Debugging and Profiling C++ Template Metaprograms

Zoltán Porkoláb[1,2]
Zoltán Borók-Nagy[1]
József Mihalicza[2]

[1] Ericsson Hungary
[2] Eötvös Loránd University

Agenda

- C++ Template Metaprogramming
- Possible debugging and profiling techiques
- Templight back-end tool
- Front-end tools
- Vision

Metaprogramming

”Metaprogramming is the writing of computer programs that write or manipulate other programs (or themselves) as their data, or that do part of the work at compile time that would otherwise be done at runtime.”

Wikipedia

C++ Template Metaprograms

```
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;
}
```

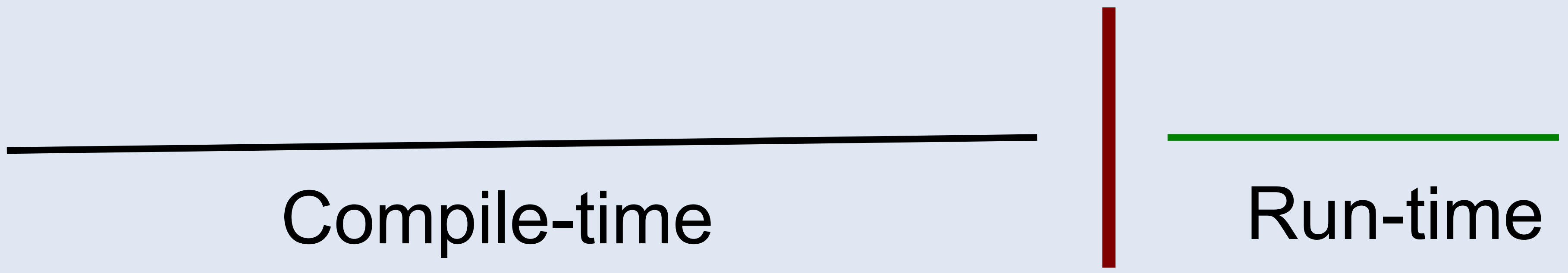
When to use metaprograms?

- Optimalisation, compile-time adaption
- Expression templates
- Static interface checking
- Simulating language extensions
- DSL embedding
- Many other areas ...

When to use metaprograms?

```
// pre C++11 code
template <class T, class S>
? max( T a, S b)
{
    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);
}
```

When to use metaprograms?



When to use metaprograms?

```
// pre C++11 code
template <class T, class S>
? max( T a, S b)
{
    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);
}
```

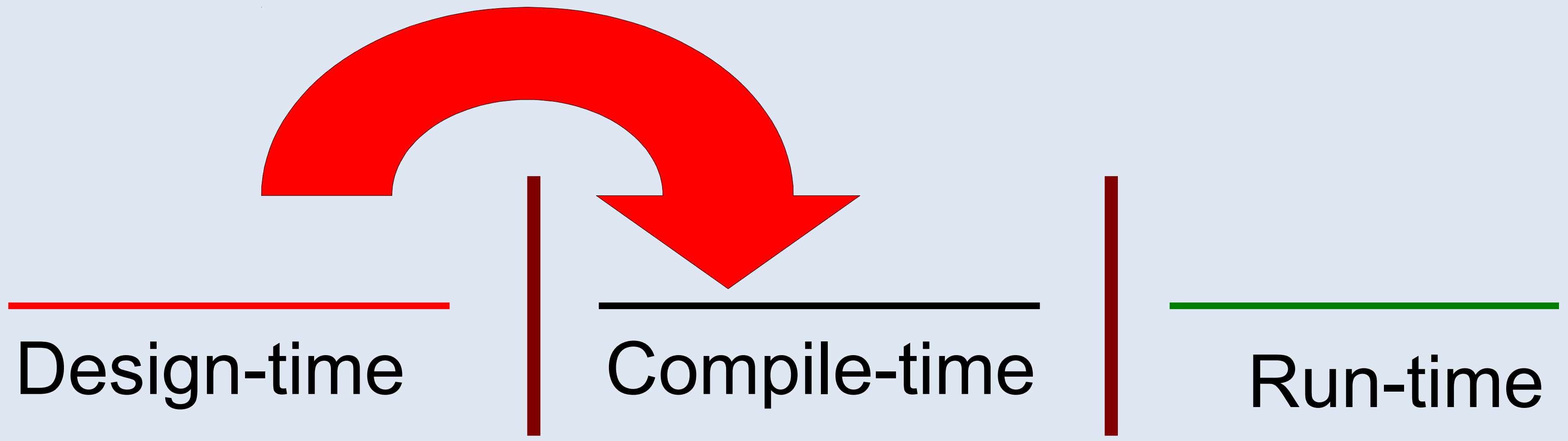
When to use metaprograms?

Design-time | Compile-time | Run-time

When to use metaprograms?

```
template <bool condition, class Then, class Else>
struct IF
{
    typedef Then RET;
};

template <class Then, class Else>
struct IF<false, Then, Else>
{
    typedef Else RET;
};

// we can be much more clever than this
template <class T, class S>
IF<sizeof(T)<sizeof(S),S,T>::RET max( T a, S b)
{
    if ( a > b )
        return a;
    else
        return b;
}
```

When to use metaprograms?



Run-time vs. Compile time

Run-time

Compile-time

Run-time vs. Compile time

Run-time

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

Compile-time

Run-time vs. Compile time

Run-time	Compile-time
Functions	Metafunctions (type)
Values, literals	Const, enum, constexpr
Data structures	Typelist (type)
If/else	Pattern matching
Loop	Recursion
Assignment	Ref. Transparency
May depend on input	Deterministic

C++ tool support

- Pretty good support for run-time C++

C++ tool support

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

C++ 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

C++ 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

Why we need tools?

- C++ syntax is not 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 programmers
- Programmers have less experience with template metaprograms

C++ Template Metaprogramming

```
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!

C++ Template Metaprogramming

```cpp
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;
}
```

C++ Template Metaprogramming

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

```
$ clang++ fact2.cpp
fact2.cpp:14:2: error: expected ';' after class
}
 ^
 ;
1 error generated.
```

C++ Template Metaprogramming

```
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;
}
```

```
$ clang++ fact2.cpp
fact2.cpp:14:2: error: expected ';' after class
}
 ^
 ;
1 error generated.
```

C++ Template Metaprogramming

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

```
$ clang++ fact6.cpp
fact6.cpp:5:34: error: no member named 'value' in 'Factorial<0>'
  enum { value = Factorial<N-1>::value * N };
                 ~~~~~~~~~~~~~~~~~^
fact6.cpp:5:18: note: in instantiation of template class 'Factorial<1>'
      requested here
  enum { value = Factorial<N-1>::value * N };
                 ^
fact6.cpp:5:18: note: in instantiation of template class 'Factorial<2>'
      requested here
  enum { value = Factorial<N-1>::value * N };
                 ^
fact6.cpp:5:18: note: in instantiation of template class 'Factorial<3>'
      requested here
  enum { value = Factorial<N-1>::value * N };
                 ^
fact6.cpp:5:18: note: in instantiation of template class 'Factorial<4>'
      requested here
  enum { value = Factorial<N-1>::value * N };
                 ^
fact6.cpp:16:21: note: in instantiation of template class 'Factorial<5>'
      requested here
  const int fact5 = Factorial<5>::value;
                    ^
1 error generated.
```

C++ Template Metaprogramming

```
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;
}
```

C++ Template Metaprogramming

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

```
$ 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.
```

C++ Template Metaprogramming

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

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

C++ Template Metaprogramming

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

```
$ 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:

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

Related

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

Run-time vs. Compile time

Run-time

- Running time
- Call stack
- Interactive

Compile-time

Run-time vs. Compile time

Run-time

- Running time
- Call stack
- Interactive

Compile-time

- Compilation time
- Instantiation chain
- Simulated interactive

Run-time vs. Compile time

Run-time	Compile-time
Running time	Compilation time
Call stack	Instantiation chain
Interactive	Simulated interactive

- Forward/backward step by step execution
- Break points
- Filters: eliminate unwanted events
- Visualization: source code + instantiations

Instrumentation

```
static int w(char *) { return 1; }

template <int N>
struct Factorial
{
  enum { begin = sizeof(w("")) };
  enum { value = Factorial<N-1>::value * N };
  enum { end = sizeof(w("")) };
};

template <>
struct Factorial<0>
{
  enum { begin = sizeof(w("")) };
  enum { value = 1 };
  enum { end = sizeof(w("")) };
};

int main()
{
  const int fact5 = Factorial<5>::value;
}
```

Instrumentation

```
static int w(char *) { return 1; }

template <int N
struct Factoria
{
   enum { begin
   enum { value
   enum { end =
};

template <>
struct Factoria
{
   enum { begin
   enum { value
   enum { end =
};

int main()
{
   const int fac
}
```

```
$ clang++ factw.cpp 2>&1 | grep -v "\^" | grep -v warning | grep -v enum | grep -v const

factw.cpp:23:21: note: in instantiation of template class 'Factorial<5>' requested here
factw.cpp:8:18: note: in instantiation of template class 'Factorial<4>' requested here
factw.cpp:23:21: note: in instantiation of template class 'Factorial<5>' requested here
factw.cpp:8:18: note: in instantiation of template class 'Factorial<3>' requested here
factw.cpp:8:18: note: in instantiation of template class 'Factorial<4>' requested here
factw.cpp:23:21: note: in instantiation of template class 'Factorial<5>' requested here
factw.cpp:8:18: note: in instantiation of template class 'Factorial<2>' requested here
factw.cpp:8:18: note: in instantiation of template class 'Factorial<3>' requested here
factw.cpp:8:18: note: in instantiation of template class 'Factorial<4>' requested here
factw.cpp:23:21: note: in instantiation of template class 'Factorial<5>' requested here
factw.cpp:8:18: note: in instantiation of template class 'Factorial<1>' requested here
factw.cpp:8:18: note: in instantiation of template class 'Factorial<2>' requested here
factw.cpp:8:18: note: in instantiation of template class 'Factorial<3>' requested here
factw.cpp:8:18: note: in instantiation of template class 'Factorial<4>' requested here
factw.cpp:23:21: note: in instantiation of template class 'Factorial<5>' requested here
factw.cpp:8:18: note: in instantiation of template class 'Factorial<2>' requested here
factw.cpp:8:18: note: in instantiation of template class 'Factorial<3>' requested here
factw.cpp:8:18: note: in instantiation of template class 'Factorial<4>' requested here
factw.cpp:23:21: note: in instantiation of template class 'Factorial<5>' requested here
factw.cpp:8:18: note: in instantiation of template class 'Factorial<3>' requested here
factw.cpp:8:18: note: in instantiation of template class 'Factorial<4>' requested here
factw.cpp:23:21: note: in instantiation of template class 'Factorial<5>' requested here
factw.cpp:8:18: note: in instantiation of template class 'Factorial<4>' requested here
factw.cpp:23:21: note: in instantiation of template class 'Factorial<5>' requested here
factw.cpp:23:21: note: in instantiation of template class 'Factorial<5>' requested here
```

Instrumentation

```
static int w(char *) { return 1; }

template <int N
struct Factoria
{
   enum { begin
   enum { value
   enum { end =
};

template <>
struct Factoria
{
   enum { begin
   enum { value
   enum { end =
};

int main()
{
   const int fac
}
```

```
$ clang++ factw.cpp 2>&1 | grep -v "\^" | grep -v warning | grep -v enum | grep -v const

factw.cpp:23:21: note: in instantiation of template class 'Factorial<5>' requested here
factw.cpp:8:18: note: in instantiation of template class 'Factorial<4>' requested here
factw.cpp:23:21: note: in instantiation of template class 'Factorial<5>' requested here
factw.cpp:8:18: note: in instantiation of template class 'Factorial<3>' requested here
factw.cpp:8:18: note: in instantiation of template class 'Factorial<4>' requested here
factw.cpp:23:21: note: in instantiation of template class 'Factorial<5>' requested here
factw.cpp:8:18: note: in instantiation of template class 'Factorial<2>' requested here
factw.cpp:8:18: note: in instantiation of template class 'Factorial<3>' requested here
factw.cpp:8:18: note: in instantiation of template class 'Factorial<4>' requested here
factw.cpp:23:21: note: in instantiation of template class 'Factorial<5>' requested here
factw.cpp:8:18: note: in instantiation of template class 'Factorial<1>' requested here
factw.cpp:8:18: note: in instantiation of template class 'Factorial<2>' requested here
factw.cpp:8:18: note: in instantiation of template class 'Factorial<3>' requested here
factw.cpp:8:18: note: in instantiation of template class 'Factorial<4>' requested here
factw.cpp:23:21: note: in instantiation of template class 'Factorial<5>' requested here
factw.cpp:8:18: note: in instantiation of template class 'Factorial<2>' requested here
factw.cpp:8:18: note: in instantiation of template class 'Factorial<3>' requested here
factw.cpp:8:18: note: in instantiation of template class 'Factorial<4>' requested here
factw.cpp:23:21: note: in instantiation of template class 'Factorial<5>' requested here
factw.cpp:8:18: note: in instantiation of template class 'Factorial<3>' requested here
factw.cpp:8:18: note: in instantiation of template class 'Factorial<4>' requested here
factw.cpp:23:21: note: in instantiation of template class 'Factorial<5>' requested here
factw.cpp:8:18: note: in instantiation of template class 'Factorial<4>' requested here
factw.cpp:23:21: note: in instantiation of template class 'Factorial<5>' requested here
factw.cpp:23:21: note: in instantiation of template class 'Factorial<5>' requested here
```

Instrumentation

```
static int w(char *) { return 1; }

template <int N
struct Factoria
{
  enum { begin
  enum { value
  enum { end =
};

template <>
struct Factoria
{
  enum { begin
  enum { value
  enum { end =
};

int main()
{
  const int fac
}
```

