So, calling `swap()` instead of swapping the values directly might result in substantial performance improvements. You should always offer a specialization of `swap()` for your own types if doing so has performance advantages.

4.5 Supplementary Comparison Operators

Four template functions define the comparison operators `!=`, `>`, `<=`, and `>=` by calling the operators `==` and `<`. These functions are defined in `<utility>` as follows:

```
namespace std {
    namespace rel_ops {
        template <class T>
        inline bool operator!= (const T& x, const T& y) {
            return !(x == y);
        }

        template <class T>
        inline bool operator> (const T& x, const T& y) {
            return y < x;
        }

        template <class T>
        inline bool operator<= (const T& x, const T& y) {
            return !(y < x);
        }

        template <class T>
        inline bool operator>= (const T& x, const T& y) {
            return !(x < y);
        }
    }
}
```

To use them, you only need to define operators `<` and `==`. By using namespace `std::rel_ops`, the other comparison operators are defined automatically. For example:

```
#include <utility>

class X {
    ...
  public:
```