Boosting MPL with Haskell elements

Ábel Sinkovics

Mpllibs

- Template Metaprogramming libraries
- http://abel.web.elte.hu/mpllibs
  - Metaparse
  - Metamonad
  - Safe printf
  - XL Xpressive

Mpllibs

- Template Metaprogramming libraries
- http://abel.web.elte.hu/mpllibs
  - Metaparse
  - Metamonad
  - Safe printf
  - XL Xpressive

Ábel Sinkovics
Endre Sajó
István Siroki
Zoltán Porkoláb

Mpllibs

- Template Metaprogramming libraries
- http://abel.web.elte.hu/mpllibs
  - Metaparse
  - Metamonad
  - Safe printf
  - XL Xpressive

Ábel Sinkovics
Endre Sajó
István Siroki
Zoltán Porkoláb

Agenda

- Laziness
- Basic building blocks
- Let/Lamba/Case expressions
- Error handling

Fact

```
template <int N> struct fact
```

```
fact n =
```

Fact

```
template <int N> struct fact
{ enum { value = N * fact<N-1>::value }; };
```

```
fact n = n * fact (n – 1)
```

Fact

```
template <int N> struct fact
{ enum { value = N * fact<N-1>::value }; };

template <> struct fact<0> { enum { value = 1 }; };
```

```
fact n = n * fact (n – 1)
fact 0 = 1
```

Fact

```
tem          N> struct
             N * fac             }; };

templat      <0> { enum { value = 1 }; };
```

```
act          – 1)
fact
```

reverse

partition

unique

list

insert

map

min

if

erase

lambda

sort

count

transform

find

iterators

max

pair

vector

fold

string

Fact

```
te          N> struct
              N * fac                    };

templat                <0> { enum {   lu  = 1 } };

                act                        - 1)
                fact
```

reverse

partition

unique

list

insert

map

min

if

case

lambda

sort

count

transform

find

iterators

max

pair

vector

fold

string

Boost.MPL

Boost.MPL

Boost.MPL

- Containers
- Iterators
- Algorithms
- Numeric data types
- Basic operations
- Lambda expressions

Boost.MPL

Boost.MPL

- Containers
- Iterators
- Algorithms
- Numeric data types
- Basic operations
- Lambda expressions

Template metaprogramming and the functional paradigm

- Values can not be changed
- Memoization
- Purity
- Higher-order metafunctions
- ...

Boost.MPL

Boost.MPL

- Containers
- Iterators
- Algorithms
- Numeric data types
- Basic operations
- Lambda expressions

- Currying
- Let expressions
- Algebraic data types
- Pattern matching
- Case expressions
- List comprehension

Boost.MPL

Boost.MPL

- Containers
- Iterators
- Algorithms
- Numeric data types
- Basic operations
- Lambda expressions

Metamonad

- Currying
- Let expressions
- Algebraic data types
- Pattern matching
- Case expressions
- List comprehension

```
template <class A, class B>
struct foo : bar<A, B, A> {};
```

Template metafunction

```
// This is a template metafunction
template <class A, class B>
struct foo : bar<A, B, A> {};
```

Template metafunction

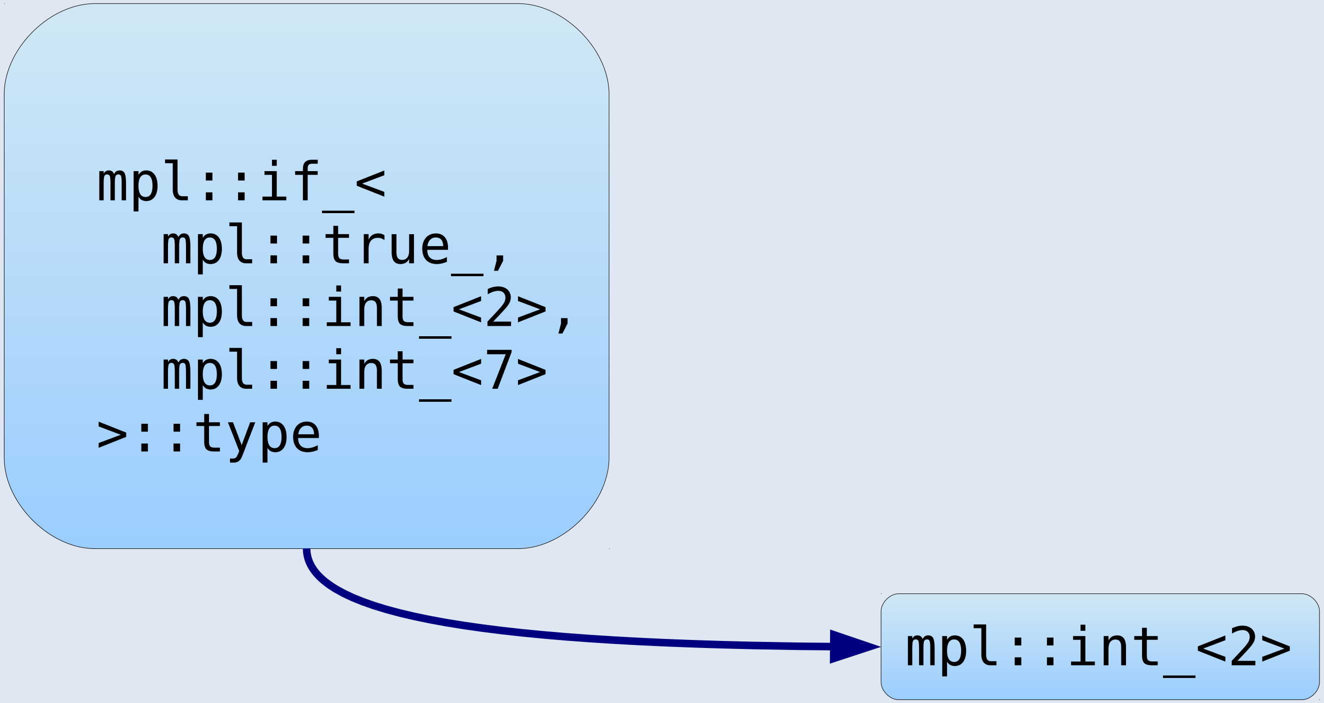
```
// This is a template metafunction
template <class A, class B>
struct foo : bar<A, B, A> {};
```

```
MPLLIBS_METAFUNCTION(foo, (A)(B))
((
  bar<A, B, A>
));
```

Times

```
mpl::if_<
  mpl::true_,
  mpl::int_<2>,
  mpl::int_<7>
>::type
```

Times



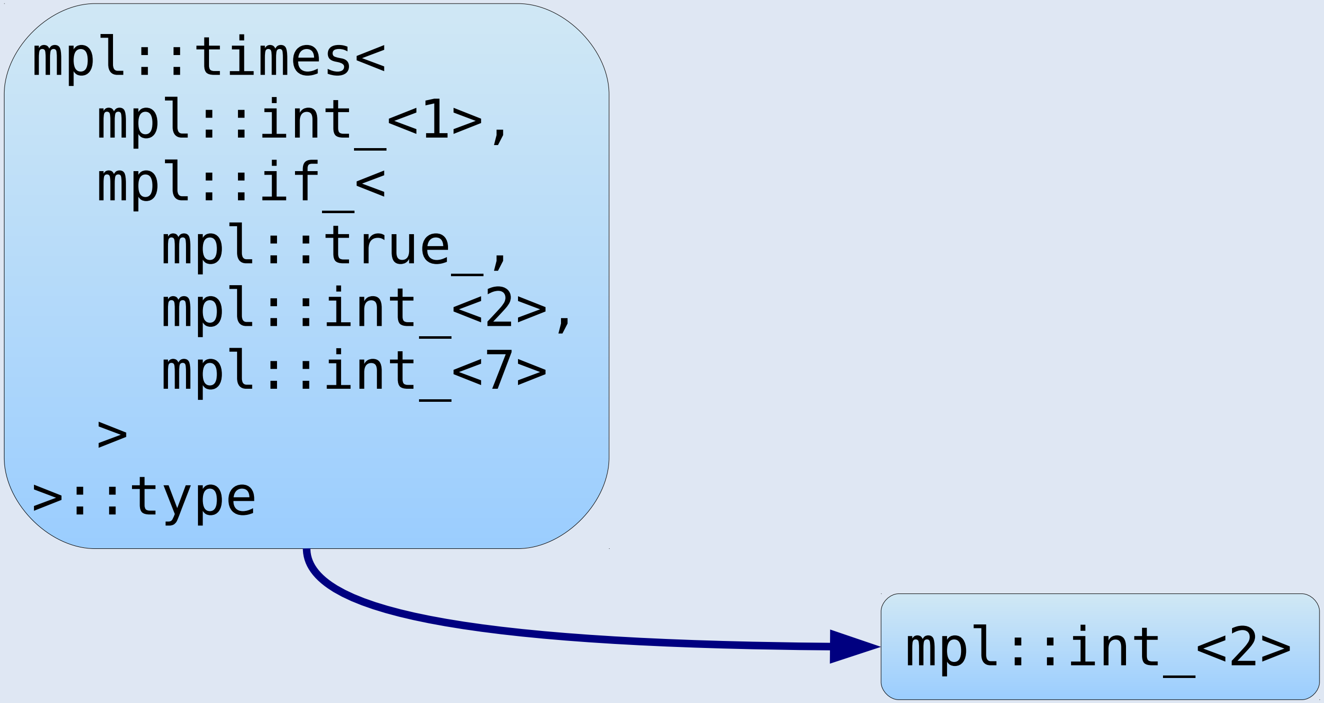
```
mpl::if_<
  mpl::true_,
  mpl::int_<2>,
  mpl::int_<7>
>::type
```

```
mpl::int_<2>
```

Times

```
mpl::times<
  mpl::int_<1>,
  mpl::if_<
    mpl::true_,
    mpl::int_<2>,
    mpl::int_<7>
  >
>::type
```

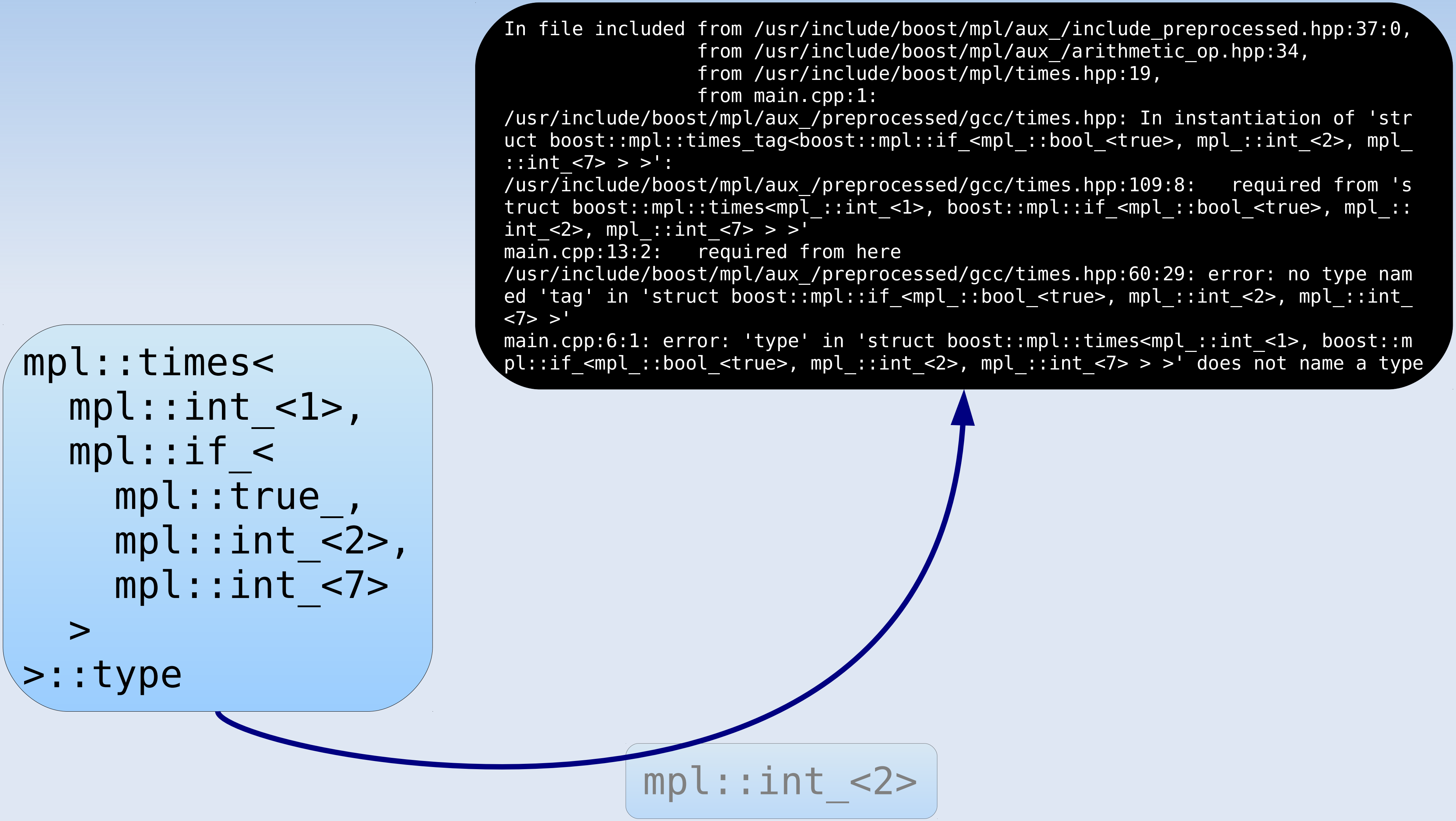
Times



```
mpl::times<
  mpl::int_<1>,
  mpl::if_<
    mpl::true_,
    mpl::int_<2>,
    mpl::int_<7>
  >
>::type
```

```
mpl::int_<2>
```

Times

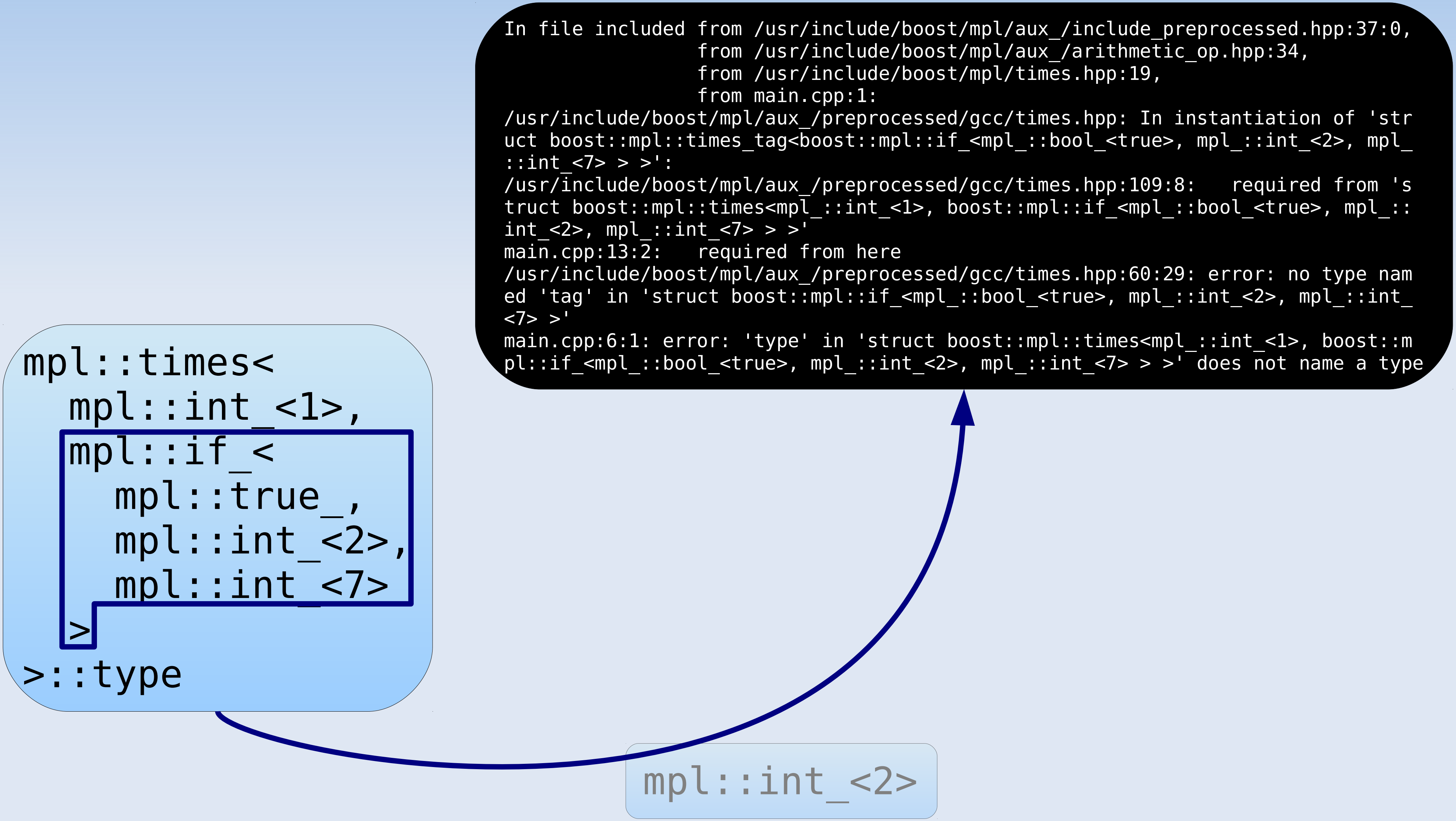


```
In file included from /usr/include/boost/mpl/aux_/include_preprocessed.hpp:37:0,
                 from /usr/include/boost/mpl/aux_/arithmetic_op.hpp:34,
                 from /usr/include/boost/mpl/times.hpp:19,
                 from main.cpp:1:
/usr/include/boost/mpl/aux_/preprocessed/gcc/times.hpp: In instantiation of 'str
uct boost::mpl::times_tag<boost::mpl::if_<mpl_::bool_<true>, mpl_::int_<2>, mpl_
::int_<7> > >':
/usr/include/boost/mpl/aux_/preprocessed/gcc/times.hpp:109:8:   required from 's
truct boost::mpl::times<mpl_::int_<1>, boost::mpl::if_<mpl_::bool_<true>, mpl_::
int_<2>, mpl_::int_<7> > >'
main.cpp:13:2:   required from here
/usr/include/boost/mpl/aux_/preprocessed/gcc/times.hpp:60:29: error: no type nam
ed 'tag' in 'struct boost::mpl::if_<mpl_::bool_<true>, mpl_::int_<2>, mpl_::int_
<7> >'
main.cpp:6:1: error: 'type' in 'struct boost::mpl::times<mpl_::int_<1>, boost::m
pl::if_<mpl_::bool_<true>, mpl_::int_<2>, mpl_::int_<7> > >' does not name a type
```

```
mpl::times<
  mpl::int_<1>,
  mpl::if_<
    mpl::true_,
    mpl::int_<2>,
    mpl::int_<7>
  >
>::type
```

```
mpl::int_<2>
```