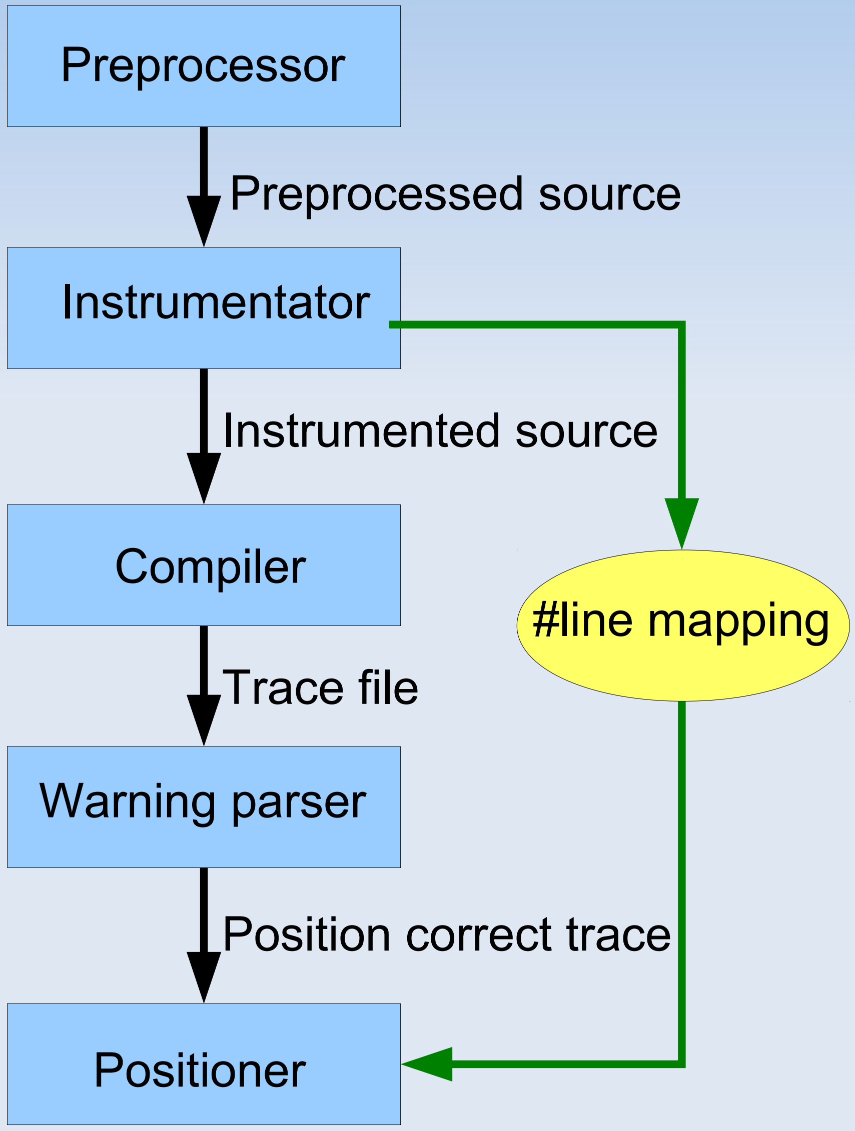
```
$ clang++ factw.cpp 2>&1 | grep -v "\^" | grep -v warning | grep -v enum | grep -v const

factw.cpp:23:21: note: in instantiation of template class 'Factorial<5>' requested here
factw.cpp:8:18: note: in instantiation of template class 'Factorial<4>' requested here

factw.cpp:8:18: note: in instantiation of template class 'Factorial<3>' requested here


factw.cpp:8:18: note: in instantiation of template class 'Factorial<2>' requested here



factw.cpp:8:18: note: in instantiation of template class 'Factorial<1>' requested here




factw.cpp:8:18: note: in instantiation of template class 'Factorial<2>' requested here



factw.cpp:8:18: note: in instantiation of template class 'Factorial<3>' requested here


factw.cpp:8:18: note: in instantiation of template class 'Factorial<4>' requested here

factw.cpp:23:21: note: in instantiation of template class 'Factorial<5>' requested here
```

Some history

- GPCE 2006, Portland:
  - Porkoláb, Mihalicza, Sipos:

    Debugging C++ template metaprograms
  - Templight 1.0
  - Based on warning injection

Templight 1.0



```
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 ------------ */
};
```

Advantages of instrumentation

- Light-way approach
  - (compared to compiler hack)
  - We used wave
- Easier to port
  - Just change the warning generator source
  - But significant differencies between compilers

Issues with instrumentation

- Parsing
- Memoization
- Inheritance
- Not easy to port the warning generator
- No profiling information
  - Collecting and timestamping warnings are delayed
  - Warning generation is costly

Compiler support

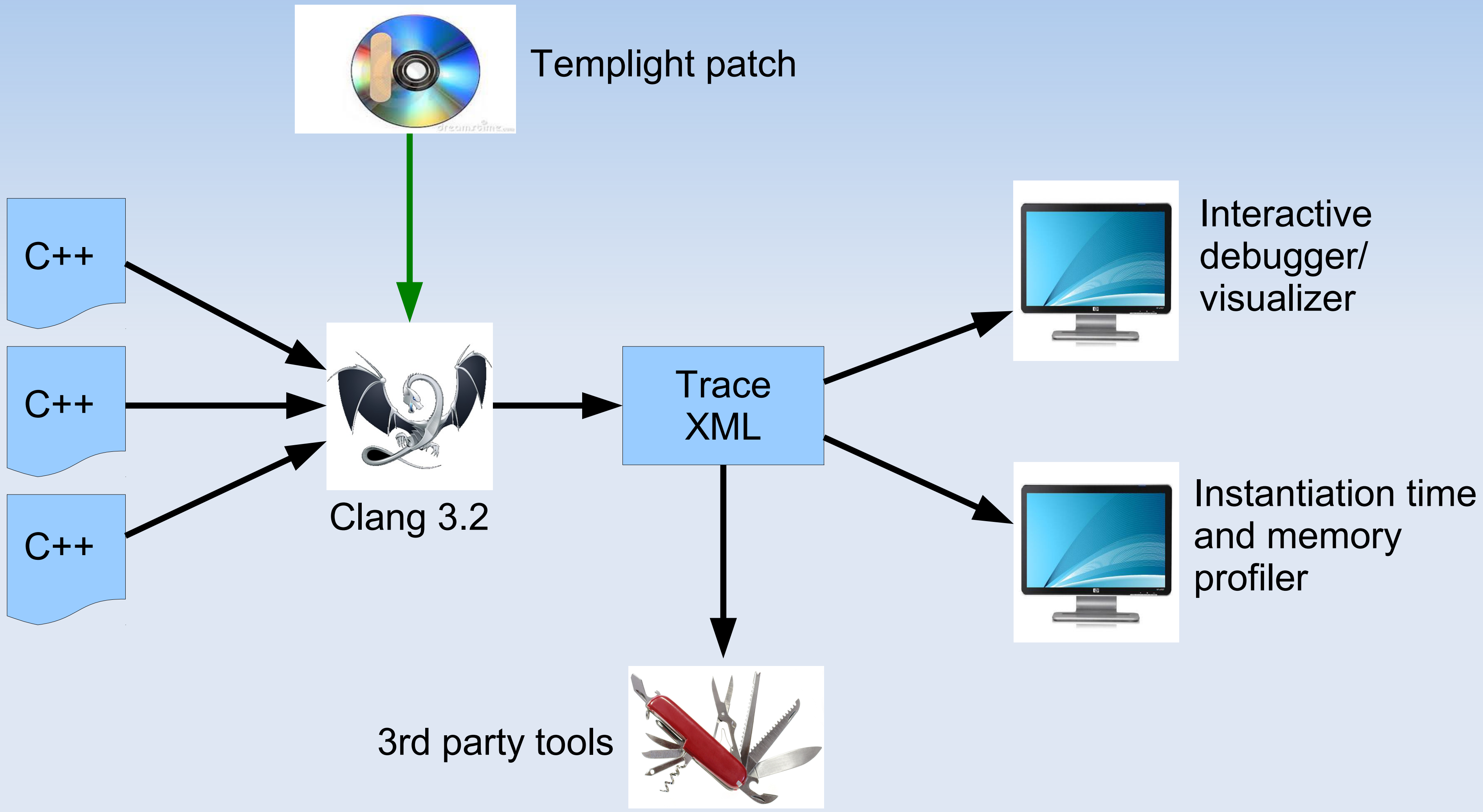
Good quality template metaprogram debugger and profiler

requires compiler support!

Templight 2.0

- Based on Clang 3.2
- Patch to
  - Detect/measure instantiation
  - Detect memoization
  - Measure memory consumption (optional)
  - Put timestamp on events
- Emit trace in XML format
- Front-end tools
  - Visual debugger
  - Profiler data viewer

Templight 2.0


Templight patch
C++
C++
C++
Clang 3.2
Trace XML
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 3.2
- Patch and build clang
- Build front-end tools (optional)
  - >=Qt 4.6 and >=Graphviz 2.28.0 required
  - $ qmake; make

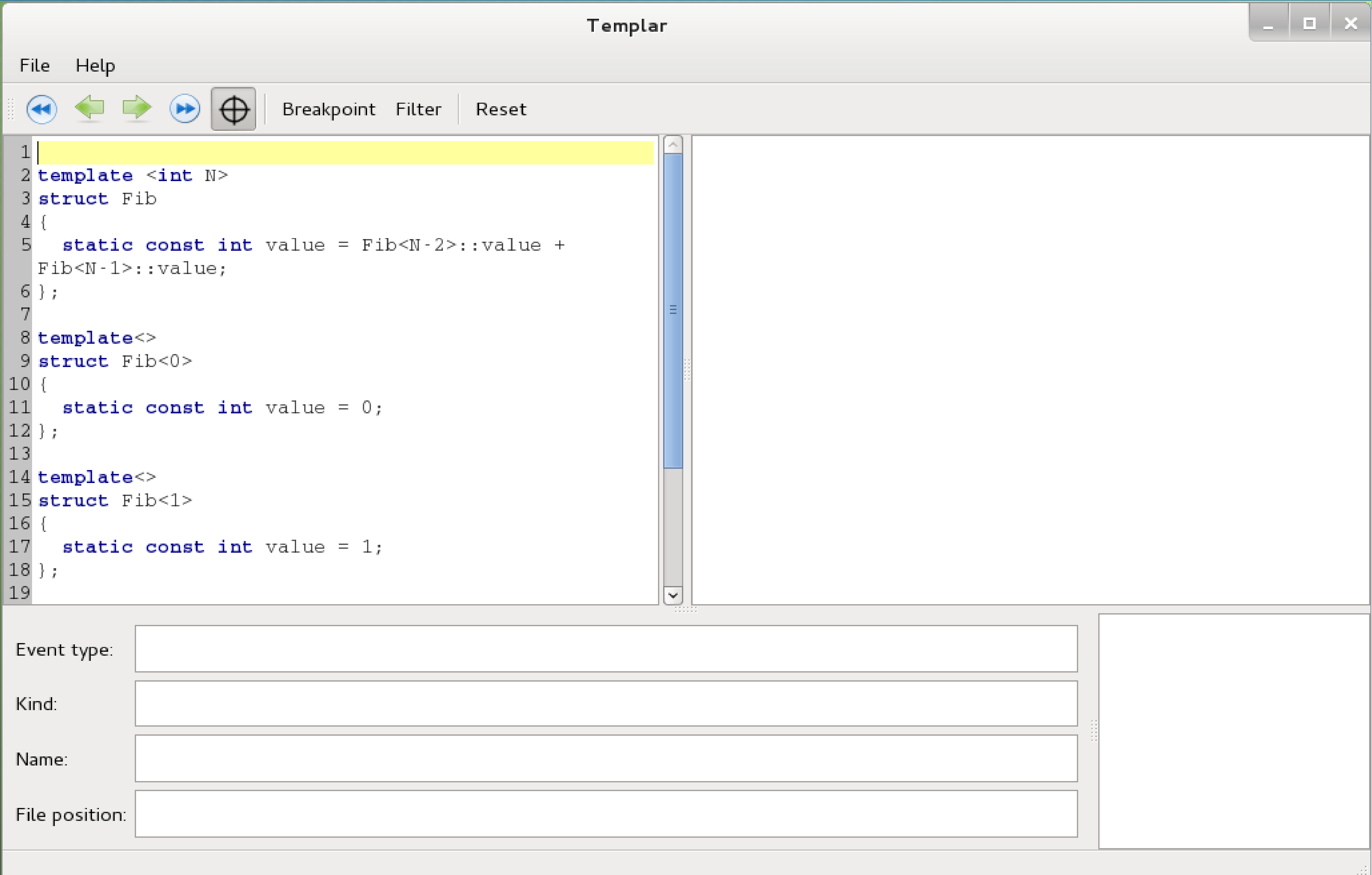
How to use

```
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

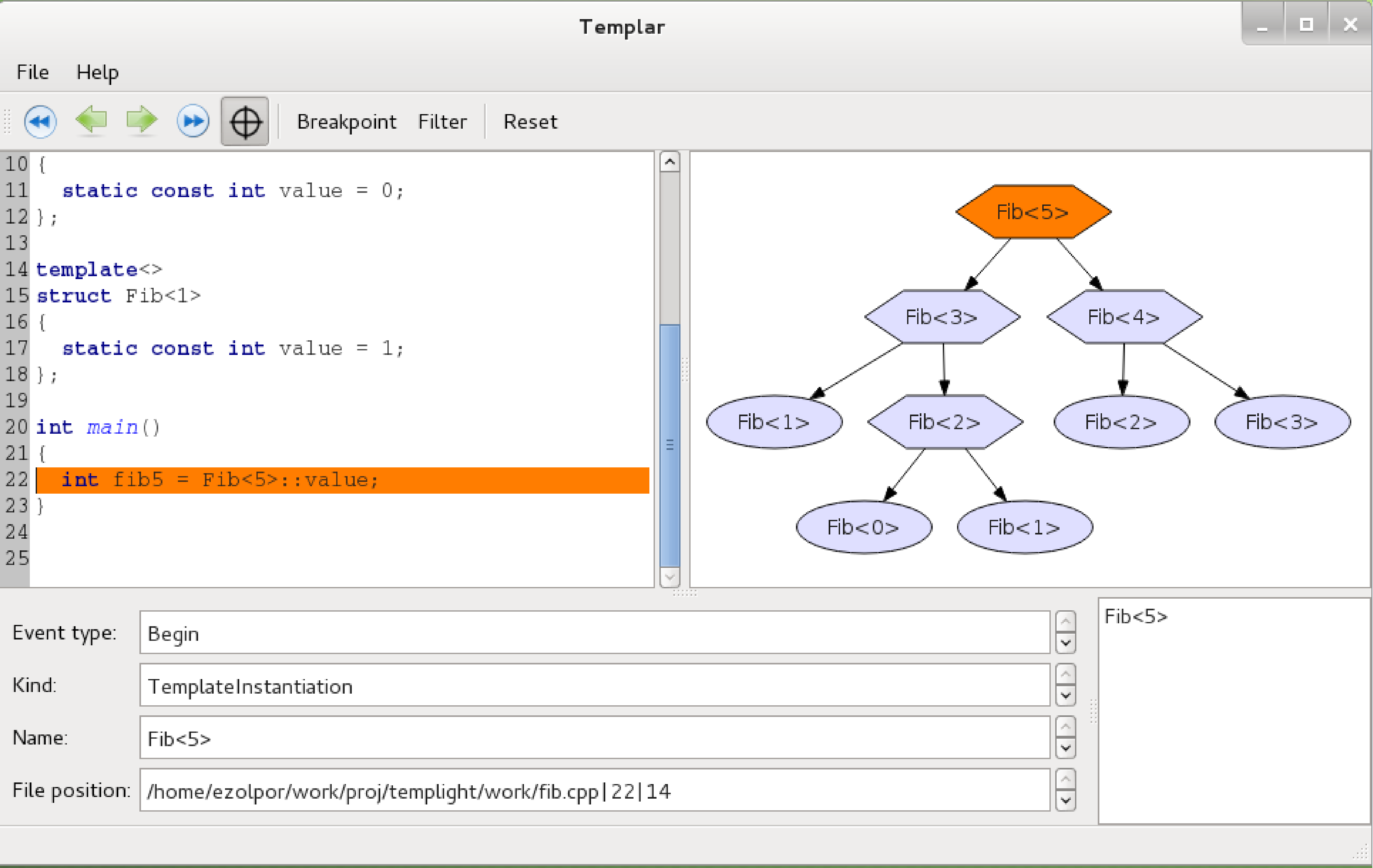
```
$ clang++ -templight fib.cpp


