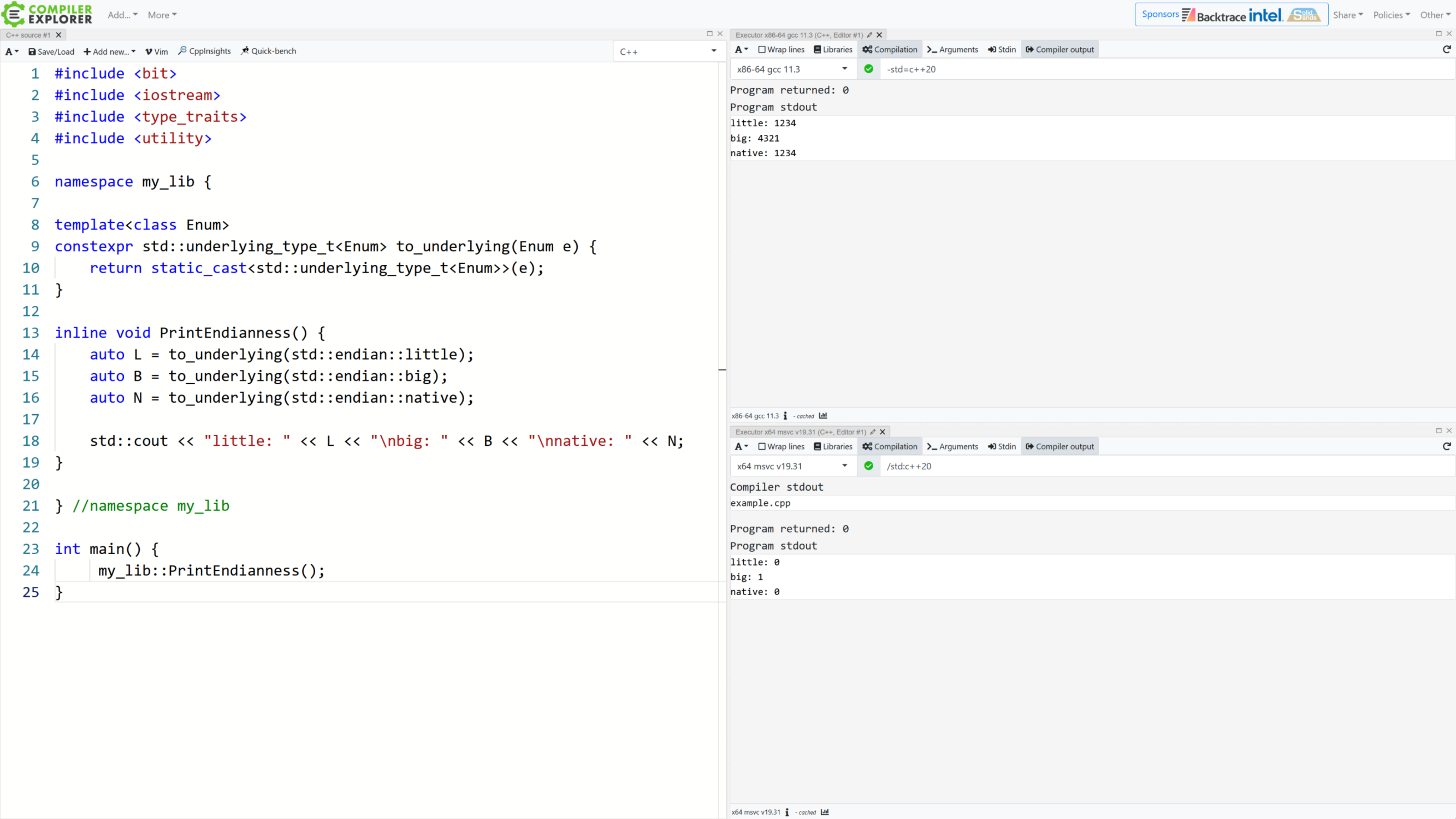
BRACE YOURSELVES

{ }

`std::to_underlying` IS COMING

by
Kilian Henneberger



```cpp
#include <bit>
#include <iostream>
#include <type_traits>
#include <utility>

namespace my_lib {

template<class Enum>
constexpr std::underlying_type_t<Enum> to_underlying(Enum e) {
    return static_cast<std::underlying_type_t<Enum>>(e);
}

inline void PrintEndianness() {
