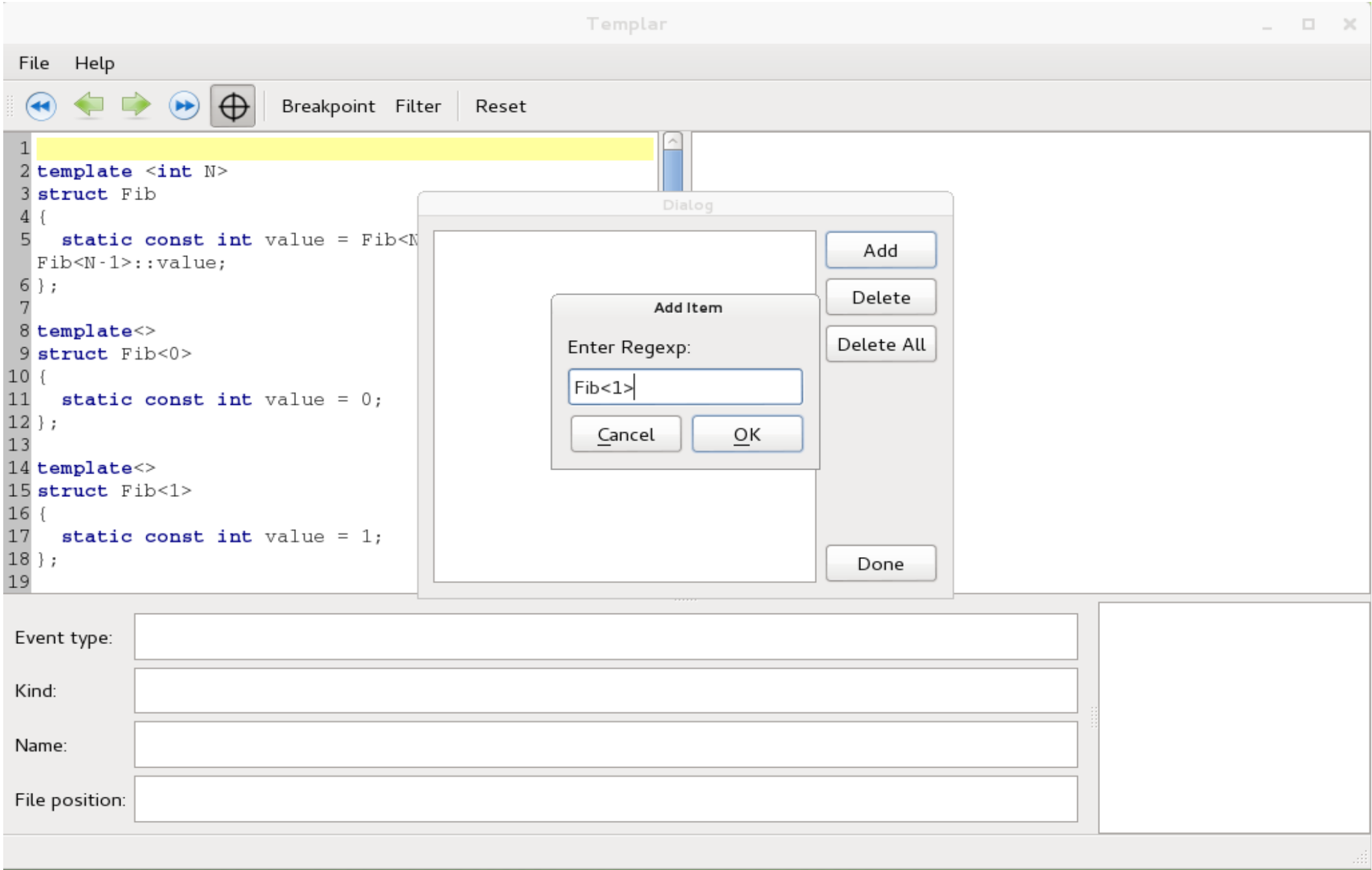
Event type: End

Kind: TemplateInstantiation

Name: Fib<5>

File position: /home/ezolpor/work/proj/templight/work/fib.cpp|22|14


Templar
File
Help
Breakpoint
Filter
Reset
1
2 template <int N>
3 struct Fib
4 {
5 static const int value = Fib<N-2>::value +
Fib<N-1>::value;
6 };
7
8 template<>
9 struct Fib<0>
10 {
11 static const int value = 0;
12 };
13
14 template<>
15 struct Fib<1>
16 {
17 static const int value = 1;
18 };
19
Event type:
Kind:
Name:
File position:

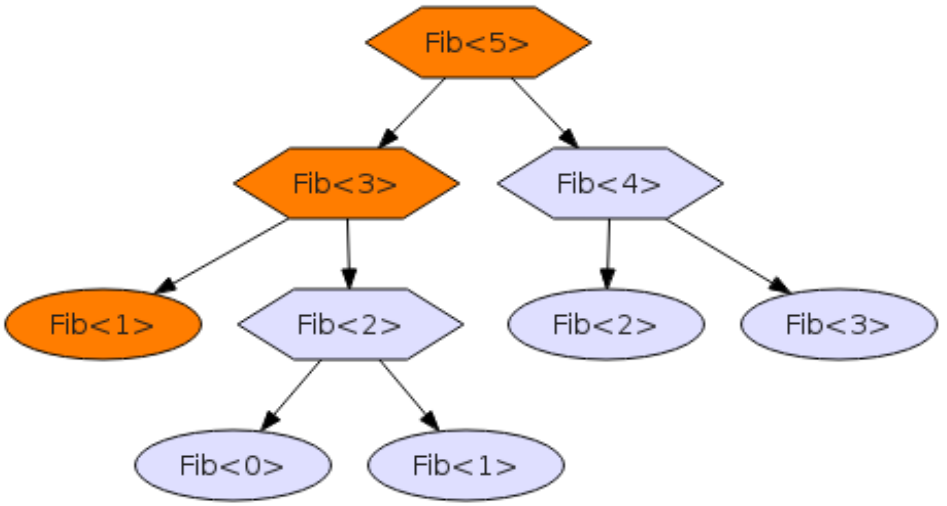

Templar
File
Help
Breakpoint
Filter
Reset
1
2 template <int N>
3 struct Fib
4 {
5 static const int value = Fib<N
Fib<N-1>::value;
6 };
7
8 template<>
9 struct Fib<0>
10 {
11 static const int value = 0;
12 };
13
14 template<>
15 struct Fib<1>
16 {
17 static const int value = 1;
18 };
19
Dialog
Add
Delete
Delete All
Done
Add Item
Enter Regexp:
Fib<1>
Cancel
OK
Event type:
Kind:
Name:
File position:

Templar

File Help

Breakpoint Filter Reset

```
1
2  template <int N>
3  struct Fib
4  {
5    static const int value = Fib<N-2>::value +
   Fib<N-1>::value;
6  };
7
8  template<>
9  struct Fib<0>
10 {
11   static const int value = 0;
12 };
13
14 template<>
15 struct Fib<1>
16 {
17   static const int value = 1;
18 };
19
```