$ 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



```

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;
};

```

Templar



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

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

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

Event type: Begin

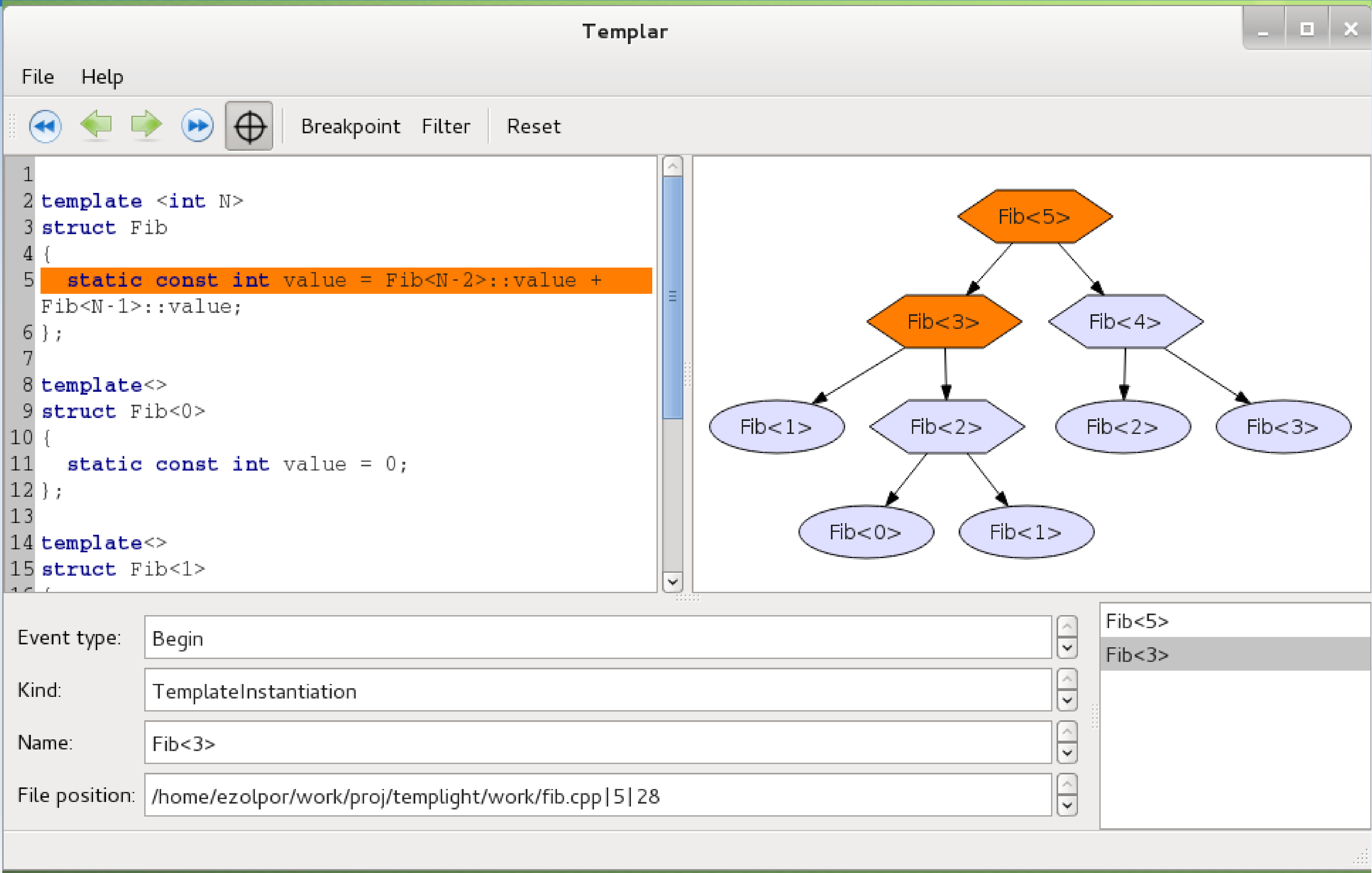
Kind: TemplateInstantiation

Name: Fib<5>

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

Fib<5>

Templar

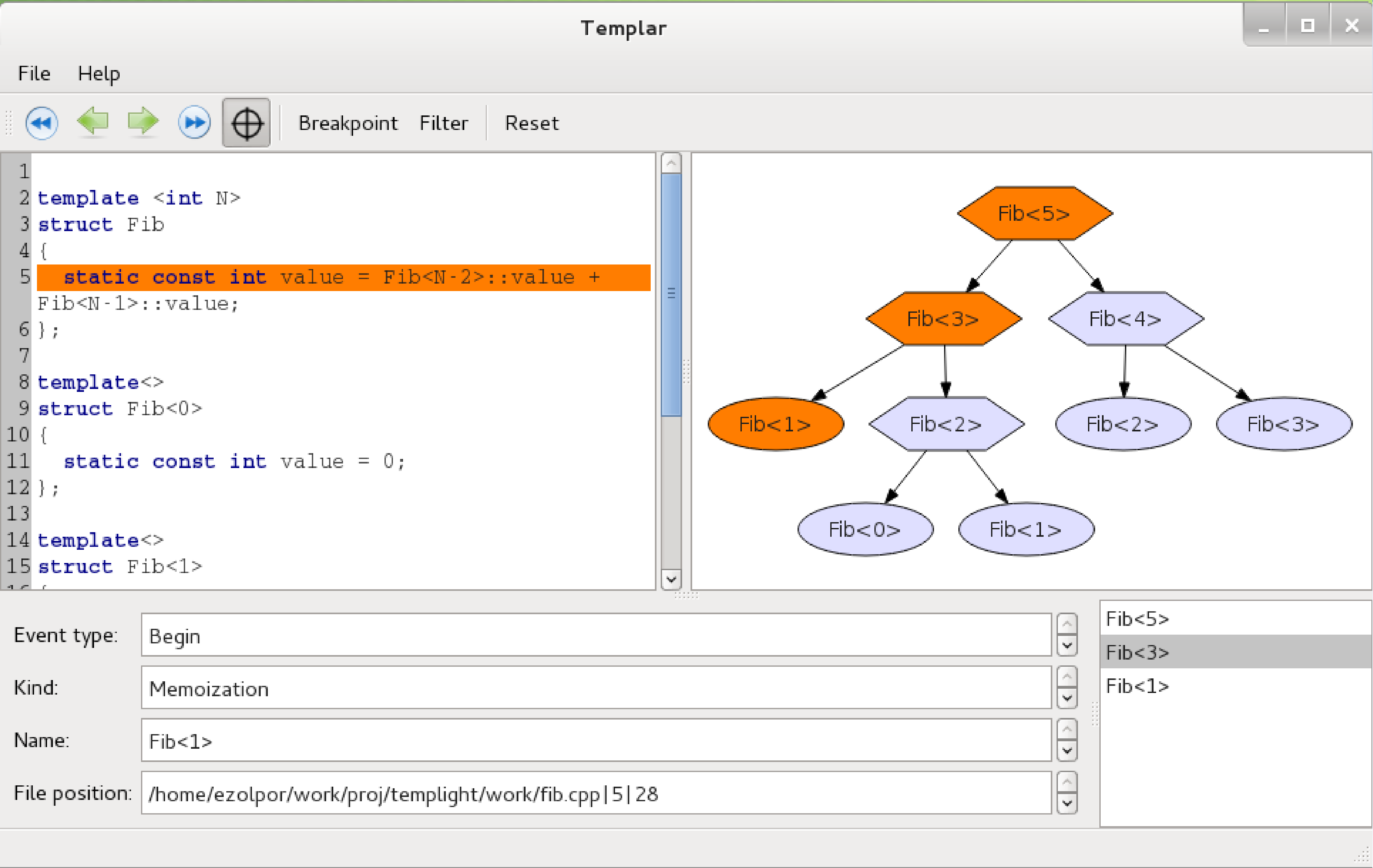


```
template <int N>
struct Fib
{
  static const int value = Fib<N-2>::value +
Fib<N-1>::value;
};

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>

Templar



