Templight: A Clang Extension for Debugging and Profiling C++ Template Metaprograms

Zoltán Porkoláb, Zoltán Borók-Nagy

Eötvös Loránd University, Budapest
Ericsson Hungary

Agenda

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

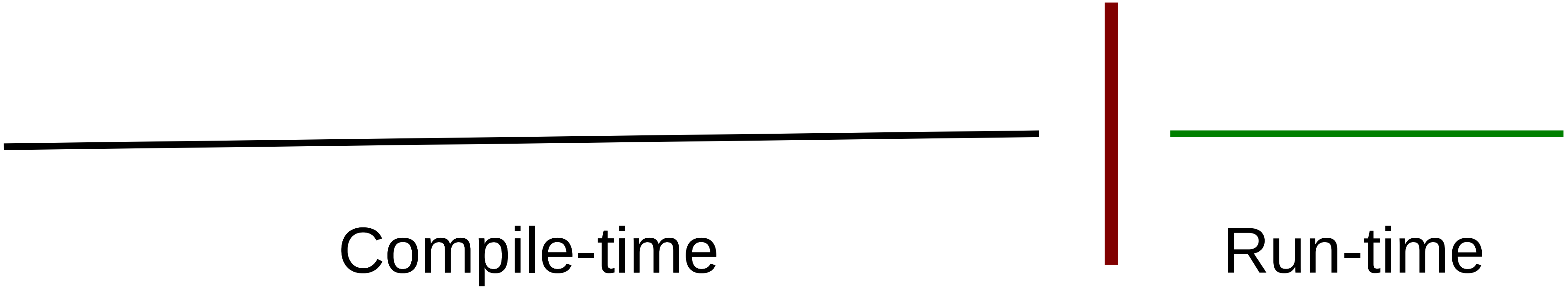
C++ Template Metaprograms

- 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 type?
{
    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);
}
```

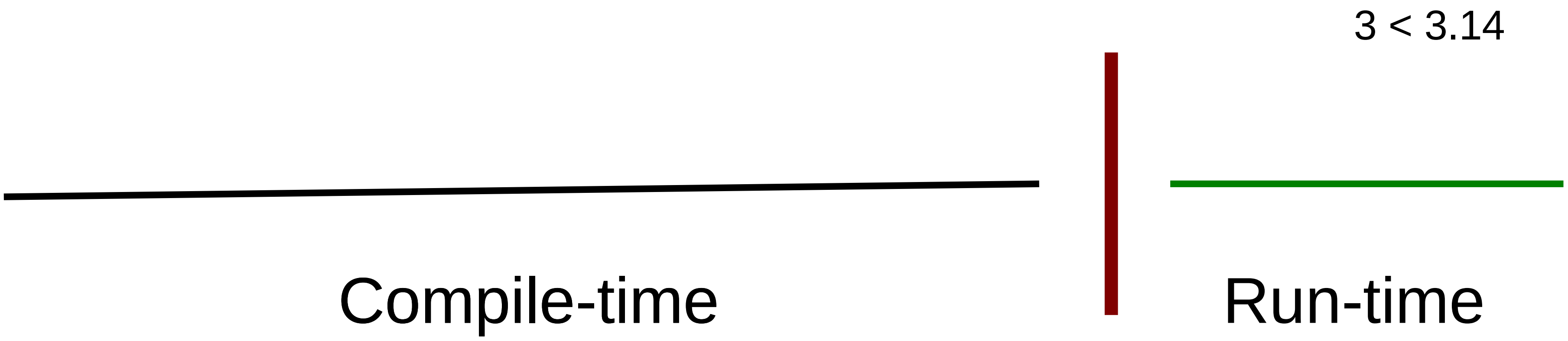
Compile-time vs. Run-time



Compile-time

Run-time

Compile-time vs. Run-time

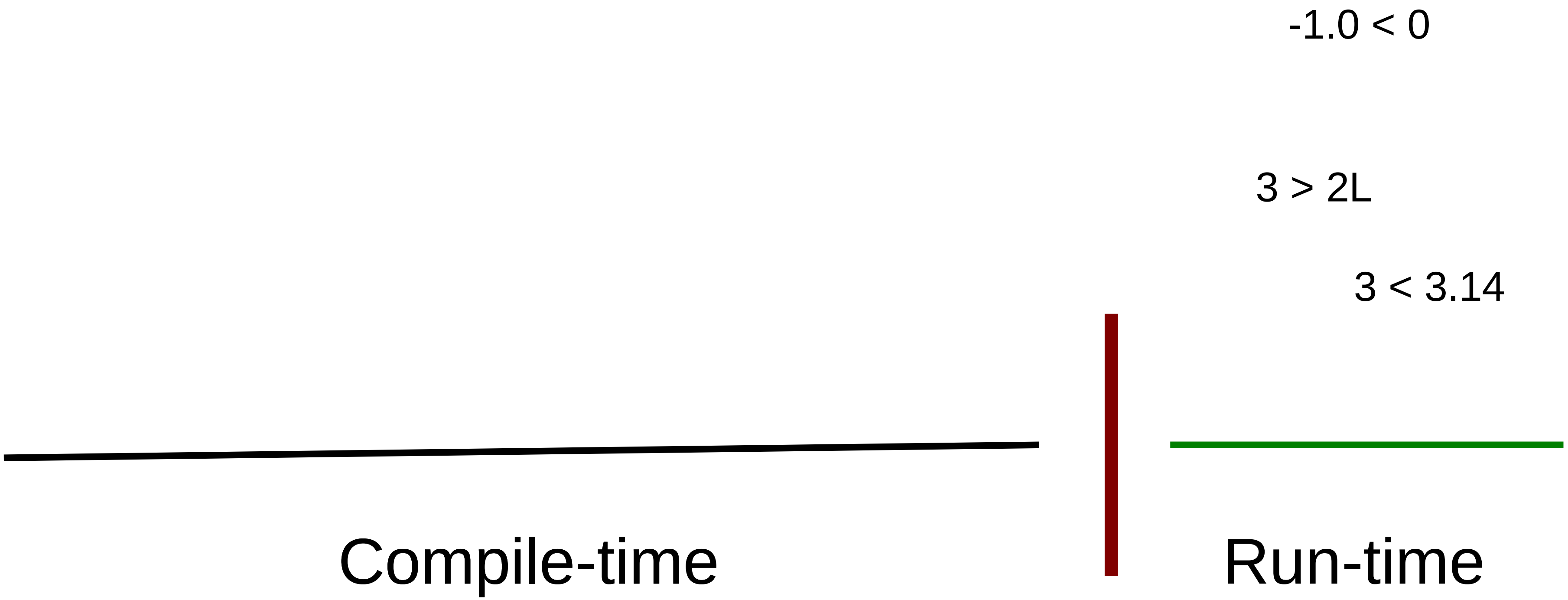


3 < 3.14

Compile-time

Run-time

Compile-time vs. Run-time



-1.0 < 0

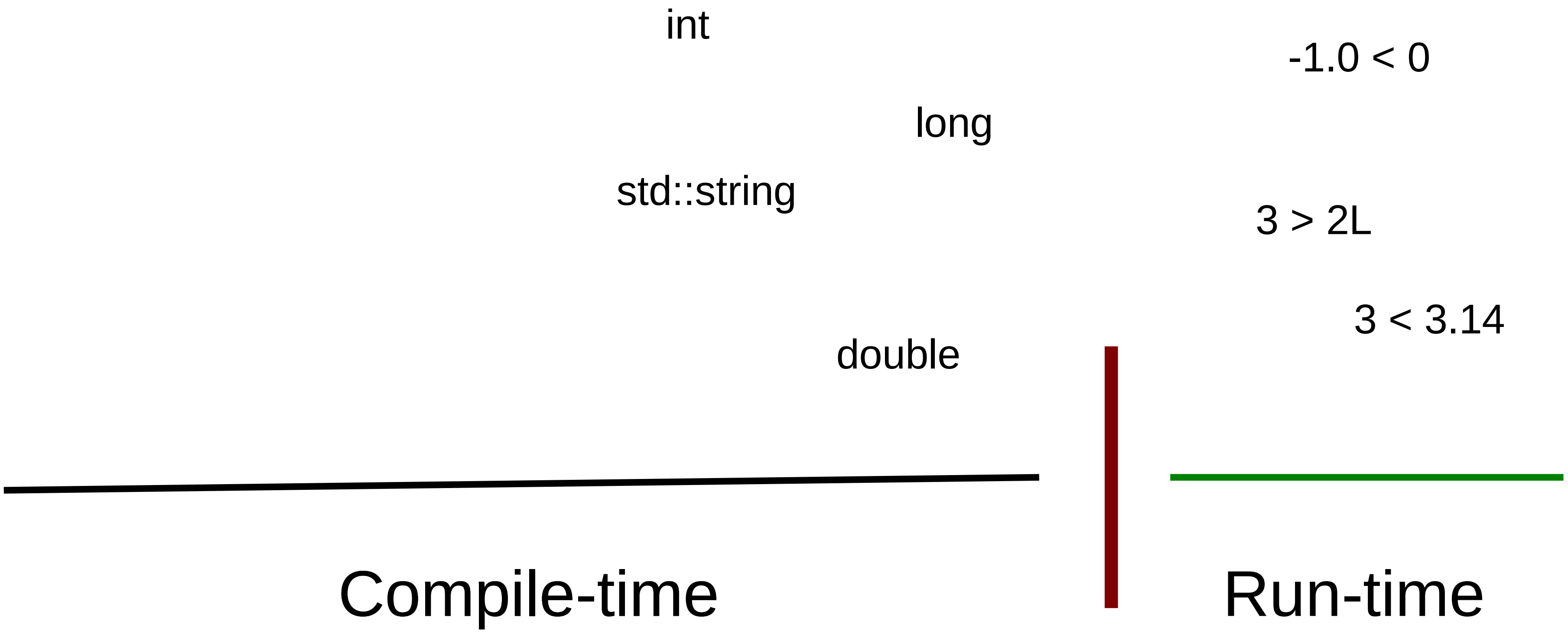
3 > 2L

3 < 3.14

Compile-time

Run-time

Compile-time vs. Run-time



int

-1.0 < 0

long

std::string

3 > 2L

3 < 3.14

double

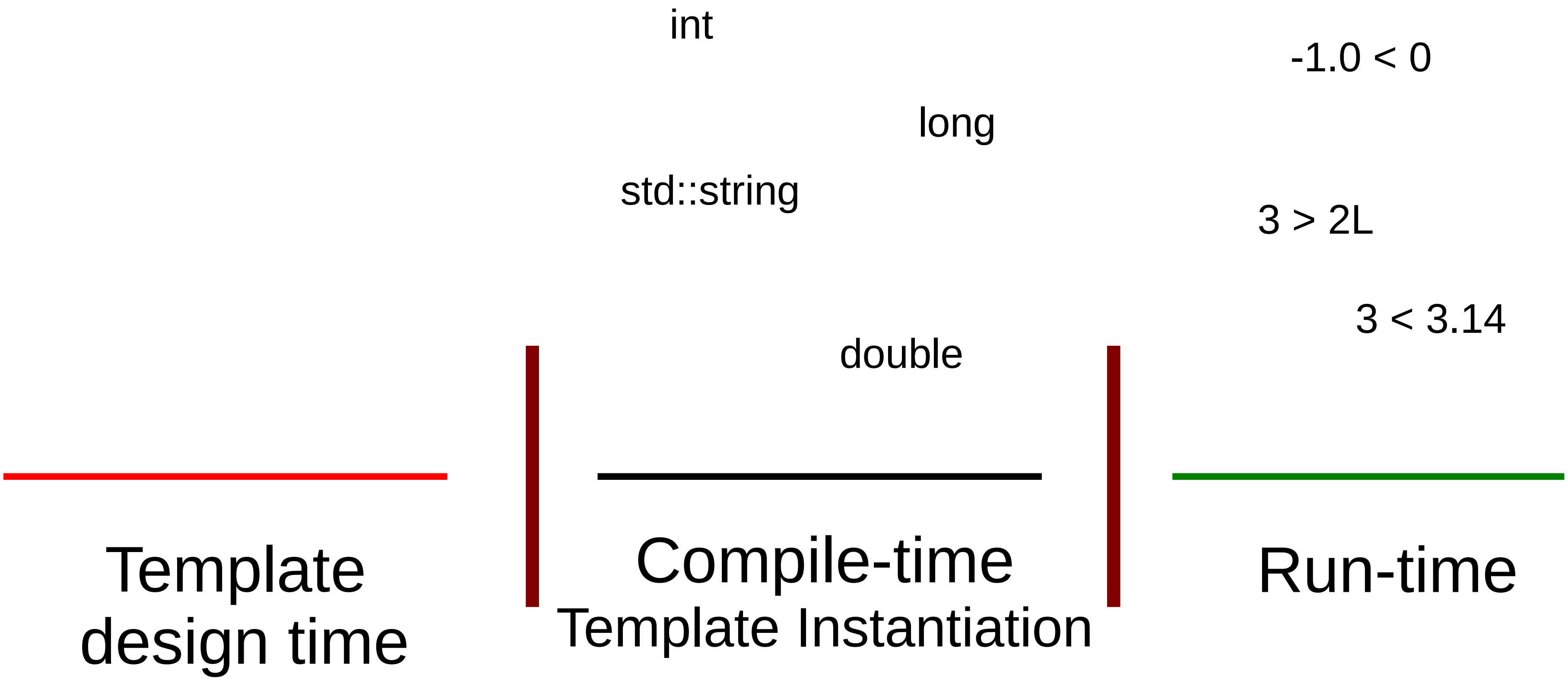
Compile-time

Run-time

Motivation

