Chapter 6. Boost.LexicalCast

Boost.LexicalCast provides a cast operator, `boost::lexical_cast`, that can convert numbers from strings to numeric types like `int` or `double` and vice versa. `boost::lexical_cast` is an alternative to functions like `std::stoi()`, `std::stod()`, and `std::to_string()`, which were added to the standard library in C++11.

Example 6.1. Using `boost::lexical_cast`

```
#include <boost/lexical_cast.hpp>
#include <string>
#include <iostream>

int main()
{
  std::string s = boost::lexical_cast<std::string>(123);
  std::cout << s << '\n';
  double d = boost::lexical_cast<double>(s);
  std::cout << d << '\n';
}
```

The cast operator `boost::lexical_cast` can convert numbers of different types. Example 6.1 first converts the integer 123 to a string, then converts the string to a floating point number. To use `boost::lexical_cast`, include the header file `boost/lexical_cast.hpp`.

`boost::lexical_cast` uses streams internally to perform the conversion. Therefore, only types with overloaded `operator<<` and `operator>>` can be converted. However, `boost::lexical_cast` can be optimized for certain types to implement a more efficient conversion.

Example 6.2. `boost::bad_lexical_cast` in case of an error

```
#include <boost/lexical_cast.hpp>
#include <string>
#include <iostream>

int main()
{
  try
  {
    int i = boost::lexical_cast<int>("abc");
    std::cout << i << '\n';
  }
  catch (const boost::bad_lexical_cast &e)
  {
    std::cerr << e.what() << '\n';
  }
}
```

If a conversion fails, an exception of type `boost::bad_lexical_cast`, which is derived from `std::bad_cast`, is thrown. Example 6.2 throws an exception because the string “abc” cannot be converted to a number of type `int`.