```
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

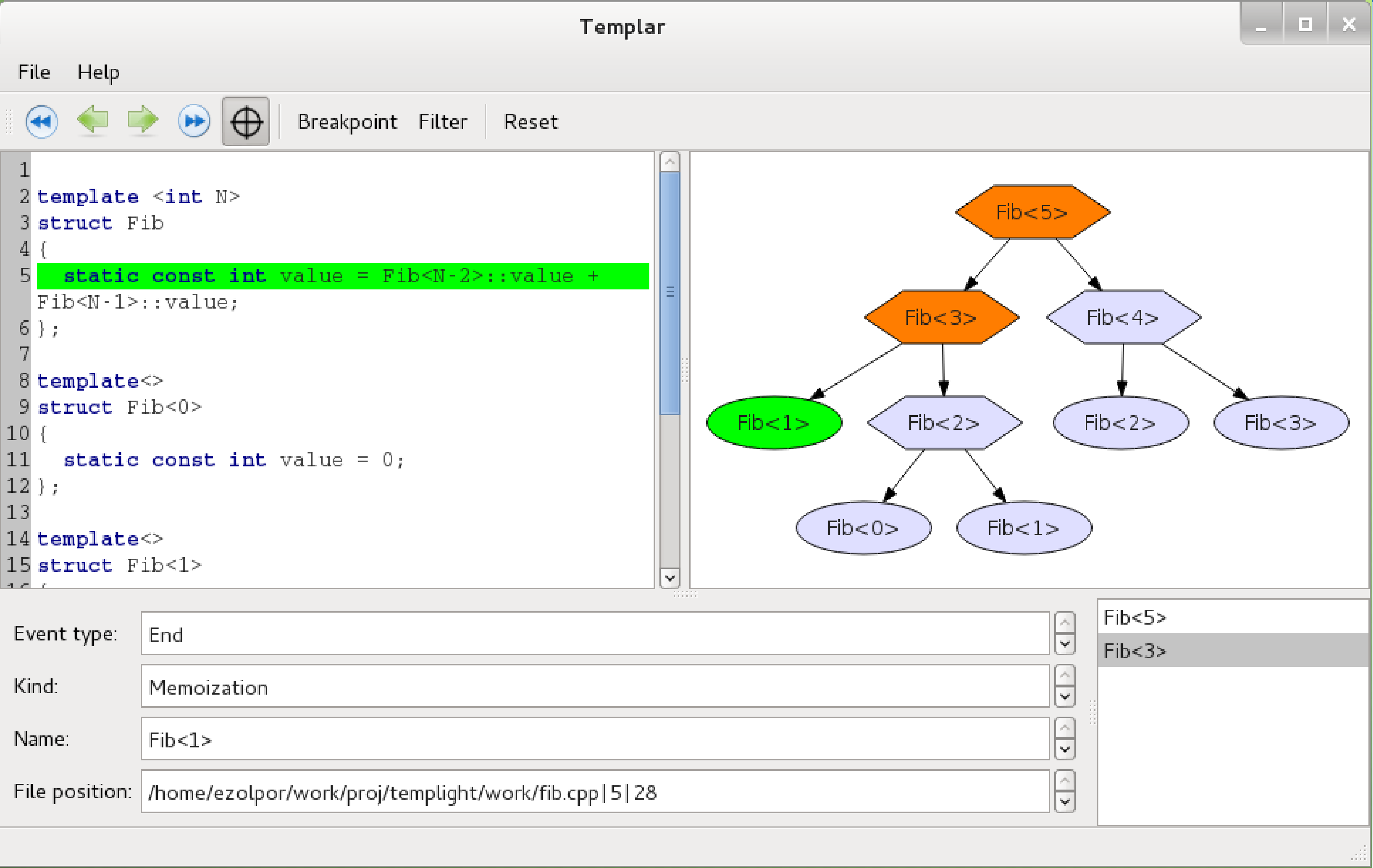
Kind: Memoization

Name: Fib<1>

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

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

Templar



```
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

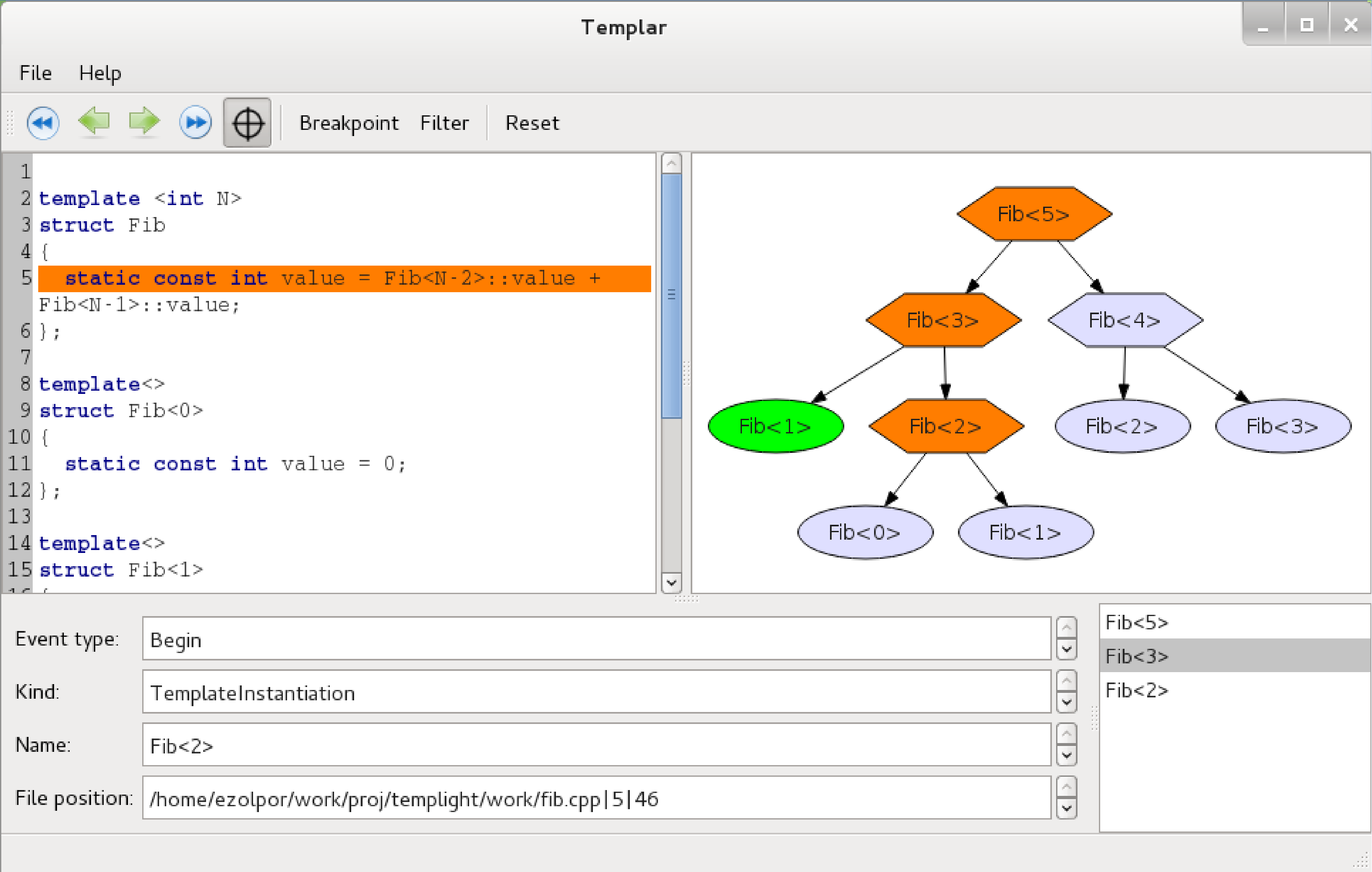
Kind: Memoization

Name: Fib<1>

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

Fib<5>
Fib<3>

Templar

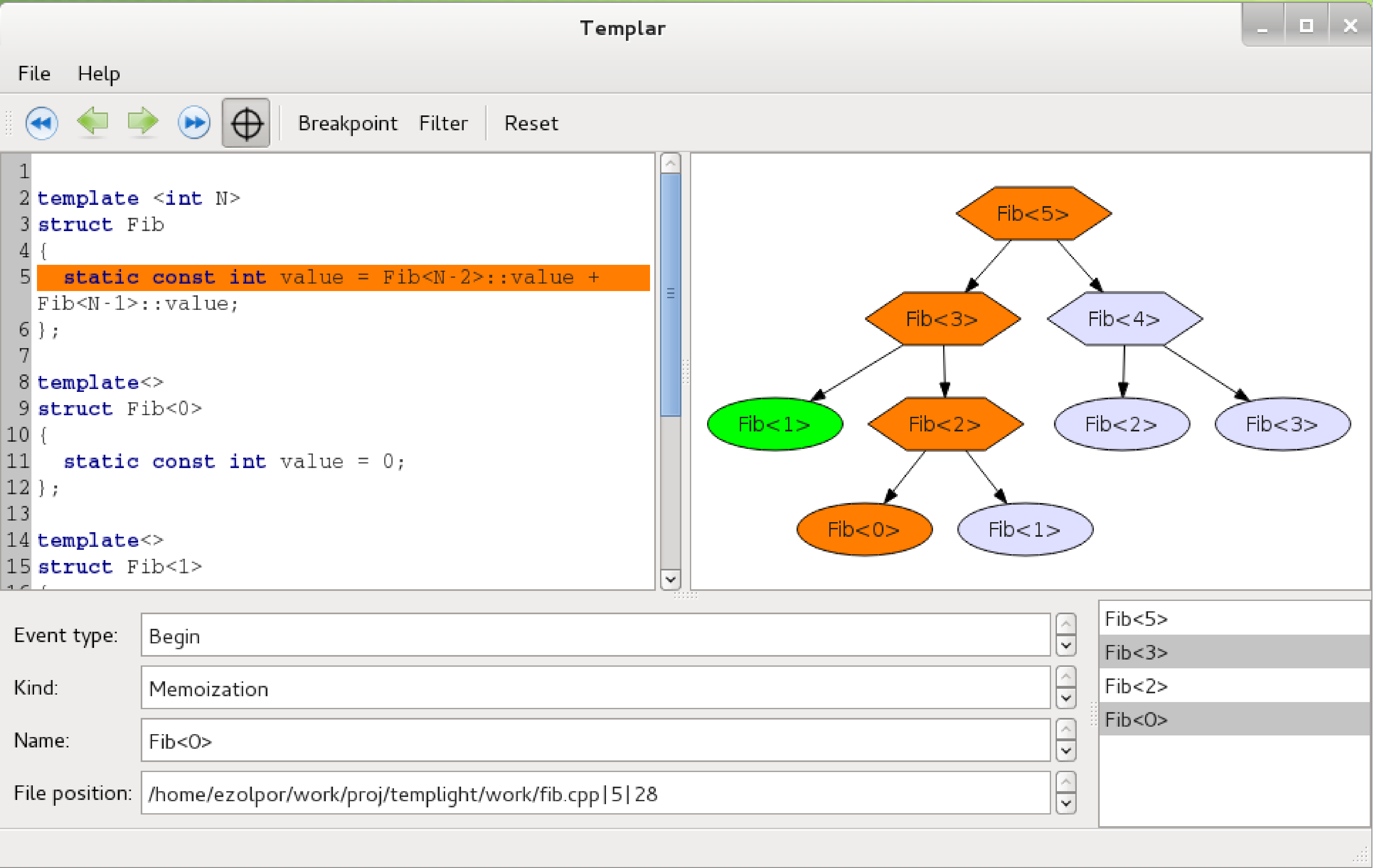


```
template <int N>
struct Fib
{
  static const int value = Fib<N-2>::value +
Fib<N-1>::value;
};

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>

Templar

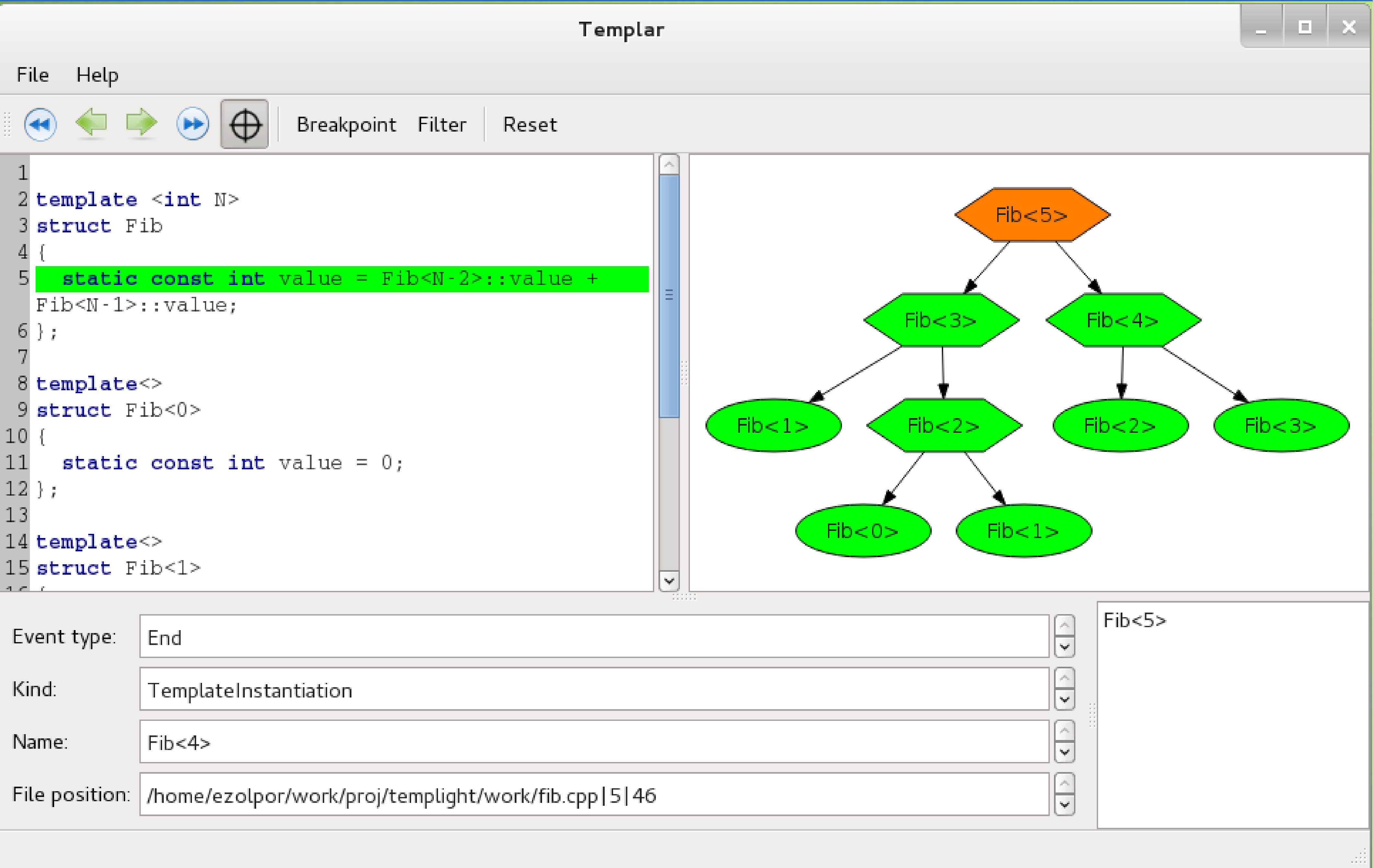


```
template <int N>
struct Fib
{
  static const int value = Fib<N-2>::value +
Fib<N-1>::value;
};

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>

Templar