```
template <class T, class S>
? max( T a, S b)  // How to define the return type?
{
    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



int

-1.0 < 0

long

std::string

3 > 2L

3 < 3.14

double

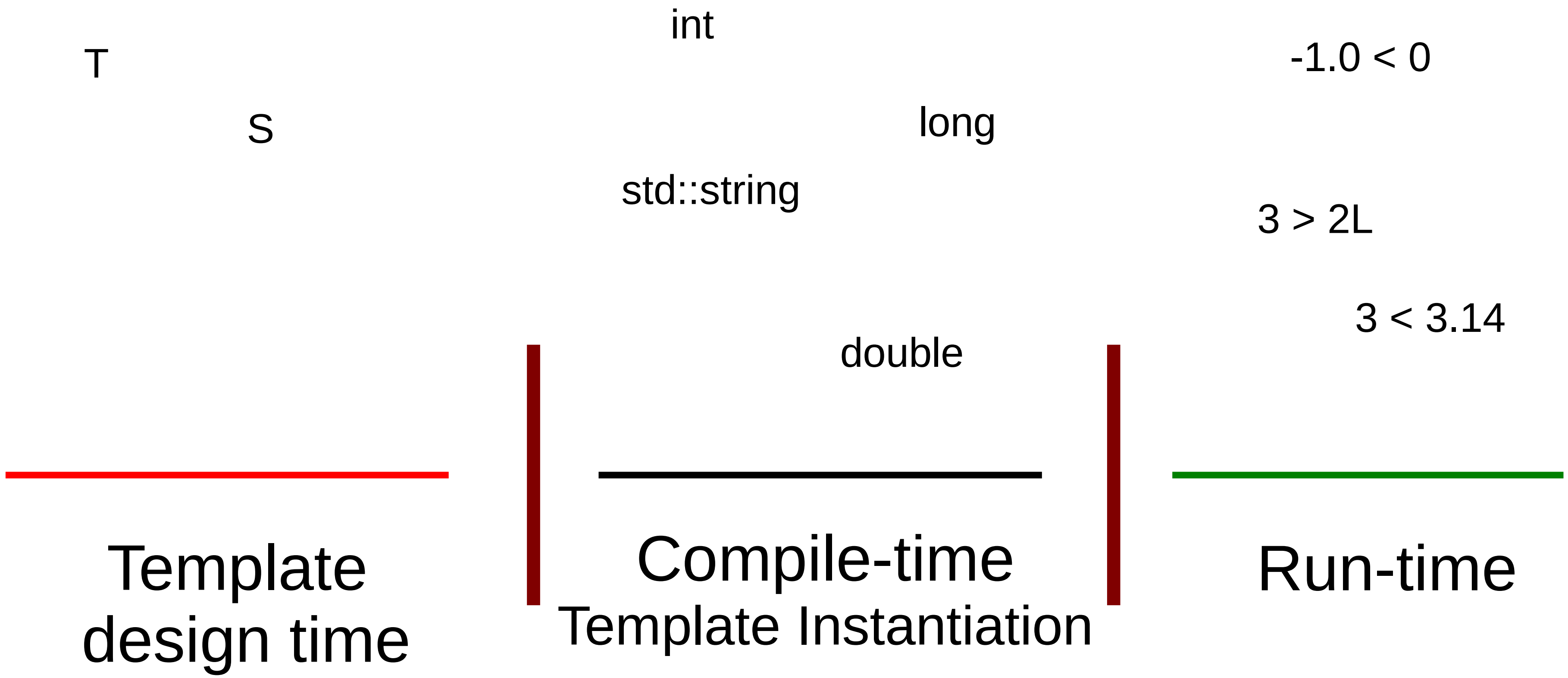
Template design time

Compile-time

Template Instantiation

Run-time

Compile-time vs. Run-time



T

S

Template design time

int

long

std::string

double

Compile-time

Template Instantiation

-1.0 < 0

3 > 2L

3 < 3.14

Run-time

Compile-time vs. Run-time

T

S

sizeof(T) < sizeof(S)

Template
design time

int

long

std::string

double

Compile-time
Template Instantiation

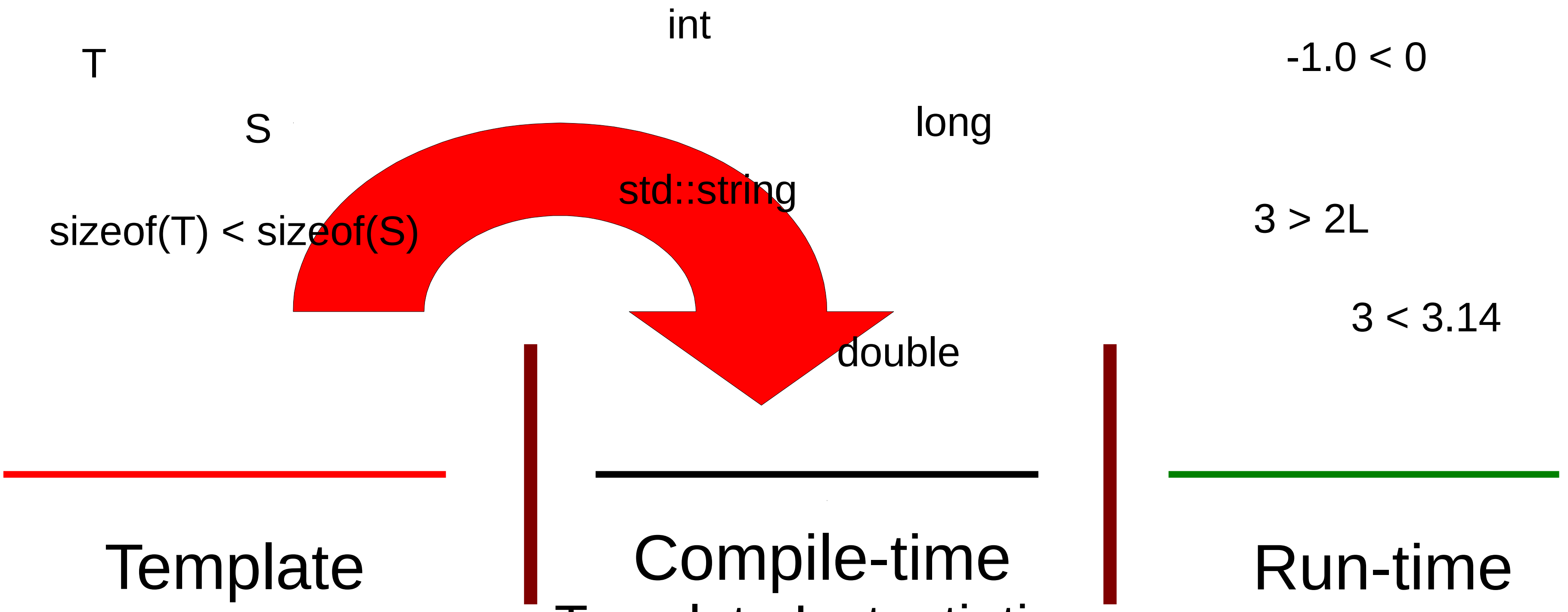
-1.0 < 0

3 > 2L

3 < 3.14

Run-time

Compile-time vs. Run-time



T

S

sizeof(T) < sizeof(S)

int

long

std::string

double

-1.0 < 0

3 > 2L

3 < 3.14

Template design time

Compile-time
Template Instantiation

Run-time

Motivation

```
template <class T, class S>
? max( T a, S b)  // How to define the return type?
{
    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>::type 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
}
```