    auto L = to_underlying(std::endian::little);
    auto B = to_underlying(std::endian::big);
    auto N = to_underlying(std::endian::native);

    std::cout << "little: " << L << "\nbig: " << B << "\nnative: " << N;
}

} //namespace my_lib

int main() {
    my_lib::PrintEndianness();
}
```

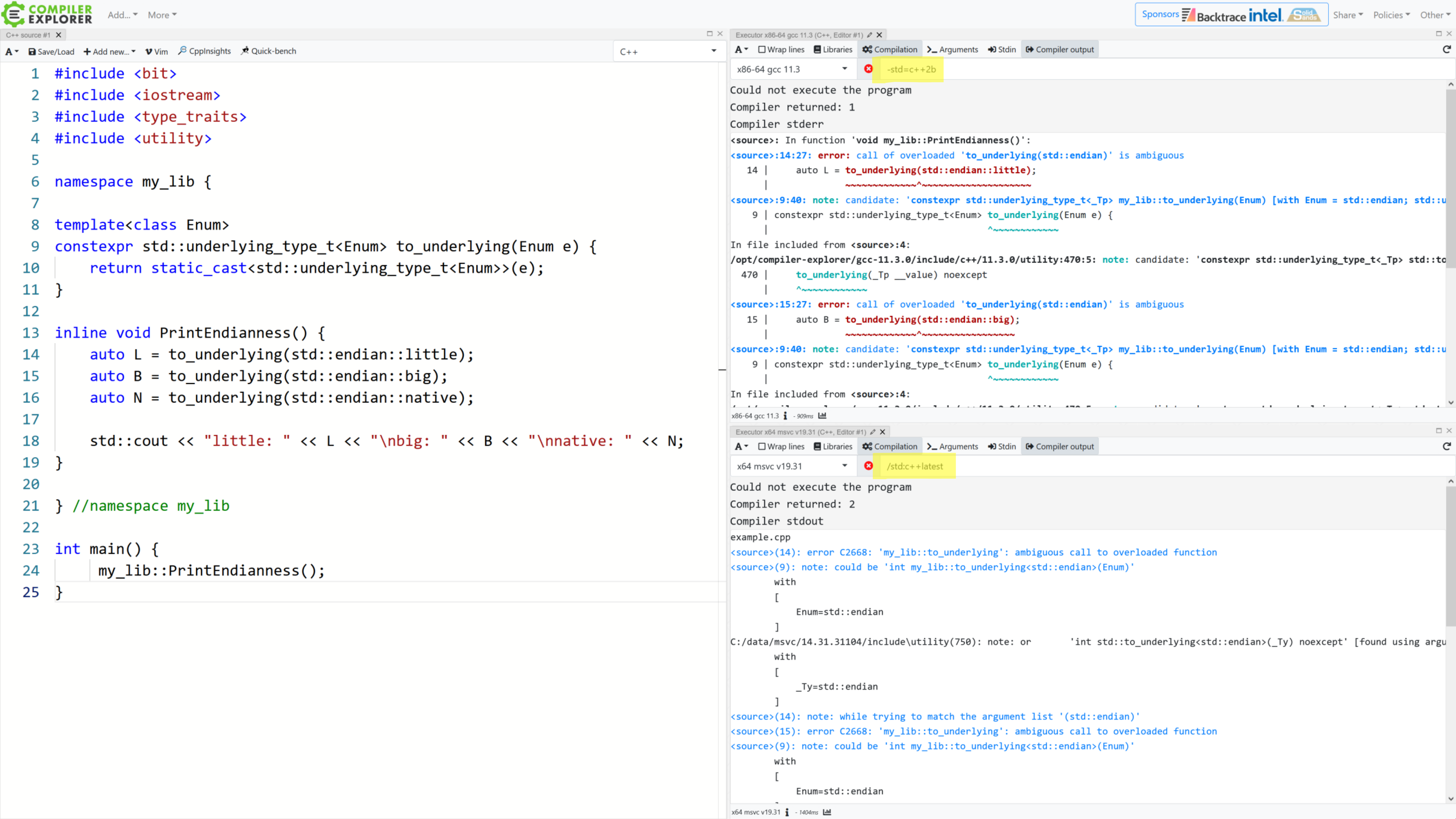
x86-64 gcc 11.3 -std=c++20

```
Program returned: 0
Program stdout
little: 1234
big: 4321
native: 1234
```