```
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

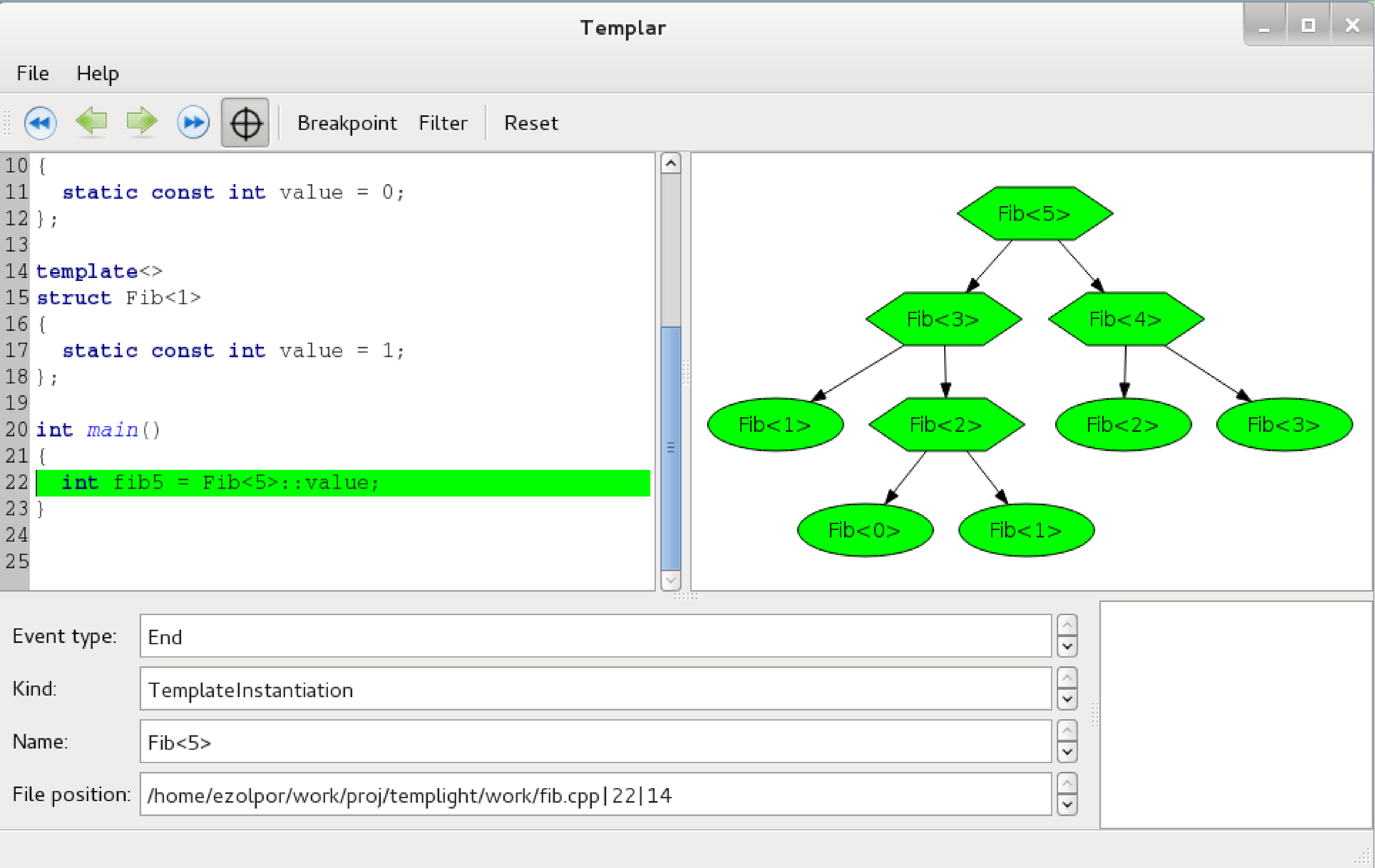
Kind: TemplateInstantiation

Name: Fib<4>

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

Fib<5>

Templar



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

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

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

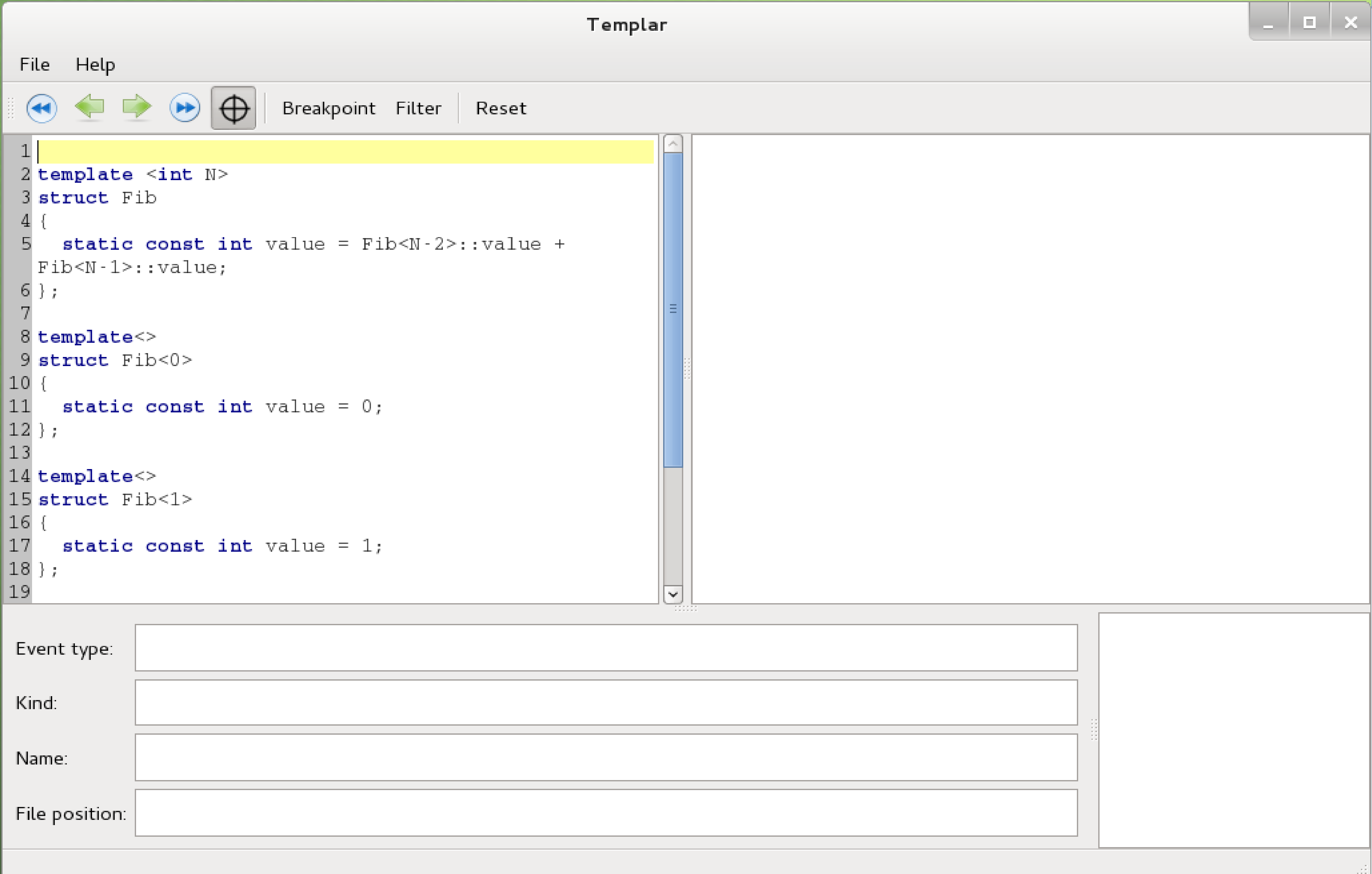
Event type: End

Kind: TemplateInstantiation

Name: Fib<5>

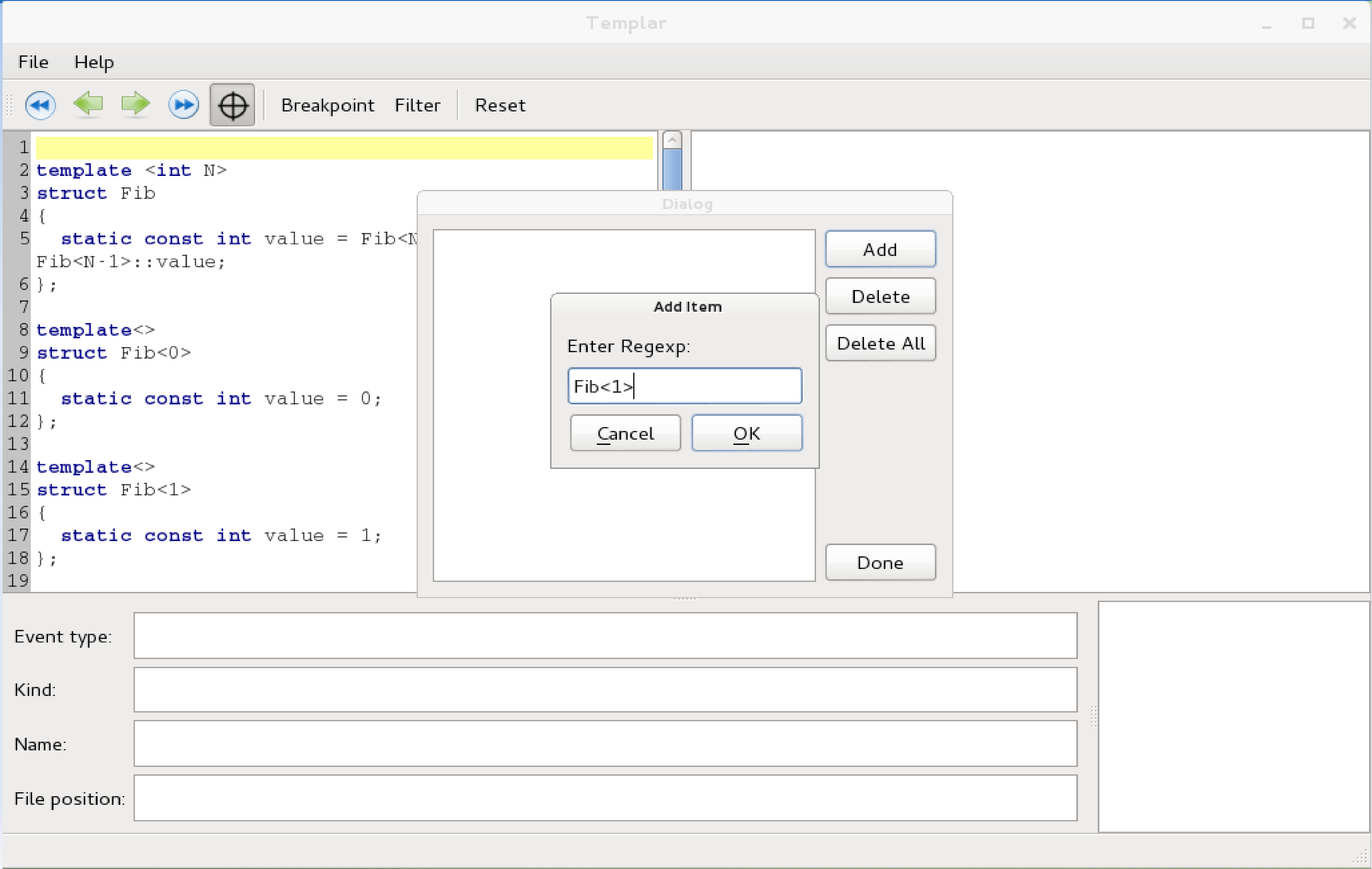
File position: /home/ezolpor/work/proj/templight/work/fib.cpp|22|14

Breakpoint



```

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;
};

```

Breakpoint



```
template <int N>
struct Fib
{
  static const int value = Fib<N
Fib<N-1>::value;
};

template<>
struct Fib<0>
{
  static const int value = 0;
};

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

Enter Regexp: Fib<1>

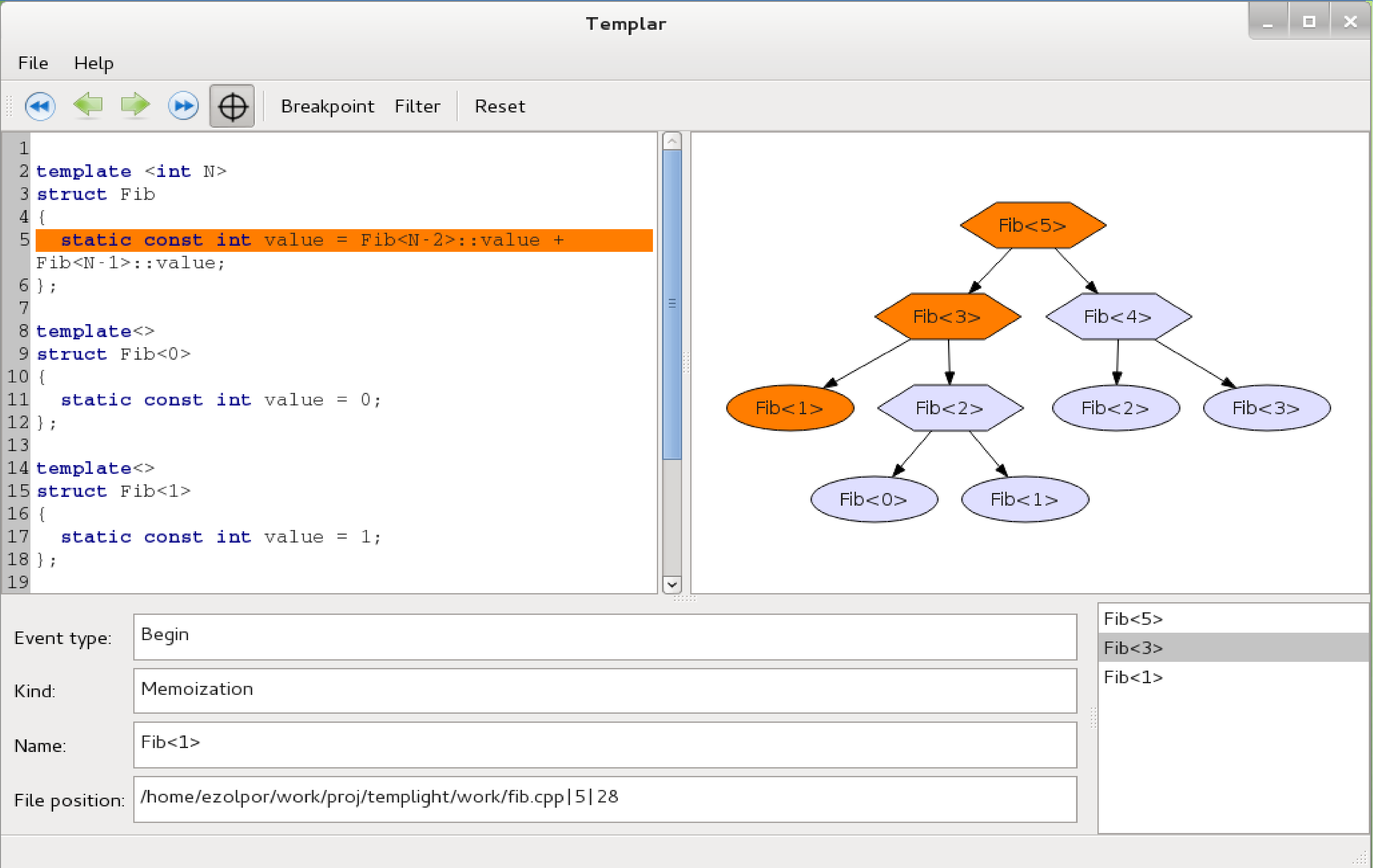
Event type:

Kind:

Name:

File position:

Breakpoint

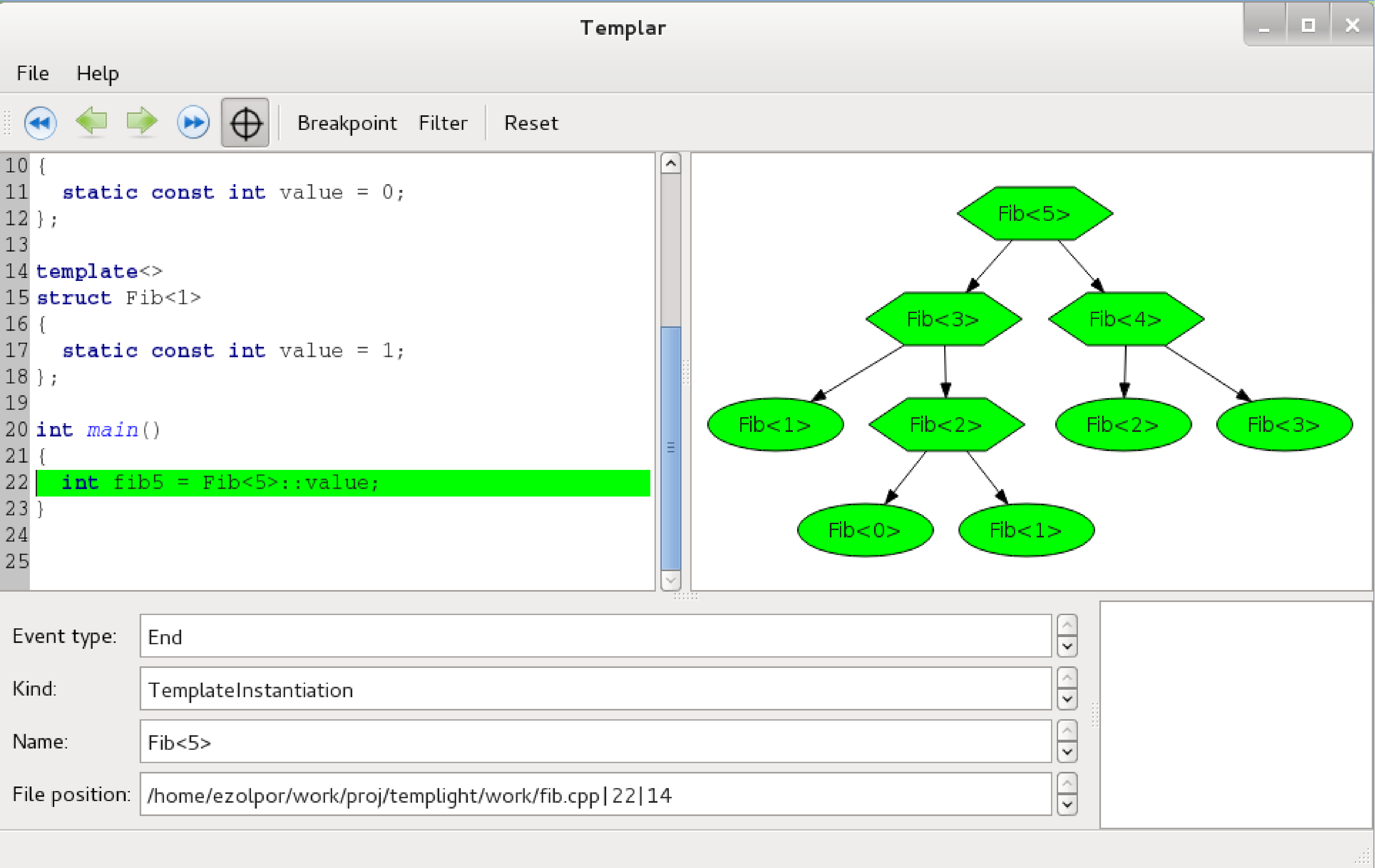