Times

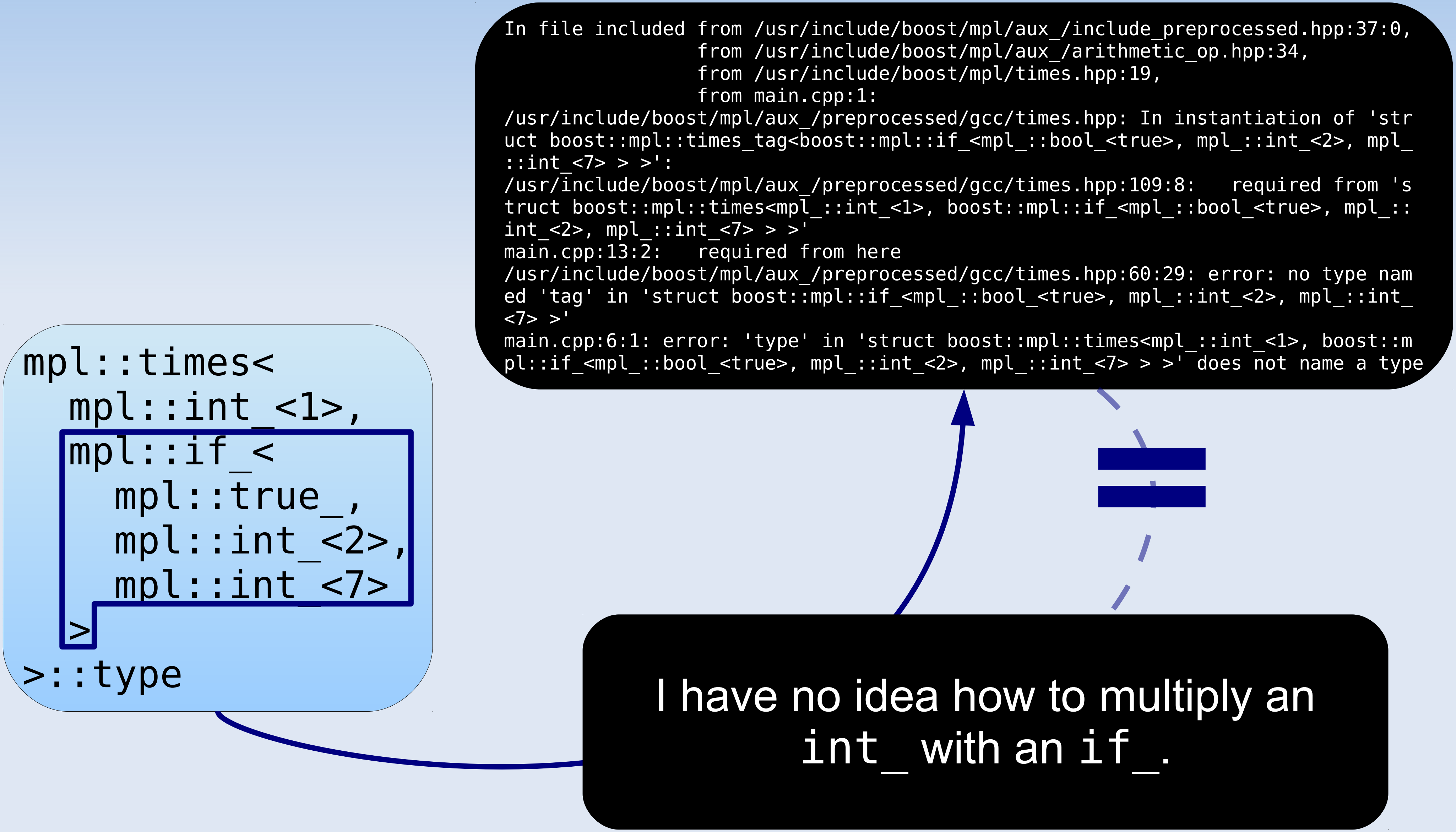


```
In file included from /usr/include/boost/mpl/aux_/include_preprocessed.hpp:37:0,
                 from /usr/include/boost/mpl/aux_/arithmetic_op.hpp:34,
                 from /usr/include/boost/mpl/times.hpp:19,
                 from main.cpp:1:
/usr/include/boost/mpl/aux_/preprocessed/gcc/times.hpp: In instantiation of 'str
uct boost::mpl::times_tag<boost::mpl::if_<mpl_::bool_<true>, mpl_::int_<2>, mpl_
::int_<7> > >':
/usr/include/boost/mpl/aux_/preprocessed/gcc/times.hpp:109:8:   required from 's
truct boost::mpl::times<mpl_::int_<1>, boost::mpl::if_<mpl_::bool_<true>, mpl_::
int_<2>, mpl_::int_<7> > >'
main.cpp:13:2:   required from here
/usr/include/boost/mpl/aux_/preprocessed/gcc/times.hpp:60:29: error: no type nam
ed 'tag' in 'struct boost::mpl::if_<mpl_::bool_<true>, mpl_::int_<2>, mpl_::int_
<7> >'
main.cpp:6:1: error: 'type' in 'struct boost::mpl::times<mpl_::int_<1>, boost::m
pl::if_<mpl_::bool_<true>, mpl_::int_<2>, mpl_::int_<7> > >' does not name a type
```

```
mpl::times<
  mpl::int_<1>,
  mpl::if_<
    mpl::true_,
    mpl::int_<2>,
    mpl::int_<7>
  >
>::type
```

```
mpl::int_<2>
```

Times

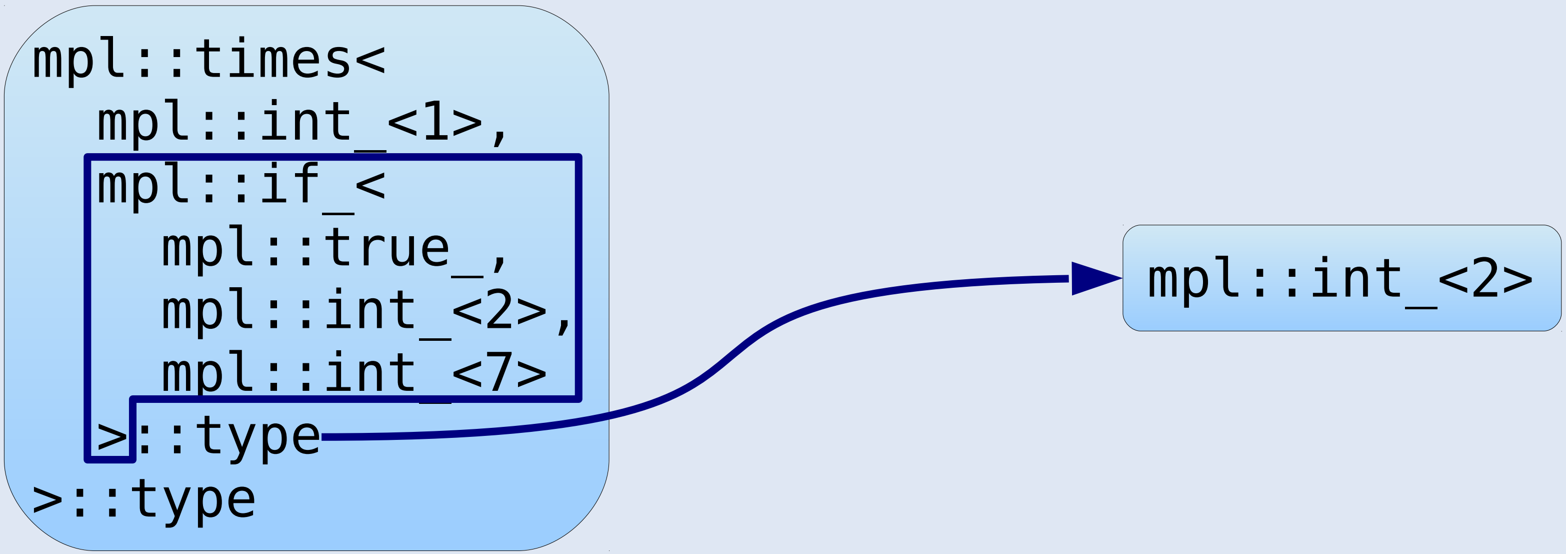


```
mpl::times<
  mpl::int_<1>,
  mpl::if_<
    mpl::true_,
    mpl::int_<2>,
    mpl::int_<7>
  >
>::type
```

```
In file included from /usr/include/boost/mpl/aux_/include_preprocessed.hpp:37:0,
                 from /usr/include/boost/mpl/aux_/arithmetic_op.hpp:34,
                 from /usr/include/boost/mpl/times.hpp:19,
                 from main.cpp:1:
/usr/include/boost/mpl/aux_/preprocessed/gcc/times.hpp: In instantiation of 'str
uct boost::mpl::times_tag<boost::mpl::if_<mpl_::bool_<true>, mpl_::int_<2>, mpl_
::int_<7> > >':
/usr/include/boost/mpl/aux_/preprocessed/gcc/times.hpp:109:8:   required from 's
truct boost::mpl::times<mpl_::int_<1>, boost::mpl::if_<mpl_::bool_<true>, mpl_::
int_<2>, mpl_::int_<7> > >'
main.cpp:13:2:   required from here
/usr/include/boost/mpl/aux_/preprocessed/gcc/times.hpp:60:29: error: no type nam
ed 'tag' in 'struct boost::mpl::if_<mpl_::bool_<true>, mpl_::int_<2>, mpl_::int_
<7> >'
main.cpp:6:1: error: 'type' in 'struct boost::mpl::times<mpl_::int_<1>, boost::m
pl::if_<mpl_::bool_<true>, mpl_::int_<2>, mpl_::int_<7> > >' does not name a type
```

I have no idea how to multiply an `int_` with an `if_`.

Times

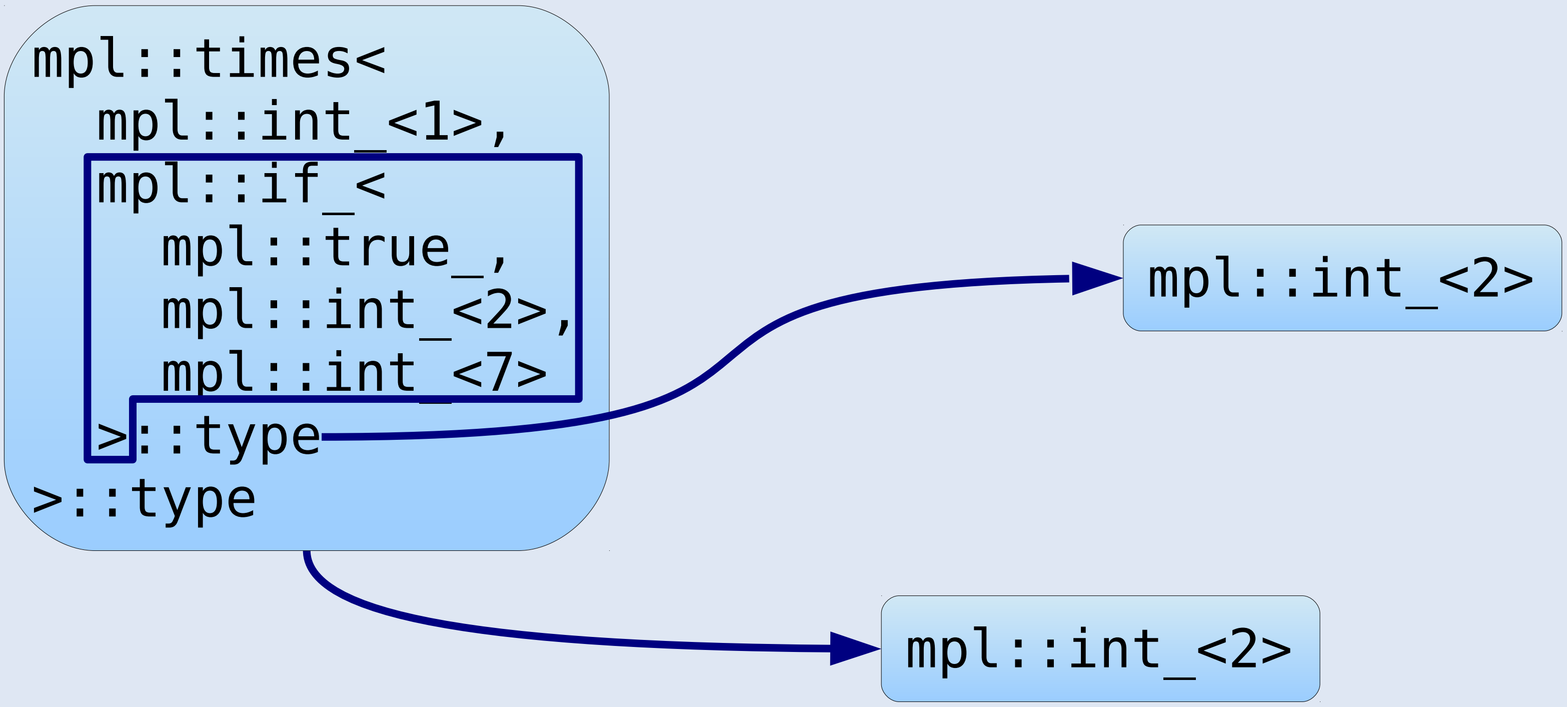


```
mpl::times<
  mpl::int_<1>,
  mpl::if_<
    mpl::true_,
    mpl::int_<2>,
    mpl::int_<7>
  >::type
>::type
```

```
mpl::int_<2>
```

```
mpl::int_<2>
```

Times

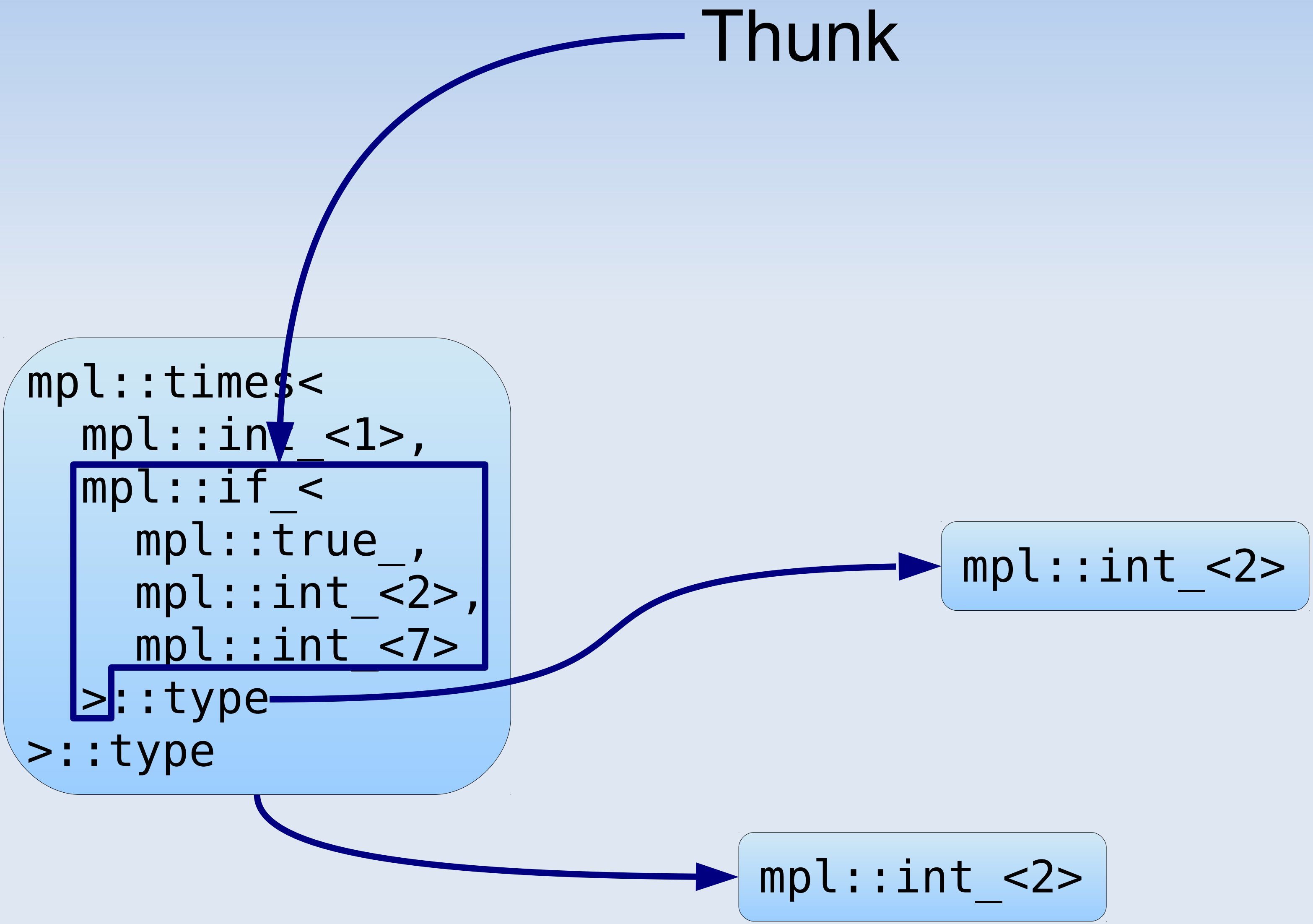


```
mpl::times<
  mpl::int_<1>,
  mpl::if_<
    mpl::true_,
    mpl::int_<2>,
    mpl::int_<7>
  >::type
>::type
```

```
mpl::int_<2>
```

```
mpl::int_<2>
```

Times



Thunk

```
mpl::times<
  mpl::int_<1>,
  mpl::if_<
    mpl::true_,
    mpl::int_<2>,
    mpl::int_<7>
  >::type
>::type
```

```
mpl::int_<2>
```

```
mpl::int_<2>
```

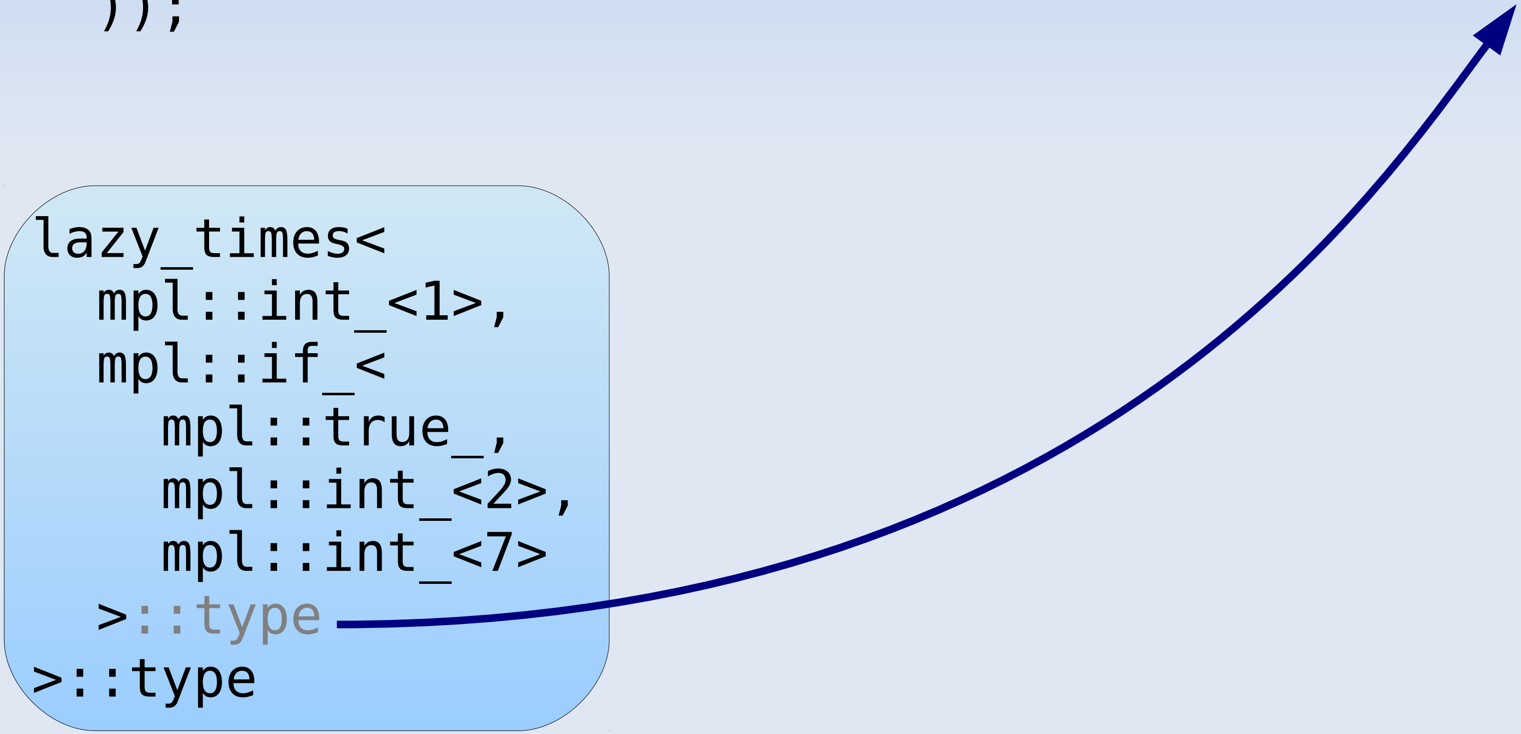
Times

```
MPLLIBS_METAFUNCTION(lazy_times, (A)(B))
((
                      A                 B
));
```

```
lazy_times<
  mpl::int_<1>,
  mpl::if_<
    mpl::true_,
    mpl::int_<2>,
    mpl::int_<7>
  >::type
>::type
```