Event type: Begin

Kind: Memoization

Name: Fib<1>

File position: /home/ezolpor/work/proj/templight/work/fib.cpp|5|28

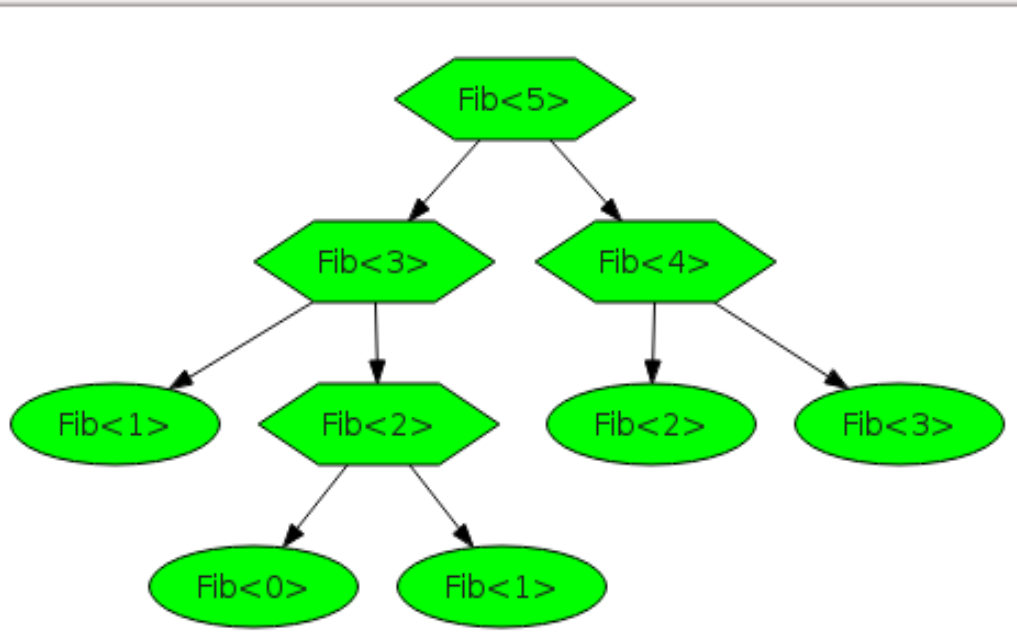
Fib<5>
Fib<3>
Fib<1>

Templar

File Help

Breakpoint Filter Reset

```
10 {
11   static const int value = 0;
12 };
13
14 template<>
15 struct Fib<1>
16 {
17   static const int value = 1;
18 };
19
20 int main()
21 {
22   int fib5 = Fib<5>::value;
23 }
24
25
```



Event type: End

Kind: TemplateInstantiation

Name: Fib<5>

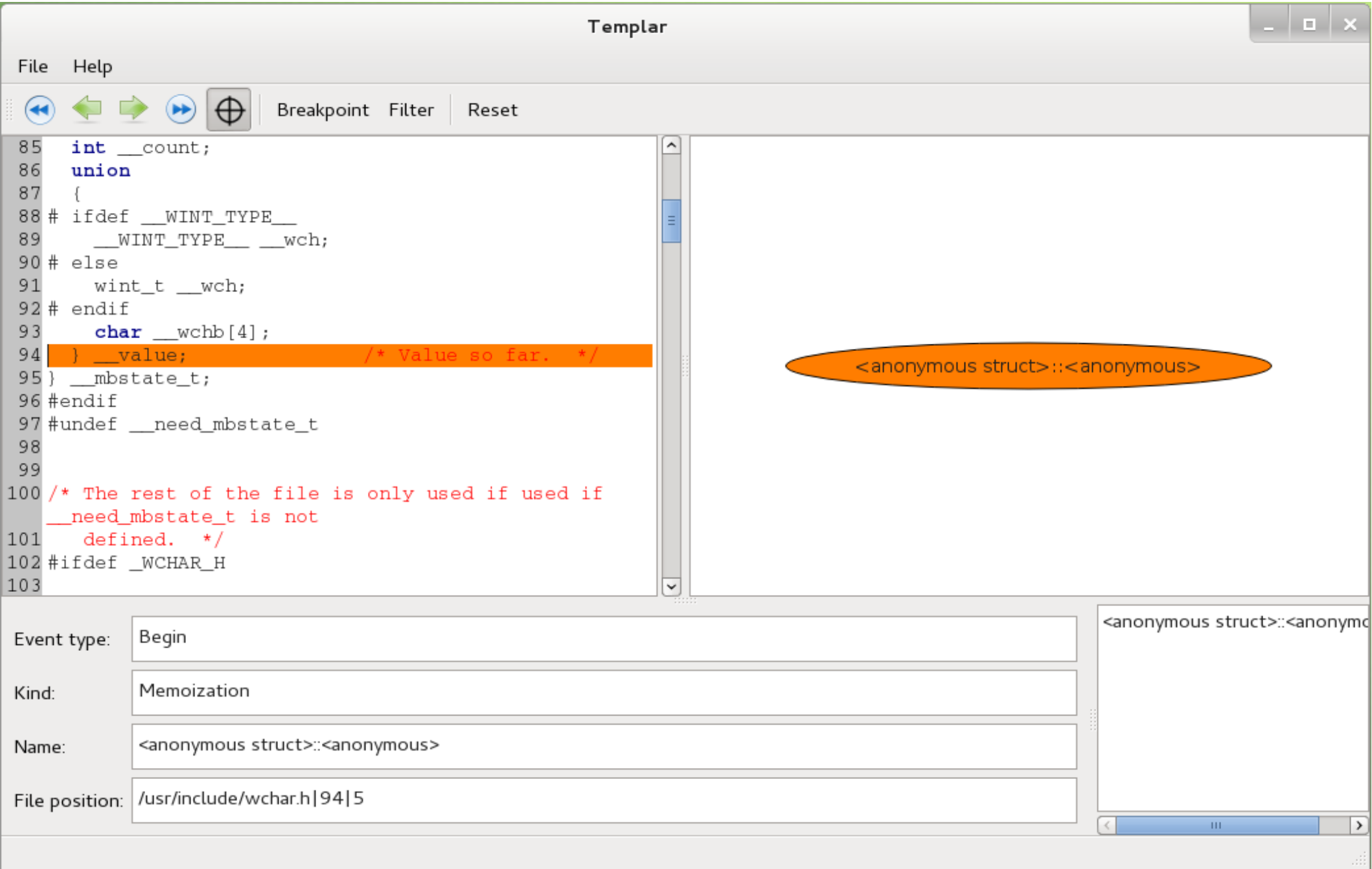
File position: /home/ezolpor/work/proj/templight/work/fib.cpp|22|14