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;
}
```

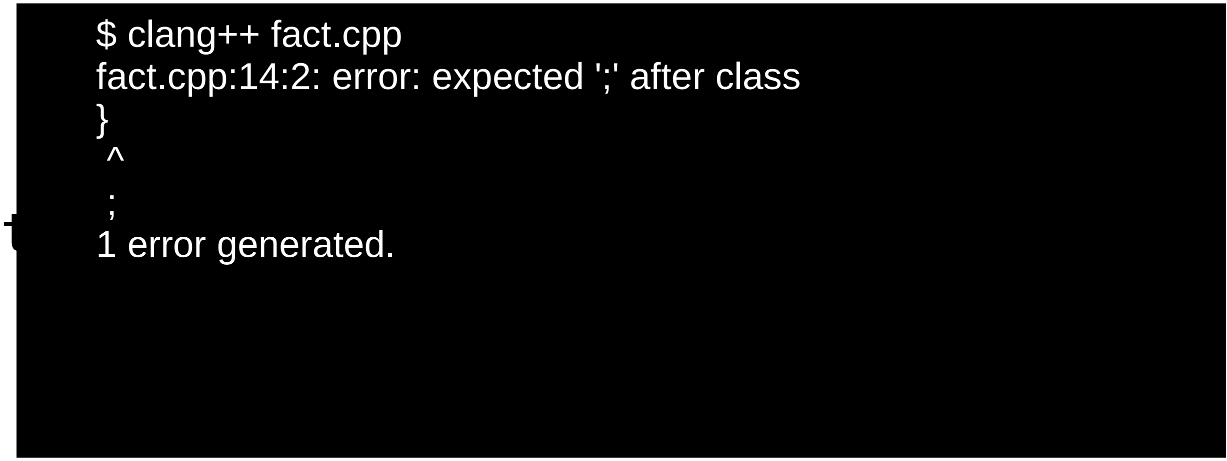
Bugs!!! ...

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.
```

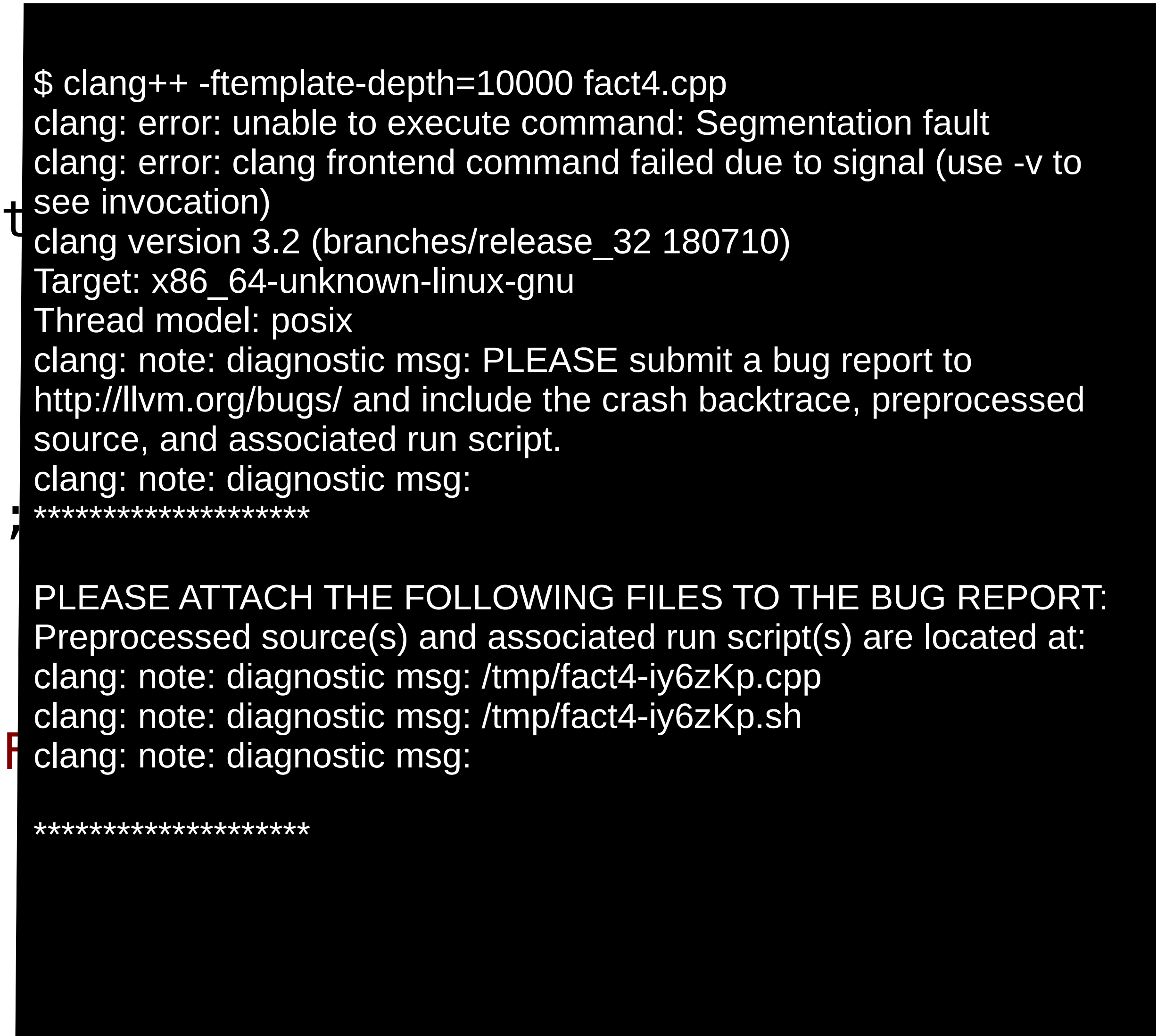
The greedy ...

```
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 optimized for detecting and reporting template metaprogram errors
- Compilers are not optimized 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

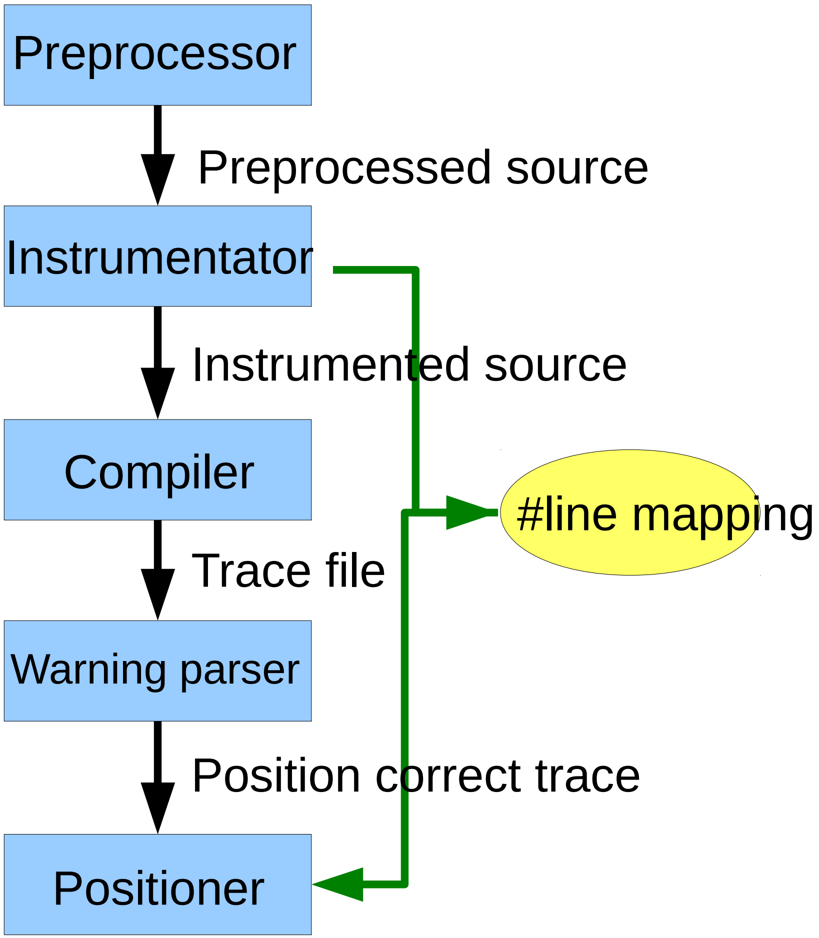
Run-time

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 appearance (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

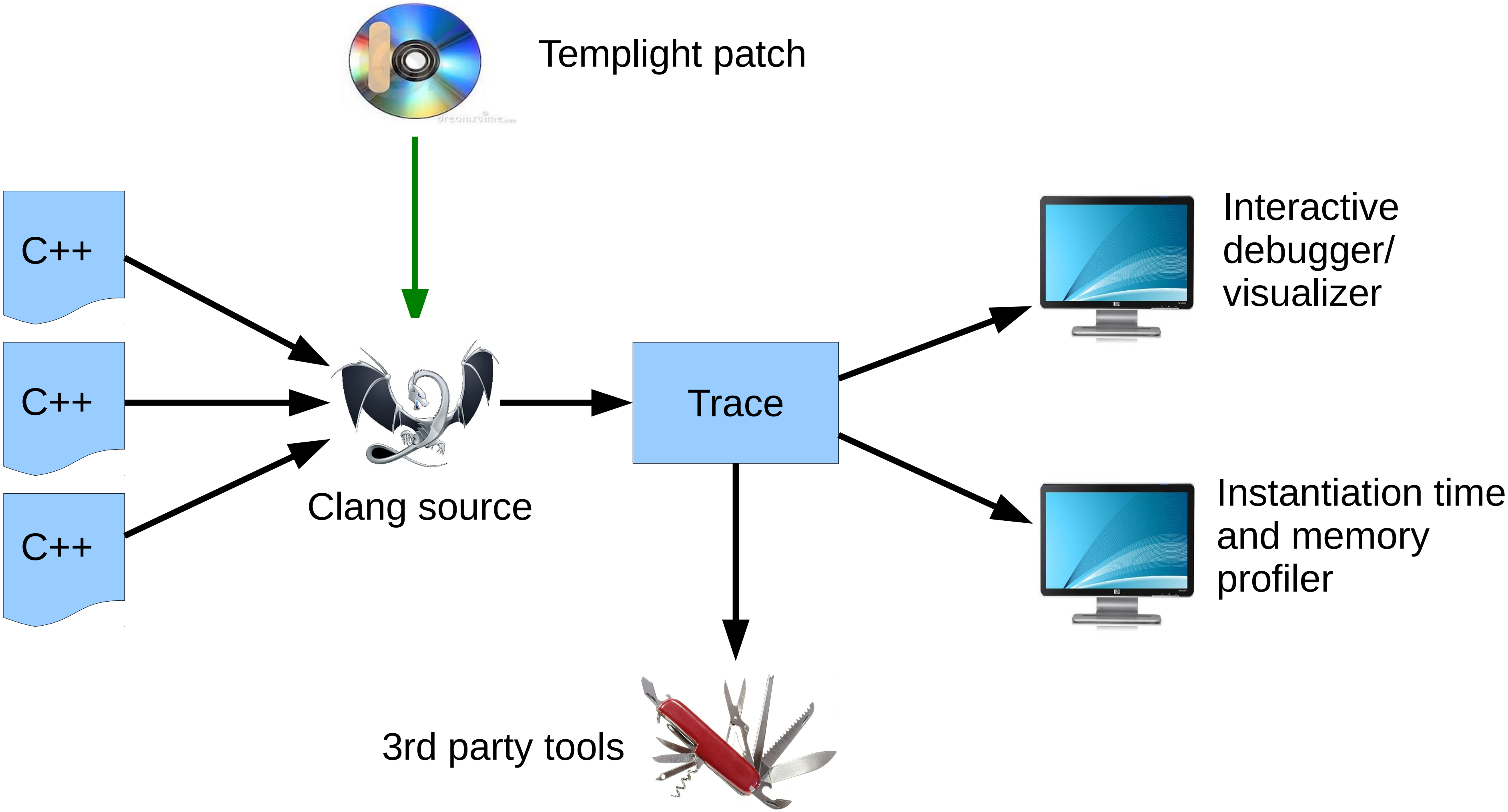
Instrumentation

- 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 is 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



Templight patch

C++

C++

C++

Clang source

Trace

Interactive debugger/ visualizer

Instantiation time and memory profiler

3rd party tools

Installation

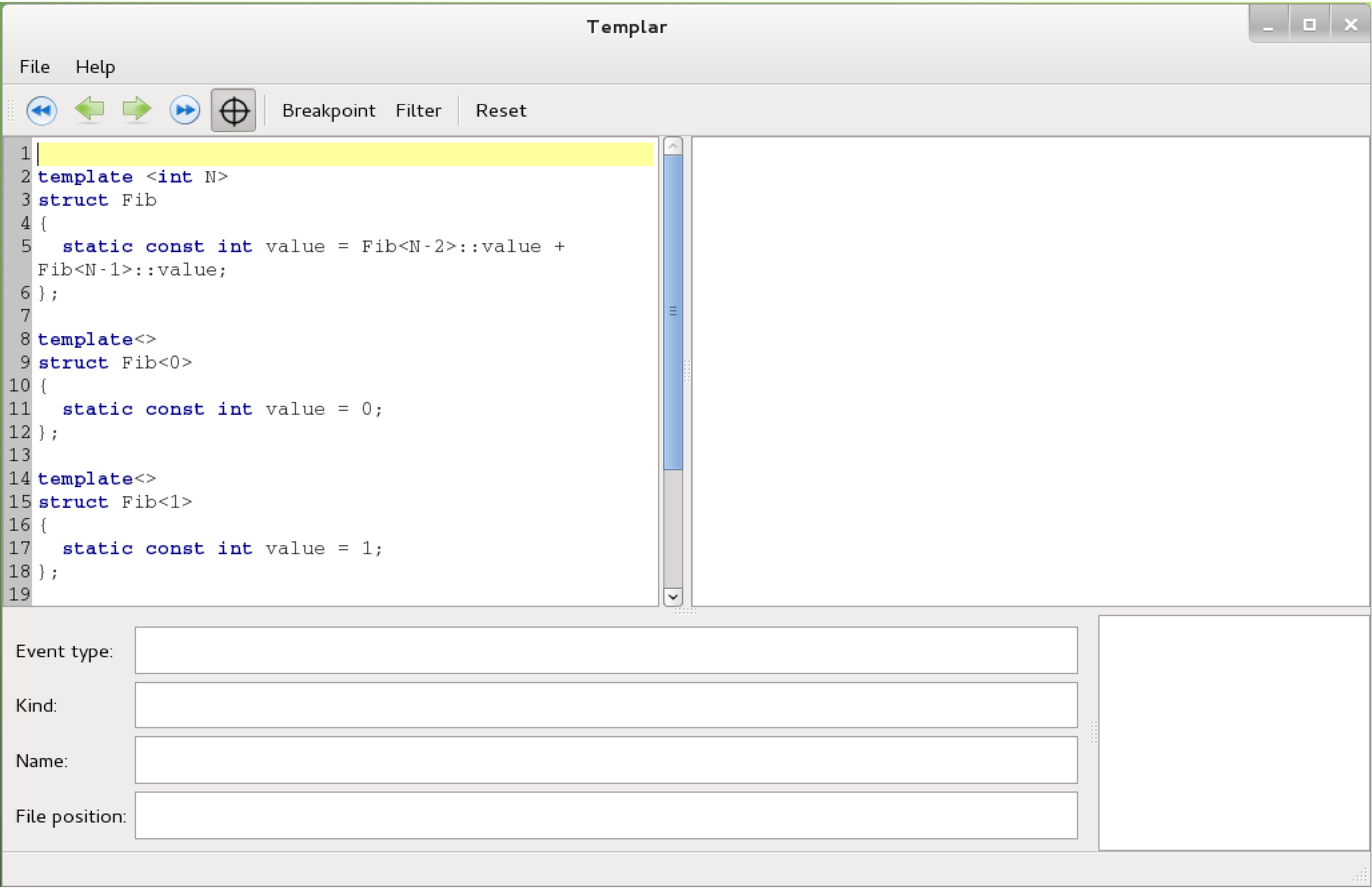
- Visit http://plc.inf.elte.hu/templight
- Download templight-<timestamp>.tar.gz
  - Contains clang patch and the two frontends
- Download Clang source
- Patch and build clang
- Build front-end tools (optional)
  - >=Qt 4.6 and >=Graphviz 2.28.0 required
  - $ qmake; make

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>
```