x64 msvc v19.31 /std:c++20

```
Compiler stdout
example.cpp

Program returned: 0
Program stdout
little: 0
big: 1
native: 0
```



```cpp
 1 #include <bit>
 2 #include <iostream>
 3 #include <type_traits>
 4 #include <utility>
 5
 6 namespace my_lib {
 7
 8 template<class Enum>
 9 constexpr std::underlying_type_t<Enum> to_underlying(Enum e) {
10     return static_cast<std::underlying_type_t<Enum>>(e);
11 }
12
13 inline void PrintEndianness() {
14     auto L = to_underlying(std::endian::little);
15     auto B = to_underlying(std::endian::big);
16     auto N = to_underlying(std::endian::native);
17
18     std::cout << "little: " << L << "\nbig: " << B << "\nnative: " << N;
19 }
20
21 } //namespace my_lib
22
23 int main() {
24     my_lib::PrintEndianness();
25 }
```

Executor x86-64 gcc 11.3 (C++, Editor #1)

x86-64 gcc 11.3 -std=c++2b

```
Could not execute the program
Compiler returned: 1
Compiler stderr
<source>: In function 'void my_lib::PrintEndianness()':
<source>:14:27: error: call of overloaded 'to_underlying(std::endian)' is ambiguous
   14 |     auto L = to_underlying(std::endian::little);
      |              ~~~~~~~~~~~~~^~~~~~~~~~~~~~~~~~~~~
<source>:9:40: note: candidate: 'constexpr std::underlying_type_t<_Tp> my_lib::to_underlying(Enum) [with Enum = std::endian; std::u
    9 | constexpr std::underlying_type_t<Enum> to_underlying(Enum e) {
      |                                        ^~~~~~~~~~~~~
In file included from <source>:4:
/opt/compiler-explorer/gcc-11.3.0/include/c++/11.3.0/utility:470:5: note: candidate: 'constexpr std::underlying_type_t<_Tp> std::to
  470 |     to_underlying(_Tp __value) noexcept
      |     ^~~~~~~~~~~~~
<source>:15:27: error: call of overloaded 'to_underlying(std::endian)' is ambiguous
   15 |     auto B = to_underlying(std::endian::big);
      |              ~~~~~~~~~~~~~^~~~~~~~~~~~~~~~~~
<source>:9:40: note: candidate: 'constexpr std::underlying_type_t<_Tp> my_lib::to_underlying(Enum) [with Enum = std::endian; std::u
    9 | constexpr std::underlying_type_t<Enum> to_underlying(Enum e) {
      |                                        ^~~~~~~~~~~~~
In file included from <source>:4:
```