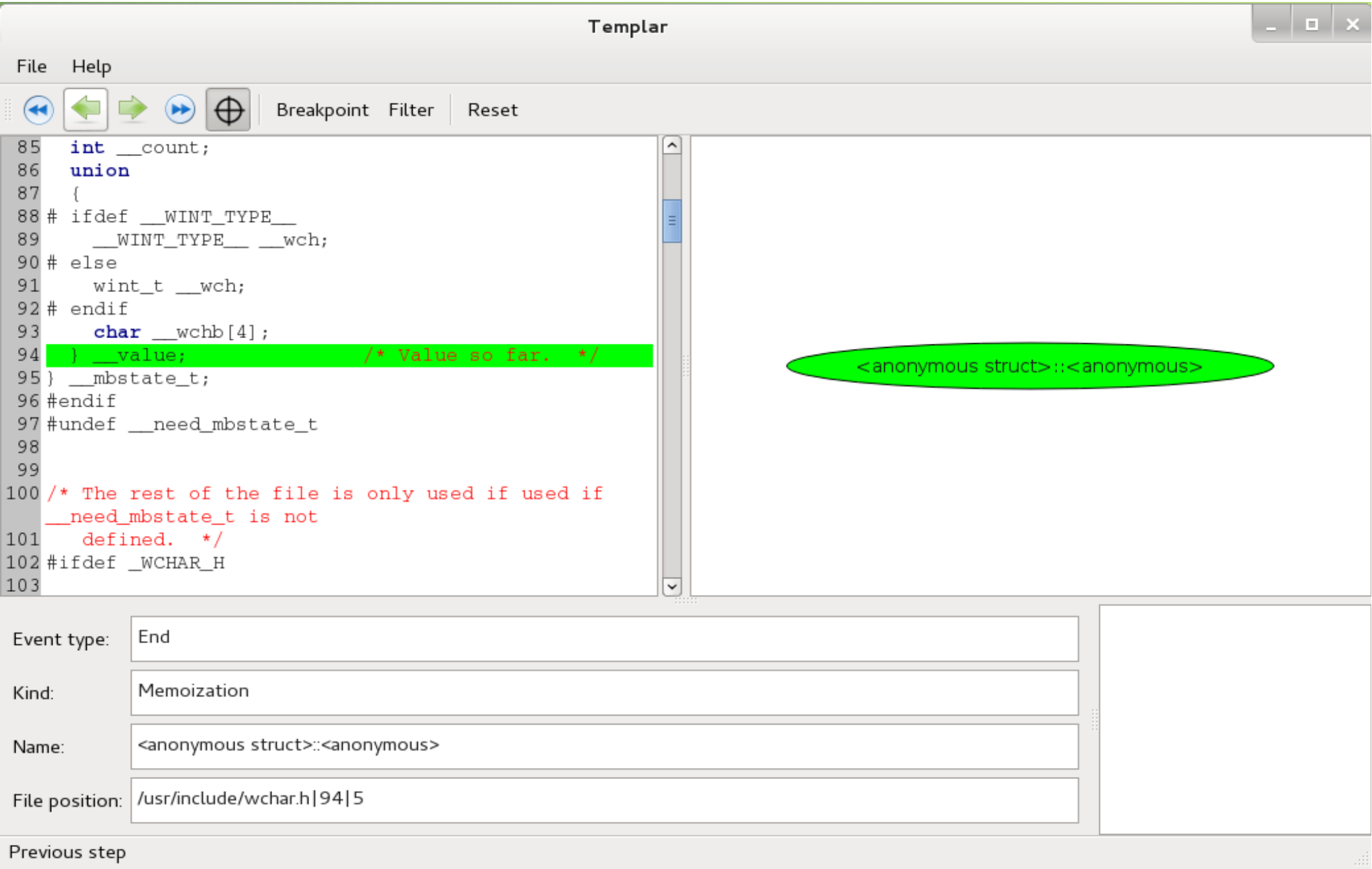
Filters

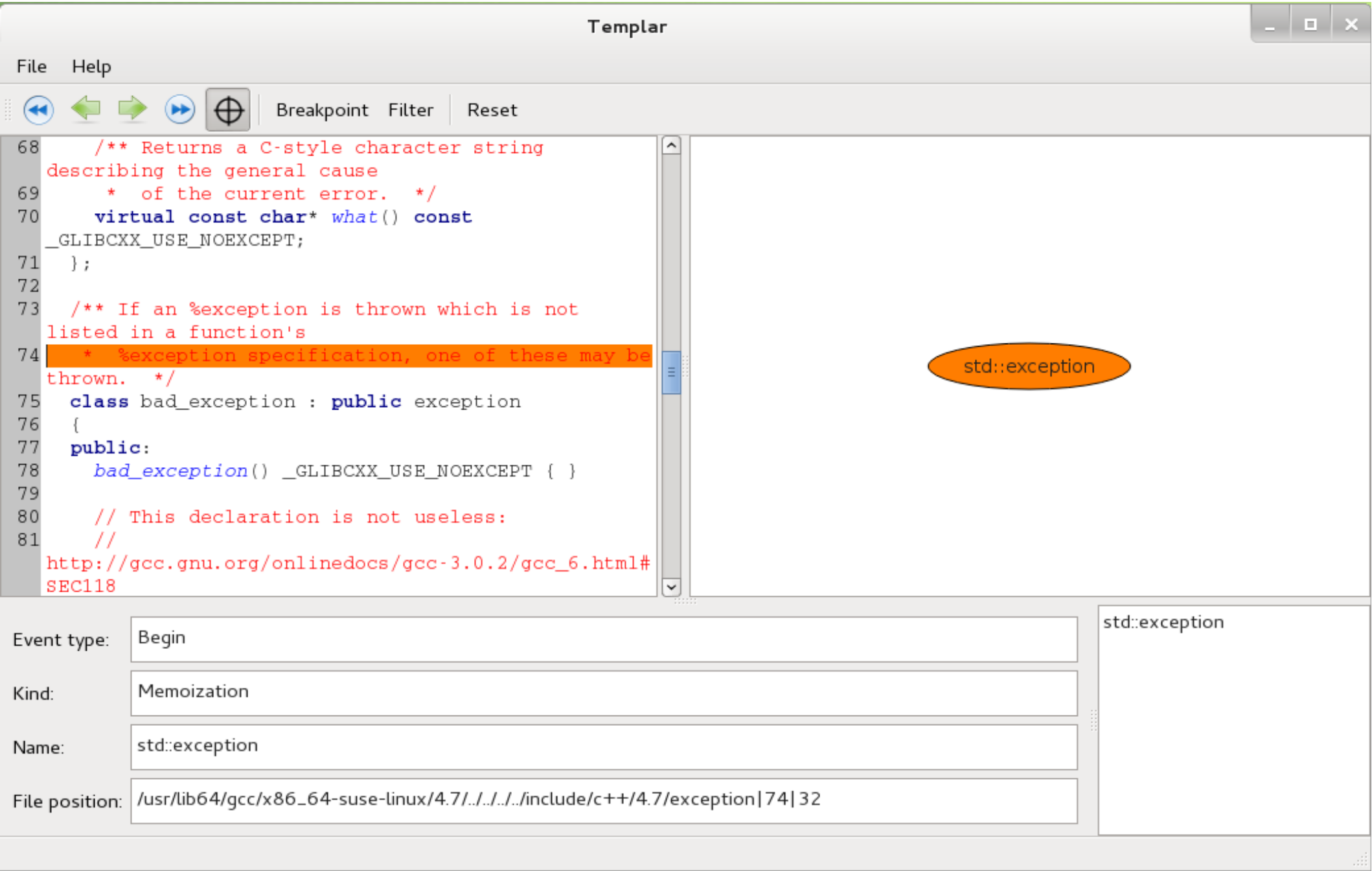
```
#include <iostream>

template<int N>
struct Fib
{
  static const int value = Fib<N-2>::value + Fib<N-1>::value;
};
template<>
struct Fib<0>
{
  static const int value = 0;
};
template<>
struct Fib<1>
{
  static const int value = 1;
};
int main()
{
  std::cout << Fib<5>::value << std::endl;
  return 0;
}
```