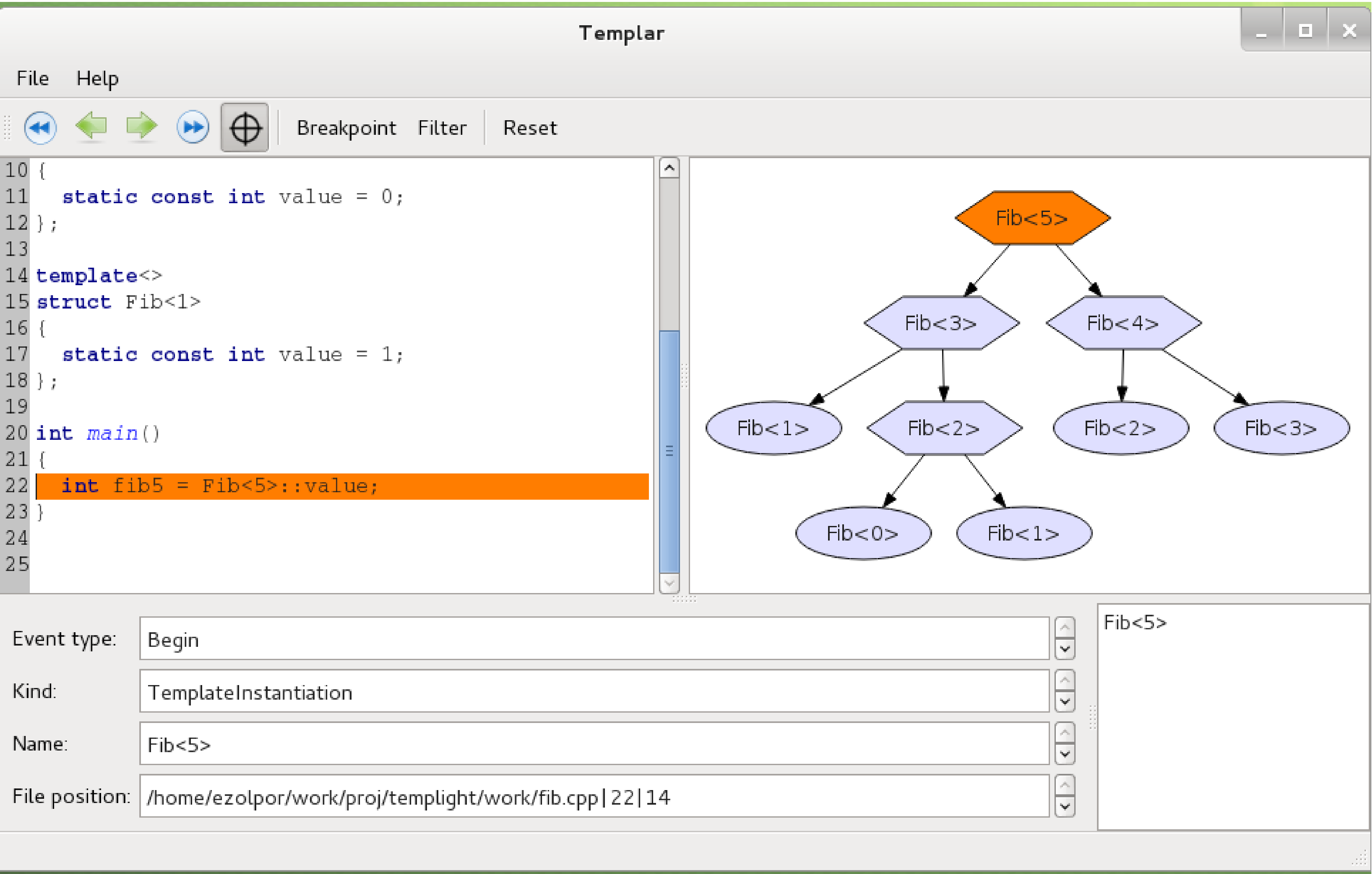
```

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;
};

```

Event type:

Kind:

Name:

File position:



```
{
  static const int value = 0;
};

template<>
struct Fib<1>
{
  static const int value = 1;
};

int main()
{
  int fib5 = Fib<5>::value;
}
```