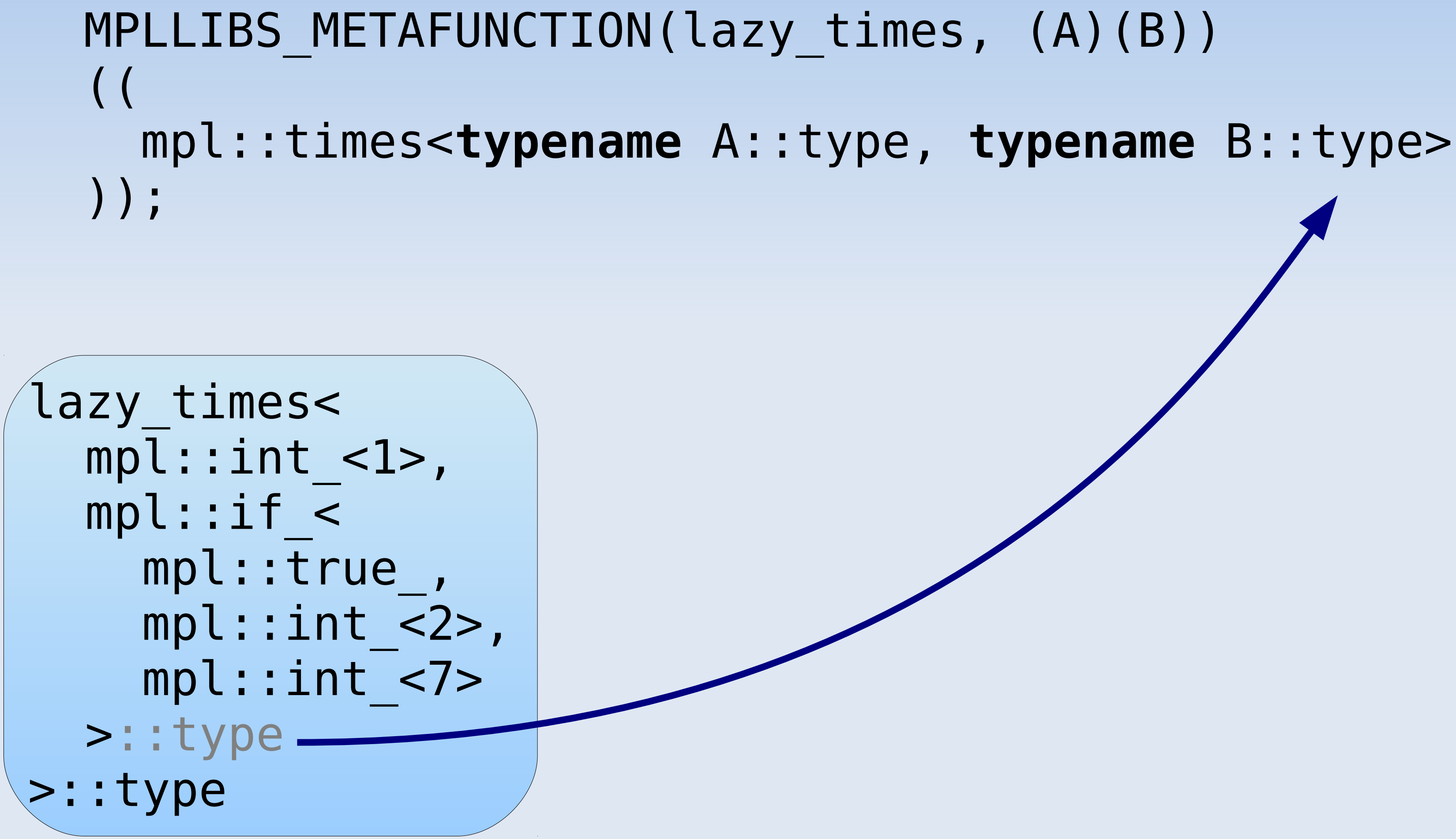
Times

```
MPLLIBS_METAFUNCTION(lazy_times, (A)(B))
((
             typename A::type  typename B::type
));
```



```
lazy_times<
  mpl::int_<1>,
  mpl::if_<
    mpl::true_,
    mpl::int_<2>,
    mpl::int_<7>
  >::type
>::type
```

Times

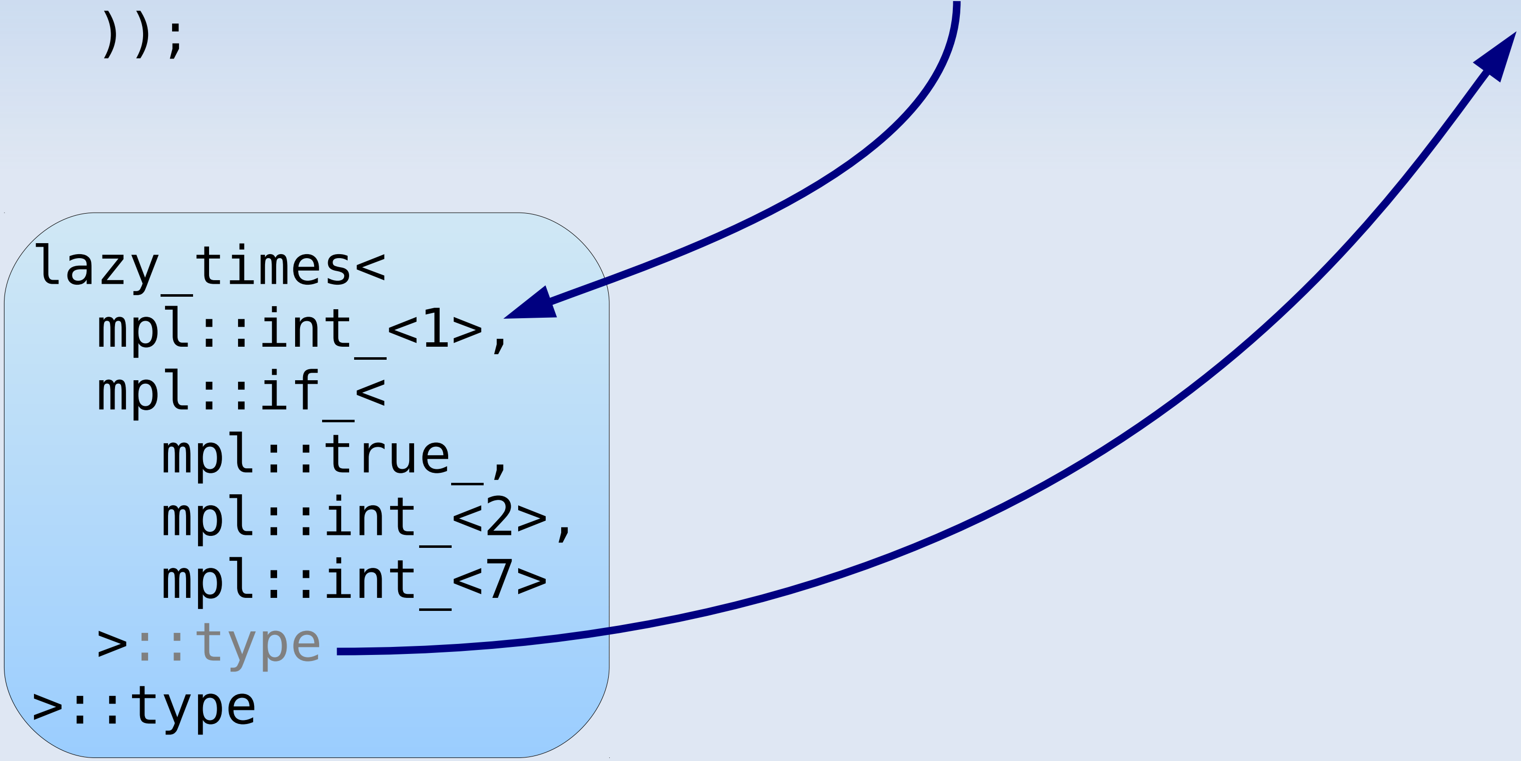


```
MPLLIBS_METAFUNCTION(lazy_times, (A)(B))
((
  mpl::times<typename A::type, typename B::type>
));
```

```
lazy_times<
  mpl::int_<1>,
  mpl::if_<
    mpl::true_,
    mpl::int_<2>,
    mpl::int_<7>
  >::type
>::type
```

Times

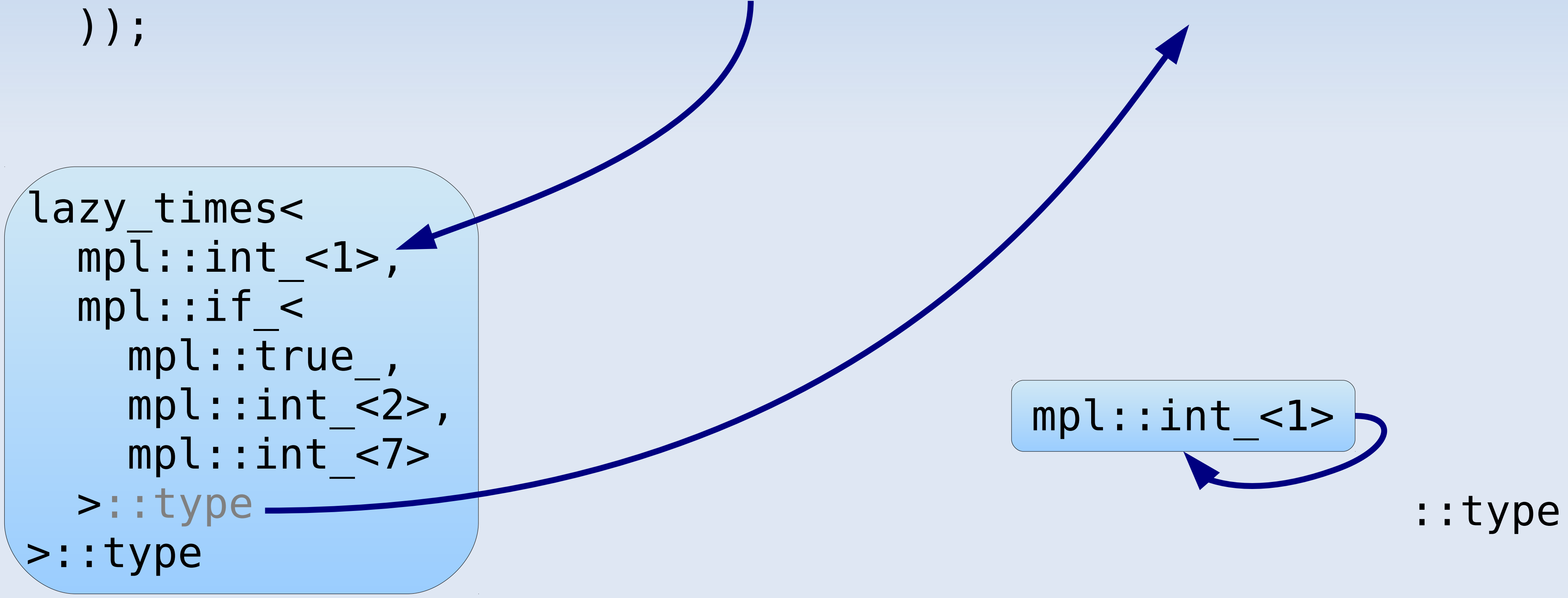
```
MPLLIBS_METAFUNCTION(lazy_times, (A)(B))
((
  mpl::times<typename A::type, typename B::type>
));
```



```
lazy_times<
  mpl::int_<1>,
  mpl::if_<
    mpl::true_,
    mpl::int_<2>,
    mpl::int_<7>
  >::type
>::type
```

Times

```
MPLLIBS_METAFUNCTION(lazy_times, (A)(B))
((
  mpl::times<typename A::type, typename B::type>
));
```

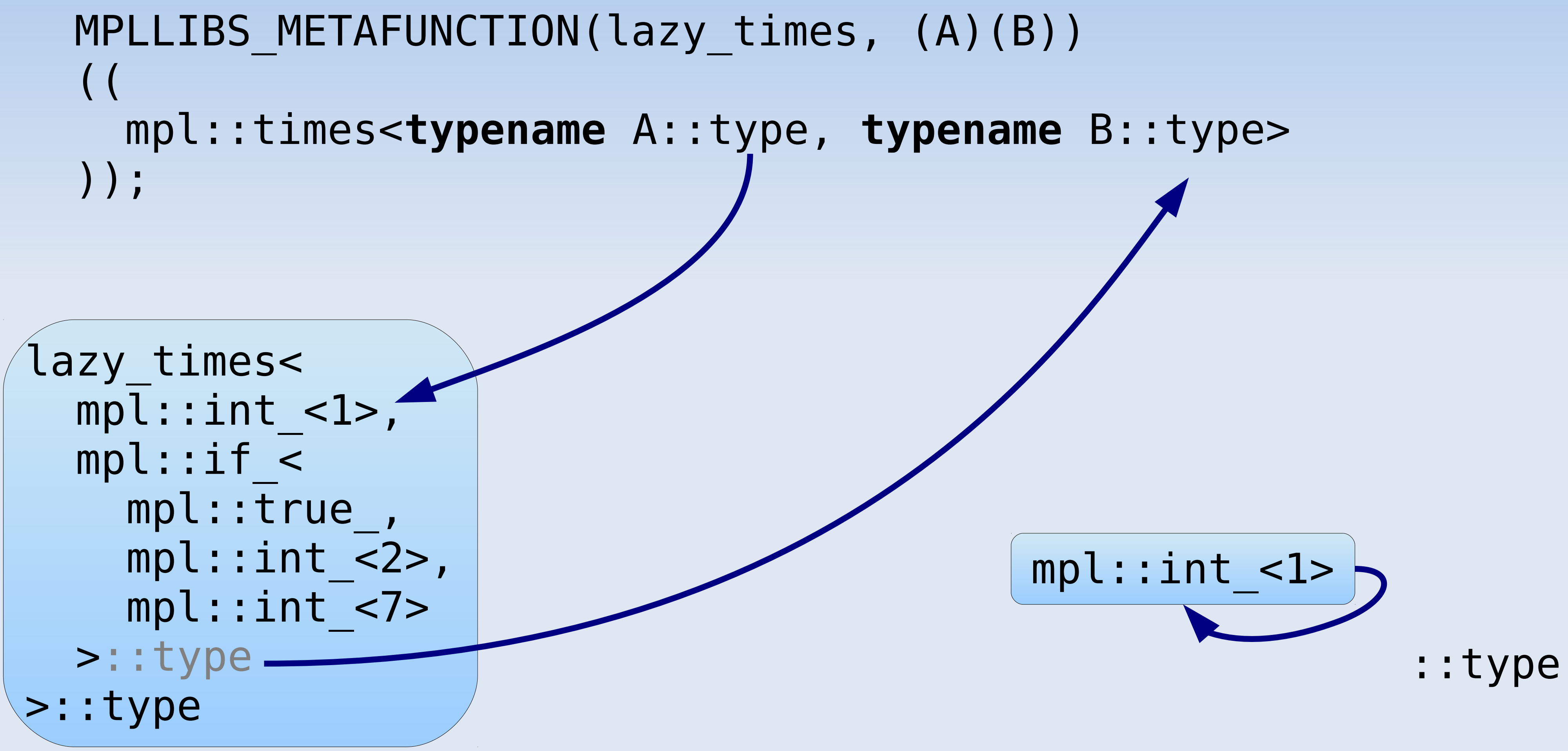


```
lazy_times<
  mpl::int_<1>,
  mpl::if_<
    mpl::true_,
    mpl::int_<2>,
    mpl::int_<7>
  >::type
>::type
```

```
mpl::int_<1>
```

```
::type
```

Times



```
MPLLIBS_METAFUNCTION(lazy_times, (A)(B))
((
  mpl::times<typename A::type, typename B::type>
));
```

```
lazy_times<
  mpl::int_<1>,
  mpl::if_<
    mpl::true_,
    mpl::int_<2>,
    mpl::int_<7>
  >::type
>::type
```

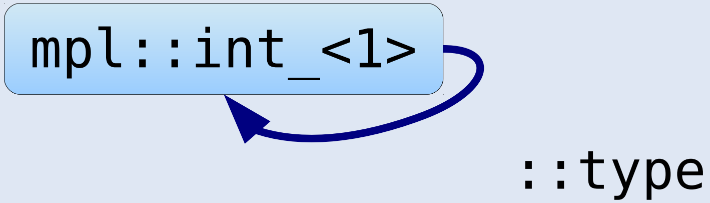
```
mpl::int_<1>
```

```
::type
```

Template metaprogramming value

Times

- Assumption: every class used as a value in a template metaprogram is a template metaprogramming value