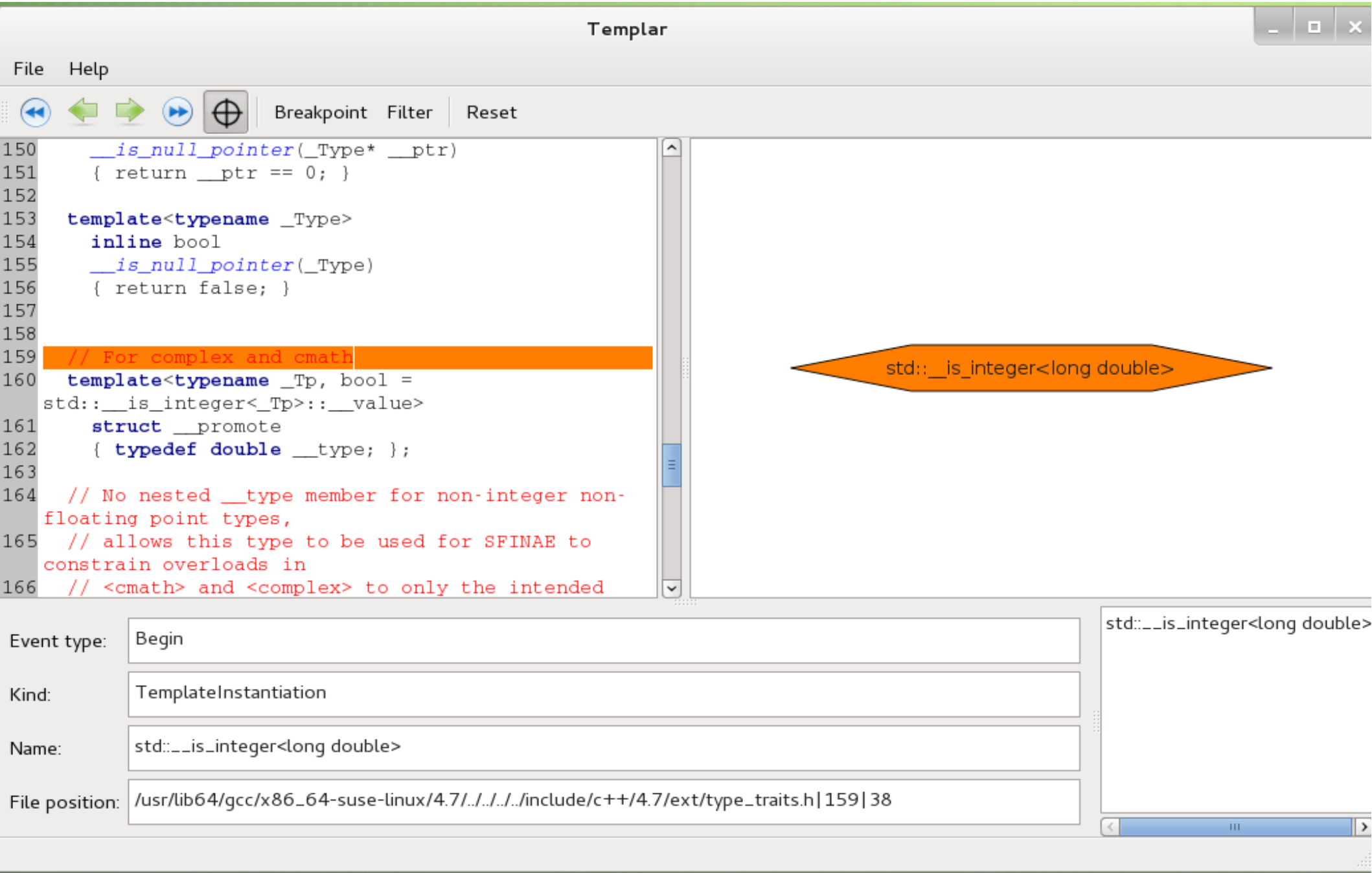
Filters

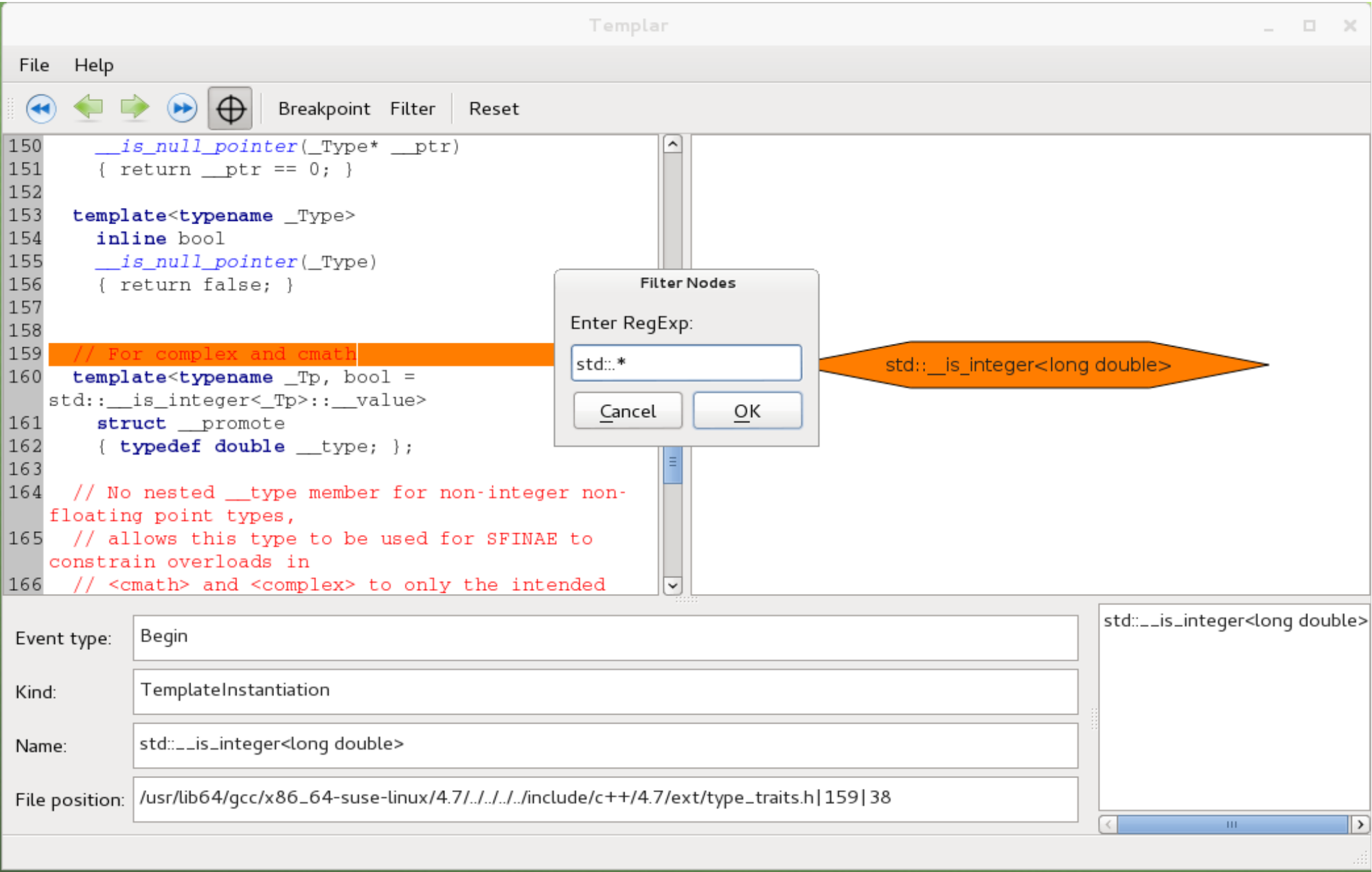
```
$ clang++ -templight fib.cpp

$ ls
fib.cpp.trace.xml

$ wc fib.cpp.trace.xml
18291  41765 738233 fib.cpp.trace.xml

$ head fib.cpp.trace.xml
<?xml version="1.0" standalone="yes"?>
<Trace>
<TemplateBegin>
    <Kind>DefaultTemplateArgumentInstantiation</Kind>
    <Context context = "std::basic_string"/>
    <PointOfInstantiation>/usr/lib64/gcc/x86_64-suse-
linux/4.7/../../../../include/c++/4.7/bits/stringfwd.h|64|
11</PointOfInstantiation>
    <TimeStamp time = "421999330.595354"/>
```


Templar
File
Help
Breakpoint
Filter
Reset
85 int __count;
86 union
87 {
88 # ifdef __WINT_TYPE__
89 __WINT_TYPE__ __wch;
90 # else
91 wint_t __wch;
92 # endif
93 char __wchb[4];
94 } __value; /* Value so far. */
95 } __mbstate_t;
96 #endif
97 #undef __need_mbstate_t
98
99
100 /* The rest of the file is only used if used if
__need_mbstate_t is not
101 defined. */
102 #ifdef _WCHAR_H
103
<anonymous struct>::<anonymous>
Event type:
Begin
Kind:
Memoization
Name:
<anonymous struct>::<anonymous>
File position:
/usr/include/wchar.h|94|5
<anonymous struct>::<anonymo


Templar
File
Help
Breakpoint
Filter
Reset
85 int __count;
86 union
87 {
88 # ifdef __WINT_TYPE__
89 __WINT_TYPE__ __wch;
90 # else
91 wint_t __wch;
92 # endif
93 char __wchb[4];
94 } __value; /* Value so far. */
95 } __mbstate_t;
96 #endif
97 #undef __need_mbstate_t
98
99
100 /* The rest of the file is only used if used if
__need_mbstate_t is not
101 defined. */
102 #ifdef _WCHAR_H
103
<anonymous struct>::<anonymous>
Event type:
End
Kind:
Memoization
Name:
<anonymous struct>::<anonymous>
File position:
/usr/include/wchar.h|94|5
Previous step


Templar
File
Help
Breakpoint
Filter
Reset
68 /** Returns a C-style character string describing the general cause
69 * of the current error. */
70 virtual const char* what() const _GLIBCXX_USE_NOEXCEPT;
71 };
72
73 /** If an %exception is thrown which is not listed in a function's
74 * %exception specification, one of these may be thrown. */
75 class bad_exception : public exception
76 {
77 public:
78 bad_exception() _GLIBCXX_USE_NOEXCEPT { }
79
80 // This declaration is not useless:
81 //
http://gcc.gnu.org/onlinedocs/gcc-3.0.2/gcc_6.html#SEC118
std::exception
Event type:
Begin
Kind:
Memoization
Name:
std::exception
File position:
/usr/lib64/gcc/x86_64-suse-linux/4.7/../../../../include/c++/4.7/exception|74|32
std::exception