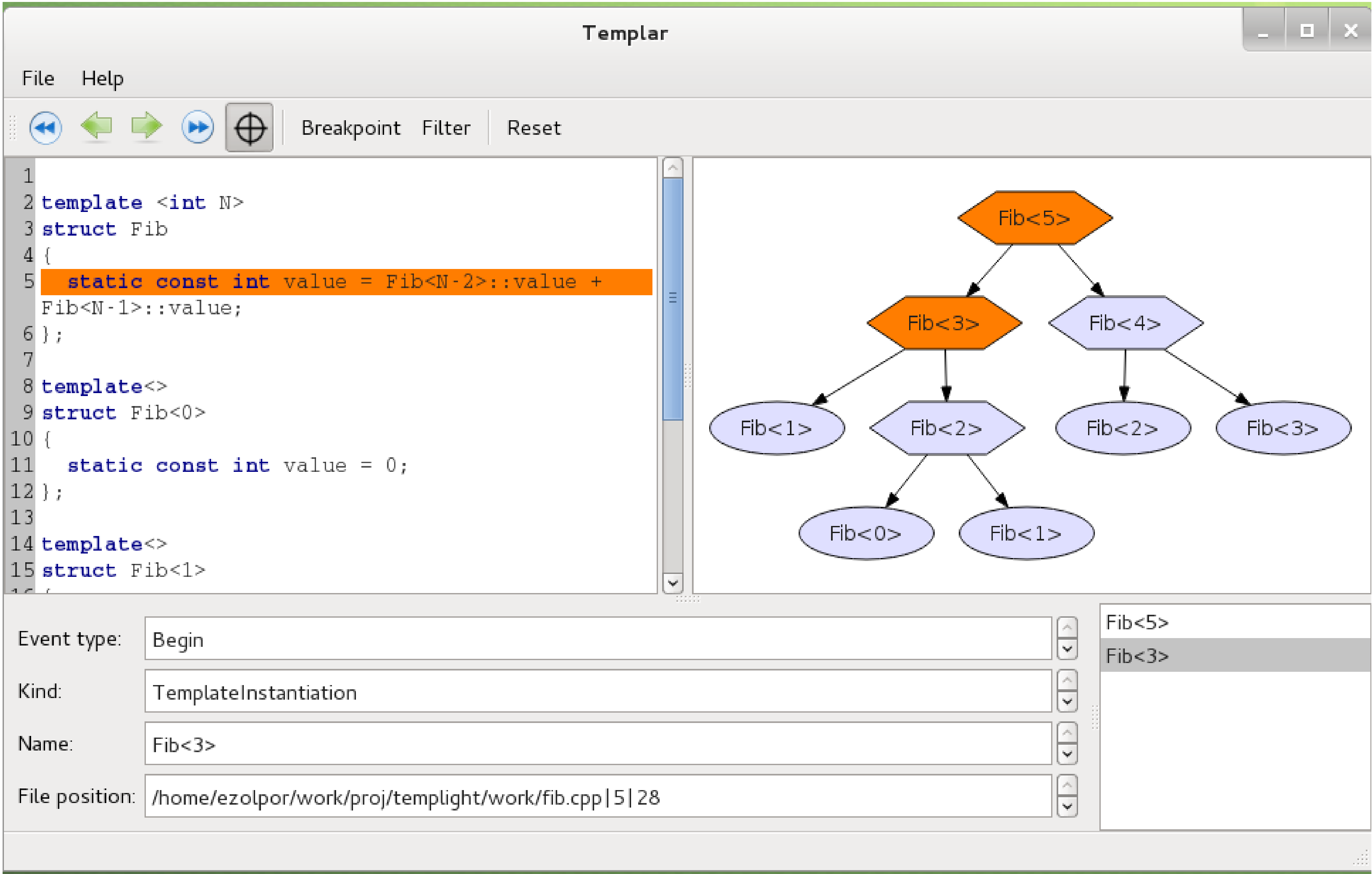
Event type: Begin

Kind: TemplateInstantiation

Name: Fib<5>

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

Fib<5>



```

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>
```

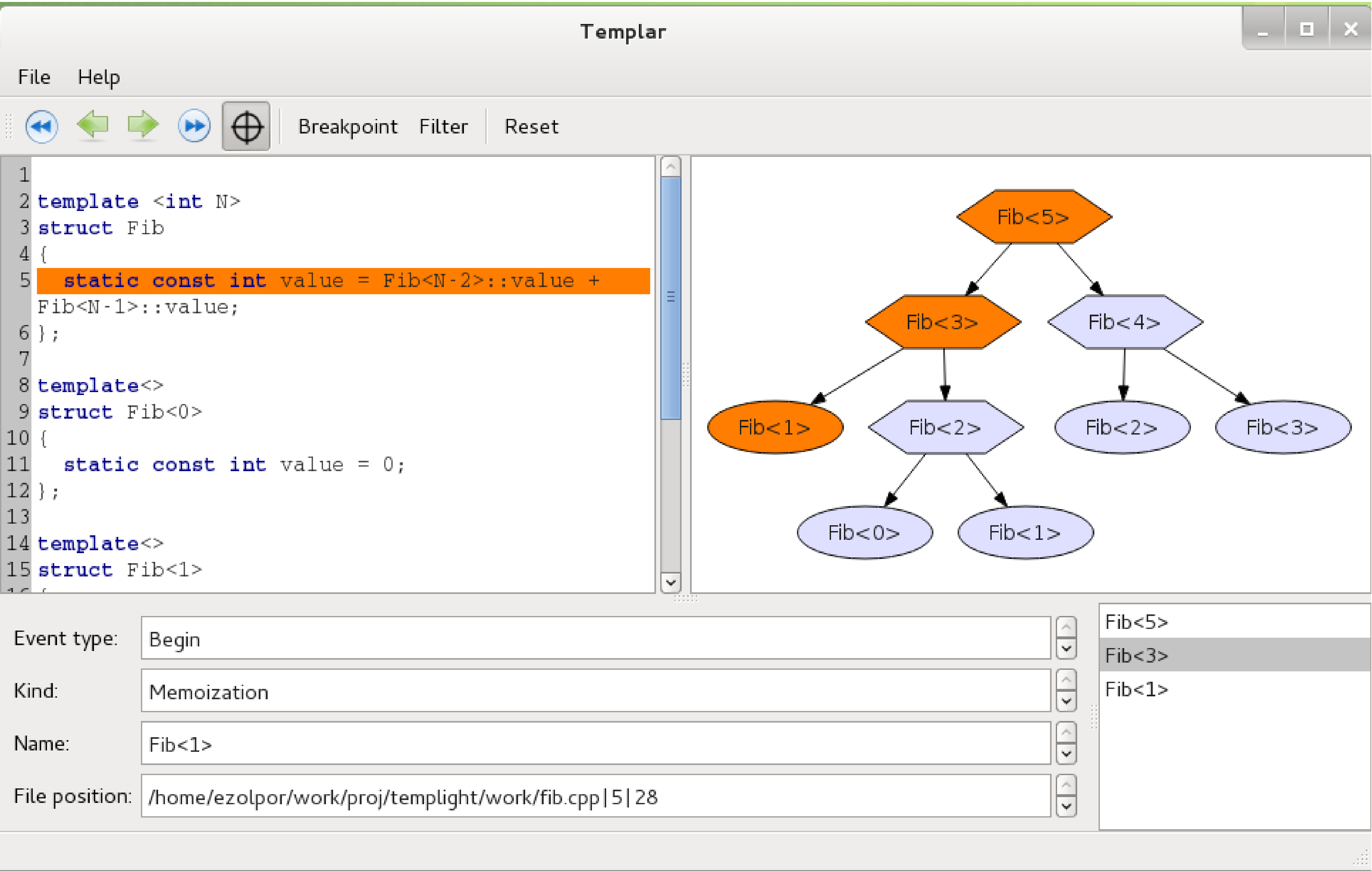
Event type: Begin

Kind: TemplateInstantiation

Name: Fib<3>

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

Fib<5>
Fib<3>



```

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>
{
```