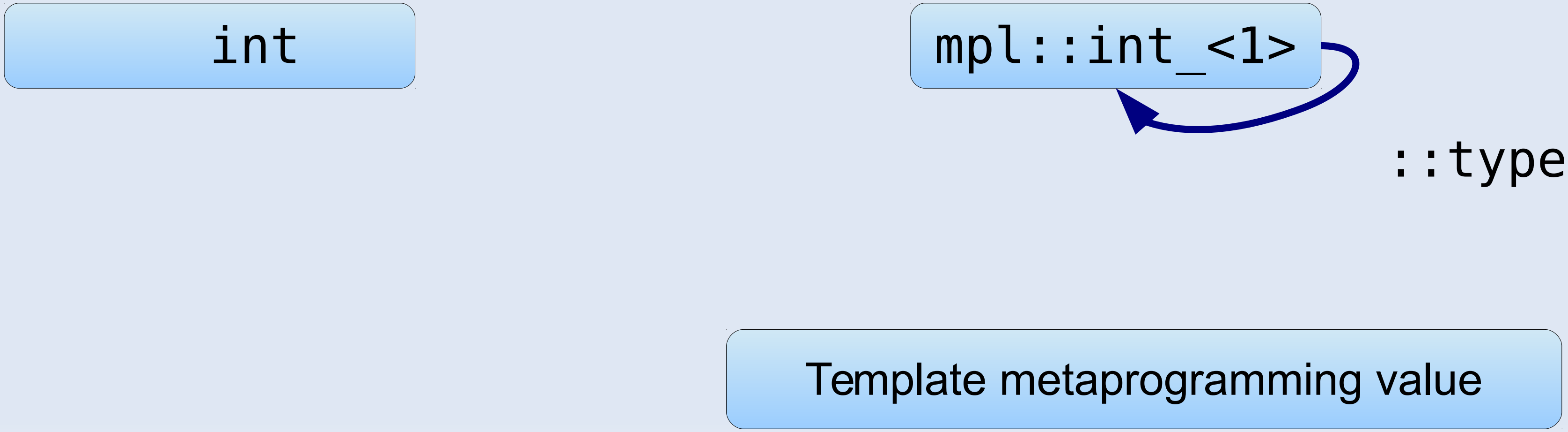
mpl::int_<1>

::type

Template metaprogramming value

Times

- Assumption: every class used as a value in a template metaprogram is a template metaprogramming value



int

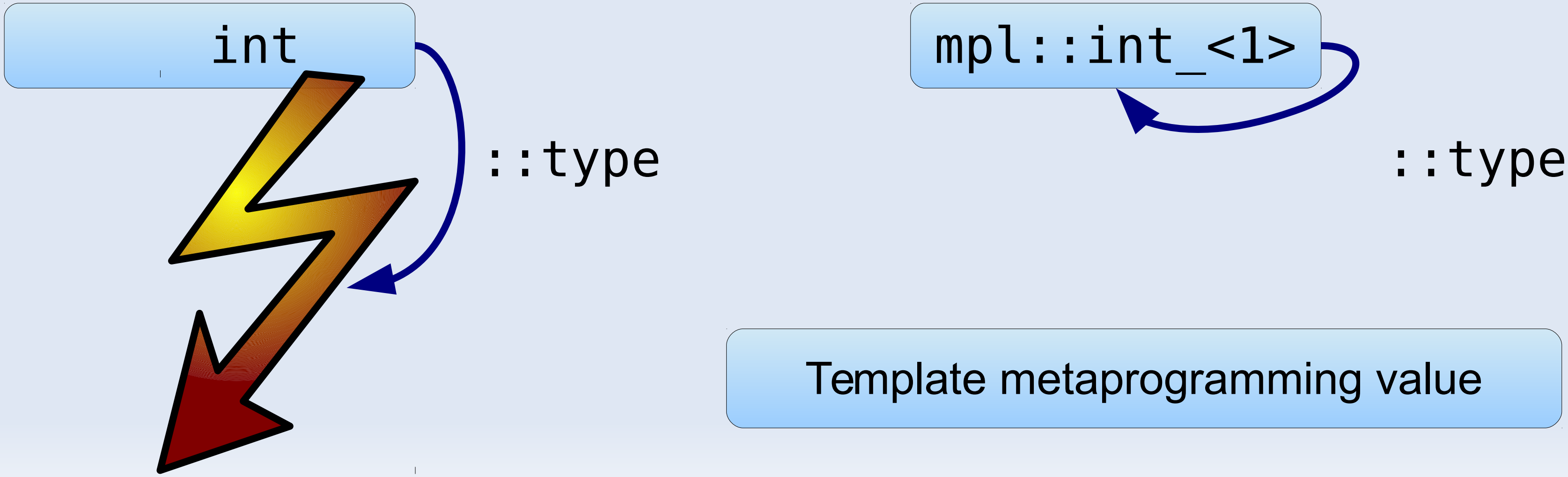
mpl::int_<1>

::type

Template metaprogramming value

Times

- Assumption: every class used as a value in a template metaprogram is a template metaprogramming value



int

::type

mpl::int_<1>

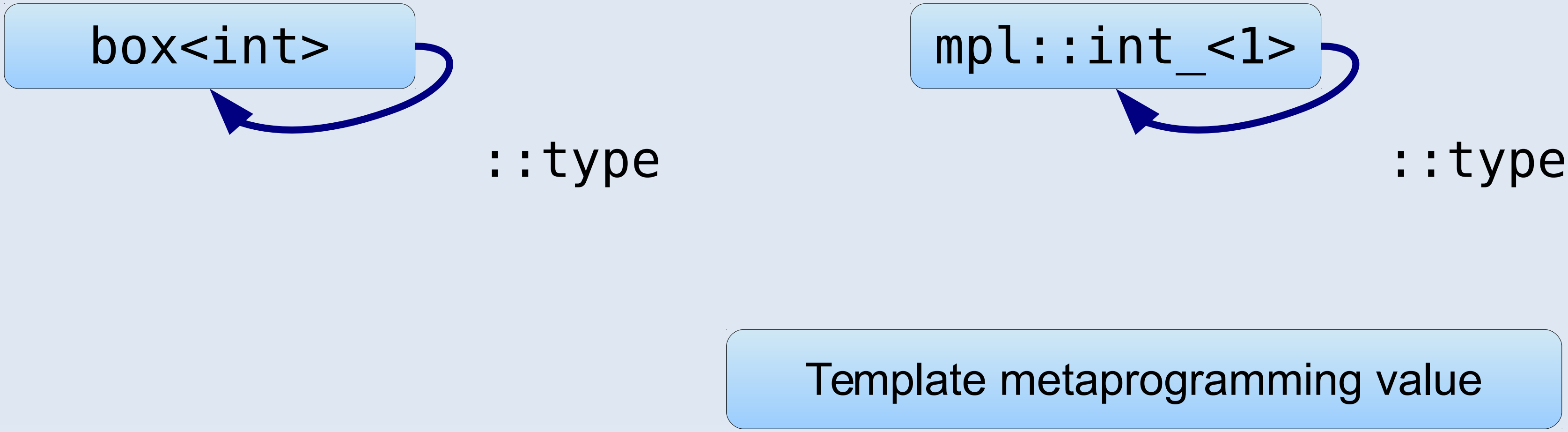
::type

Template metaprogramming value

Times

- Assumption: every class used as a value in a template metaprogram is a template metaprogramming value

```
template <class T>
struct box {
  typedef box type;
};
```



Fact

```
MPLLIBS_METAFUNCTION(fact, (N))
((
));
```

```
int fact(int N)
{
  return 0 == N ? 1 : N * fact(N – 1);
}
```

Fact

```
MPLLIBS_METAFUNCTION(fact, (N))
((
  mpl::eval_if<



     ,
                ,









  >
));
```

```
int fact(int N)
{
  return 0 == N ? 1 : N * fact(N - 1);
}
```

Fact

```
MPLLIBS_METAFUNCTION(fact, (N))
((
  mpl::eval_if<
    mpl::equal_to<
      mpl::int_<0>,
      N
    >,
                ,





  >
));
```

```
int fact(int N)
{
  return 0 == N ? 1 : N * fact(N – 1);
}
```

Fact

```
MPLLIBS_METAFUNCTION(fact, (N))
((
  mpl::eval_if<
    mpl::equal_to<
      mpl::int_<0>,
      N
    >,
    mpl::int_<1>,
  >
));
```

```
int fact(int N)
{
  return 0 == N ? 1 : N * fact(N – 1);
}
```

Fact

```
MPLLIBS_METAFUNCTION(fact, (N))
((
  mpl::eval_if<
    mpl::equal_to<
      mpl::int_<0>,
      N
    >,
    mpl::int_<1>,
    mpl::times<





       ,
      N
    >
  >
));
```

```
int fact(int N)
{
  return 0 == N ? 1 : N * fact(N - 1);
}
```

Fact

```
MPLLIBS_METAFUNCTION(fact, (N))
((
  mpl::eval_if<
    mpl::equal_to<
      mpl::int_<0>,
      N
    >,
    mpl::int_<1>,
    mpl::times<
      fact<





      >,
      N
    >
  >
));
```

```
int fact(int N)
{
  return 0 == N ? 1 : N * fact(N – 1);
}
```

Fact

```
MPLLIBS_METAFUNCTION(fact, (N))
((
  mpl::eval_if<
    mpl::equal_to<
      mpl::int_<0>,
      N
    >,
    mpl::int_<1>,
    mpl::times<
      fact<
        mpl::minus<
          N,
          mpl::int_<1>
        >
      >,
      N
    >
  >
));
```

```
int fact(int N)
{
  return 0 == N ? 1 : N * fact(N – 1);
}
```

Fact

```
MPLLIBS_METAFUNCTION(fact, (N))
((
  mpl::eval_if<
    typename mpl::equal_to<
      mpl::int_<0>,
      N
    >::type,
    mpl::int_<1>,
    mpl::times<
      typename fact<
        typename mpl::minus<
          N,
          mpl::int_<1>
        >::type
      >::type,
      N
    >
  >
));
```

```
int fact(int N)
{
  return 0 == N ? 1 : N * fact(N – 1);
}
```

Fact

```
MPLLIBS_METAFUNCTION(fact, (N))
((
  mpl::eval_if<
    typename mpl::equal_to<
      mpl::int_<0>,
      N
    >::type,
    mpl::int_<1>,
    mpl::times<
      typename fact<
        typename mpl::minus<
          N,
          mpl::int_<1>
        >::type
      >::type,
      N
    >
  >
));
```

```
fact<mpl::int_<0>>::type
```

Fact

```
MPLLIBS_METAFUNCTION(fact, (N))
((
  mpl::eval_if<
    typename mpl::equal_to<
      mpl::int_<0>,
      mpl::int_<0>
    >::type,
    mpl::int_<1>,
    mpl::times<
      typename fact<
        typename mpl::minus<
          mpl::int_<0>,
          mpl::int_<1>
        >::type
      >::type,
      mpl::int_<0>
    >
  >::type
));
```

```
fact<mpl::int_<0>>::type
```

Fact

```
MPLLIBS_METAFUNCTION(fact, (N))
((
  mpl::eval_if<
    typename mpl::equal_to<
      mpl::int_<0>,
      mpl::int_<0>
    >::type,
    mpl::int_<1>,
    mpl::times<
      typename fact<
        typename mpl::minus<
          mpl::int_<0>,
          mpl::int_<1>
        >::type
      >::type,
      mpl::int_<0>
    >
  >::type
));
```

```
fact<mpl::int_<0>>::type
```

Fact

```
MPLLIBS_METAFUNCTION(fact, (N))
((
  mpl::eval_if<
    mpl::true_,



    mpl::int_<1>,
    mpl::times<
      typename fact<
        typename mpl::minus<
          mpl::int_<0>,
          mpl::int_<1>
        >::type
      >::type,
      mpl::int_<0>
    >
  >::type
));
```

`fact<mpl::int_<0>>::type`

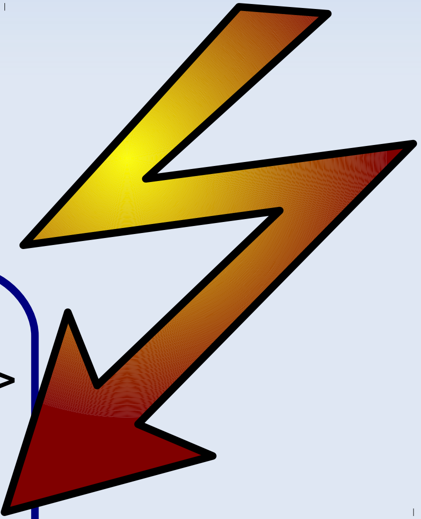
Fact

```
MPLLIBS_METAFUNCTION(fact, (N))
((
  mpl::eval_if<
    mpl::true_,



    mpl::int_<1>,
    mpl::times<
      typename fact<
        mpl::int_<-1>



      >::type,
      mpl::int_<0>
    >
  >::type
));
```

fact<mpl::int_<0>>::type

Fact

```
MPLLIBS_METAFUNCTION(fact, (N))
((
  mpl::eval_if<
    mpl::true_,




    mpl::int_<1>,
    mpl::times<
      typename fact<
        mpl::int_<-1>



      >::type,
      mpl::int_<0>
    >
  >::type
));
```



```
fact<mpl::int_<0>>::type
```

Fact

```
MPLLIBS_METAFUNCTION(fact, (N))
((
  lazy_eval_if<
    lazy_equal_to<
      mpl::int_<0>,
      N
    >,
    mpl::int_<1>,
    lazy_times<
      fact<
        lazy_minus<
          N,
          mpl::int_<1>
        >
      >,
      N
    >
  >
));
```

```
fact<mpl::int_<0>>::type
```

Fact

```
MPLLIBS_METAFUNCTION(fact, (N))
((
  lazy_eval_if<
    lazy_equal_to<
      mpl::int_<0>,
      mpl::int_<0>
    >,
    mpl::int_<1>,
    lazy_times<
      fact<
        lazy_minus<
          mpl::int_<0>,
          mpl::int_<1>
        >
      >,
      mpl::int_<0>
    >
  >::type
));
```

```
fact<mpl::int_<0>>::type
```

Fact

```
MPLLIBS_METAFUNCTION(fact, (N))
((
  lazy_eval_if<
    lazy_equal_to<
      mpl::int_<0>,
      mpl::int_<0>
    >,
    mpl::int_<1>,
    lazy_times<
      fact<
        lazy_m
          mpl:
          mpl:
        >
      >,
      mpl::int_<0>
    >
  >::type
));
```

```
MPLLIBS_METAFUNCTION(lazy_eval_if, (C)(T)(F))
((
  mpl::eval_if<typename C::type, T, F>
));
```

```
fact<mpl::int_<0>>::type
```

Fact

```
MPLLIBS_METAFUNCTION(fact, (N))
((
  lazy_eval_if<
    lazy_equal_to<
      mpl::int_<0>,
      mpl::int_<0>
    >,
    mpl::int_<1>,
    lazy_times<
      fact<
        lazy_m
          mpl:
          mpl:
        >
      >,
      mpl::int_<0>
    >
  >::type
));
```

```
MPLLIBS_METAFUNCTION(lazy_eval_if, (C)(T)(F))
((
  mpl::eval_if<typename C::type, T, F>
));
```

```
fact<mpl::int_<0>>::type
```

Fact

```
MPLLIBS_METAFUNCTION(fact, (N))
((
  mpl::eval_if<
    mpl::true_,



    mpl::int_<1>,
    lazy_times<
      fact<
        lazy_minus<
          mpl::int_<0>,
          mpl::int_<1>
        >
      >,
      mpl::int_<0>
    >
  >::type
));
```

```
fact<mpl::int_<0>>::type
```

Fact

```
MPLLIBS_METAFUNCTION(fact, (N))
((

    mpl::int_<1>

));
```

```
fact<mpl::int_<0>>::type
```

Fact

```
MPL
((
```

```
template <class N>
struct fact_impl;

MPLLIBS_METAFUNCTION(fact, (N))
((
  mpl::eval_if<
    typename mpl::equal_to<mpl::int_<0>, typename N::type>::type,
    mpl::int_<1>,
    fact_impl<N>
  >
));

MPLLIBS_METAFUNCTION(fact_impl, (N))
((
  mpl::times<
    typename fact<mpl::minus<typename N::type, mpl::int_<1>>>::type,
    typename N::type
  >
));
```

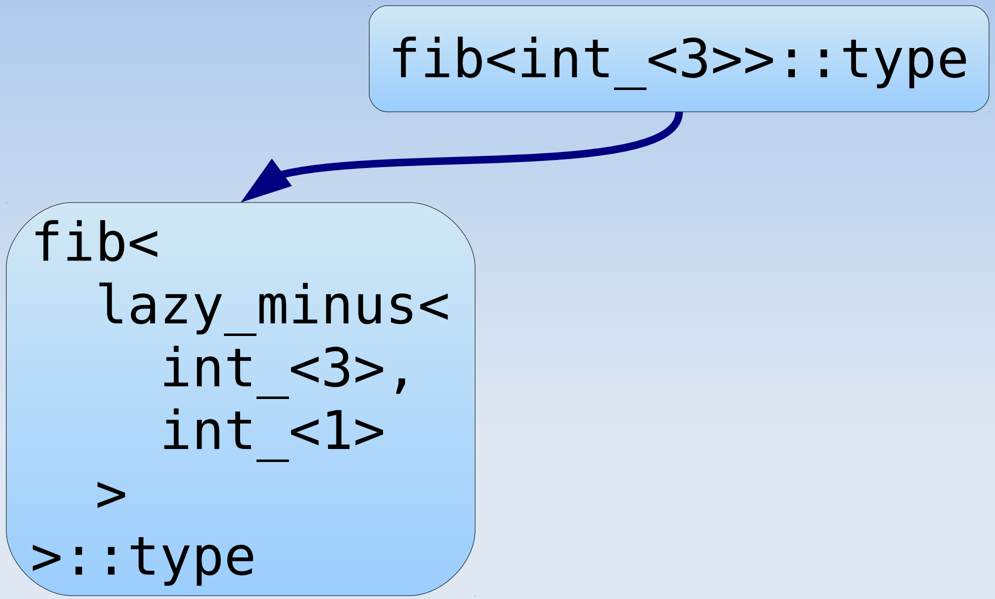
```
fact<mpl::int_<0>>::type
```

```
));
```

The price of laziness

```
fib<int_<3>>::type
```

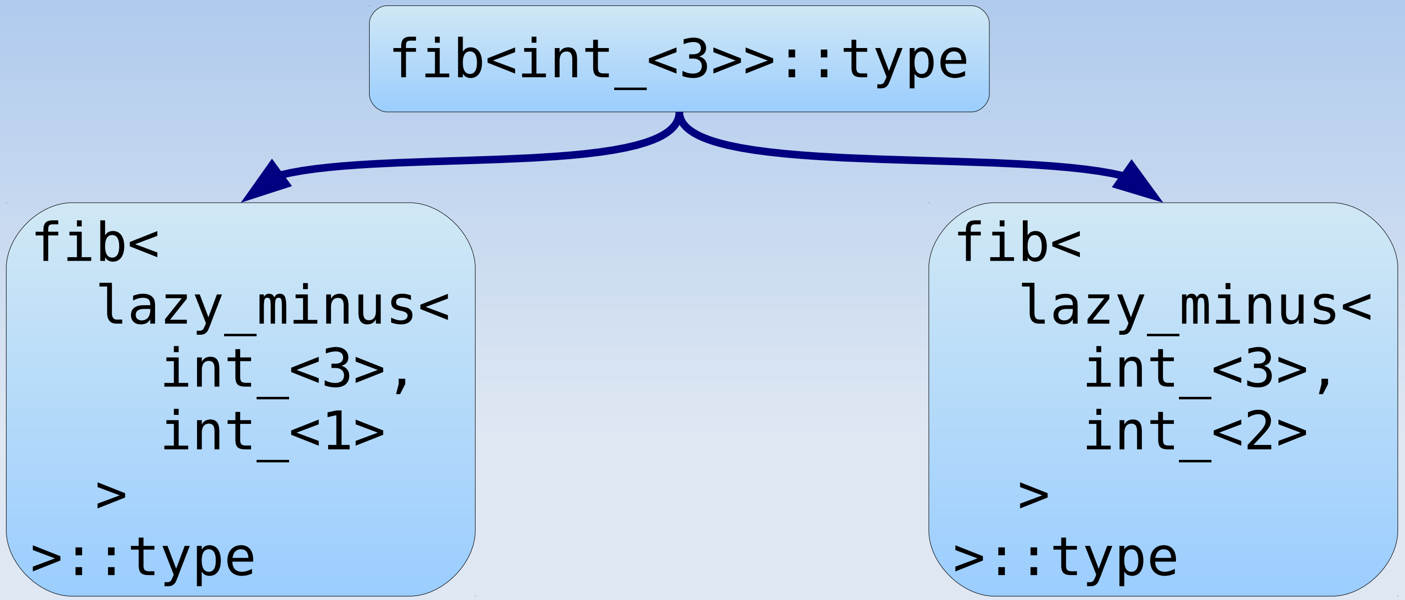
The price of laziness



```
fib<int_<3>>::type
```

```
fib<
  lazy_minus<
    int_<3>,
    int_<1>
  >
>::type
```