Templar
File
Help
Breakpoint
Filter
Reset
150 __is_null_pointer(_Type* __ptr)
151 { return __ptr == 0; }
152
153 template<typename _Type>
154 inline bool
155 __is_null_pointer(_Type)
156 { return false; }
157
158
159 // For complex and cmath
160 template<typename _Tp, bool =
std::__is_integer<_Tp>::__value>
161 struct __promote
162 { typedef double __type; };
163
164 // No nested __type member for non-integer non-
floating point types,
165 // allows this type to be used for SFINAE to
constrain overloads in
166 // <cmath> and <complex> to only the intended
std::__is_integer<long double>
Event type:
Begin
Kind:
TemplateInstantiation
Name:
std::__is_integer<long double>
File position:
/usr/lib64/gcc/x86_64-suse-linux/4.7/../../../../include/c++/4.7/ext/type_traits.h|159|38
std::__is_integer<long double>

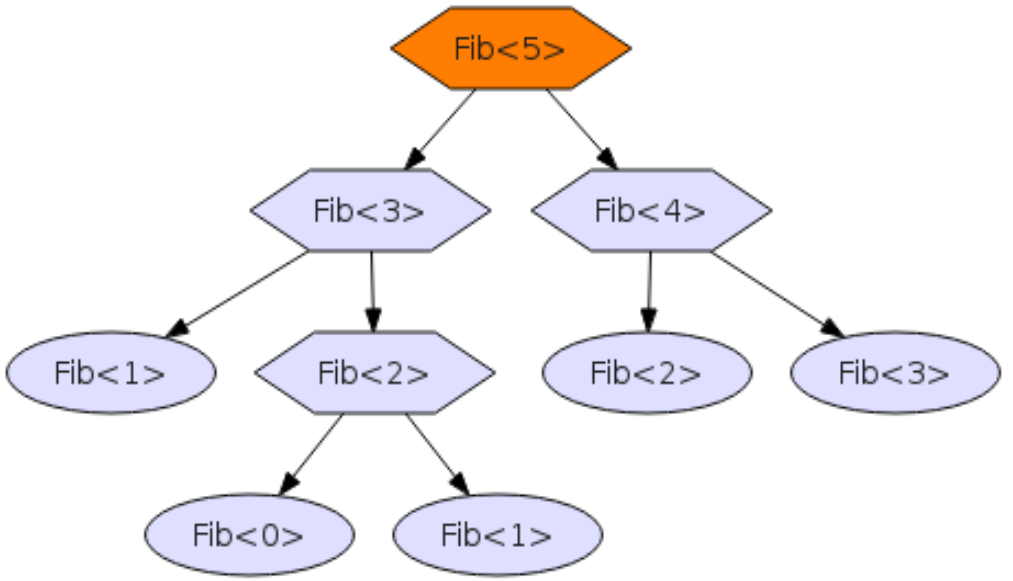

Templar
File
Help
Breakpoint
Filter
Reset
150 __is_null_pointer(_Type* __ptr)
151 { return __ptr == 0; }
152
153 template<typename _Type>
154 inline bool
155 __is_null_pointer(_Type)
156 { return false; }
157
158
159 // For complex and cmath
160 template<typename _Tp, bool =
std::__is_integer<_Tp>::__value>
161 struct __promote
162 { typedef double __type; };
163
164 // No nested __type member for non-integer non-
floating point types,
165 // allows this type to be used for SFINAE to
constrain overloads in
166 // <cmath> and <complex> to only the intended
Filter Nodes
Enter RegExp:
std::.*
Cancel
OK
std::__is_integer<long double>
Event type:
Begin
Kind:
TemplateInstantiation
Name:
std::__is_integer<long double>
File position:
/usr/lib64/gcc/x86_64-suse-linux/4.7/../../../../include/c++/4.7/ext/type_traits.h|159|38
std::__is_integer<long double>

Templar

File Help

Breakpoint Filter Reset

```
10 {
11   static const int value = 0;
12 };
13
14 template<>
15 struct Fib<1>
16 {
17   static const int value = 1;
18 };
19
20 int main()
21 {
22   int fib5 = Fib<5>::value;
23 }
24
25
```



