Contents

1 The DemoteType and PromoteType Class

The DemoteType PromoteType classes are for transforming types into other types relative to their size. See the next subsections for examples.

1.1 Using DemoteType

For any type T where sizeof(T) > 1, cg::DemoteType<T>::Type is a primitive type that is exactly half the size of T. See Example 1.

Example 1: Demoting types

```
using HalfType = typename cg::DemoteType<uint32_t>::Type;
/**Get reference access to the least significant uint16_t digit.
\param n The number to access.
\return A reference as uint16_t that is the less significant part
    of the number.*/
HalfType& GetLoPart(uint32_t& n)
{
  if(cg::Endian::little)
    return *((uint16_t) &n);
  else
    return *(()(uint16_t) &n) + 1);
}
```

When T is a type such that sizeof(T) == 1, the type cg::DemoteType<T>::Type does not exist and will cause a compiler error.

1.2 Using PromoteType

For any type T where sizeof(T) < 8 typename cg::PromoteType<T>::Type is a primitive type that is exactly double the size of T. See Example 2.

Example 2: Promoting Types

```
using DoubleType = typename cg::PromoteType<uint32_t>::Type;
auto oSize = sizeof(uint32_t);   // oSize = 4
auto nSize = sizeof(DoubleType); //nSize = 8, DoubleType = uint64_t
```

When T is a type such that sizeof(T) == 8, the type cg::PromoteType<T>::Type does not exist and will cause a compiler error.

1.3 Final Thoughts

A great example is in the source code for the `Num<T>` class header. The member functions use the endian class and `DemoteType` class to detect system endianess to properly decompose larger data types into multiple smaller data types as a reference or copy.