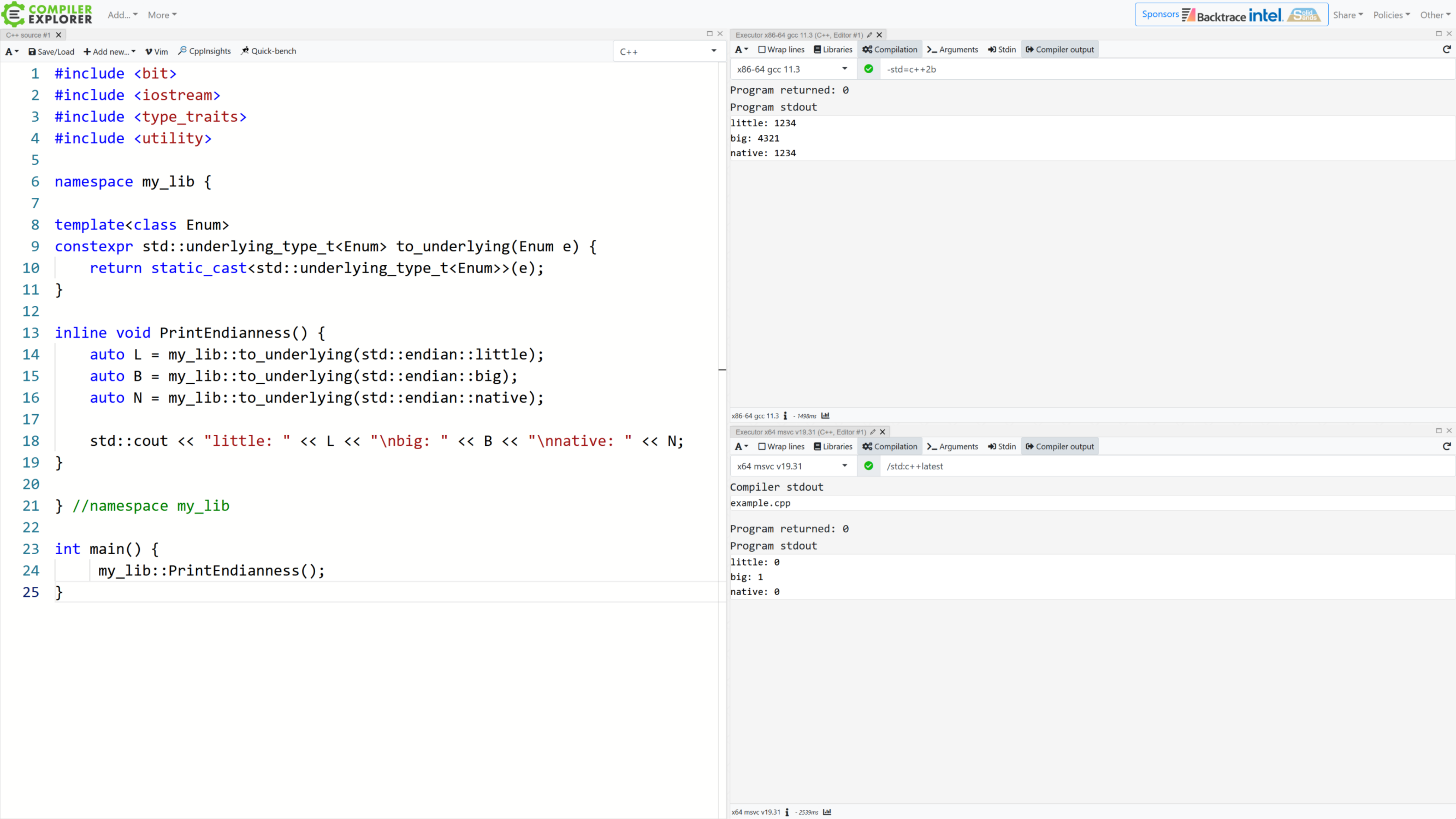
x86-64 gcc 11.3 - 909ms

Executor x64 msvc v19.31 (C++, Editor #1)