```
template <int N>
struct Fib
{
  static const int value = Fib<N-2>::value +
Fib<N-1>::value;
};

template<>
struct Fib<0>
{
  static const int value = 0;
};

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

Event type: Begin
Kind: Memoization
Name: Fib<1>
File position: /home/ezolpor/work/proj/templight/work/fib.cpp|5|28

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

Breakpoint



```
{
  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

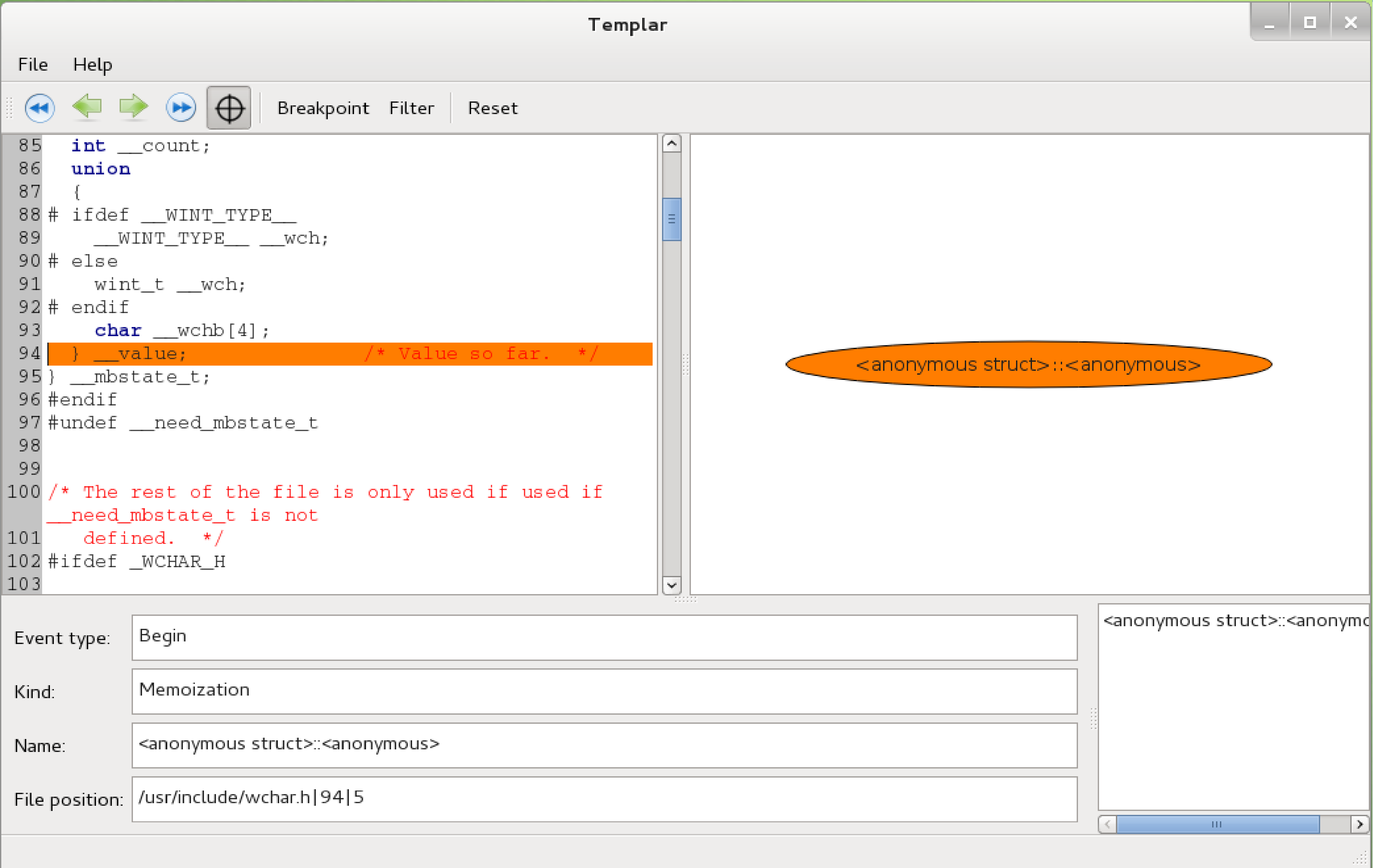
Filter

```
#include <iostream>

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;
}
```

Filter

```
$ 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"/>
```

Filter

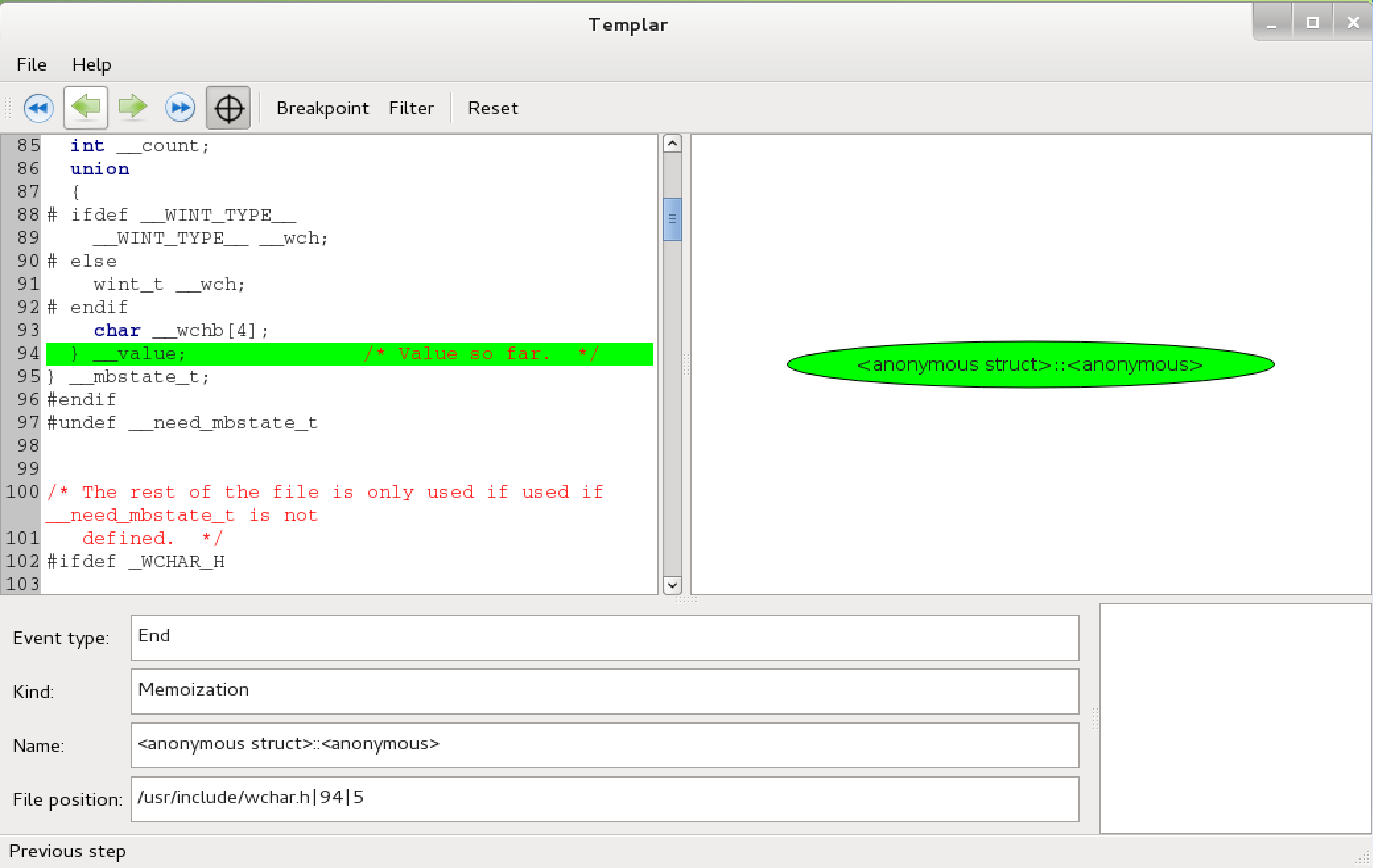


```
  int __count;
  union
  {
# ifdef __WINT_TYPE__
    __WINT_TYPE__ __wch;
# else
    wint_t __wch;
# endif
    char __wchb[4];
  } __value;              /* Value so far.  */
} __mbstate_t;
#endif
#undef __need_mbstate_t


/* The rest of the file is only used if used if
__need_mbstate_t is not
   defined.  */
#ifdef _WCHAR_H
```

Event type:	Begin
Kind:	Memoization
Name:	<anonymous struct>::<anonymous>
File position:	/usr/include/wchar.h|94|5

<anonymous struct>::<anonymo

Filter



```
  int __count;
  union
  {
# ifdef __WINT_TYPE__
    __WINT_TYPE__ __wch;
# else
    wint_t __wch;
# endif
    char __wchb[4];
  } __value;              /* Value so far.  */
} __mbstate_t;
#endif
#undef __need_mbstate_t


/* The rest of the file is only used if used if
__need_mbstate_t is not
   defined.  */
#ifdef _WCHAR_H

```

Event type: End

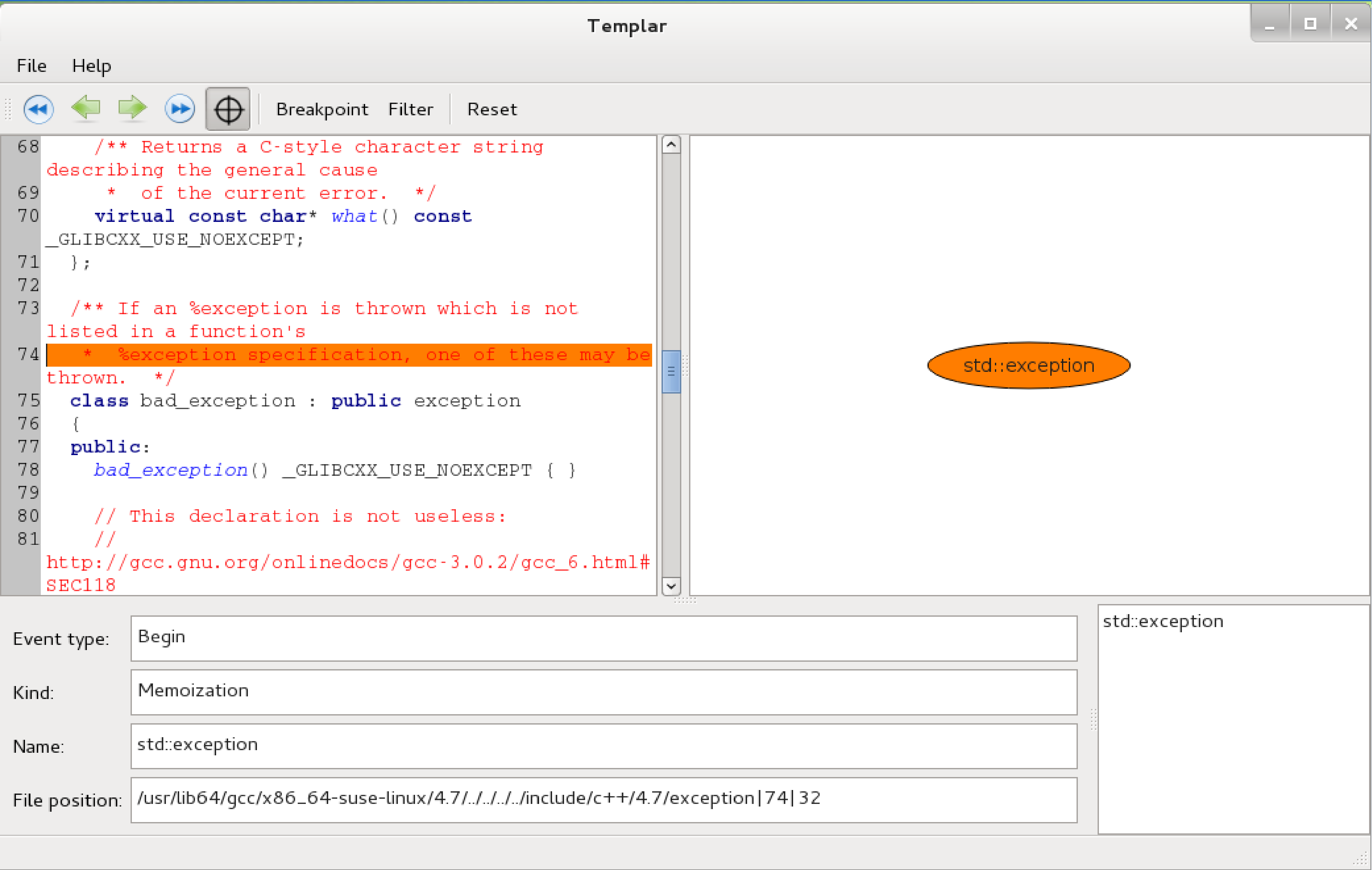
Kind: Memoization

Name: <anonymous struct>::<anonymous>

File position: /usr/include/wchar.h|94|5

Previous step

Filter



```
    /** Returns a C-style character string
describing the general cause
     *  of the current error.  */
    virtual const char* what() const
_GLIBCXX_USE_NOEXCEPT;
  };

  /** If an %exception is thrown which is not
listed in a function's
   *  %exception specification, one of these may be
thrown.  */
  class bad_exception : public exception
  {
  public:
    bad_exception() _GLIBCXX_USE_NOEXCEPT { }