The price of laziness

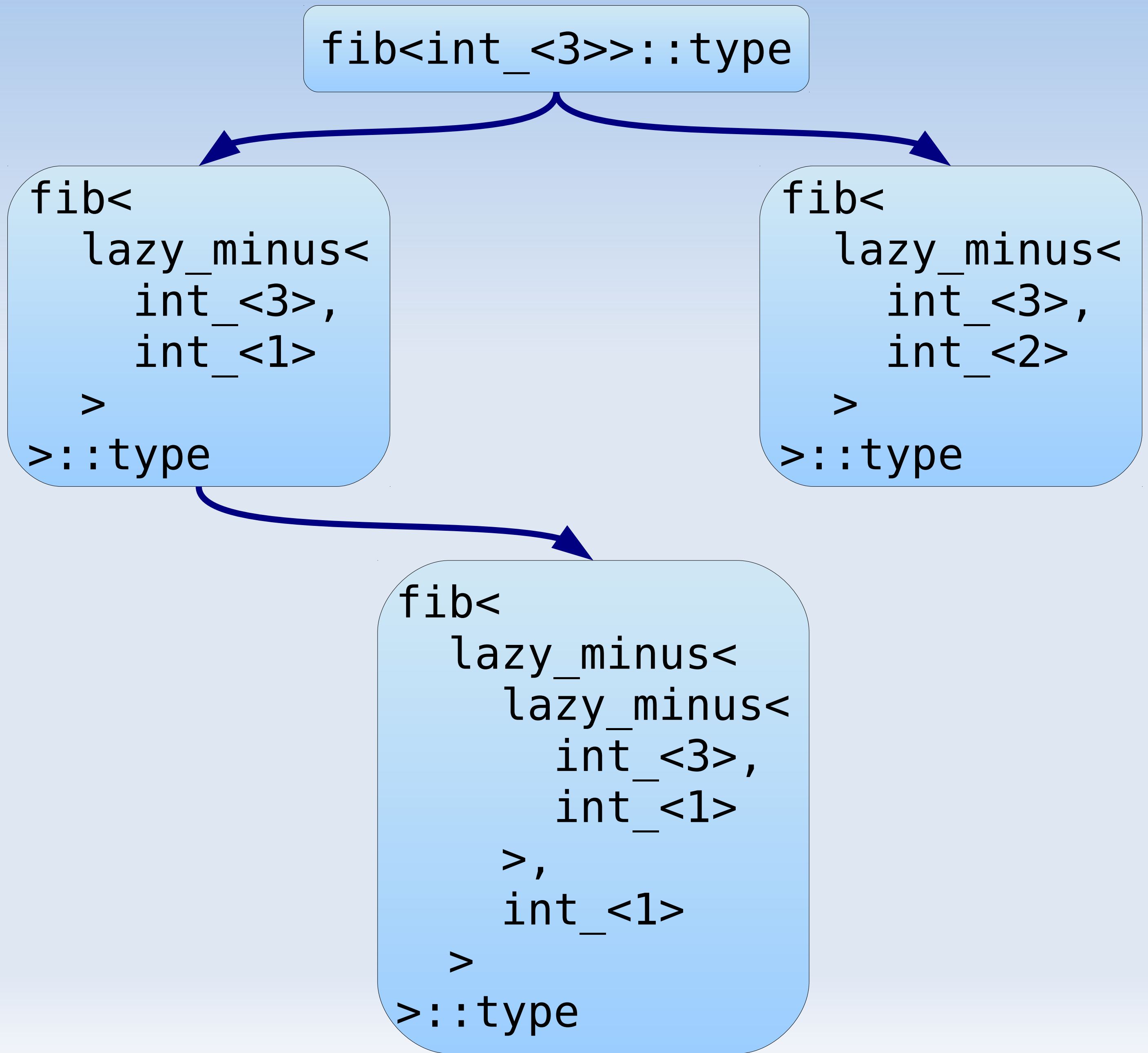


```
fib<int_<3>>::type
```

```
fib<
  lazy_minus<
    int_<3>,
    int_<1>
  >
>::type
```

```
fib<
  lazy_minus<
    int_<3>,
    int_<2>
  >
>::type
```

The price of laziness



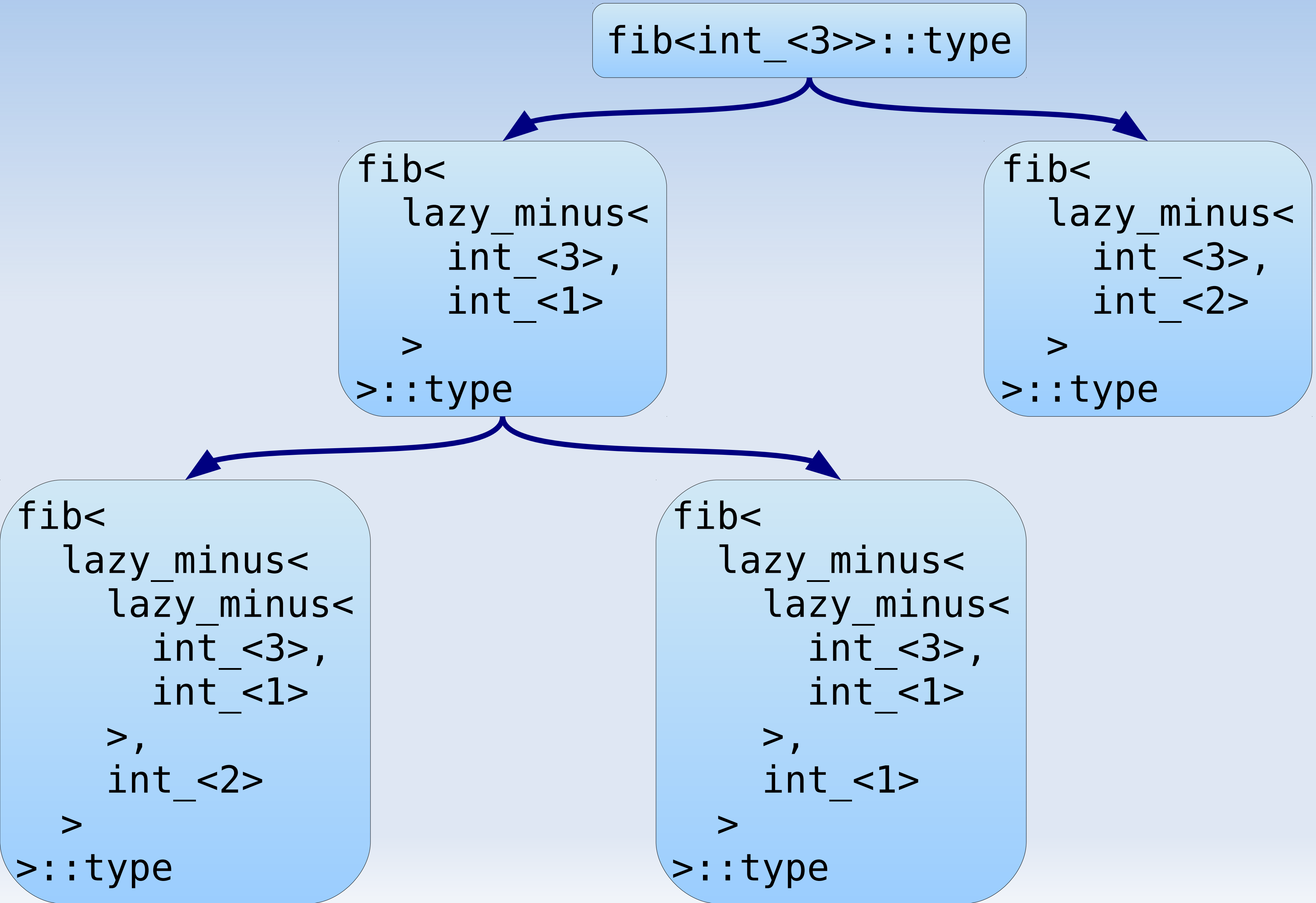
```
fib<int_<3>>::type
```

```
fib<
  lazy_minus<
    int_<3>,
    int_<1>
  >
>::type
```

```
fib<
  lazy_minus<
    int_<3>,
    int_<2>
  >
>::type
```

```
fib<
  lazy_minus<
    lazy_minus<
      int_<3>,
      int_<1>
    >,
    int_<1>
  >
>::type
```

The price of laziness



```
fib<int_<3>>::type
```

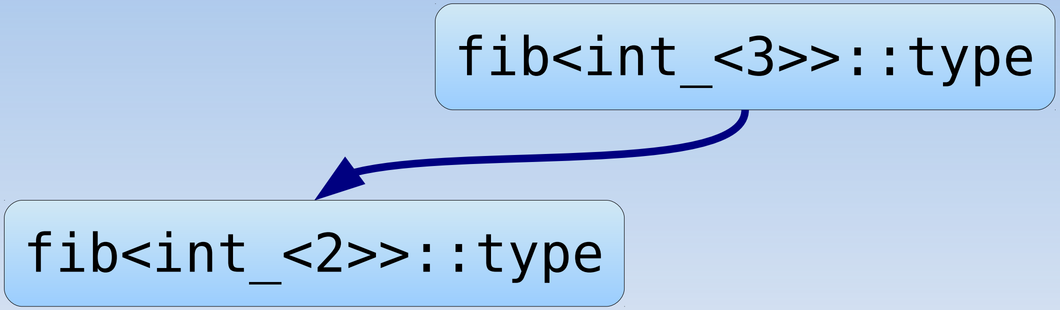
```
fib<
  lazy_minus<
    int_<3>,
    int_<1>
  >
>::type
```

```
fib<
  lazy_minus<
    int_<3>,
    int_<2>
  >
>::type
```

```
fib<
  lazy_minus<
    lazy_minus<
      int_<3>,
      int_<1>
    >,
    int_<2>
  >
>::type
```

```
fib<
  lazy_minus<
    lazy_minus<
      int_<3>,
      int_<1>
    >,
    int_<1>
  >
>::type
```