x64 msvc v19.31 /std:c++latest

```
Could not execute the program
Compiler returned: 2
Compiler stdout
example.cpp
<source>(14): error C2668: 'my_lib::to_underlying': ambiguous call to overloaded function
<source>(9): note: could be 'int my_lib::to_underlying<std::endian>(Enum)'
        with
        [
            Enum=std::endian
        ]
C:/data/msvc/14.31.31104/include\utility(750): note: or       'int std::to_underlying<std::endian>(_Ty) noexcept' [found using argu
        with
        [
            _Ty=std::endian
        ]
<source>(14): note: while trying to match the argument list '(std::endian)'
<source>(15): error C2668: 'my_lib::to_underlying': ambiguous call to overloaded function
<source>(9): note: could be 'int my_lib::to_underlying<std::endian>(Enum)'
        with
        [
            Enum=std::endian
```

x64 msvc v19.31 - 1404ms



```cpp
#include <bit>
#include <iostream>
#include <type_traits>
#include <utility>

namespace my_lib {

template<class Enum>
constexpr std::underlying_type_t<Enum> to_underlying(Enum e) {
    return static_cast<std::underlying_type_t<Enum>>(e);
}

inline void PrintEndianness() {
    auto L = my_lib::to_underlying(std::endian::little);
    auto B = my_lib::to_underlying(std::endian::big);
    auto N = my_lib::to_underlying(std::endian::native);

    std::cout << "little: " << L << "\nbig: " << B << "\nnative: " << N;
}

} //namespace my_lib

int main() {
     my_lib::PrintEndianness();
}
```

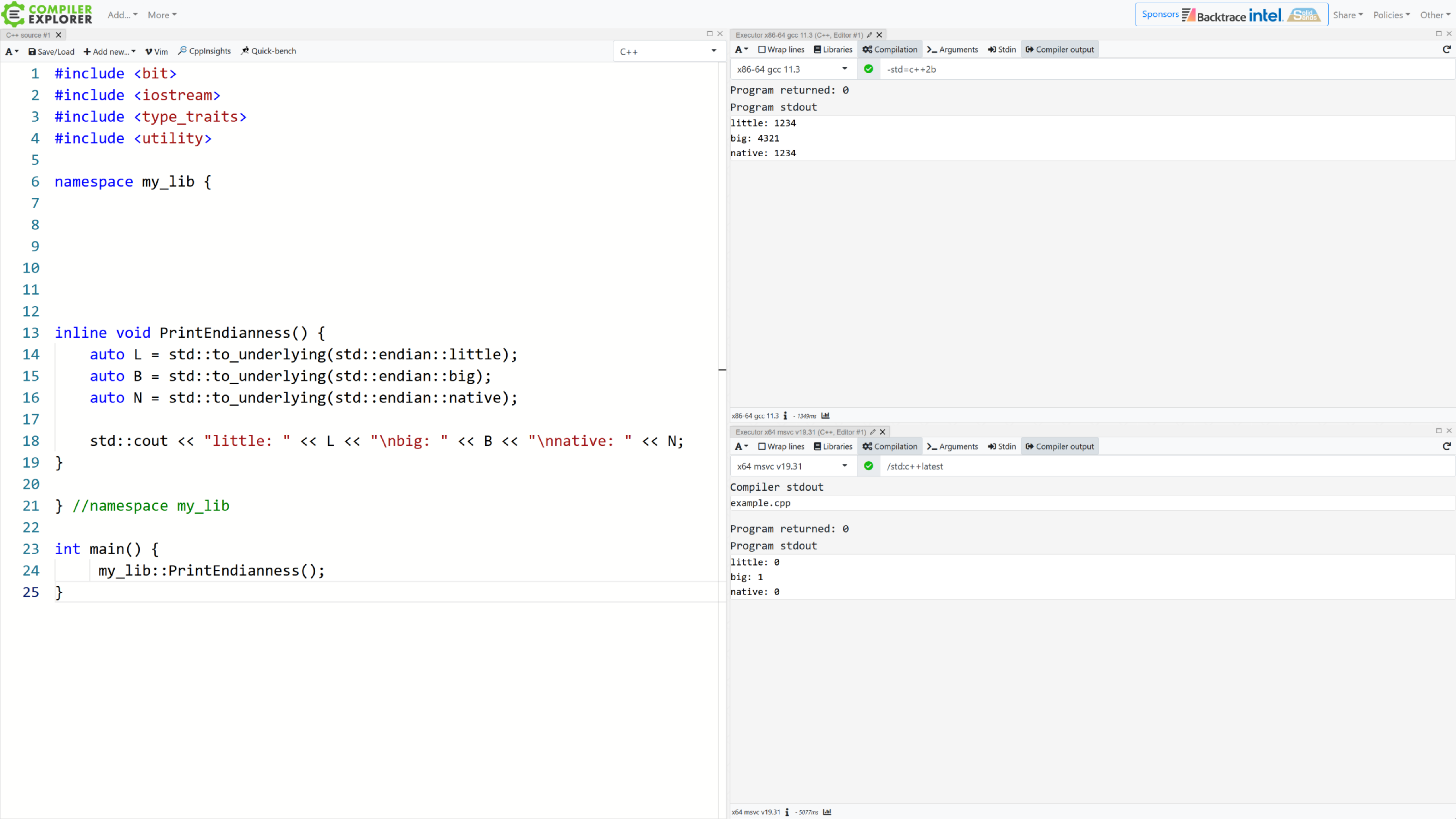
x86-64 gcc 11.3 -std=c++2b

```
Program returned: 0
Program stdout
little: 1234
big: 4321
native: 1234
```