    // This declaration is not useless:
    //
http://gcc.gnu.org/onlinedocs/gcc-3.0.2/gcc_6.html#
SEC118
```

Event type: Begin

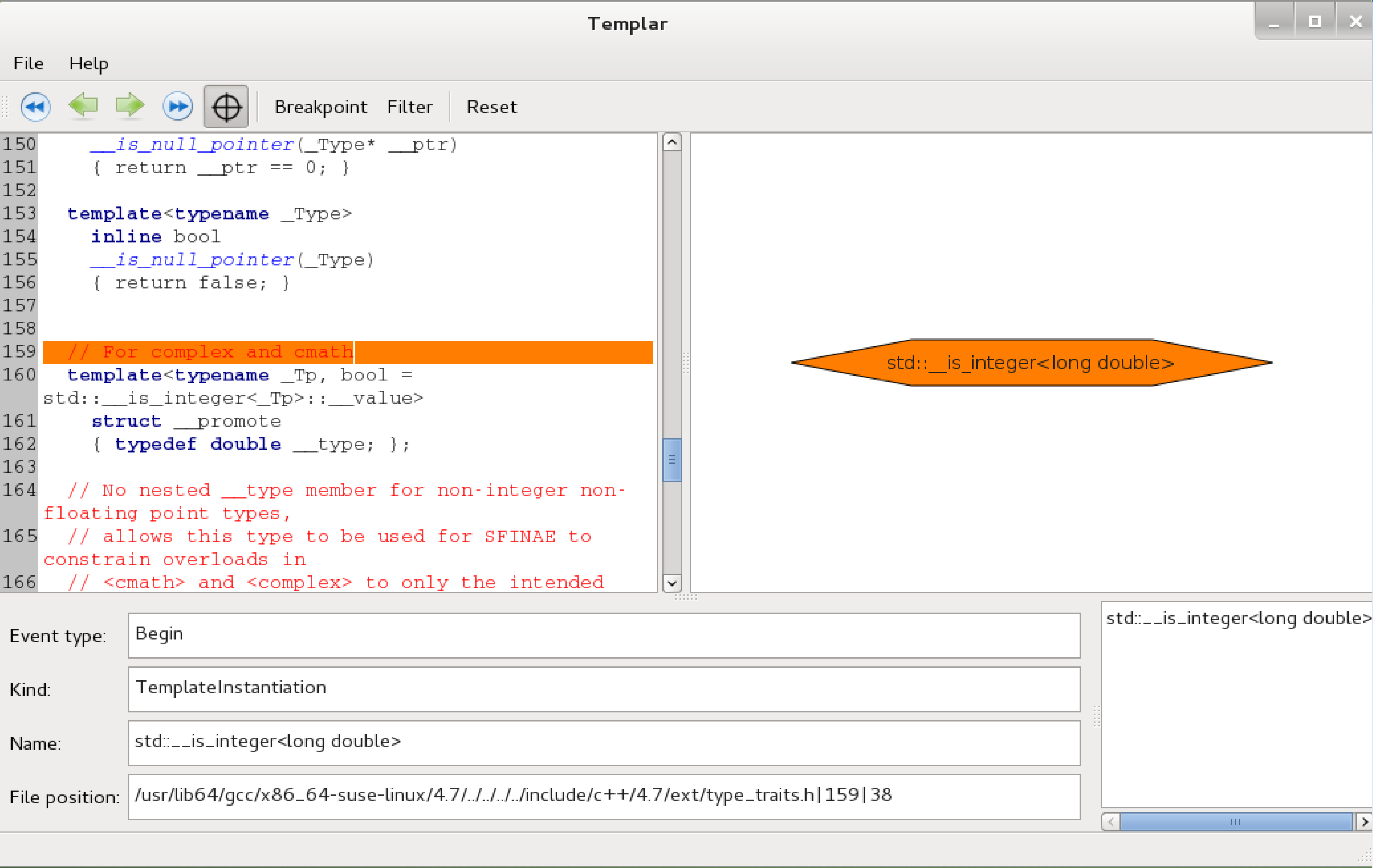
Kind: Memoization

Name: std::exception

File position: /usr/lib64/gcc/x86_64-suse-linux/4.7/../../../../include/c++/4.7/exception|74|32

std::exception

Filter



```
    __is_null_pointer(_Type* __ptr)
    { return __ptr == 0; }

  template<typename _Type>
    inline bool
    __is_null_pointer(_Type)
    { return false; }


  // For complex and cmath
  template<typename _Tp, bool =
std::__is_integer<_Tp>::__value>
    struct __promote
    { typedef double __type; };

  // No nested __type member for non-integer non-
floating point types,
  // allows this type to be used for SFINAE to
constrain overloads in
  // <cmath> and <complex> to only the intended
```

Event type: Begin

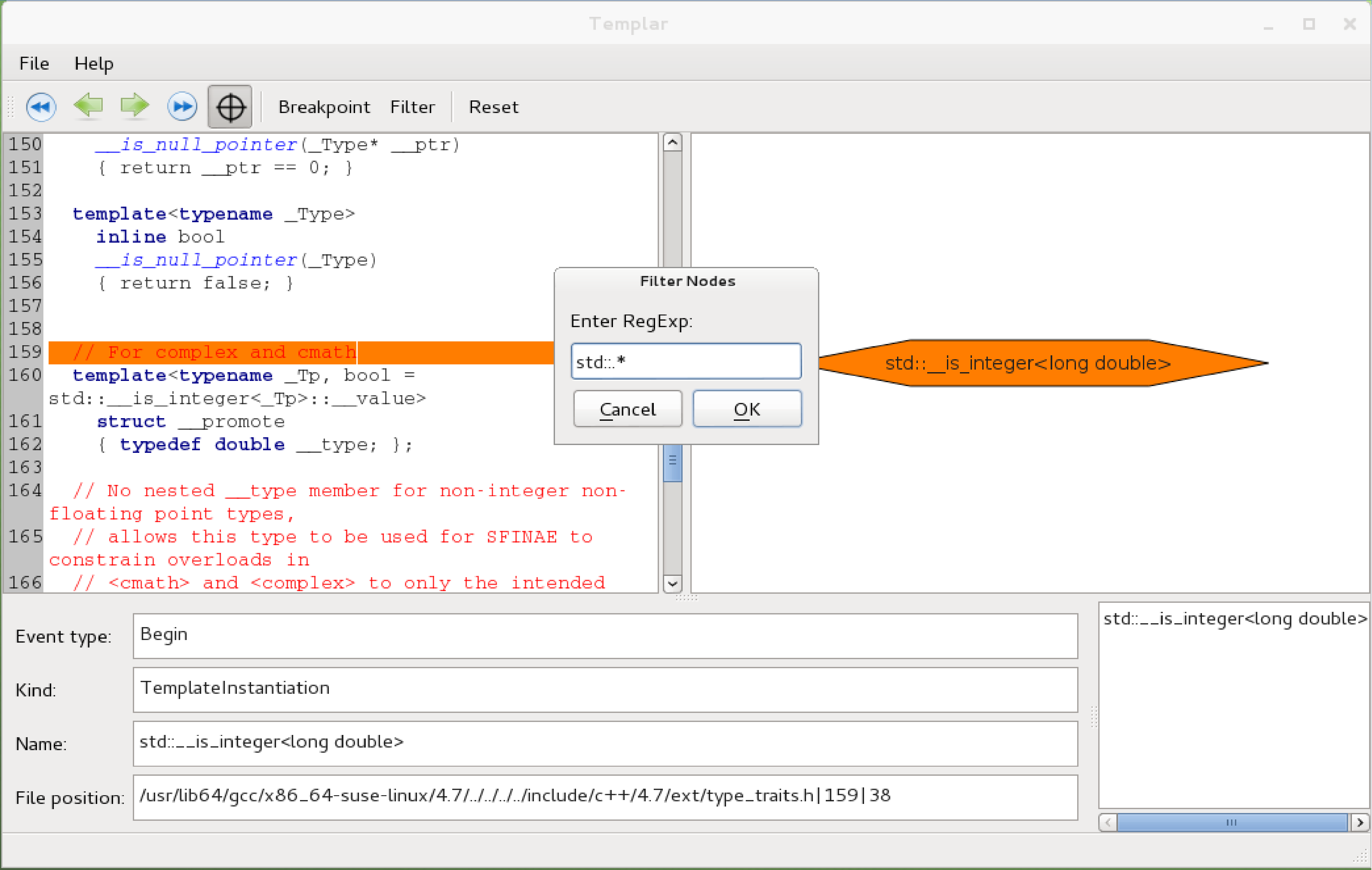
Kind: TemplateInstantiation

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>

Filter



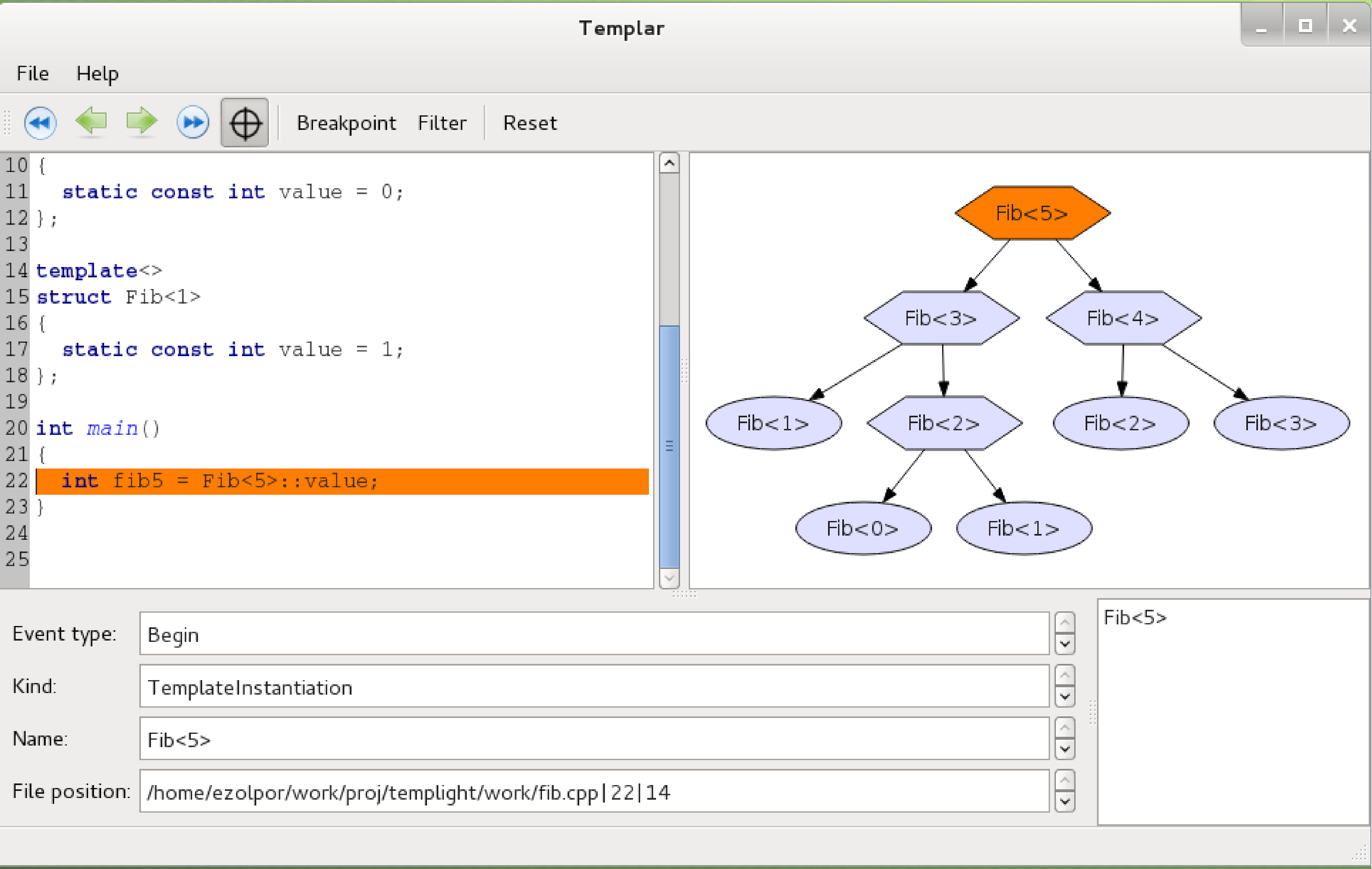
```
    __is_null_pointer(_Type* __ptr)
    { return __ptr == 0; }

  template<typename _Type>
    inline bool
    __is_null_pointer(_Type)
    { return false; }


  // For complex and cmath
  template<typename _Tp, bool =
std::__is_integer<_Tp>::__value>
    struct __promote
    { typedef double __type; };