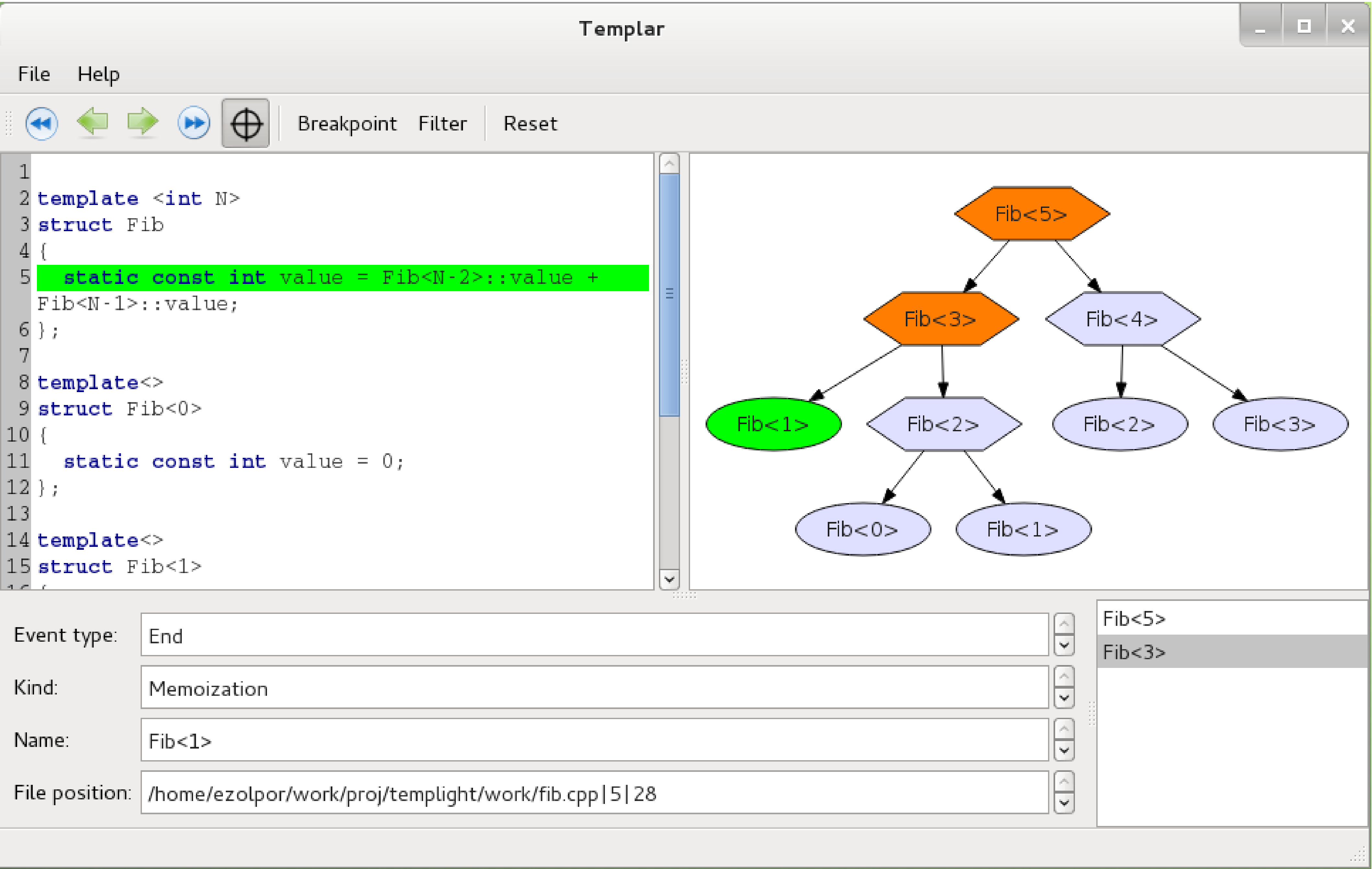
Event type: Begin

Kind: Memoization

Name: Fib<1>

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

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



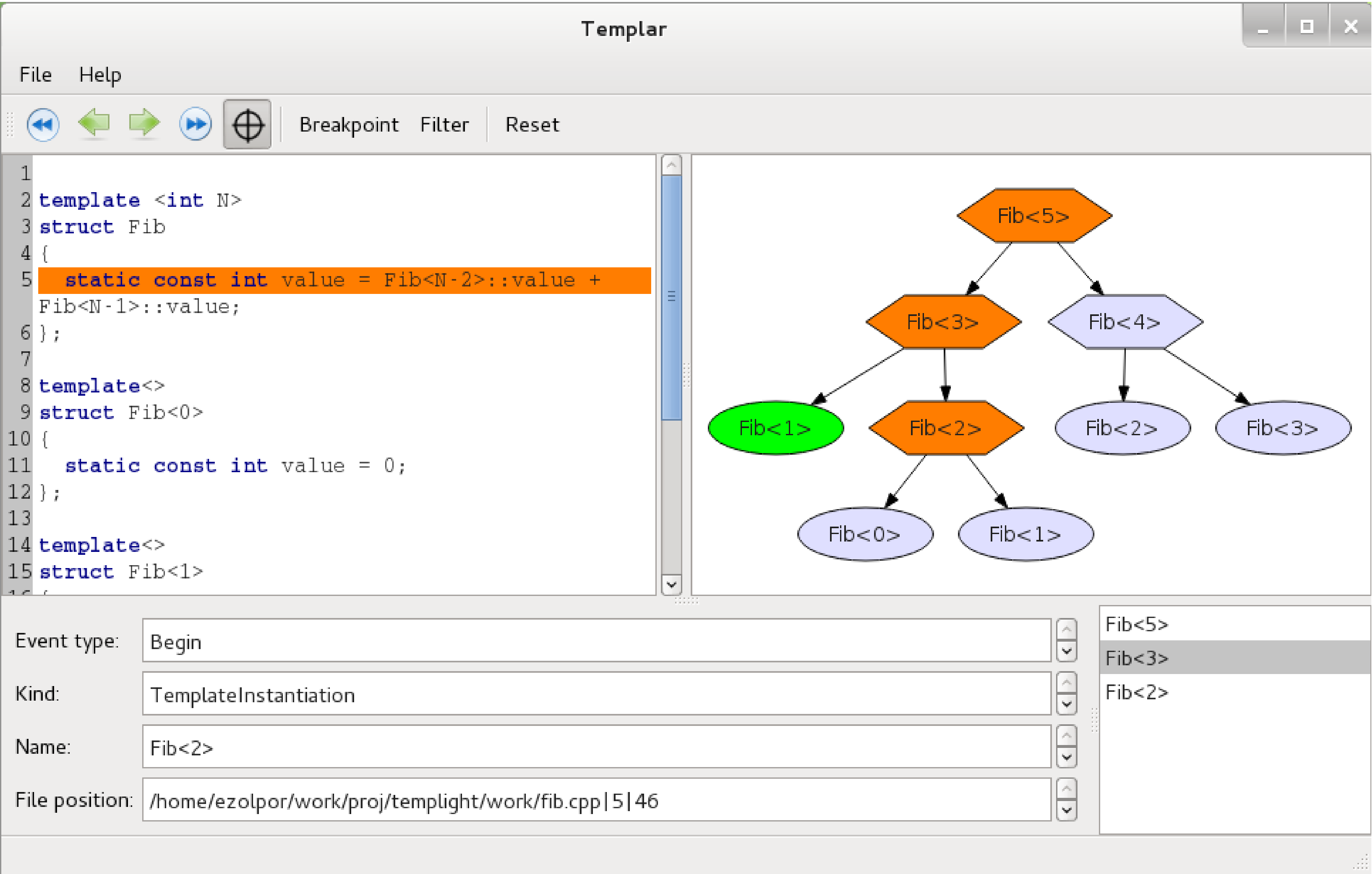
```

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>
```

Event type: End

Kind: Memoization

Name: Fib<1>

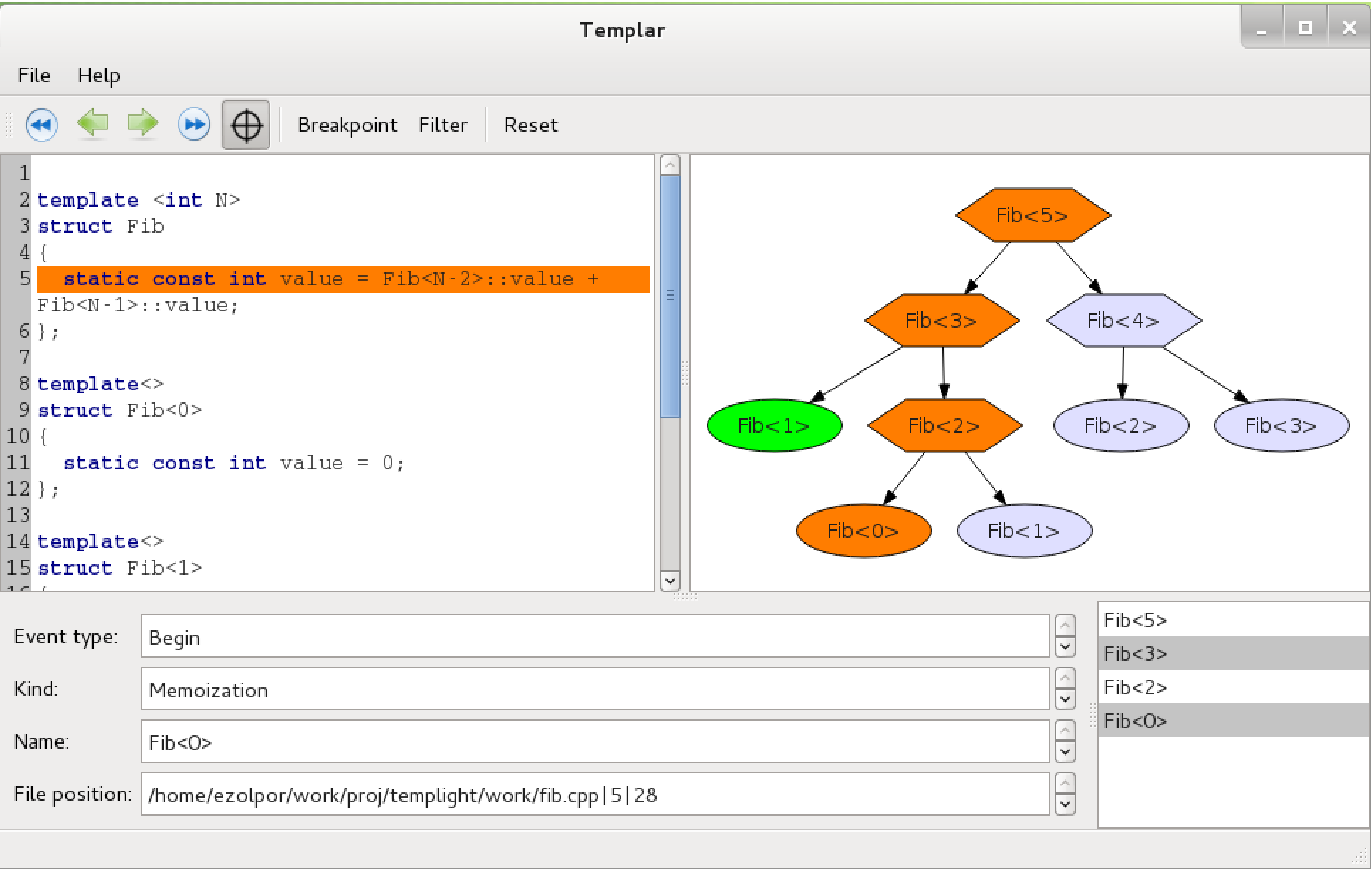
File position: /home/ezolpor/work/proj/templight/work/fib.cpp|5|28



```

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>
{
```

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>



```

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>
```