x64 msvc v19.31 /std:c++latest

```
Compiler stdout
example.cpp

Program returned: 0
Program stdout
little: 0
big: 1
native: 0
```



```cpp
#include <bit>
#include <iostream>
#include <type_traits>
#include <utility>

namespace my_lib {







inline void PrintEndianness() {
    auto L = std::to_underlying(std::endian::little);
    auto B = std::to_underlying(std::endian::big);
    auto N = std::to_underlying(std::endian::native);

    std::cout << "little: " << L << "\nbig: " << B << "\nnative: " << N;
}

} //namespace my_lib

int main() {
     my_lib::PrintEndianness();
}
```

x86-64 gcc 11.3 -std=c++2b

```
Program returned: 0
Program stdout
little: 1234
big: 4321
native: 1234
```

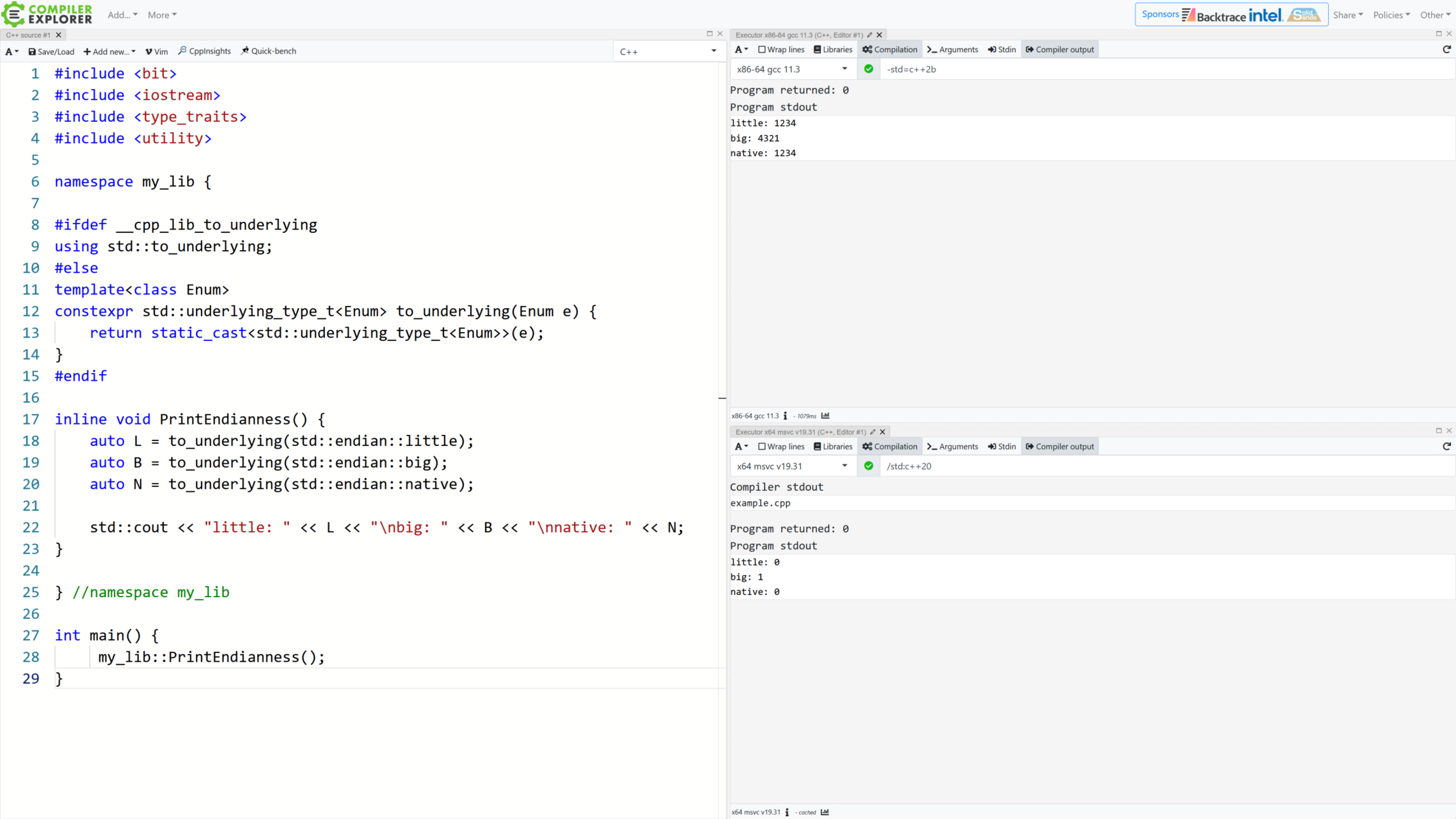
x86-64 gcc 11.3 - 1349ms

x64 msvc v19.31 /std:c++latest

```
Compiler stdout
example.cpp

Program returned: 0
Program stdout
little: 0
big: 1
native: 0
```

x64 msvc v19.31 - 5077ms



C++ source #1

C++

```
#include <bit>
#include <iostream>
#include <type_traits>
#include <utility>

namespace my_lib {

#ifdef __cpp_lib_to_underlying
using std::to_underlying;
#else
template<class Enum>
constexpr std::underlying_type_t<Enum> to_underlying(Enum e) {
    return static_cast<std::underlying_type_t<Enum>>(e);
}
#endif

inline void PrintEndianness() {
    auto L = to_underlying(std::endian::little);
    auto B = to_underlying(std::endian::big);
    auto N = to_underlying(std::endian::native);

    std::cout << "little: " << L << "\nbig: " << B << "\nnative: " << N;
}

} //namespace my_lib

int main() {
     my_lib::PrintEndianness();
}
```

Executor x86-64 gcc 11.3 (C++, Editor #1)

x86-64 gcc 11.3

-std=c++2b

```
Program returned: 0
Program stdout
little: 1234
big: 4321
native: 1234
```

x86-64 gcc 11.3 - 1079ms

Executor x64 msvc v19.31 (C++, Editor #1)

x64 msvc v19.31

/std:c++20

```
Compiler stdout
example.cpp

Program returned: 0
Program stdout
little: 0
big: 1
native: 0
```

x64 msvc v19.31 - cached

Conclusion

Every unqualified function call, involving an argument type from another namespace, might lead to a different overload set at some point in time.