Chapter 54. Boost.Conversion

Boost.Conversion defines the cast operators `boost::polymorphic_cast` and `boost::polymorphic_downcast` in the header file `boost/cast.hpp`. They are designed to handle type casts – usually done with `dynamic_cast` – more precisely.

Example 54.1. Down and cross casts with `dynamic_cast`

```
struct base1 { virtual ~base1() = default; };
struct base2 { virtual ~base2() = default; };
struct derived : public base1, public base2 {};

void downcast(base1 *b1)
{
  derived *d = dynamic_cast<derived*>(b1);
}

void crosscast(base1 *b1)
{
  base2 *b2 = dynamic_cast<base2*>(b1);
}

int main()
{
  derived *d = new derived;
  downcast(d);

  base1 *b1 = new derived;
  crosscast(b1);
}
```

Example 54.1 uses the cast operator `dynamic_cast` twice: In `downcast()`, it transforms a pointer pointing to a base class to one pointing to a derived class. In `crosscast()`, it transforms a pointer pointing to a base class to one pointing to a different base class. The first transformation is a *downcast*, and the second is a *cross cast*. The cast operators from Boost.Conversion let you distinguish a downcast from a cross cast.

Example 54.2. Down and cross casts with `polymorphic_downcast` and `polymorphic_cast`

```
#include <boost/cast.hpp>

struct base1 { virtual ~base1() = default; };
struct base2 { virtual ~base2() = default; };
struct derived : public base1, public base2 {};

void downcast(base1 *b1)
{
  derived *d = boost::polymorphic_downcast<derived*>(b1);
}

void crosscast(base1 *b1)
{
  base2 *b2 = boost::polymorphic_cast<base2*>(b1);
}

int main()
{
  derived *d = new derived;
  downcast(d);

  base1 *b1 = new derived;
  crosscast(b1);
}
```

boost::polymorphic_downcast (see Example 54.2) can only be used for downcasts because it uses static_cast to perform the cast. Because static_cast does not dynamically check the cast for validity, boost::polymorphic_downcast must only be used if the cast is safe. In debug builds, boost::polymorphic_downcast uses dynamic_cast and assert() to make sure the type cast is valid. This test is only performed if the macro NDEBUG is not defined, which is usually the case for debug builds.

boost::polymorphic_cast is required for cross casts. boost::polymorphic_cast uses dynamic_cast, which is the only cast operator that can perform a cross cast. It is better to use boost::polymorphic_cast instead of dynamic_cast because the former throws an exception of type std::bad_cast in case of an error, while dynamic_cast returns a null pointer if the type cast fails.

Use boost::polymorphic_downcast and boost::polymorphic_cast only to convert pointers; otherwise, use dynamic_cast. Because boost::polymorphic_downcast is based on static_cast, it cannot convert objects of a base class to objects of a derived class. Also, it does not make sense to use boost::polymorphic_cast to convert types other than pointers because dynamic_cast will throw an exception of type std::bad_cast if a cast fails.