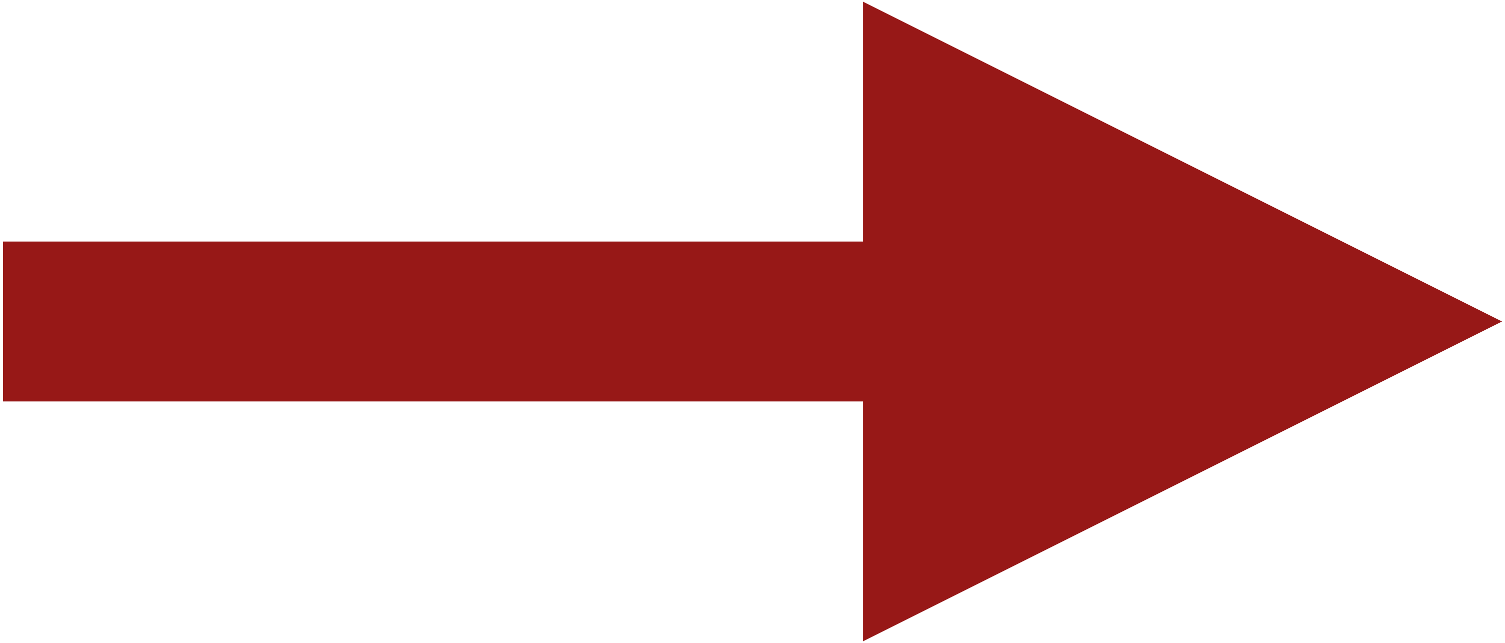
```cpp
void transfer(int from, int to, int sum)
{
        // no deadlock
        auto& acc1 = _accounts[from];
        auto& acc2 = _accounts[to];

        std::lock(acc1.get_mutex(), acc2.get_mutex());
        lock_guard lk1(acc1.get_mutex(), std::adopt_lock);
        lock_guard lk2(acc2.get_mutex(), std::adopt_lock);

        std::cout << "Moving money from = " << from
       << " to = " << to << " sum = " << sum << "\n";

        if (acc1.balance() >= sum)
        {
            acc1.deposit(-sum);
            acc2.deposit(sum);
        }
 }
```



```cpp
void transfer(int from, int to, int sum)
{
        // no deadlock
        auto& acc1 = _accounts[from];
        auto& acc2 = _accounts[to];

        std::lock(acc1.get_mutex(), acc2.get_mutex());
        lock_guard lk1(acc1.get_mutex(), std::adopt_lock);
        lock_guard lk2(acc2.get_mutex(), std::adopt_lock);

        std::cout << "Moving money from = " << from
       << " to = " << to << " sum = " << sum << "\n";

        if (acc1.balance() >= sum)
        {
            acc1.deposit(-sum);
            acc2.deposit(sum);
        }
 }
```

```
nik@Nicolas-MacBook-Air:~/GitHub/cpp_sandbox/multithreading/thread_sync$
clang++ -std=c++14 deadlock.cpp
nik@Nicolas-MacBook-Air:~/GitHub/cpp_sandbox/multithreading/thread_sync$
./a.out
Moving money from = 0 to = 1 sum = 10
Moving money from = 1 to = 0 sum = 10
Program end ... Balance 0 = 100 - Balance 1 = 0
nik@Nicolas-MacBook-Air:~/GitHub/cpp_sandbox/multithreading/thread_sync$
```