std::atomic

```
template< class T > struct atomic
template<> struct atomic<Integral>;
template< class T > struct atomic<T*>;
```

- Each instantiation and full specialization of the std::atomic template defines an atomic type.
- Objects of atomic types are the only C++ objects that are free from data races; that is
- if one thread writes to an atomic object while another thread reads from it, the behaviour is well-defined.

from http://en.cppreference.com/w/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;
```