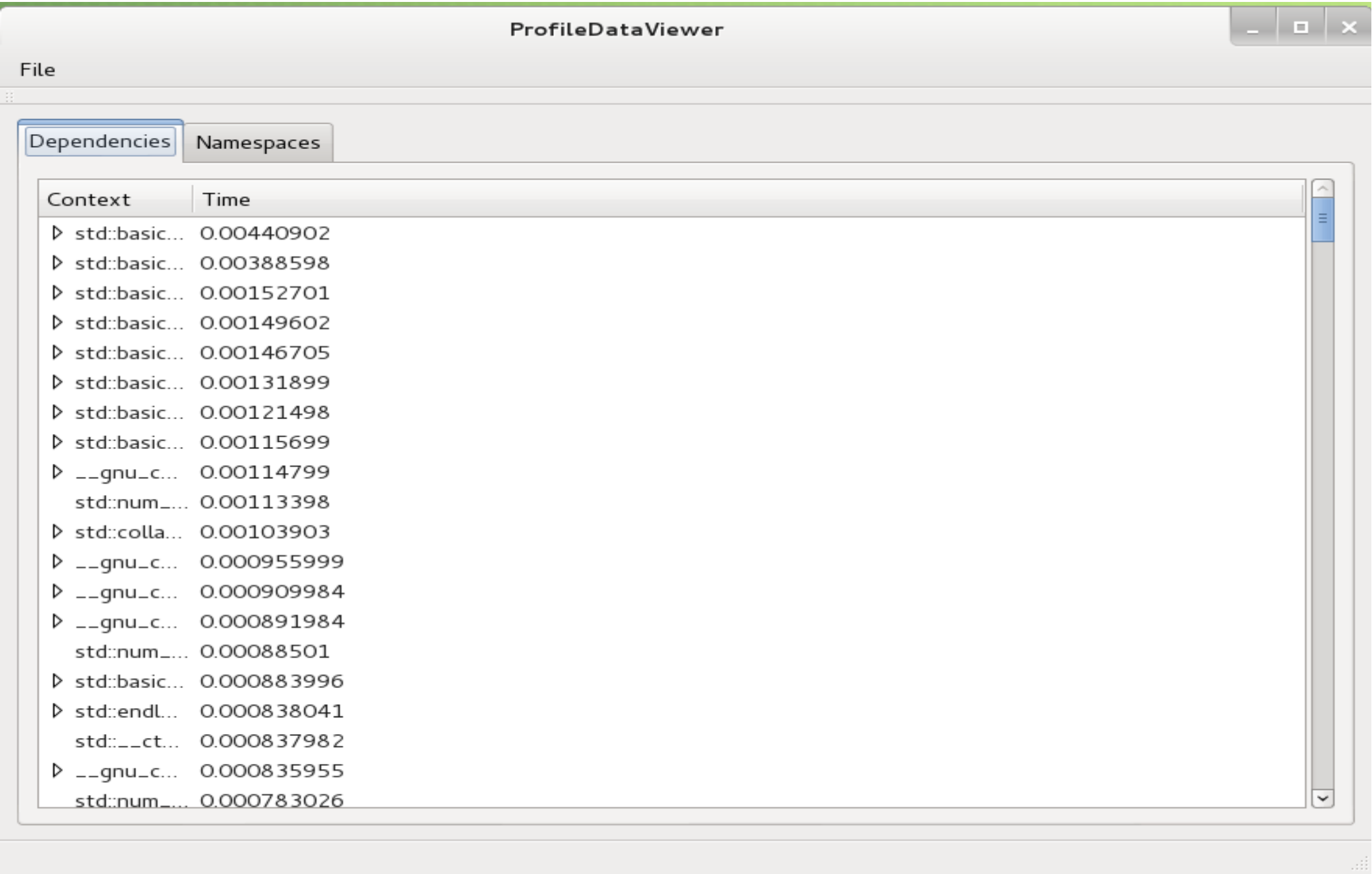
Event type: Begin

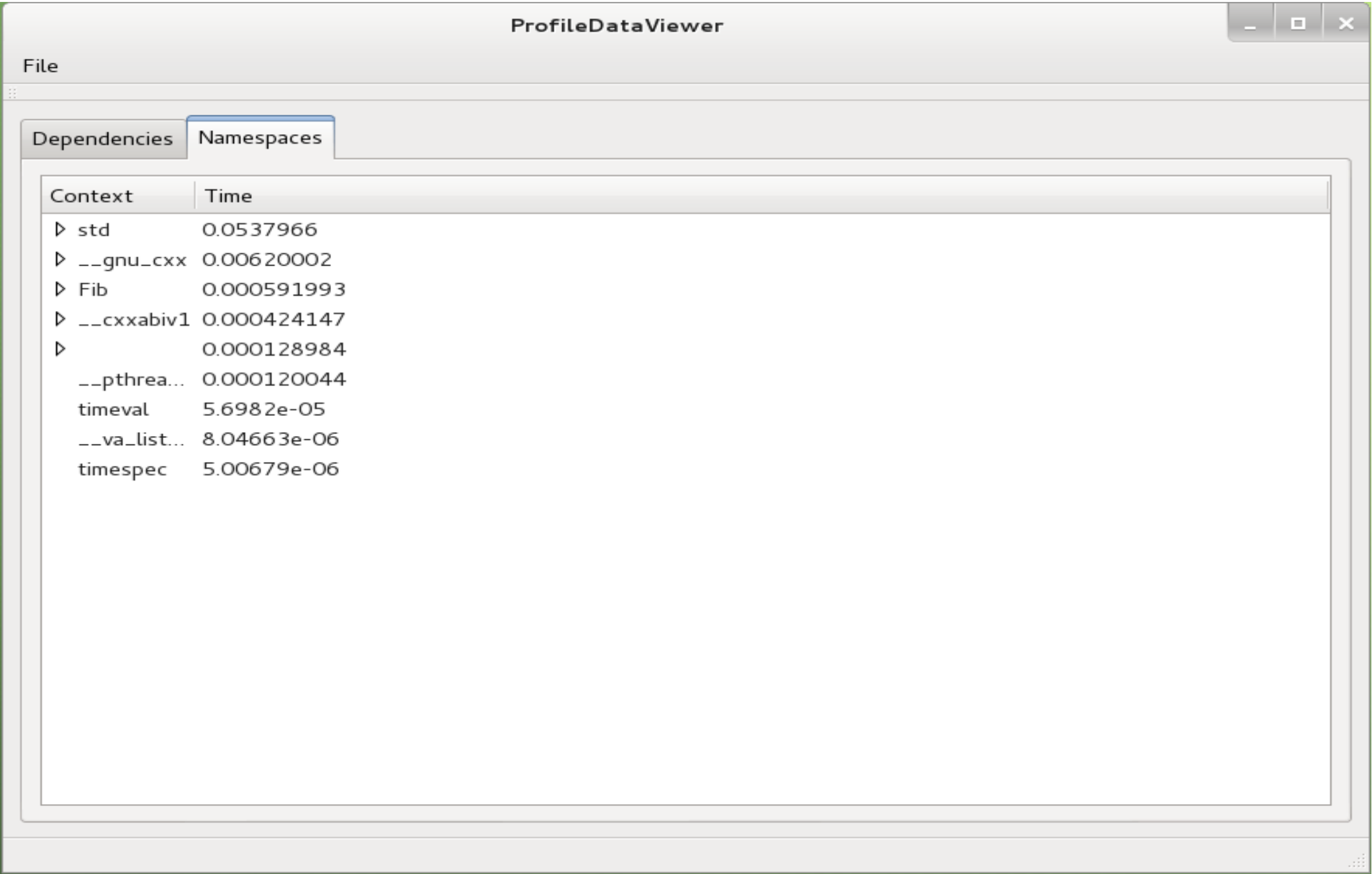
Kind: TemplateInstantiation

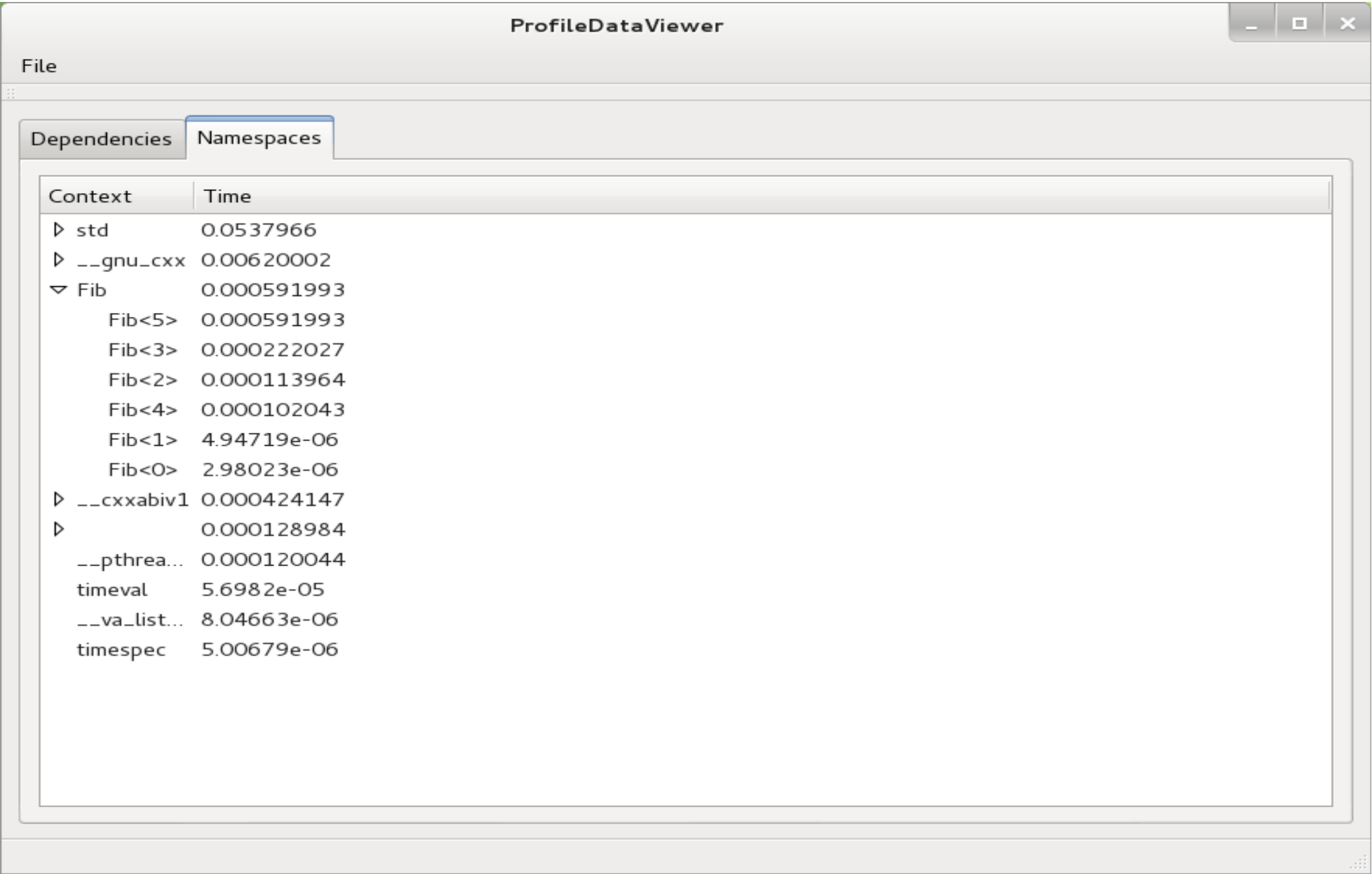
Name: Fib<5>

File position: /home/ezolpor/work/proj/templight/work/fib.cpp|22|14

Fib<5>


ProfileDataViewer
File
Dependencies
Namespaces
Context
Time
std::basic... 0.00440902
std::basic... 0.00388598
std::basic... 0.00152701
std::basic... 0.00149602
std::basic... 0.00146705
std::basic... 0.00131899
std::basic... 0.00121498
std::basic... 0.00115699
__gnu_c... 0.00114799
std::num_... 0.00113398
std::colla... 0.00103903
__gnu_c... 0.000955999
__gnu_c... 0.000909984
__gnu_c... 0.000891984
std::num_... 0.00088501
std::basic... 0.000883996
std::endl... 0.000838041
std::__ct... 0.000837982
__gnu_c... 0.000835955
std::num_... 0.000783026