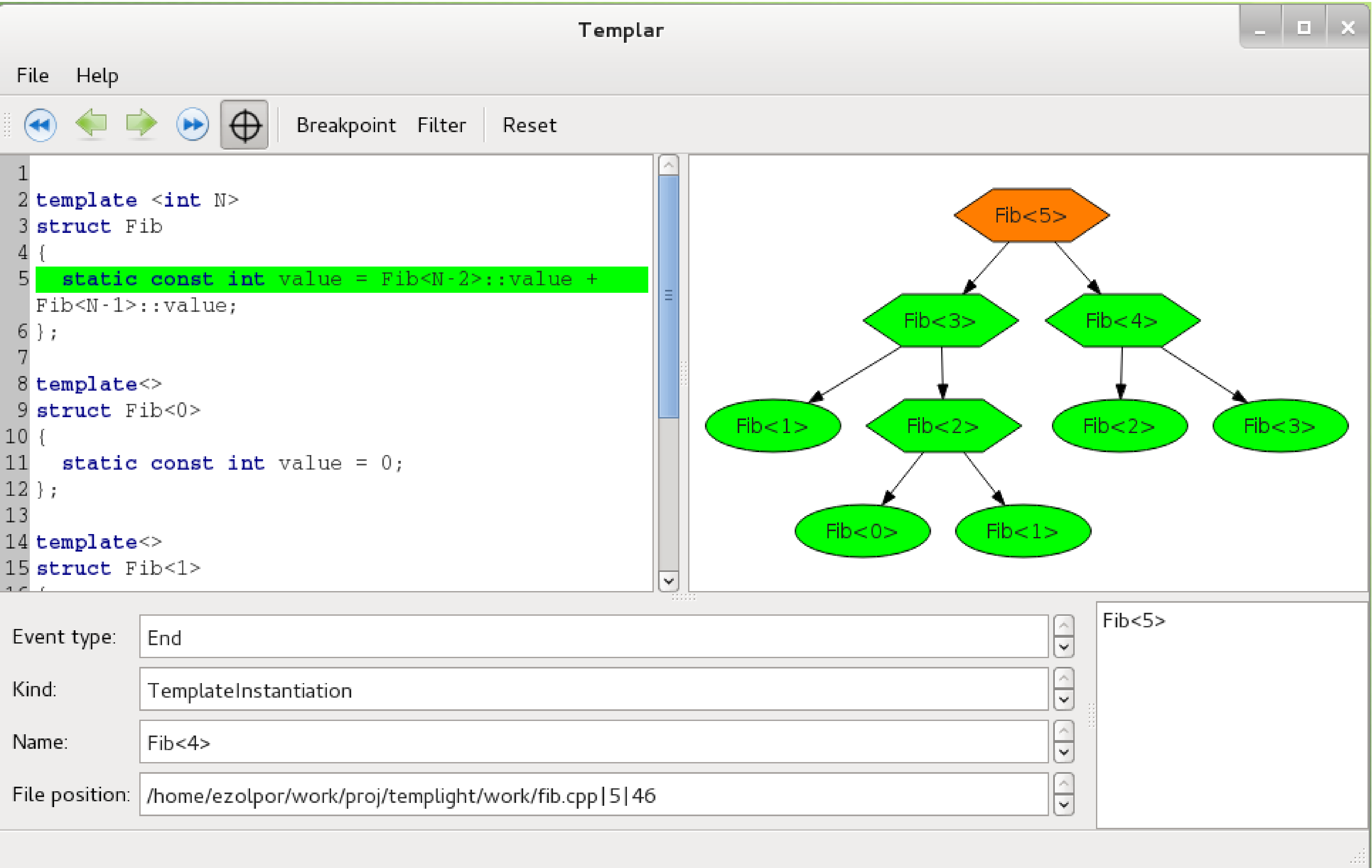
Event type: Begin

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>



```
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>
```

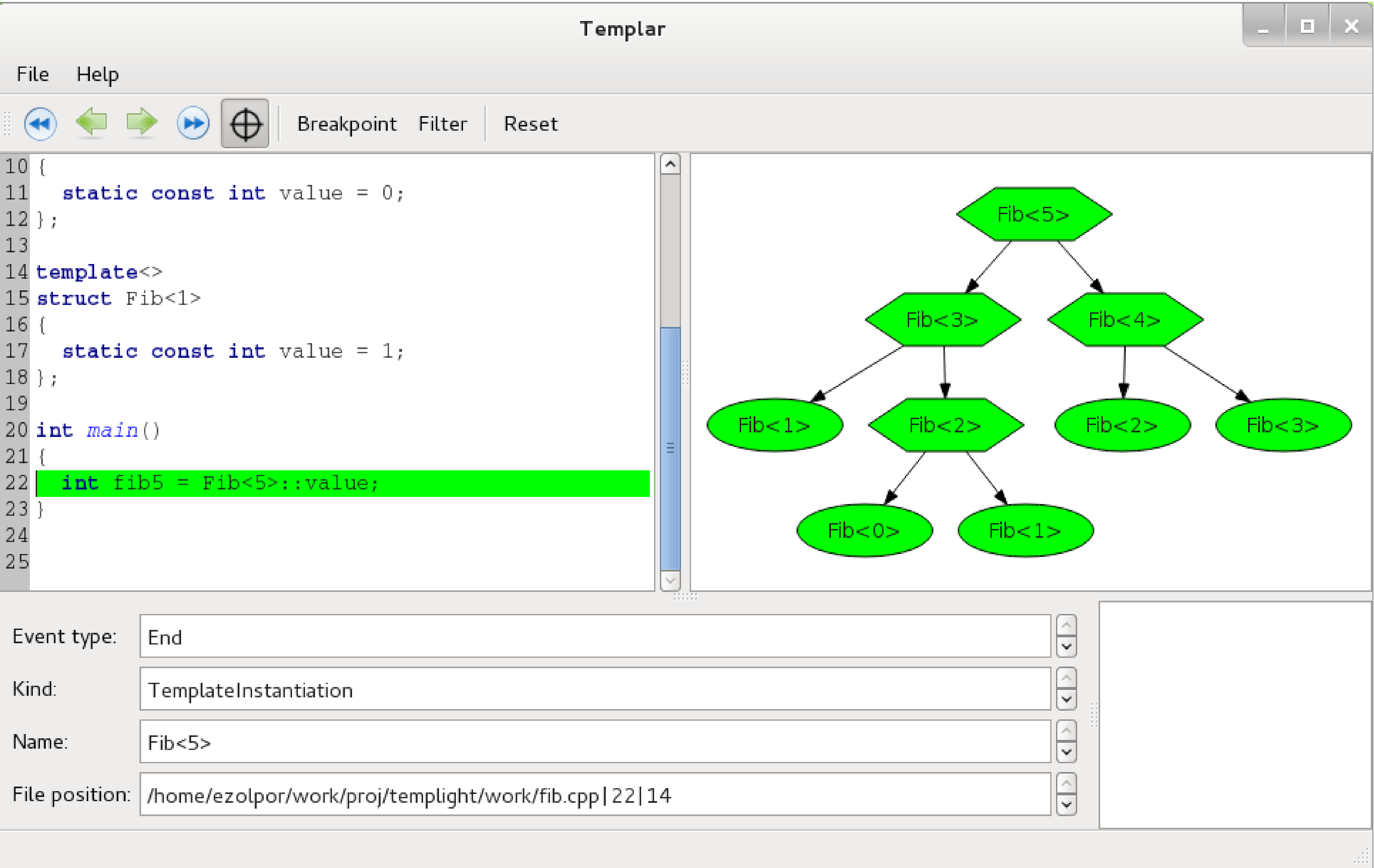
Event type: End

Kind: TemplateInstantiation

Name: Fib<4>

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

Fib<5>



```
{
 static const int value = 0;
};

template<>
struct Fib<1>
{
 static const int value = 1;
};

int main()
{
 int fib5 = Fib<5>::value;
}
```

Event type:	End
Kind:	TemplateInstantiation
Name:	Fib<5>
File position:	/home/ezolpor/work/proj/templight/work/fib.cpp|22|14

Major features

- Debugging
  - Breakpoints: Step in/out/over, forward or backward
  - Filtering out unwanted events
  - Safe mode – flush output after each events
- Profiling
  - Cumulative instantiation times
  - Memory usage at each events
  - Distortion < 3%
    - Heap allocated, not growing, default size is 500.000
    - Flush at the end of compilation

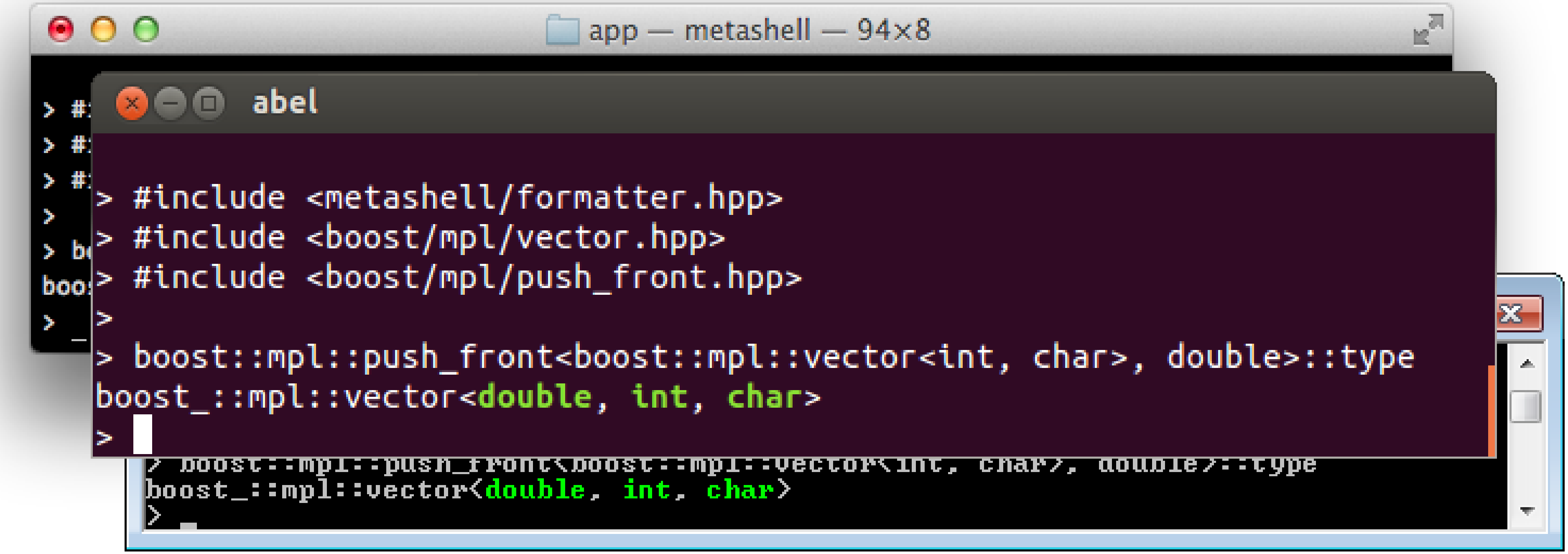
Forks, Applications

- 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/

Metashell – interactive TMP REPL

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

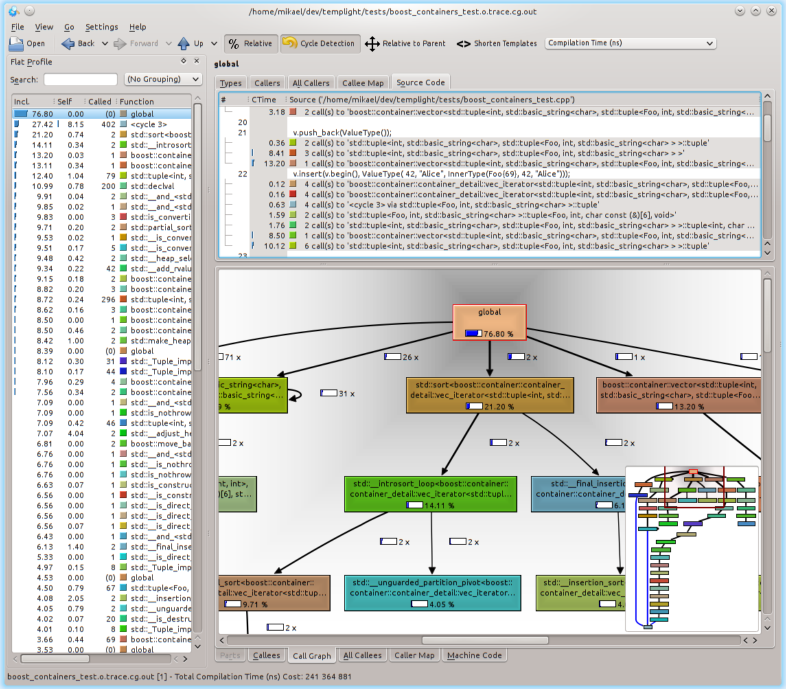


```
> #include <metashell/formatter.hpp>
> #include <boost/mpl/vector.hpp>
> #include <boost/mpl/push_front.hpp>
>
> boost::mpl::push_front<boost::mpl::vector<int, char>, double>::type
boost_::mpl::vector<double, int, char>
> 
```

