```
std::atomic<double> b;
std::atomic<long> c;
std::atomic<int> d;
std::atomic<short> e;
std::atomic<char> f;
std::atomic<long long> g;

std::cout << "double is atomic = "<< std::boolalpha <<  b.is_lock_free() << std::endl;
std::cout << "long is atomic = "<< std::boolalpha <<  c.is_lock_free() << std::endl;
std::cout << "int is atomic = "<< std::boolalpha <<  d.is_lock_free() << std::endl;
std::cout << "short is atomic = "<< std::boolalpha <<  e.is_lock_free() << std::endl;
std::cout << "char is atomic = "<< std::boolalpha <<  f.is_lock_free() << std::endl;
std::cout << "long long is atomic = "<< std::boolalpha <<  g.is_lock_free() <<
std::endl;
```

```
clang++ -std=c++14 atomic.cpp
nik@Nicolas-MacBook-Air:~/GitHub/cpp_sandbox/multithreading/thread_sync$ ./a.out
double is atomic = true
long is atomic = true
int is atomic = true
short is atomic = true
char is atomic = true
long long is atomic = true
```

```cpp
std::atomic<double> b;
std::atomic<long> c;
std::atomic<int> d;
std::atomic<short> e;
std::atomic<char> f;
std::atomic<long long> g;

std::cout << "double is atomic = "<< std::boolalpha <<  b.is_lock_free() << std::endl;
std::cout << "long is atomic = "<< std::boolalpha <<  c.is_lock_free() << std::endl;
std::cout << "int is atomic = "<< std::boolalpha <<  d.is_lock_free() << std::endl;
std::cout << "short is atomic = "<< std::boolalpha <<  e.is_lock_free() << std::endl;
std::cout << "char is atomic = "<< std::boolalpha <<  f.is_lock_free() << std::endl;
std::cout << "long long is atomic = "<< std::boolalpha <<  g.is_lock_free() <<
std::endl;
```

```
clang++ -std=c++14 atomic.cpp
nik@Nicolas-MacBook-Air:~/GitHub/cpp_sandbox/multithreading/thread_sync$ ./a.out
double is atomic = true
long is atomic = true
int is atomic = true
short is atomic = true
char is atomic = true
long long is atomic = true
```

```cpp
std::atomic<int> ay{0};
int x = 0;

void atomic_read_barrier()
{
    std::cout << "y = " << ay.load() << std::endl;
    std::cout << "x = " << x << std::endl;
    std::cout << std::endl;
}

void atomic_write_barrier()
{
    x = 42;
    ay.store(20);
}

std::thread t2(atomic_read_barrier);
std::thread t1(atomic_write_barrier);
t1.join();
t2.join();
```