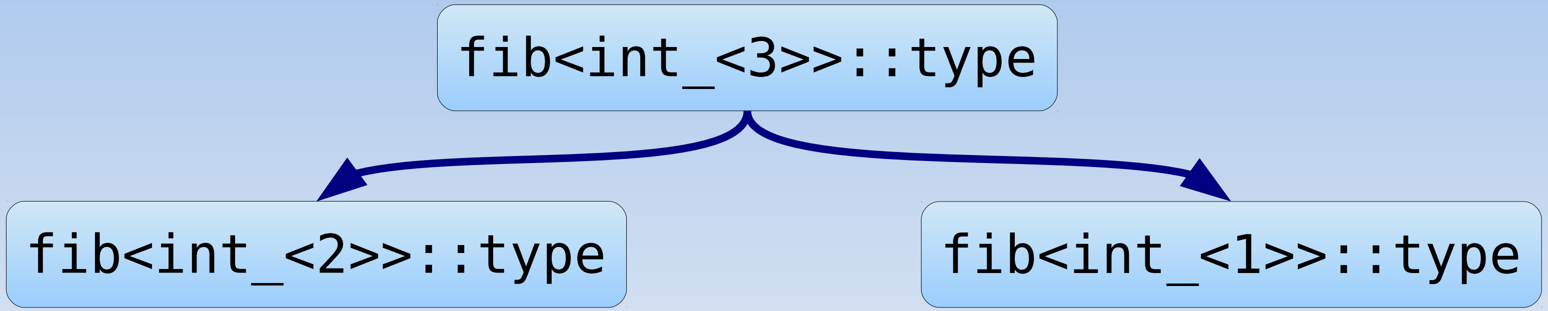
The price of laziness



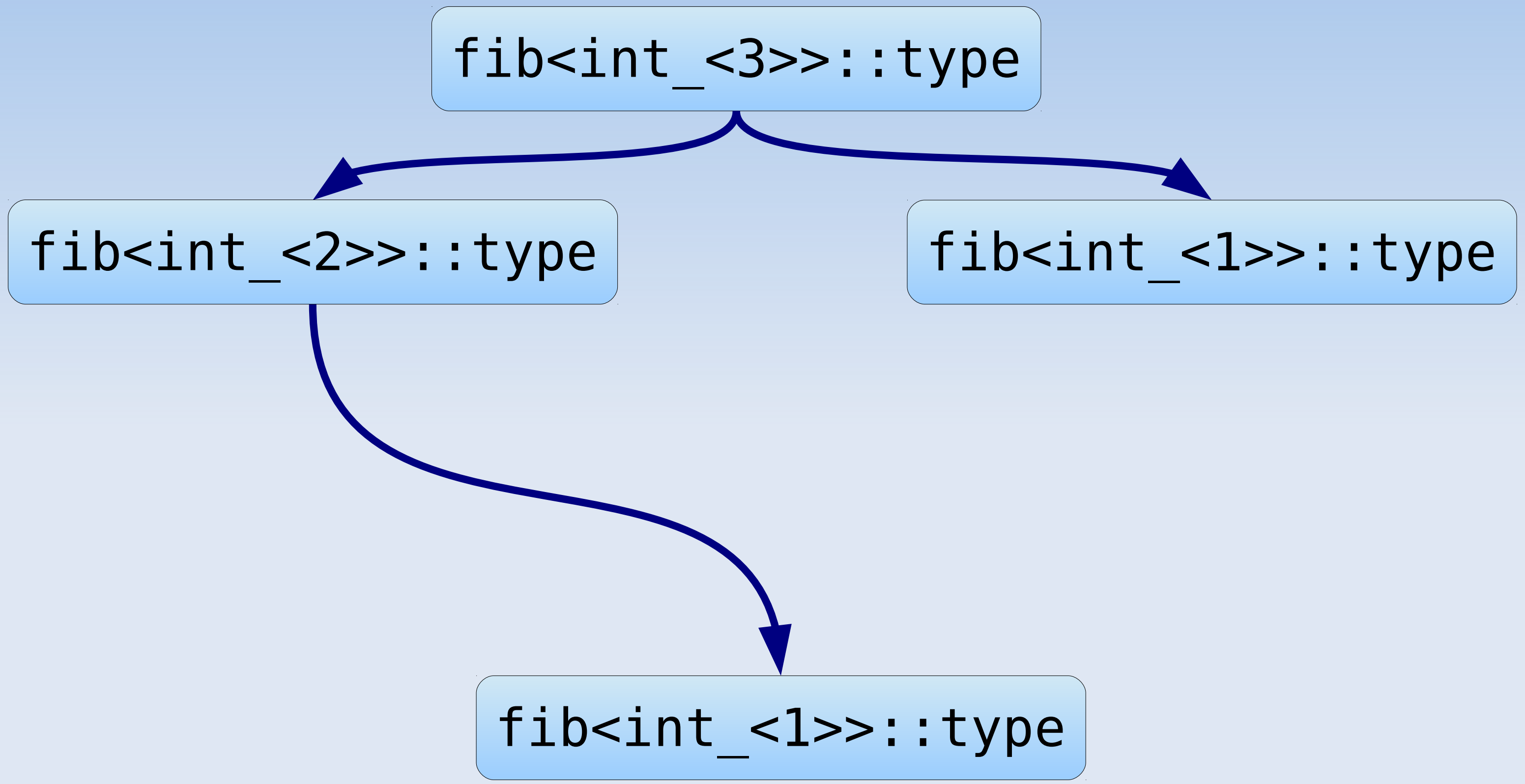
fib<int_<3>>::type

fib<int_<2>>::type

The price of laziness



The price of laziness



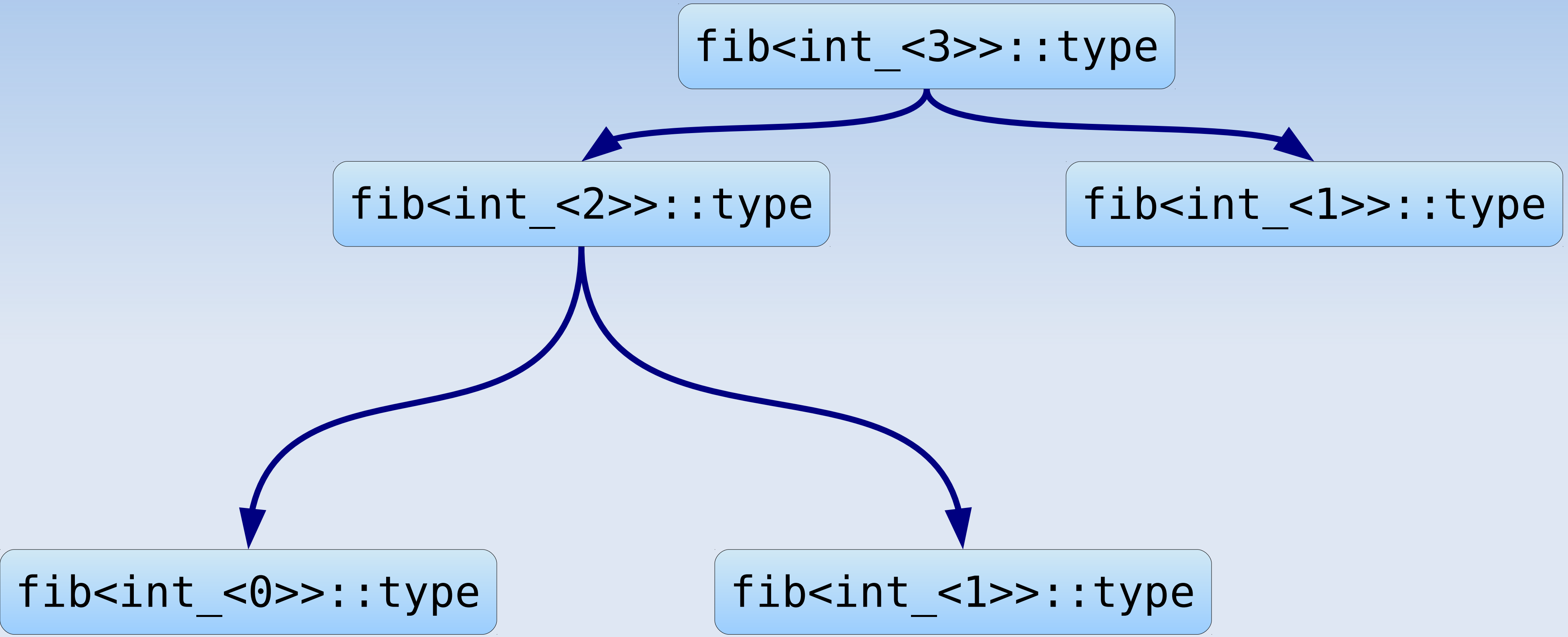
fib<int_<3>>::type

fib<int_<2>>::type

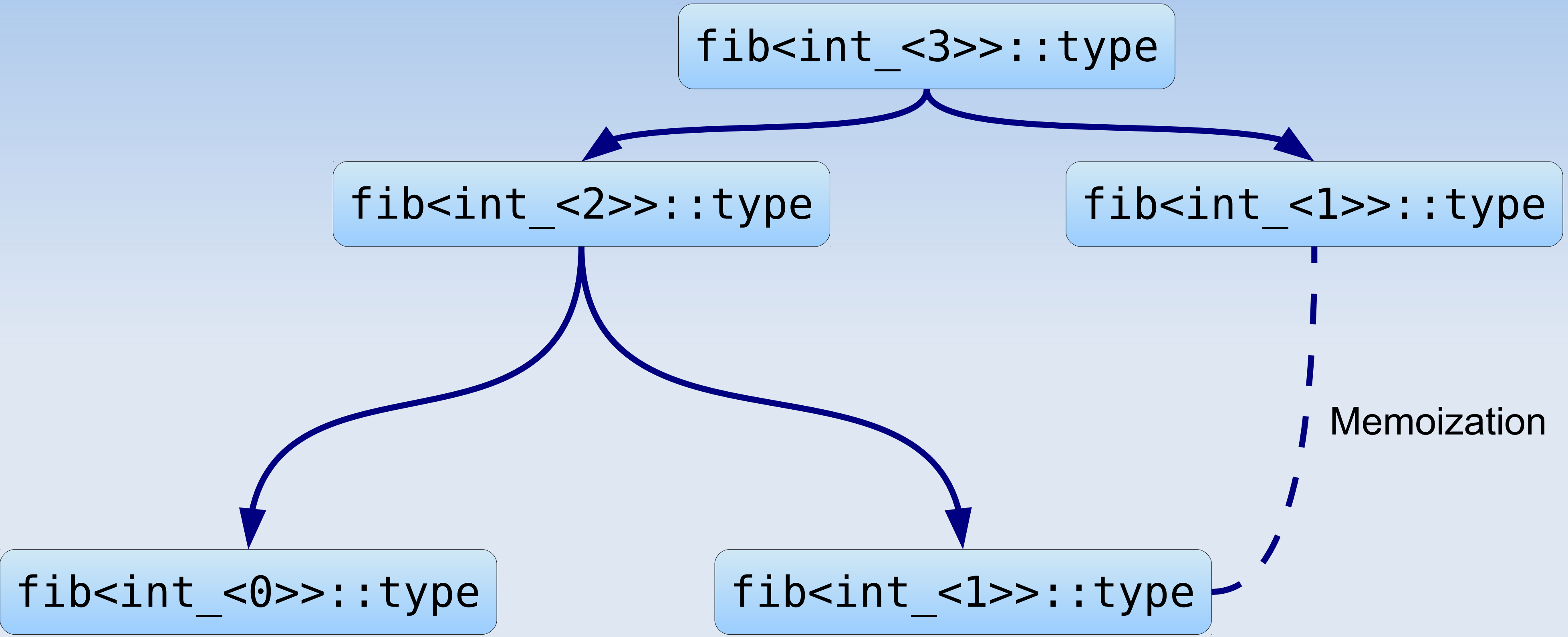
fib<int_<1>>::type

fib<int_<1>>::type

The price of laziness



The price of laziness



fib<int_<3>>::type

fib<int_<2>>::type

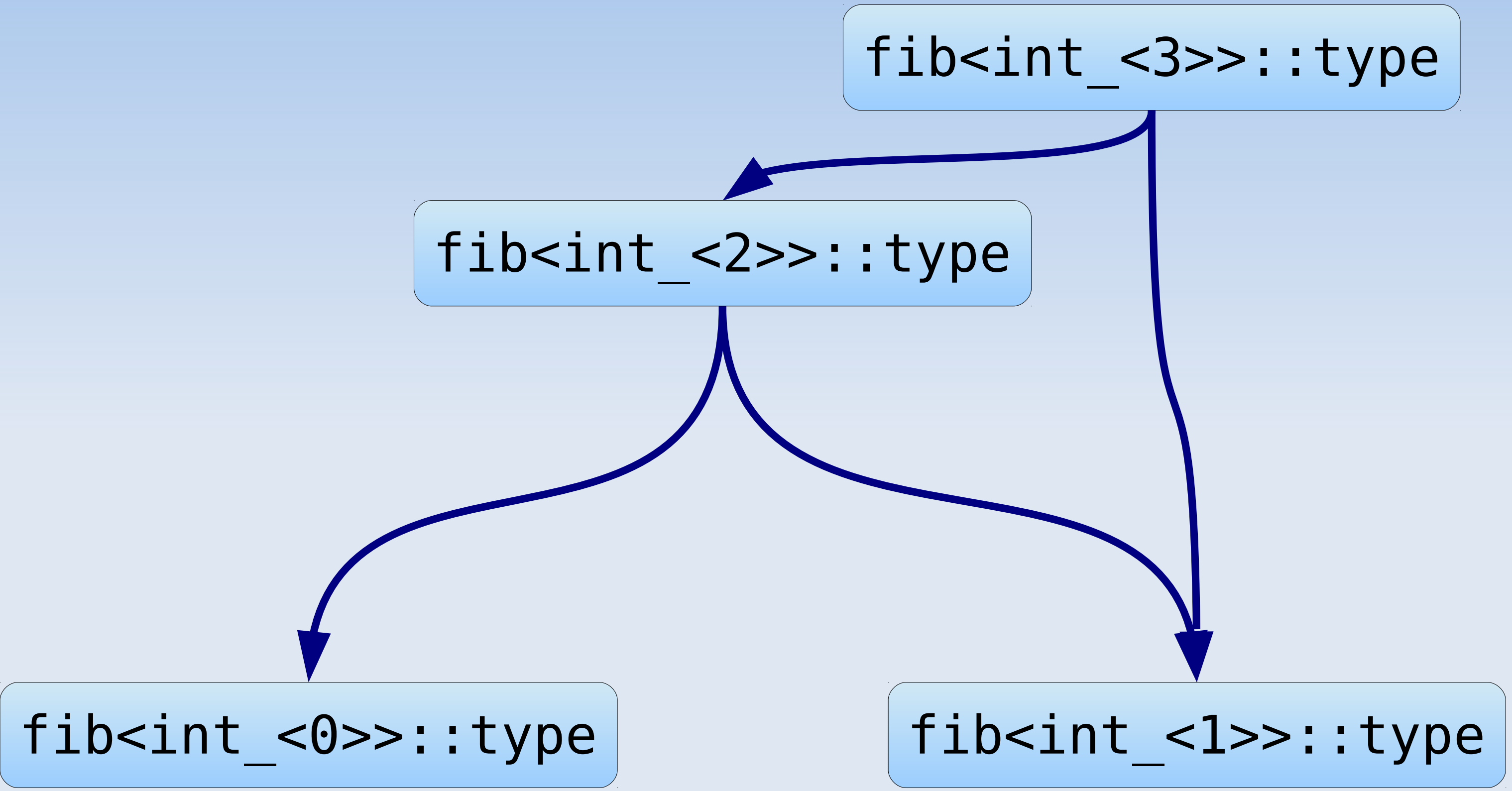
fib<int_<1>>::type

fib<int_<0>>::type

fib<int_<1>>::type

Memoization

The price of laziness



fib<int_<3>>::type

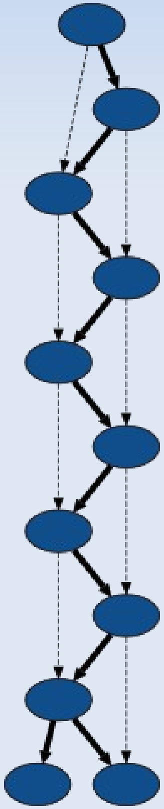
fib<int_<2>>::type

fib<int_<0>>::type

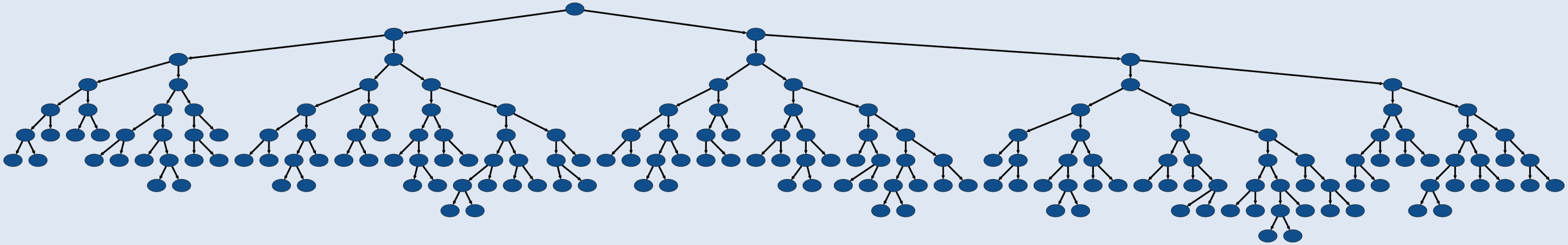
fib<int_<1>>::type

The price of laziness

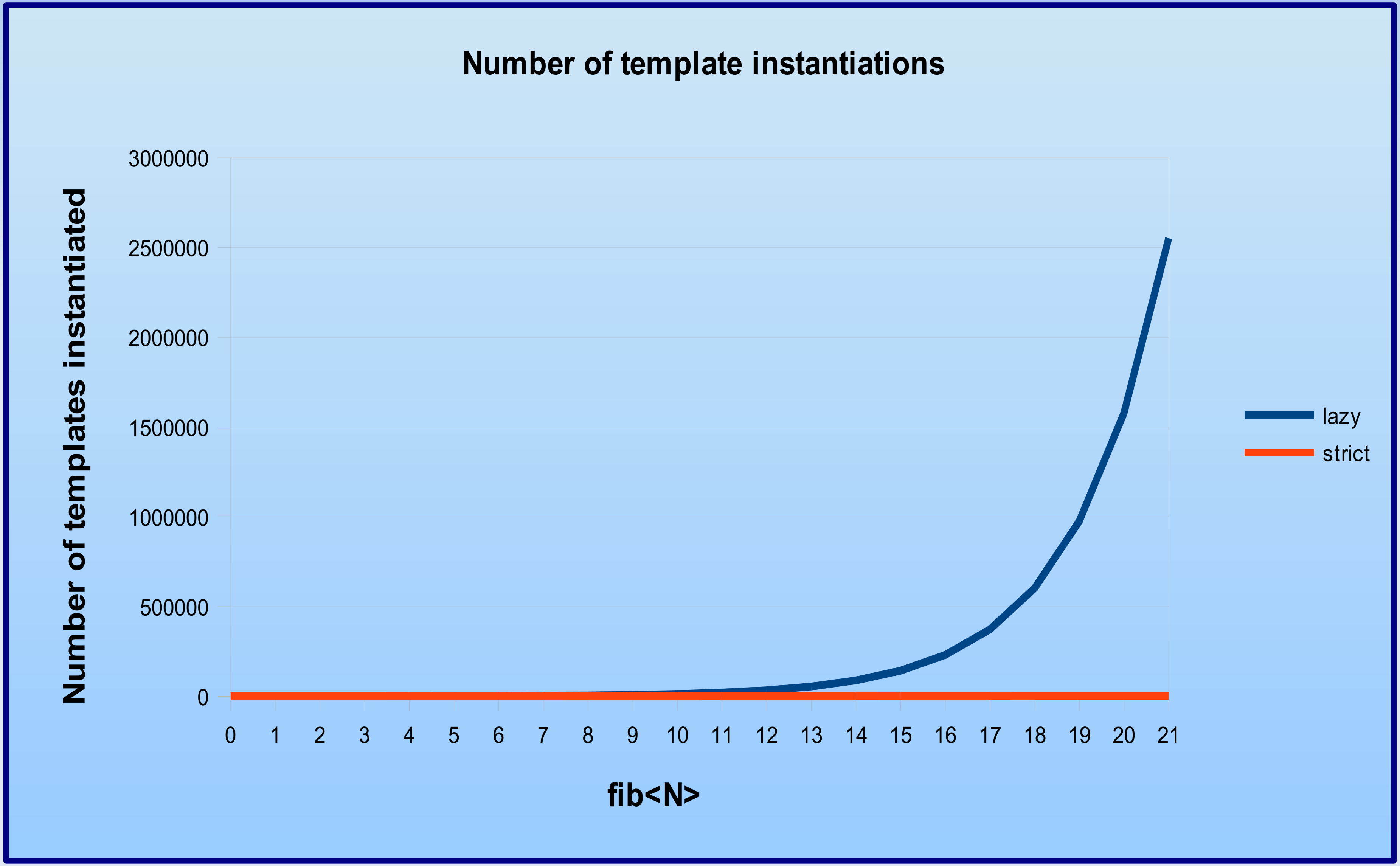
```
strict_fib<int_<10>>::type
```



```
lazy_fib<int_<10>>::type
```



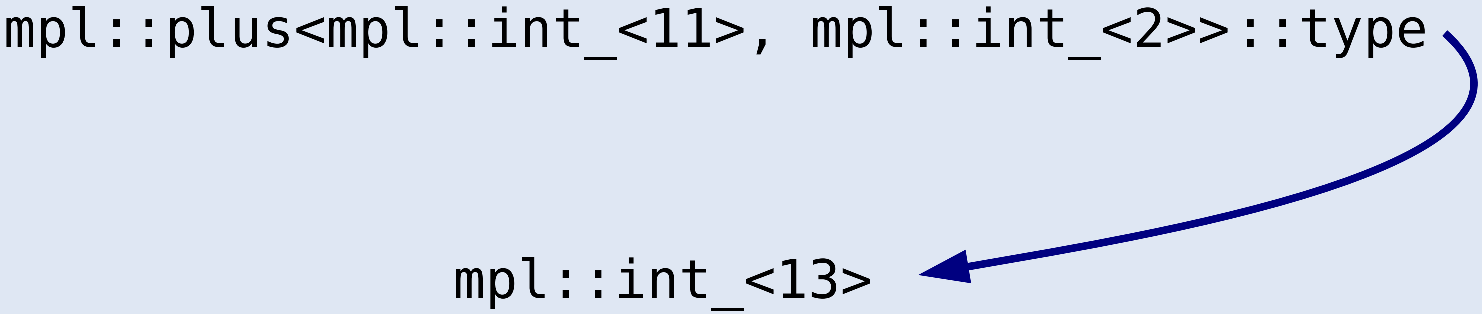
The price of laziness



Syntaxes

```
mpl::plus<mpl::int_<11>, mpl::int_<2>>
```

Syntaxes



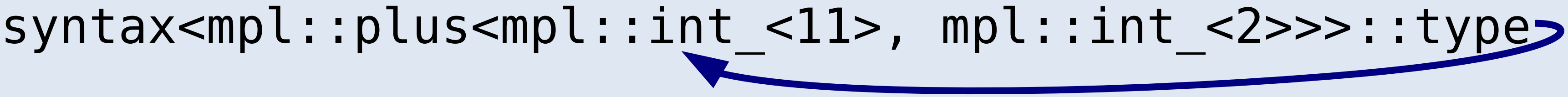
```
mpl::plus<mpl::int_<11>, mpl::int_<2>>::type
```

```
mpl::int_<13>
```

Syntaxes

```
syntax<mpl::plus<mpl::int_<11>, mpl::int_<2>>>
```

Syntaxes

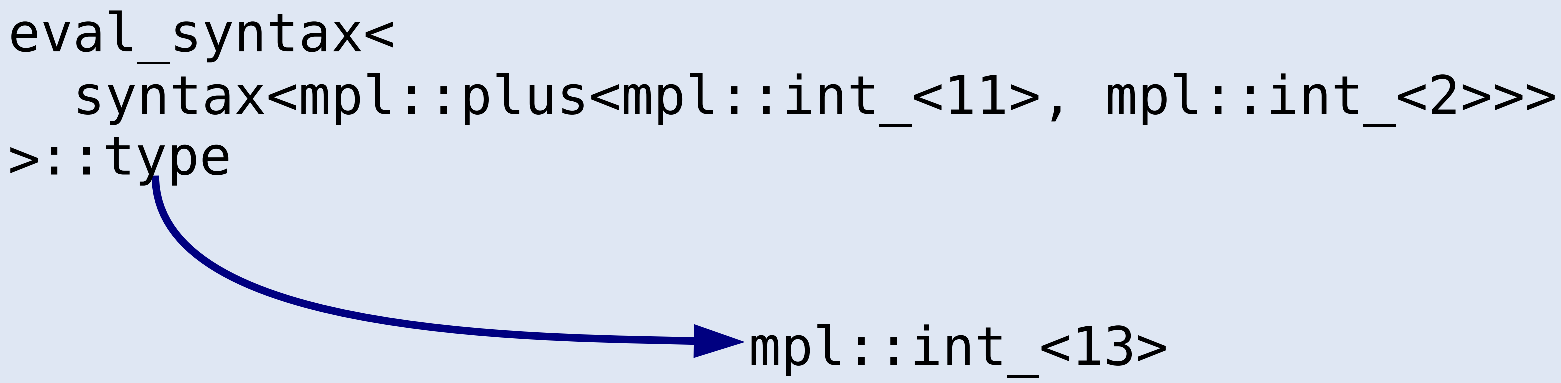


```
syntax<mpl::plus<mpl::int_<11>, mpl::int_<2>>>::type
```

Syntaxes

```
eval_syntax<
  syntax<mpl::plus<mpl::int_<11>, mpl::int_<2>>>
>
```

Syntaxes



```
eval_syntax<
  syntax<mpl::plus<mpl::int_<11>, mpl::int_<2>>>
>::type
```

```
mpl::int_<13>
```

Syntaxes

```
struct a_;



    syntax<mpl::plus<mpl::int_<11>,      var<a_>>>
```

Syntaxes

```
struct a_;
typedef var<a_> a;
```

```
    syntax<mpl::plus<mpl::int_<11>,          a  >>
```

Syntaxes

```
struct a_;
typedef var<a_> a;
// b, c, d, …, z
```

```
    syntax<mpl::plus<mpl::int_<11>,          a  >>
```

Syntaxes

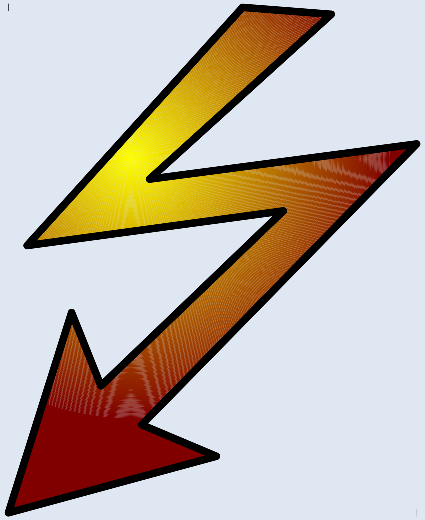
```
struct a_;
typedef var<a_> a;
// b, c, d, …, z




  eval_syntax<
    syntax<mpl::plus<mpl::int_<11>,          a  >>
  >::type
```

Syntaxes

```
struct a_;
typedef var<a_> a;
// b, c, d, …, z
```

```
  eval_syntax<
    syntax<mpl::plus<mpl::int_<11>,           a  >>
  >::type
```

```
mpl::plus<mpl::int_<11>, a>
```



Syntaxes

```
struct a_;
typedef var<a_> a;
// b, c, d, …, z


  let<
    a, syntax<mpl::int_<2>>,
    syntax<mpl::plus<mpl::int_<11>,          a  >>
  >
```

Syntaxes

```
struct a_;
typedef var<a_> a;
// b, c, d, …, z
```

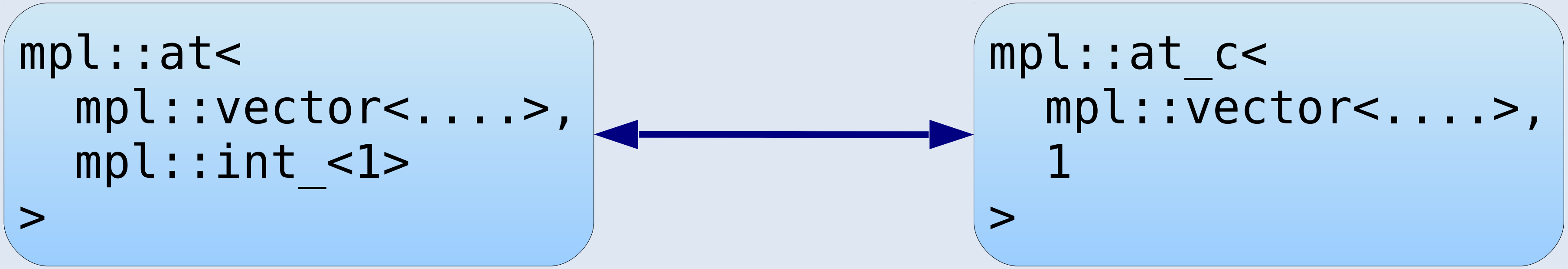
```
  let<
    a, syntax<mpl::int_<2>>,
    syntax<mpl::plus<mpl::int_<11>,          a  >>
  >::type
       ↓

    syntax<mpl::plus<mpl::int_<11>, mpl::int_<2>>>
```

Syntaxes

```
struct a_;
typedef var<a_> a;
// b, c, d, …, z
```

```
  let<
    a, syntax<mpl::int_<2>>,
    syntax<mpl::plus<mpl::int_<11>,          a  >>
  >::type
```

```
    syntax<mpl::plus<mpl::int_<11>, mpl::int_<2>>>
```

