Chapter 28. Boost.CompressedPair

Boost.CompressedPair provides `boost::compressed_pair`, a class that behaves like `std::pair`. However, if one or both template parameters are empty classes, `boost::compressed_pair` consumes less memory. `boost::compressed_pair` uses a technique known as empty base class optimization.

To use `boost::compressed_pair`, include the header file `boost/compressed_pair.hpp`.

Example 28.1. Reduced memory requirements with `boost::compressed_pair`

```
#include <boost/compressed_pair.hpp>
#include <utility>
#include <iostream>

struct empty {};

int main()
{
  std::pair<int, empty> p;
  std::cout << sizeof(p) << '\n';

  boost::compressed_pair<int, empty> cp;
  std::cout << sizeof(cp) << '\n';
}
```

Example 28.1 illustrates this by using `boost::compressed_pair` for **cp** and `std::pair` for **p**. When compiled using Visual C++ 2013 and run on a 64-bit Windows 7 system, the example returns 4 for `sizeof(cp)` and 8 for `sizeof(p)`.

Please note that there is another difference between `boost::compressed_pair` and `std::pair`: the values stored in `boost::compressed_pair` are accessed through the member functions `first()` and `second()`. `std::pair` uses two identically named member variables instead.