  // No nested __type member for non-integer non-
floating point types,
  // allows this type to be used for SFINAE to
constrain overloads in
  // <cmath> and <complex> to only the intended
```

Enter RegExp: std::.*

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

Filter



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

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

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

Event type: Begin

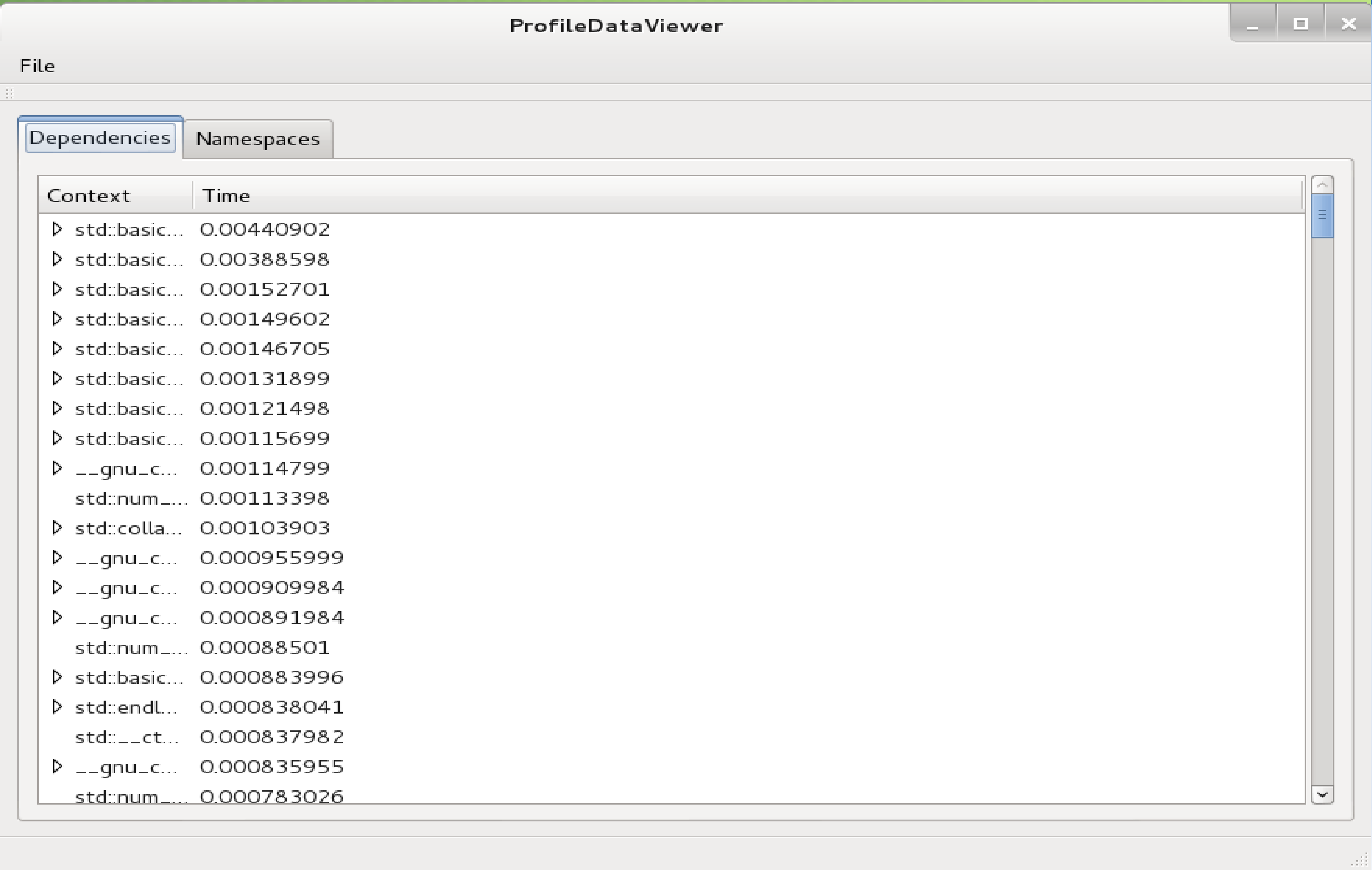
Kind: TemplateInstantiation

Name: Fib<5>

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

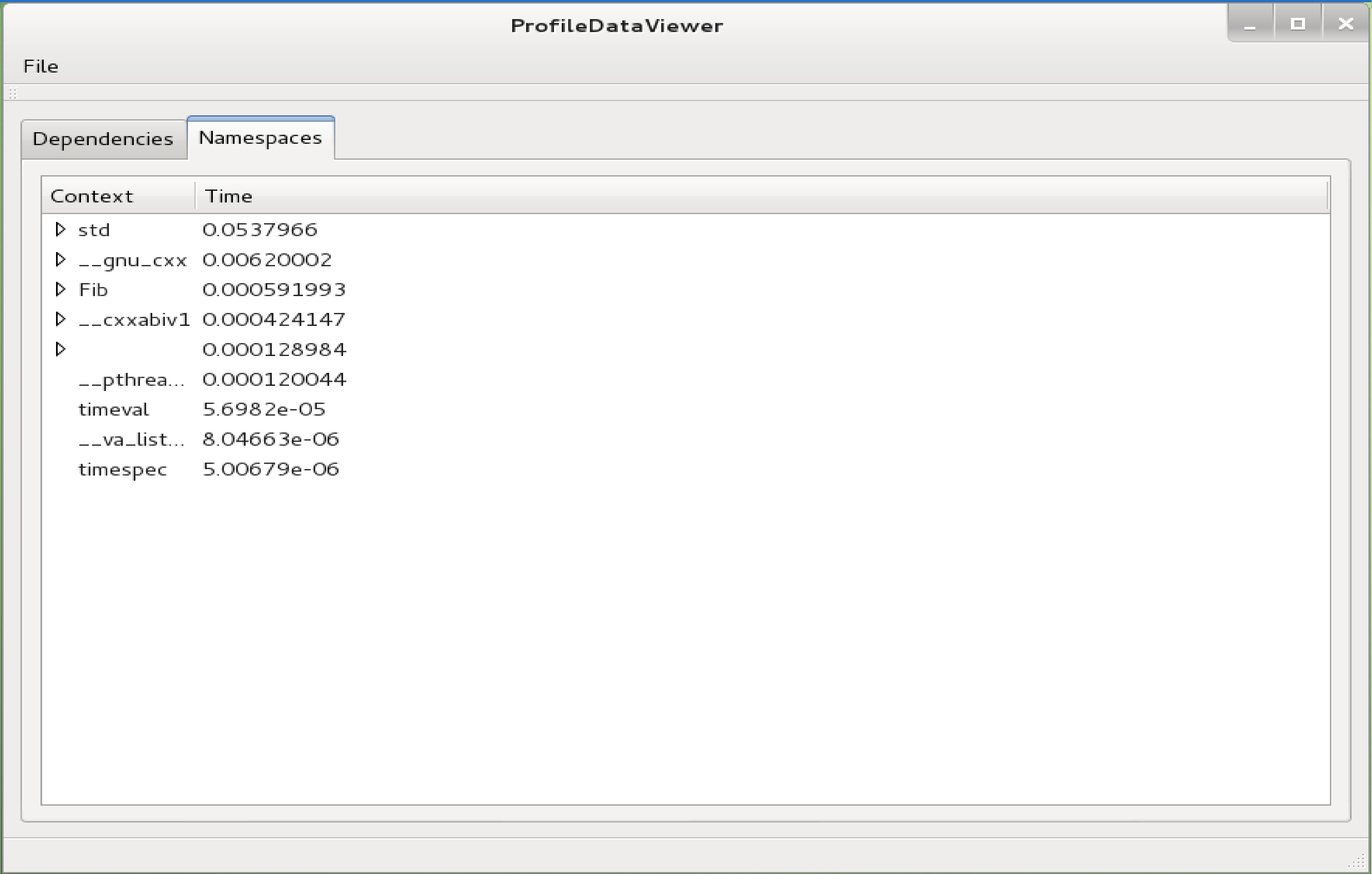
Fib<5>

Profiler



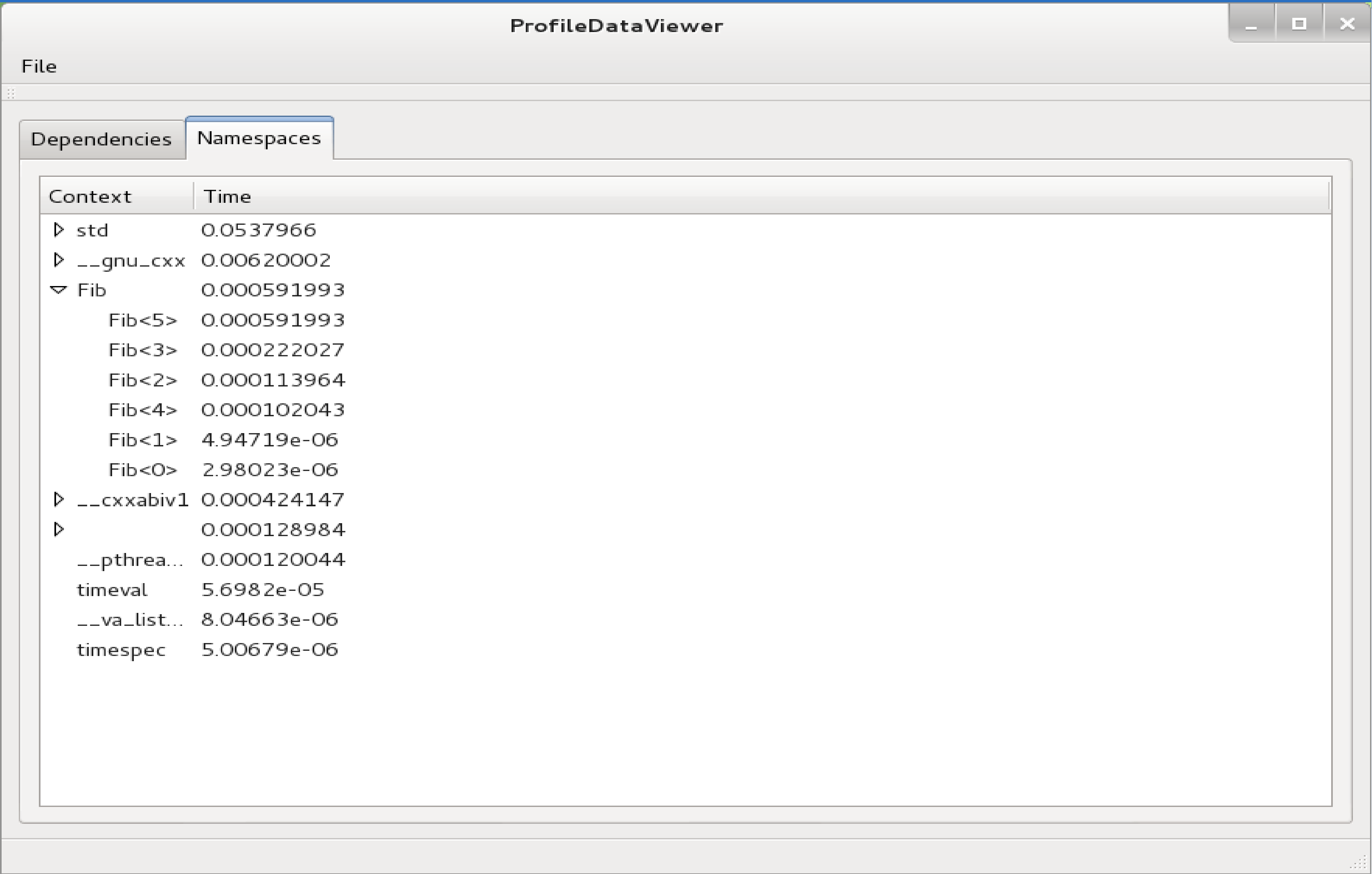
Context	Time
std::basic...	0.00440902
std::basic...	0.00388598
std::basic...	0.00152701
std::basic...	0.00149602
std::basic...	0.00146705
std::basic...	0.00131899
std::basic...	0.00121498
std::basic...	0.00115699
__gnu_c...	0.00114799
std::num_...	0.00113398
std::colla...	0.00103903
__gnu_c...	0.000955999
__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

Profiler



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

Profiler



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

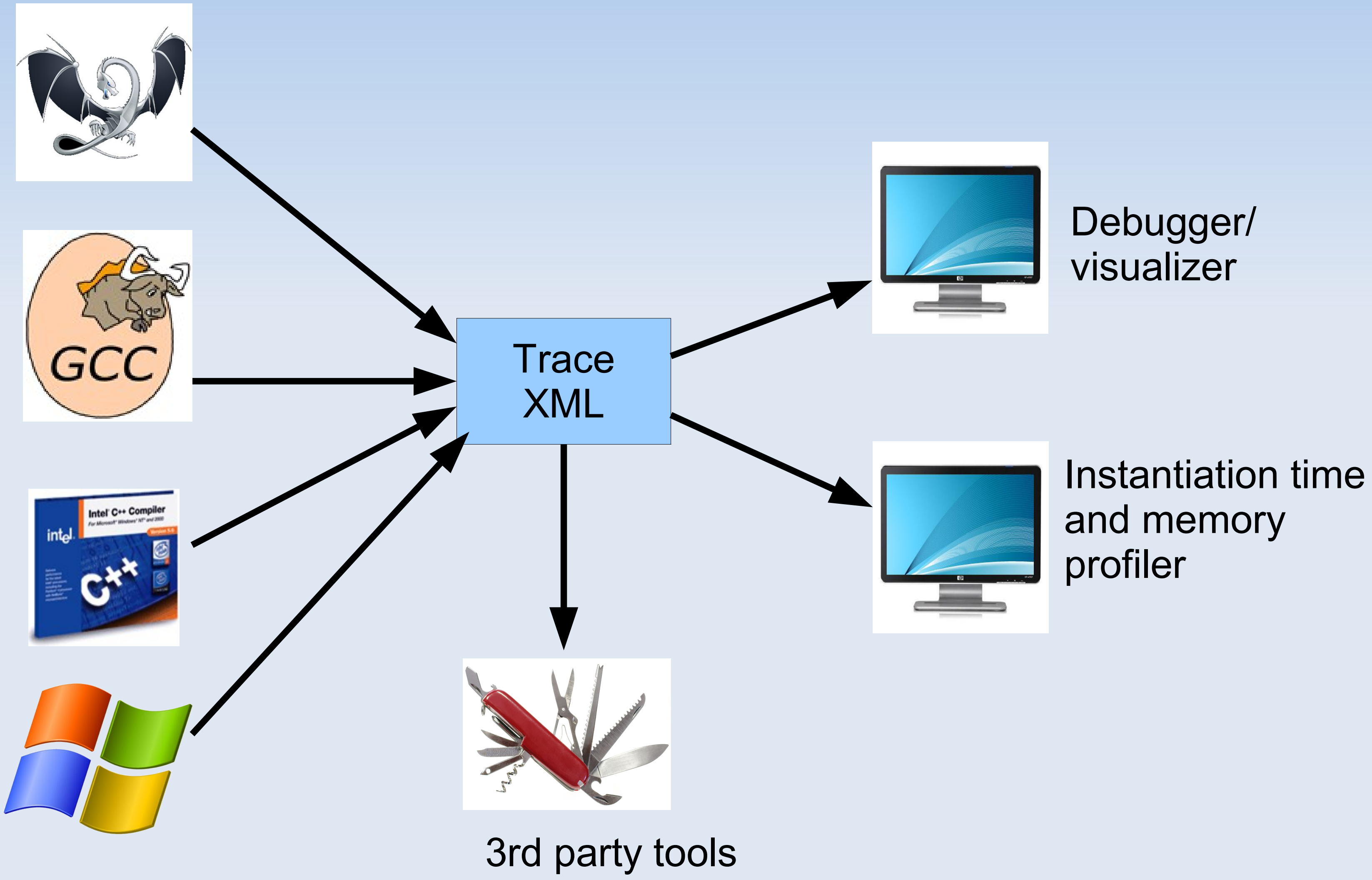
Memory usage

```
$ 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>
```

Distorsion

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

Vision


GCC
Intel C++ Compiler
intel
C++
Trace XML
Debugger/ visualizer
Instantiation time and memory profiler
3rd party tools

Summary

- Tool support for C++ metaprogramming
- Debugger/profiler requires compiler support
- Templight 2.0
- Please use it, give us feadback
- Compilers, will you support us?

Thank you

Debugging and Profiling
C++ Template Metaprograms

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