ProfileDataViewer
File
Dependencies
Namespaces
Context
Time
std
0.0537966
__gnu_cxx
0.00620002
Fib
0.000591993
__cxxabiv1
0.000424147
0.000128984
__pthrea...
0.000120044
timeval
5.6982e-05
__va_list...
8.04663e-06
timespec
5.00679e-06


ProfileDataViewer
File
Dependencies
Namespaces
Context
Time
std 0.0537966
__gnu_cxx 0.00620002
Fib 0.000591993
Fib<5> 0.000591993
Fib<3> 0.000222027
Fib<2> 0.000113964
Fib<4> 0.000102043
Fib<1> 4.94719e-06
Fib<0> 2.98023e-06
__cxxabiv1 0.000424147
0.000128984
__pthrea... 0.000120044
timeval 5.6982e-05
__va_list... 8.04663e-06
timespec 5.00679e-06

Memory usage

```
$ clang++ -templight-memory fib.cpp

$ ls
fib.cpp.memory.trace.xml

$ wc fib.cpp.memory.trace.xml
  18291  41765 756365 fib5.cpp.memory.trace.xml

0$ head fib.cpp.trace.xml
<?xml version="1.0" standalone="yes"?>
<Trace>
<TemplateBegin>
    <Kind>TemplateInstantiation</Kind>
    <Context context = "Fib&lt;5&gt;"/>
    <PointOfInstantiation>fib.cpp|22|
14</PointOfInstantiation>
    <TimeStamp time = "421998401.188854"/>
    <MemoryUsage bytes = "647664"/>
</TemplateBegin>
<TemplateBegin>
```

Distortion

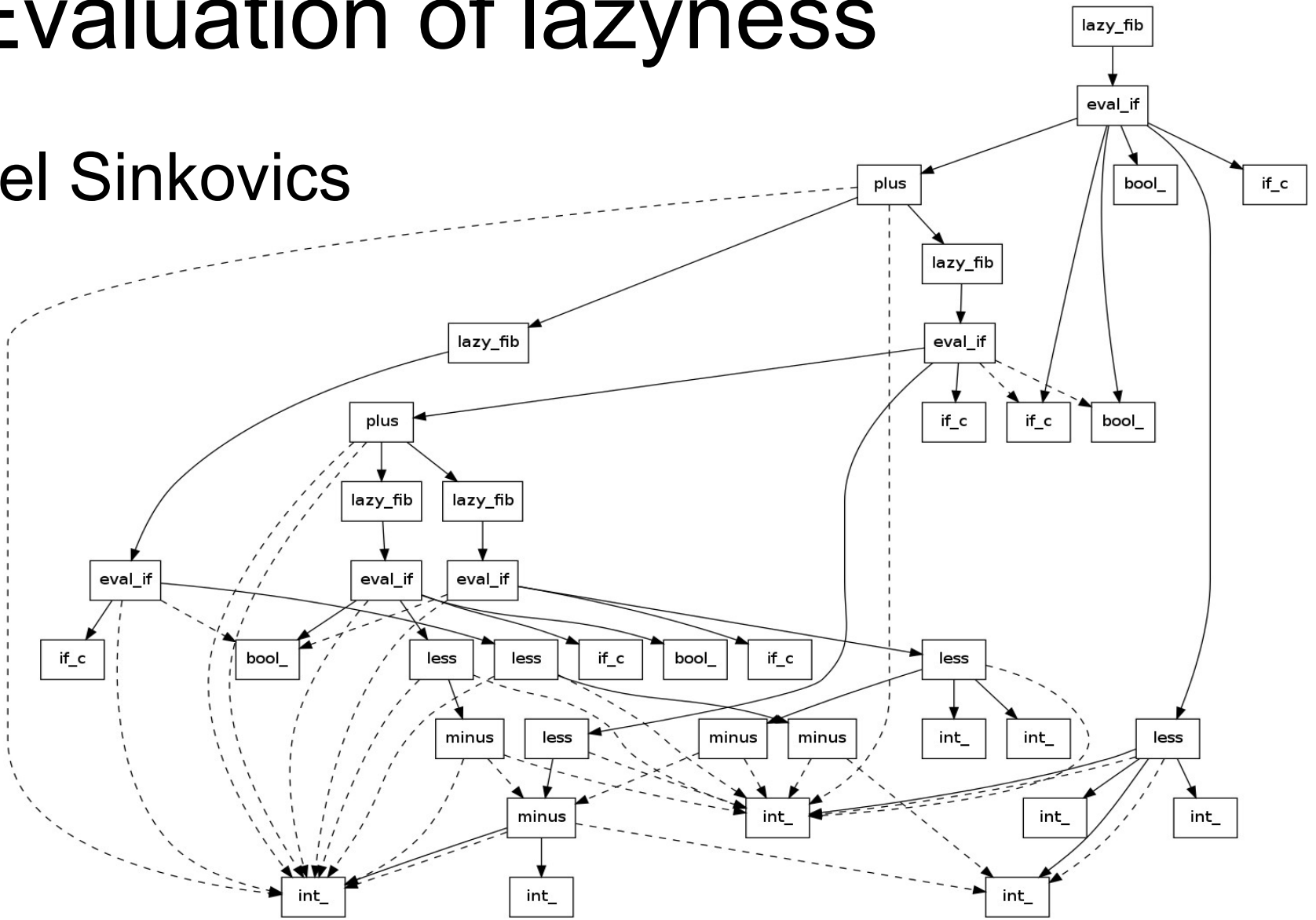
- Safe-mode is installed
  - Invalid profiling info
  - Flush messages even the compiler crashes
- Internal buffer collects events
  - Heap allocated, not growing, size = 500.000
  - Flush at end of compilation
  - Distorsion < 3%
  - clang++ -templight -trace-capacity=1000000

Forks, Applications

- Martin Schulze modified client tools http://github.com/schulmar/Templar
- Malte Skarupke's blog: comparing instantiation time of unique_ptr, boost::flat_map, etc. http://probablydance.com/2014/04/05/reinventing-the-wheel-for-better-compile-time/
- Ábel Sinkovics: comparing lazy and greedy metaprograms