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



/home/mikael/dev/templight/tests/boost_containers_test.o.trace.cg.out

Search: (No Grouping)

Compilation Time (ns)

Incl.	Self	Called	Function
76.80	0.00	(0)	global
27.42	8.15	402	<cycle 3>
21.20	0.74	2	std::sort<boost
14.11	0.34	2	std::__introsort
13.20	0.03	1	boost::containe
13.11	0.34	1	boost::containe
12.40	1.04	79	std::tuple<int, s
10.99	0.78	200	std::declval
9.91	0.04	2	std::__and_<std
9.85	0.02	1	std::__and_<std
9.83	0.00	3	std::is_converti
9.71	0.20	2	std::partial_sort
9.53	0.02	1	std::__is_conver
9.51	0.17	5	std::__is_conver
9.48	0.42	2	std::__heap_sel
9.34	0.22	42	std::__add_rvalu
9.15	0.18	2	boost::containe
8.82	0.20	3	boost::containe
8.72	0.24	296	std::tuple<int, s
8.62	0.16	3	boost::containe
8.50	0.00	1	boost::containe
8.50	0.46	2	boost::containe
8.42	1.00	2	std::make_heap
8.39	0.00	(0)	global
8.12	0.30	31	std::_Tuple_imp
8.10	0.17	44	std::_Tuple_imp
7.96	0.29	4	boost::containe
7.56	0.34	2	boost::containe
7.09	0.00	1	std::__and_<std
7.09	0.00	1	std::is_nothrow
7.09	0.42	46	std::tuple<int, s
7.07	4.04	2	std::__adjust_he
6.81	0.00	2	boost::move_ba
6.76	0.00	1	std::__and_<std
6.76	0.00	1	std::__is_nothro
6.76	0.00	1	std::is_nothrow
6.63	0.07	1	std::is_construc
6.56	0.00	1	std::__is_constr
6.56	0.00	1	std::__is_direct
6.56	0.00	1	std::__is_direct
6.56	0.07	1	std::__is_direct
6.43	0.00	1	std::__and_<std
6.13	1.40	2	std::__final_inse
5.33	0.00	1	std::__is_direct
4.97	0.15	8	std::_Tuple_imp
4.53	0.00	(0)	global
4.50	0.79	67	std::tuple<Foo,
4.08	2.05	2	std::__insertion
4.05	0.79	2	std::__unguarde
4.02	0.07	20	std::__is_destru
4.01	0.10	8	std::_Tuple_imp
3.66	0.44	69	boost::containe
3.53	0.00	(0)	global

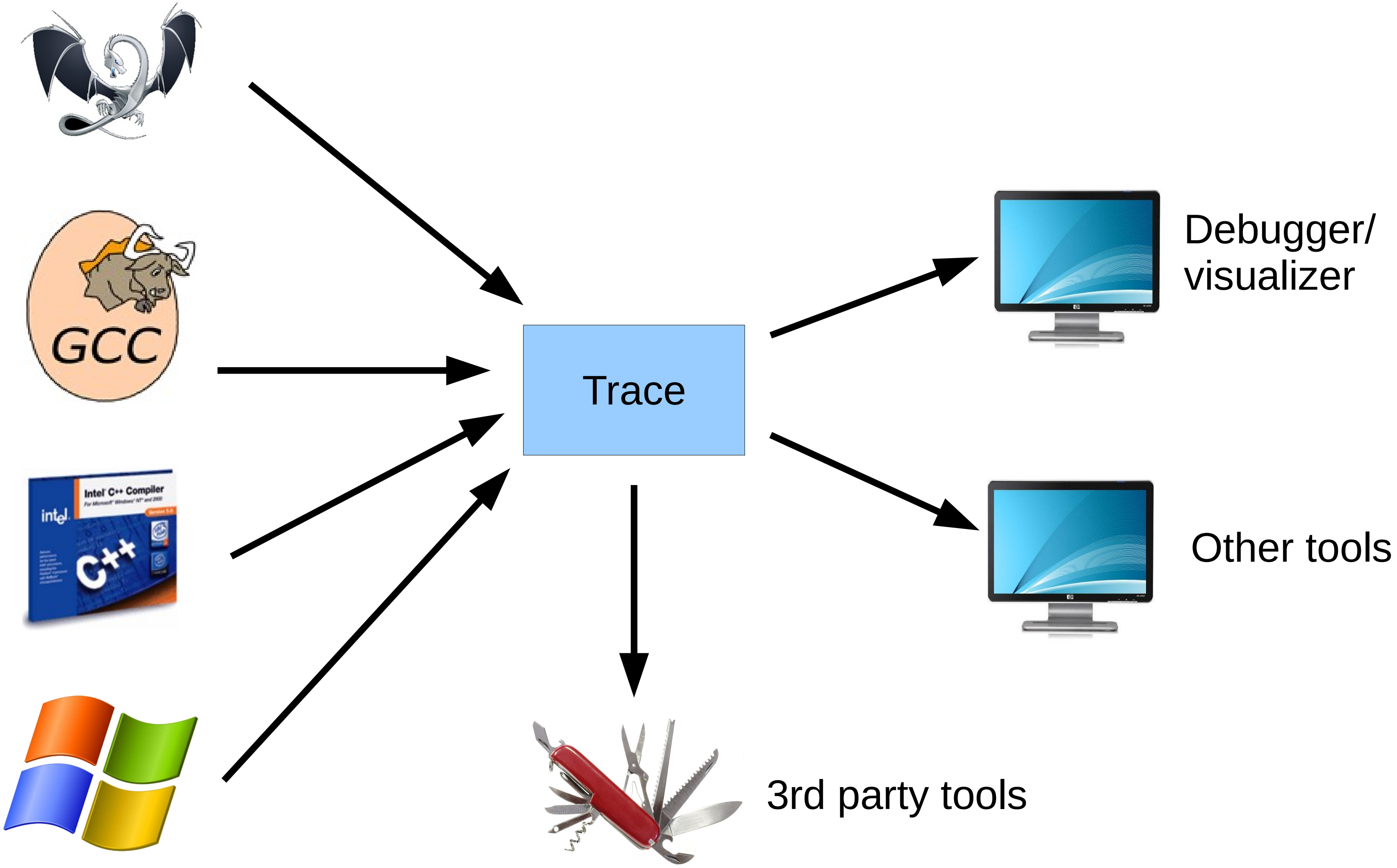
global

```
#    CTime   Source ('/home/mikael/dev/templight/tests/boost_containers_test.cpp')
     3.18    2 call(s) to 'boost::container::vector<std::tuple<int, std::basic_string<char>, std::tuple<Foo, int, std::basic_string<...
20
21           v.push_back(ValueType());
     0.36    2 call(s) to 'std::tuple<int, std::basic_string<char>, std::tuple<Foo, int, std::basic_string<char> > >::tuple'
     8.41    3 call(s) to 'std::tuple<int, std::basic_string<char>, std::tuple<Foo, int, std::basic_string<char> > >'
    13.20    1 call(s) to 'boost::container::vector<std::tuple<int, std::basic_string<char>, std::tuple<Foo, int, std::basic_string<...
22           v.insert(v.begin(), ValueType( 42, "Alice", InnerType(Foo{69}, 42, "Alice")));
     0.12    4 call(s) to 'boost::container::container_detail::vec_iterator<std::tuple<int, std::basic_string<char>, std::tuple<Foo,...
     0.16    4 call(s) to 'boost::container::container_detail::vec_iterator<std::tuple<int, std::basic_string<char>, std::tuple<Foo,...
     0.63    4 call(s) to '<cycle 3> via std::tuple<Foo, int, std::basic_string<char> >::tuple'
     1.59    2 call(s) to 'std::tuple<Foo, int, std::basic_string<char> >::tuple<Foo, int, char const (&)[6], void>'
     1.76    2 call(s) to 'std::tuple<int, std::basic_string<char>, std::tuple<Foo, int, std::basic_string<char> > >::tuple<int, char ...
     8.50    1 call(s) to 'boost::container::vector<std::tuple<int, std::basic_string<char>, std::tuple<Foo, int, std::basic_string<...
    10.12    6 call(s) to 'std::tuple<int, std::basic_string<char>, std::tuple<Foo, int, std::basic_string<char> > >::tuple'
23
```

global
76.80 %
71 x
26 x
2 x
1 x
ic_string<char>,
::basic_string<...
9 %
31 x
std::sort<boost::container::container_
detail::vec_iterator<std::tuple<int, std:...
21.20 %
boost::container::vector<std::tuple<int,
std::basic_string<char>, std::tuple<Foo...
13.20 %
2 x
2 x
2 x
nt, int>,
)[6], st...
std::__introsort_loop<boost::container::
container_detail::vec_iterator<std::tupl...
14.11 %
std::__final_insertic
container::container_d
6.1
2 x
2 x
_sort<boost::container::
tail::vec_iterator<std::tup...
9.71 %
std::__unguarded_partition_pivot<boost::
container::container_detail::vec_iterator...
4.05 %
std::__insertion_sort
container_detail::vec_
4.
2 x

boost_containers_test.o.trace.cg.out [1] - Total Compilation Time (ns) Cost: 241 364 881

Our vision



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?

Q/A

Templight: A Clang Extension for
Debugging and Profiling C++ Template Metaprograms
http://plc.inf.elte.hu/templight
gsd@elte.hu