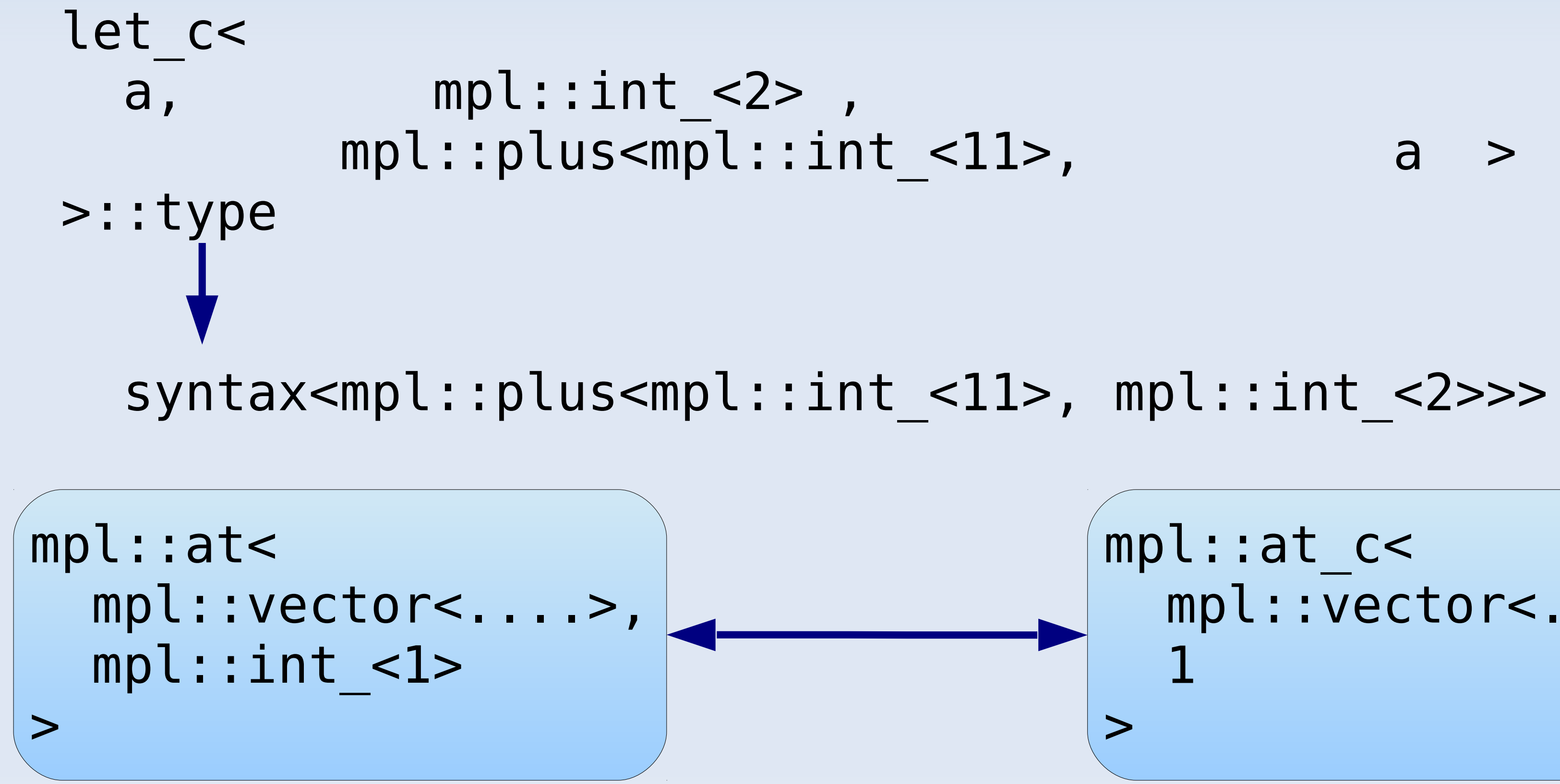


```
mpl::at<
  mpl::vector<....>,
  mpl::int_<1>
>
```

```
mpl::at_c<
  mpl::vector<....>,
  1
>
```

Syntaxes

```
struct a_;
typedef var<a_> a;
// b, c, d, …, z
```



```
let_c<
  a,        mpl::int_<2> ,
         mpl::plus<mpl::int_<11>,          a  >
>::type
```

```
  syntax<mpl::plus<mpl::int_<11>, mpl::int_<2>>>
```

```
mpl::at<
  mpl::vector<....>,
  mpl::int_<1>
>
```

```
mpl::at_c<
  mpl::vector<....>,
  1
>
```

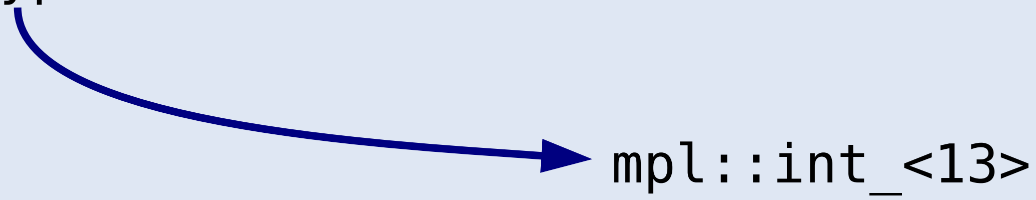
Syntaxes

```
struct a_;
typedef var<a_> a;
// b, c, d, …, z

eval_syntax<
  let_c<
    a,        mpl::int_<2> ,
           mpl::plus<mpl::int_<11>,          a  >
  >
>::type
                       → mpl::int_<13>
```

Syntaxes

```
struct a_;
typedef var<a_> a;
// b, c, d, …, z
```

```
  eval_let_c<
    a,        mpl::int_<2> ,
           mpl::plus<mpl::int_<11>,          a  >
  >::type
```



```
mpl::int_<13>
```

```
syntax<mpl::plus<a,            b>>
```

Lambdas

```
lambda<      syntax<mpl::plus<a,            b>>>
```

Lambdas

```
lambda<a, b, syntax<mpl::plus<a,            b>>>
```

Lambdas

```
typedef lambda<a, b, syntax<mpl::plus<a,            b>>> add;
```

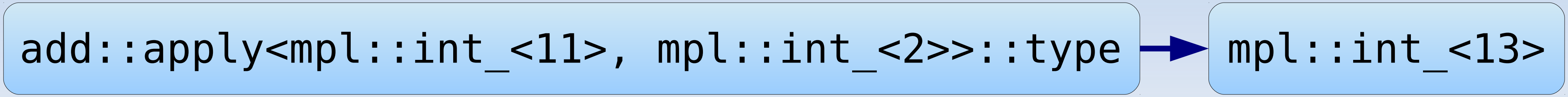
Lambdas

```
typedef lambda<a, b, syntax<mpl::plus<a,            b>>> add;
```

```
add::apply<mpl::int_<11>, mpl::int_<2>>::type
```

Lambdas

```
typedef lambda<a, b, syntax<mpl::plus<a,             b>>> add;
```

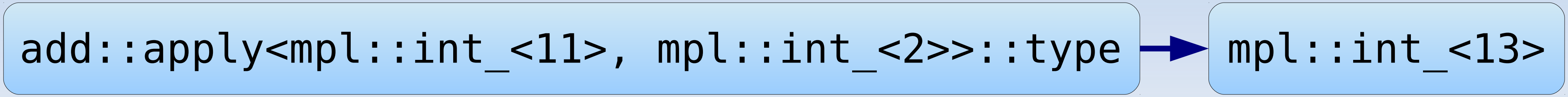


add::apply<mpl::int_<11>, mpl::int_<2>>::type

mpl::int_<13>

Lambdas

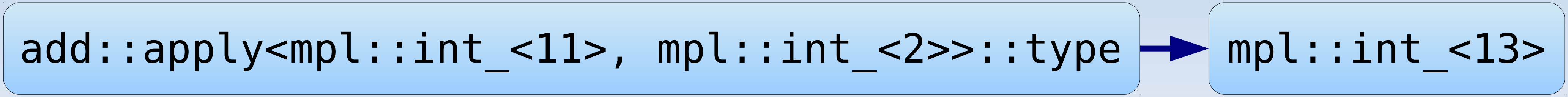
```
typedef lambda_c<a, b,      mpl::plus<a,            b> > add;
```



add::apply<mpl::int_<11>, mpl::int_<2>>::type → mpl::int_<13>

Lambdas

```
typedef lambda_c<a, b,      mpl::plus<a,            b> > add;
```



```
add::apply<mpl::int_<11>, mpl::int_<2>>::type
```

```
mpl::int_<13>
```

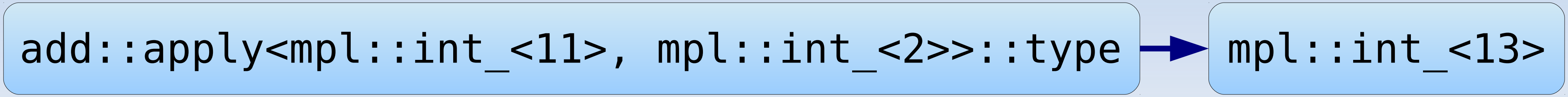
```
add::apply<mpl::int_<1>>::type
```

Lambdas

```
  typedef lambda_c<a, b,      mpl::plus<a,            b> > add;
          lambda_c<   b,      mpl::plus<mpl::int_<1>, b> >

add::apply<mpl::int_<11>, mpl::int_<2>>::type → mpl::int_<13>

          add::apply<mpl::int_<1>>::type
```

Lambdas

```
typedef lambda_c<a, b,      mpl::plus<a,            b> > add;
        lambda_c<   b,      mpl::plus<mpl::int_<1>, b> >
```



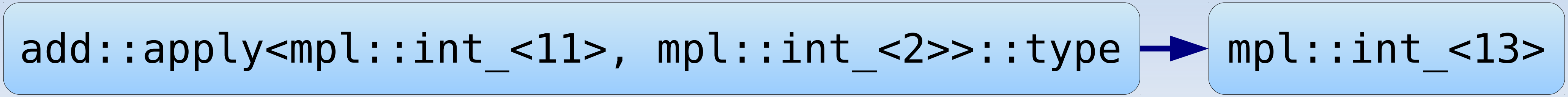
```
add::apply<mpl::int_<11>, mpl::int_<2>>::type
```

```
mpl::int_<13>
```

```
typedef add::apply<mpl::int_<1>>::type inc;
```

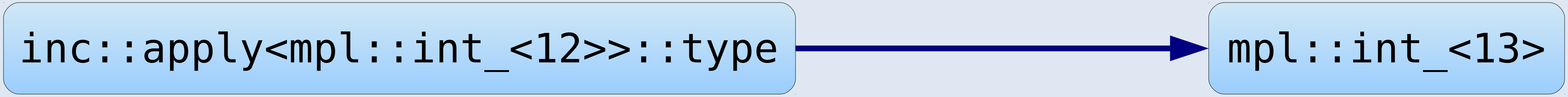
Lambdas

```
typedef lambda_c<a, b,      mpl::plus<a,            b> > add;
        lambda_c<   b,      mpl::plus<mpl::int_<1>, b> >
```



```
add::apply<mpl::int_<11>, mpl::int_<2>>::type → mpl::int_<13>
```

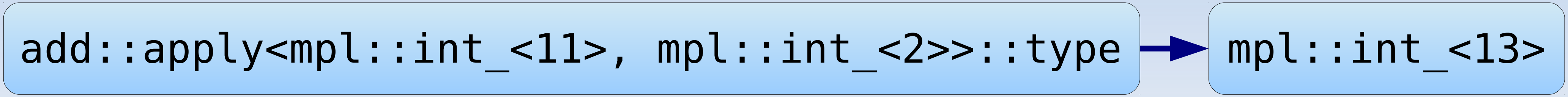
```
typedef add::apply<mpl::int_<1>>::type inc;
```



```
inc::apply<mpl::int_<12>>::type → mpl::int_<13>
```

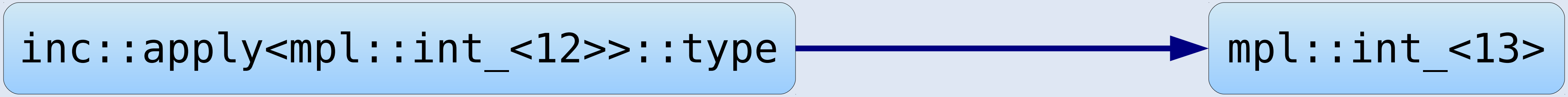
Lambdas

```
typedef lambda_c<a, b,     mpl::plus<a,               b> > add;
        lambda_c<   b,     mpl::plus<mpl::int_<1>, b> >
```



```
add::apply<mpl::int_<11>, mpl::int_<2>>::type
mpl::int_<13>
```

```
typedef add::apply<mpl::int_<1>>::type inc;
```



```
inc::apply<mpl::int_<12>>::type
mpl::int_<13>
```

```
MPLLIBS_METAFUNCTION(my_plus, (A)(B)) ((mpl::plus<A, B>));
```

Lambdas

```
typedef lambda_c<a, b,      mpl::plus<a,            b> > add;
        lambda_c<   b,      mpl::plus<mpl::int_<1>, b> >
```

```
add::apply<mpl::int_<11>, mpl::int_<2>>::type → mpl::int_<13>
```

```
typedef add::apply<mpl::int_<1>>::type inc;
```

```
inc::apply<mpl::int_<12>>::type → mpl::int_<13>
```

```
MPLLIBS_METAFUNCTION(my_plus, (A)(B)) ((mpl::plus<A, B>));

        my_plus<mpl::int_<1>>::type
```

Lambdas

```
typedef lambda_c<a, b,      mpl::plus<a,            b> > add;
        lambda_c<   b,      mpl::plus<mpl::int_<1>, b> >
```

```
add::apply<mpl::int_<11>, mpl::int_<2>>::type → mpl::int_<13>
```

```
typedef add::apply<mpl::int_<1>>::type inc;
```

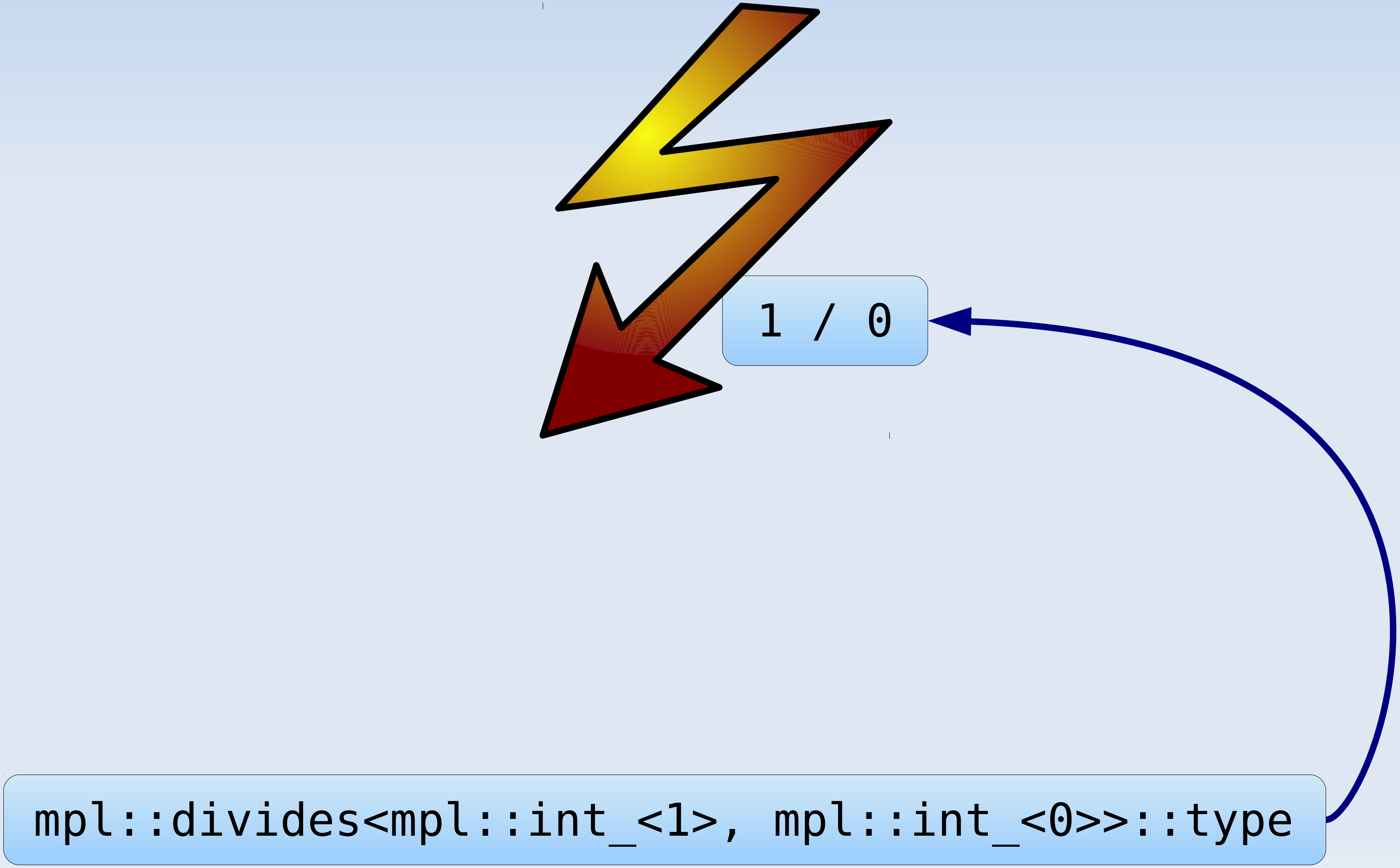
```
inc::apply<mpl::int_<12>>::type                → mpl::int_<13>
```

```
MPLLIBS_METAFUNCTION(my_plus, (A)(B)) ((mpl::plus<A, B>));

typedef my_plus<mpl::int_<1>>::type inc;
```

Error handling

```
mpl::divides<mpl::int_<1>, mpl::int_<0>>::type
```

Error handling



1 / 0

```
mpl::divides<mpl::int_<1>, mpl::int_<0>>::type
```

Error handling

```
MPLLIBS_METAFUNCTION(safe_divides, (A)(B))
((





));
```

```
safe_divides<mpl::int_<1>, mpl::int_<0>>::type
```

Error handling

```
MPLLIBS_METAFUNCTION(safe_divides, (A)(B))
((
  if_<
    lazy_equal_to<mpl::int_<0>, B>,


  >
));
```

```
safe_divides<mpl::int_<1>, mpl::int_<0>>::type
```

Error handling

```
struct nothing;


MPLLIBS_METAFUNCTION(safe_divides, (A)(B))
((
  if_<
    lazy_equal_to<mpl::int_<0>, B>,
    nothing,

  >
));
```

```
safe_divides<mpl::int_<1>, mpl::int_<0>>::type
```

Error handling

```
struct nothing;
template <class T> struct just;

MPLLIBS_METAFUNCTION(safe_divides, (A)(B))
((
  if_<
    lazy_equal_to<mpl::int_<0>, B>,
    nothing,
    just<lazy_divides<A, B>>
  >
));
```

```
safe_divides<mpl::int_<1>, mpl::int_<0>>::type
```

Error handling

```
// Maybe
struct nothing;
template <class T> struct just;

MPLLIBS_METAFUNCTION(safe_divides, (A)(B))
((
  if_<
    lazy_equal_to<mpl::int_<0>, B>,
    nothing,
    just<lazy_divides<A, B>>
  >
));
```

```
safe_divides<mpl::int_<1>, mpl::int_<0>>::type
```

Error handling

```
// Maybe
MPLLIBS_DATA(maybe, ((nothing, 0))((just, 1)));


MPLLIBS_METAFUNCTION(safe_divides, (A)(B))
((
  if_<
    lazy_equal_to<mpl::int_<0>, B>,
    nothing,
    just<lazy_divides<A, B>>
  >
));
```

```
safe_divides<mpl::int_<1>, mpl::int_<0>>::type
```

Error handling

```
f<A, B>   →   A + 12 / B
```

Error handling

```
MPLLIBS_METAFUNCTION(f, (A)(B))
((

                 safe_divides<mpl::int_<12>, B>




));
```

```
f<A, B>   →   A + 12 / B
```

Error handling

```
MPLLIBS_METAFUNCTION(f, (A)(B))
((
  if_<
    lazy_is_same<safe_divides<mpl::int_<12>, B>, nothing>,


  >
));
```

```
f<A, B>   →   A + 12 / B
```

Error handling

```
MPLLIBS_METAFUNCTION(f, (A)(B))
((
  if_<
    lazy_is_same<safe_divides<mpl::int_<12>, B>, nothing>,
    nothing,

  >
));
```

```
f<A, B>    →    A + 12 / B
```

Error handling

```
MPLLIBS_METAFUNCTION(f, (A)(B))
((
  if_<
    lazy_is_same<safe_divides<mpl::int_<12>, B>, nothing>,
    nothing,
    ???
  >
));
```

```
f<A, B>   →   A + 12 / B
```

Error handling

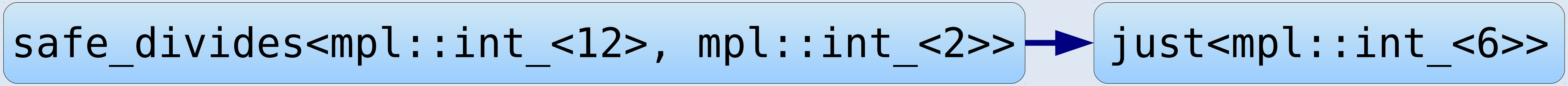
```
MPLLIBS_METAFUNCTION(f, (A)(B))
((
  if_<
    lazy_is_same<safe_divides<mpl::int_<12>, B>, nothing>,
    nothing,
    ???
  >
));
```

```
safe_divides<mpl::int_<12>, mpl::int_<2>>
```

```
f<A, B>    →    A + 12 / B
```

Error handling

```
MPLLIBS_METAFUNCTION(f, (A)(B))
((
  if_<
    lazy_is_same<safe_divides<mpl::int_<12>, B>, nothing>,
    nothing,
    ???
  >
));
```