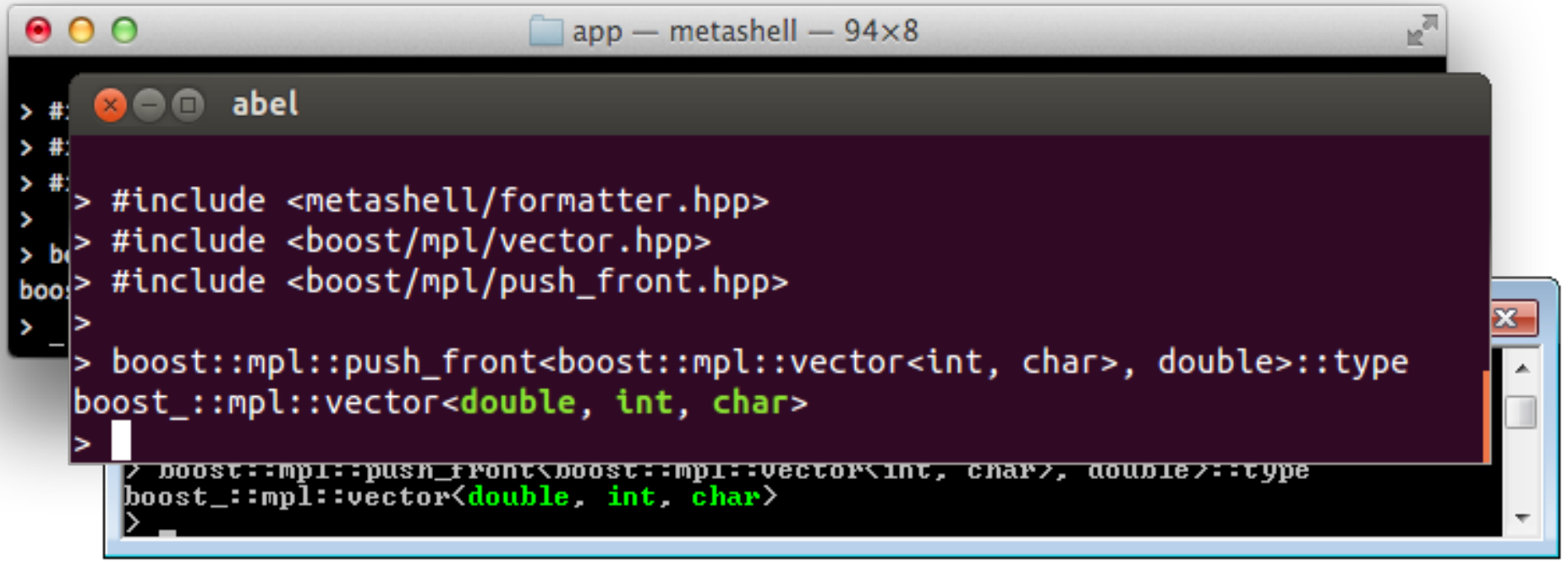
Evaluation of lazyness

- Ábel Sinkovics



Metashell – interactive TMP REPL

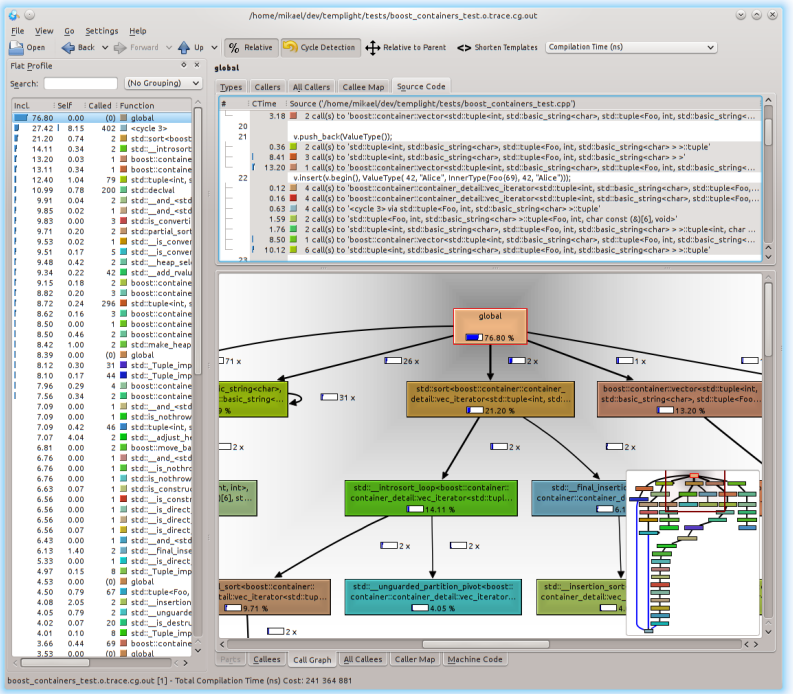
- Ábel Sinkovics and András Kucsma
- Metashell https://github.com/sabel83/metashell
- Online demo: http://abel.web.elte.hu/shell



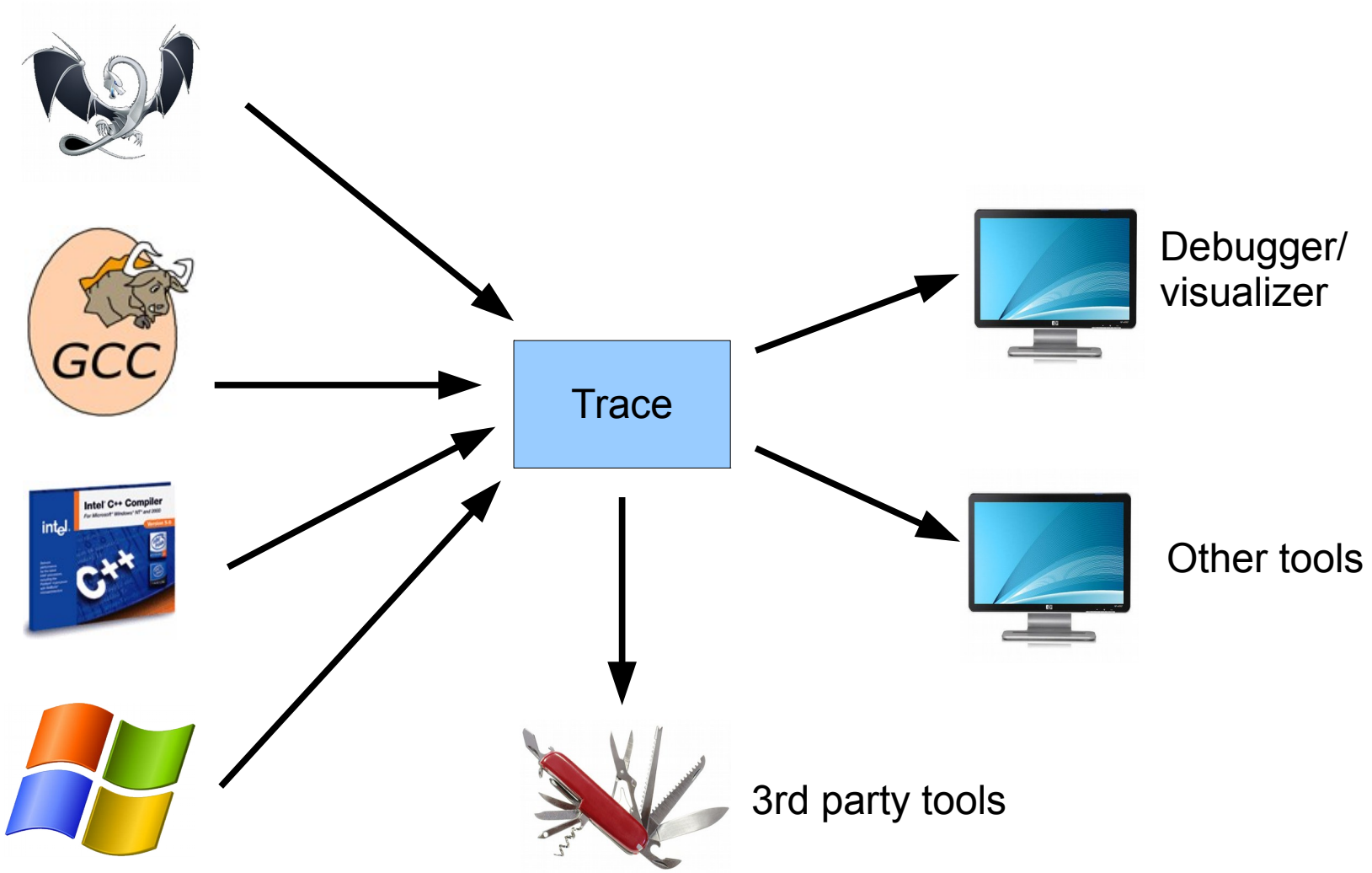
Mikael Persson's Templight fork

- https://github.com/mikael-s-persson/templight
- https://github.com/mikael-s-persson/templight-tools

- Refactored and greatly improved Templight
- Patch is under review
- Tools: KCacheGrind format



Our vision


GCC
Intel C++ Compiler
intel
C++
Trace
Debugger/
visualizer
Other tools
3rd party tools

Summary

- Tool support for C++ metaprogramming
- Debugger/profiler requires compiler support
- Templight 2.0 based on clang
- Mikael's patch for clang is under review
- Please use it, give us feadback
- Compiler vendors, will you support Templight?

Thank you!

Debugging and Profiling C++ Template Metaprograms

http://plc.inf.elte.hu/templight

gsd@elte.hu