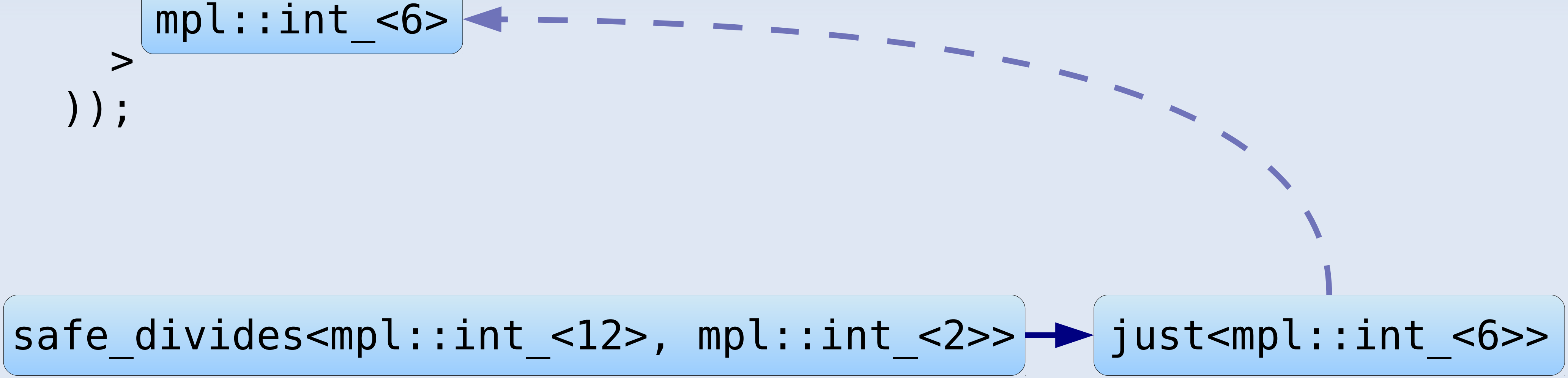
```
safe_divides<mpl::int_<12>, mpl::int_<2>>
```

```
just<mpl::int_<6>>
```

```
f<A, B>   →   A + 12 / B
```

Error handling

```
MPLLIBS_METAFUNCTION(f, (A)(B))
((
  if_<
    lazy_is_same<safe_divides<mpl::int_<12>, B>, nothing>,
    nothing,
    mpl::int_<6>
  >
));
```

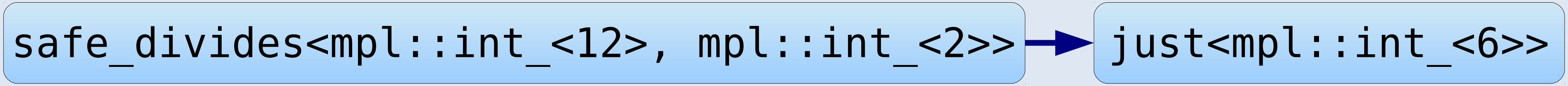


```
safe_divides<mpl::int_<12>, mpl::int_<2>> → just<mpl::int_<6>>
```

```
f<A, B>   →   A + 12 / B
```

Error handling

```
MPLLIBS_METAFUNCTION(f, (A)(B))
((
       case safe_divides<mpl::int_<12>, B> of
              nothing → nothing
              just<x> → mpl::plus<A, x>



));
```

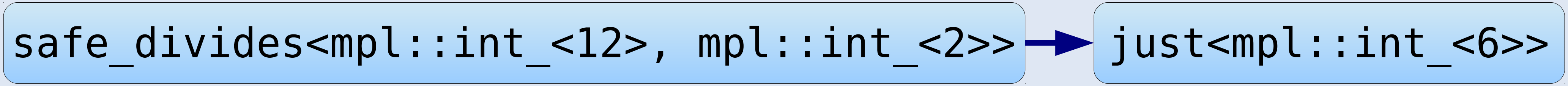


```
safe_divides<mpl::int_<12>, mpl::int_<2>> → just<mpl::int_<6>>
```

```
f<A, B>    →    A + 12 / B
```

Error handling

```
MPLLIBS_METAFUNCTION(f, (A)(B))
((
  eval_case<safe_divides<mpl::int_<12>, B>,
    matches_c<nothing,  nothing>,
    matches_c<just<x>,  mpl::plus<A, x>>
  >

));
```

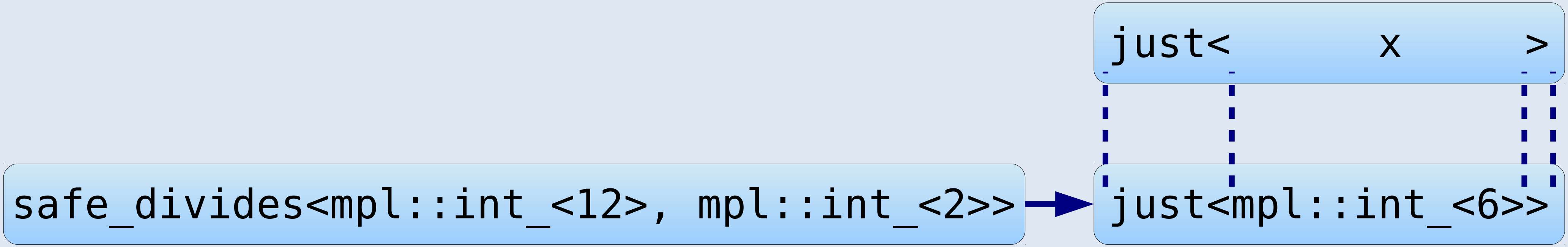


safe_divides<mpl::int_<12>, mpl::int_<2>> → just<mpl::int_<6>>

f<A, B>   →   A + 12 / B

Error handling

```
MPLLIBS_METAFUNCTION(f, (A)(B))
((
  eval_case<safe_divides<mpl::int_<12>, B>,
    matches_c<nothing,  nothing>,
    matches_c<just<x>,  mpl::plus<A, x>>
  >

));
```



just<   x   >

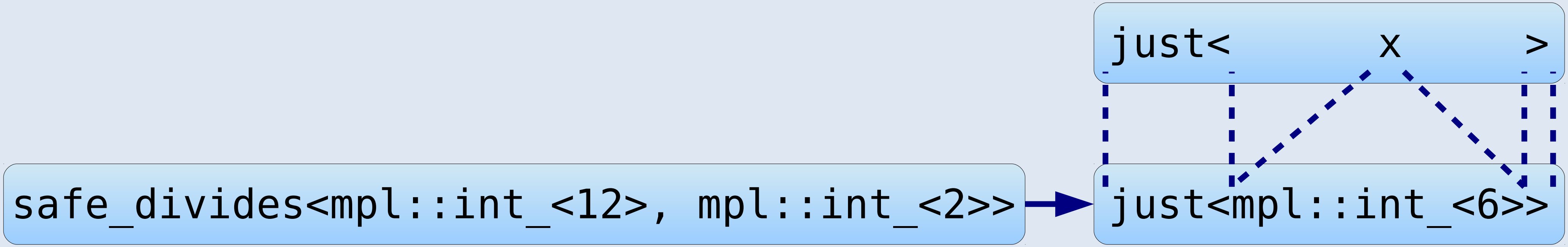
safe_divides<mpl::int_<12>, mpl::int_<2>>

just<mpl::int_<6>>

f<A, B> → A + 12 / B

Error handling

```
MPLLIBS_METAFUNCTION(f, (A)(B))
((
  eval_case<safe_divides<mpl::int_<12>, B>,
    matches_c<nothing,  nothing>,
    matches_c<just<x>,  mpl::plus<A, x>>
  >

));
```

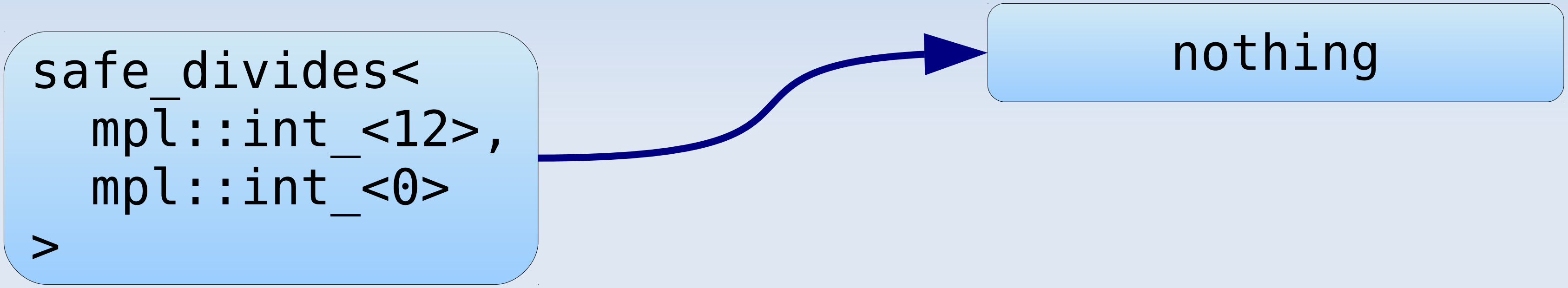


```
just<      x      >
```

```
safe_divides<mpl::int_<12>, mpl::int_<2>> → just<mpl::int_<6>>
```

```
f<A, B>   →   A + 12 / B
```

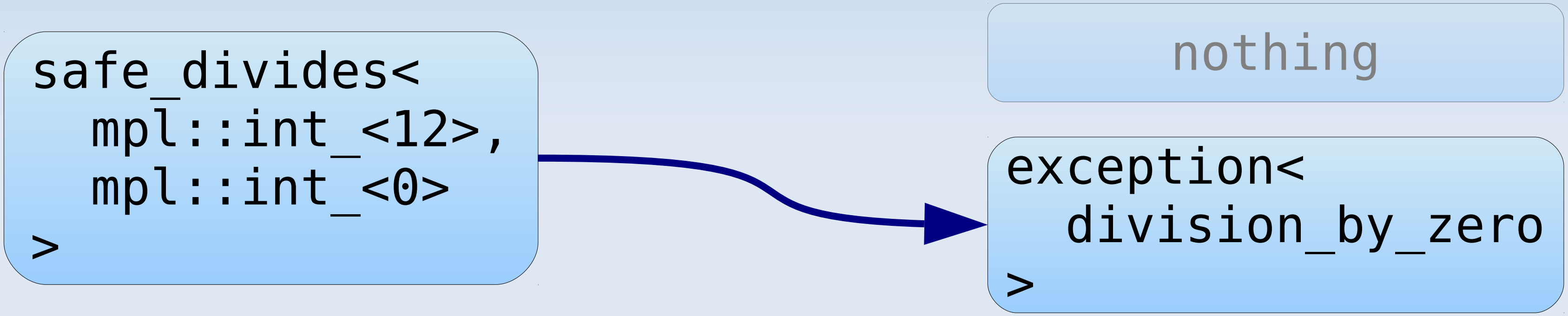
Error handling



```
safe_divides<
  mpl::int_<12>,
  mpl::int_<0>
>
```

Error handling

```
struct division_by_zero;
```



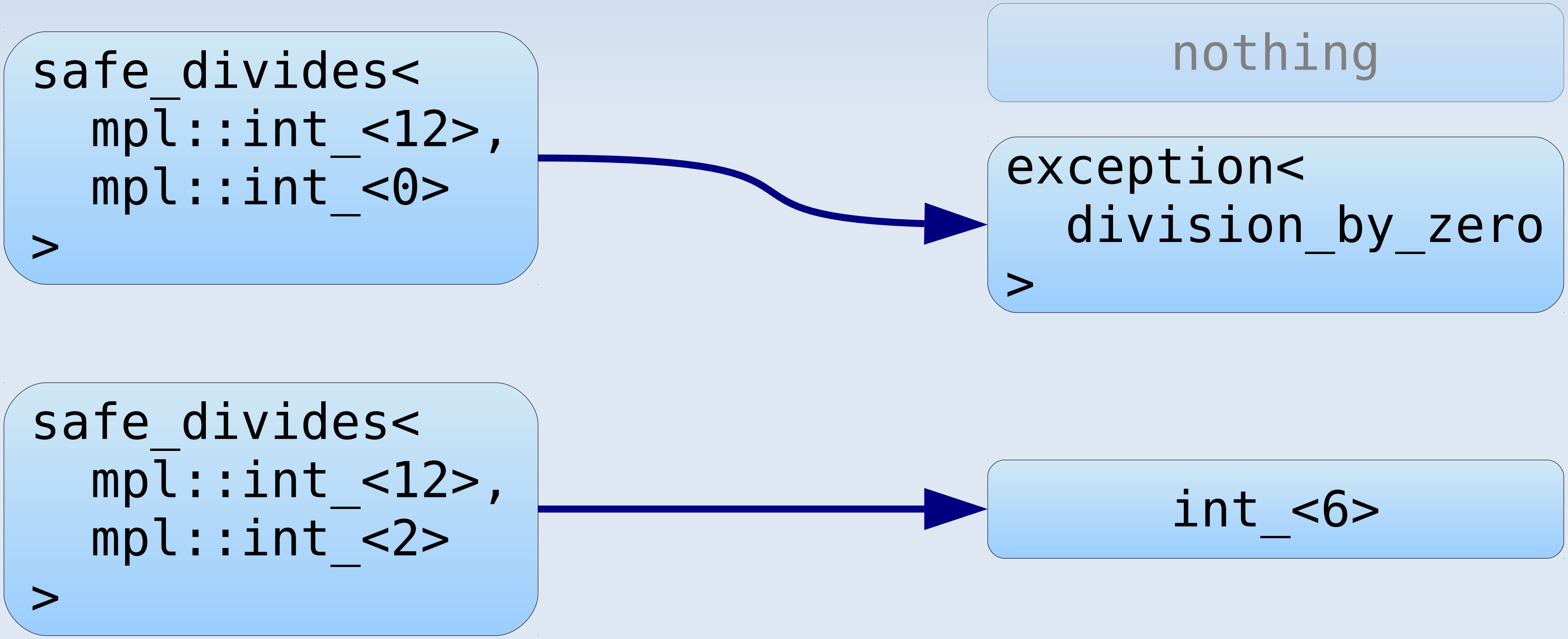
```
safe_divides<
  mpl::int_<12>,
  mpl::int_<0>
>
```

```
nothing
```

```
exception<
  division_by_zero
>
```

Error handling

```
struct division_by_zero;
```



```
safe_divides<
  mpl::int_<12>,
  mpl::int_<0>
>
```

```
nothing
```

```
exception<
  division_by_zero
>
```

```
safe_divides<
  mpl::int_<12>,
  mpl::int_<2>
>
```

```
int_<6>
```

Error handling

```
MPLLIBS_METAFUNCTION(f, (A)(B))
((
  eval_case<safe_divides<mpl::int_<12>, B>



  >
));
```

```
f<A, B>   →   A + 12 / B
```

Error handling

```
MPLLIBS_METAFUNCTION(f, (A)(B))
((
  eval_case<safe_divides<mpl::int_<12>, B>,
    matches_c<exception<e>, exception<e>>

  >
));
```

```
f<A, B>   →   A + 12 / B
```

Error handling

```
MPLLIBS_METAFUNCTION(f, (A)(B))
((
  eval_case<safe_divides<mpl::int_<12>, B>,
    matches_c<exception<e>, exception<e>>,
    matches_c<x,            mpl::plus<A, x>>
  >
));
```

f<A, B> → A + 12 / B

Error handling

```
MPLLIBS_METAFUNCTION(f, (A)(B))
((
  eval_case<safe_divides<mpl::int_<12>, B>,
    matches_c<exception<e>, exception<e>>,
    matches_c<x,            mpl::plus<A, x>>
  >
));
```

```
MPLLIBS_METAFUNCTION(f, (A)(B))
((

                 safe_divides<mpl::int_<12>, B>




));
```

```
f<A, B>   →   A + 12 / B
```

Error handling

```
MPLLIBS_METAFUNCTION(f, (A)(B))
((
  eval_case<safe_divides<mpl::int_<12>, B>,
    matches_c<exception<e>, exception<e>>,
    matches_c<x,            mpl::plus<A, x>>
  >
));
```

```
MPLLIBS_METAFUNCTION(f, (A)(B))
((

    mpl::plus<A, safe_divides<mpl::int_<12>, B>>




));
```

```
f<A, B>   →   A + 12 / B
```

Error handling

```
MPLLIBS_METAFUNCTION(f, (A)(B))
((
  eval_case<safe_divides<mpl::int_<12>, B>,
    matches_c<exception<e>, exception<e>>,
    matches_c<x,            mpl::plus<A, x>>
  >
));
```

```
MPLLIBS_METAFUNCTION(f, (A)(B))
((
  try_c<
    mpl::plus<A, safe_divides<mpl::int_<12>, B>>



  >
));
```

`f<A, B>    →    A + 12 / B`

Error handling

```
MPLLIBS_METAFUNCTION(f, (A)(B))
((
  eval_case<safe_divides<mpl::int_<12>, B>,
    matches_c<exception<e>, exception<e>>,
    matches_c<x,            mpl::plus<A, x>>
  >
));
```

```
MPLLIBS_METAFUNCTION(f, (A)(B))
((
  try_c<
    mpl::plus<A, safe_divides<mpl::int_<12>, B>>,

    catch_c<e,                                       >

  >
));
```

f<A, B> → A + 12 / B

Error handling

```
MPLLIBS_METAFUNCTION(f, (A)(B))
((
  eval_case<safe_divides<mpl::int_<12>, B>,
    matches_c<exception<e>, exception<e>>,
    matches_c<x,            mpl::plus<A, x>>
  >
));
```

```
MPLLIBS_METAFUNCTION(f, (A)(B))
((
  try_c<
    mpl::plus<A, safe_divides<mpl::int_<12>, B>>,

    catch_c<e, boost::is_same<e, division_by_zero>,  >

  >
));
```

```
f<A, B>   →   A + 12 / B
```

Error handling

```
MPLLIBS_METAFUNCTION(f, (A)(B))
((
  eval_case<safe_divides<mpl::int_<12>, B>,
    matches_c<exception<e>, exception<e>>,
    matches_c<x,            mpl::plus<A, x>>
  >
));
```

```
MPLLIBS_METAFUNCTION(f, (A)(B))
((
  try_c<
    mpl::plus<A, safe_divides<mpl::int_<12>, B>>,

    catch_c<e, boost::is_same<e, division_by_zero>, A>

  >
));
```

f<A, B>   →   A + 12 / B

Error handling

```
MPLLIBS_METAFUNCTION(f, (A)(B))
((
  eval_case<safe_divides<mpl::int_<12>, B>,
    matches_c<exception<e>, exception<e>>,
    matches_c<x,            mpl::plus<A, x>>
  >
));
```

```
MPLLIBS_METAFUNCTION(f, (A)(B))
((
  try_c<
    mpl::plus<A, safe_divides<mpl::int_<12>, B>>,

    catch_c<e, boost::is_same<e, division_by_zero>, A>,
    catch_c<e, boost::true_, /* … */>
  >
));
```

```
f<A, B>   →   A + 12 / B
```

Summary

- Laziness
- Syntaxes
- Let/Lambda/Case expressions
- Algebraic data-types
- Exceptions

Fact

```
template <class N>
struct fact;

template <class N>
struct fact_impl :
  times<
    N,
    typename fact<typename minus<N, int_<1>>::type>::type
  >
{};

template <class N>
struct fact :
  eval_if<
    typename equal_to<N, int_<1>>::type,
    int_<1>,
    fact_impl<N>
  >
{};
```

Fact

```
template <class N>
struct fact;

template <class N>
s
```

```
MPLLIBS_METAFUNCTION(fact, (N))
((
  eval_case< N,
    matches_c<int_<0>, int_<1>>,
    matches_c<_,       times<N, fact<minus<N, int_<1>>>>
  >
));
```

```
struct fact :
  eval_if<
    typename equal_to<N, int_<1>>::type,
    int_<1>,
    fact_impl<N>
  >
{};
```

Q & A

Mpllibs.Metamonad

http://abel.web.elte.hu/mpllibs