**Отчёт**

**Лабораторная работа №7**

**«Самодельный mutex»**

**Системы Реального Времени**

*выполнил студент 3 курса МО*

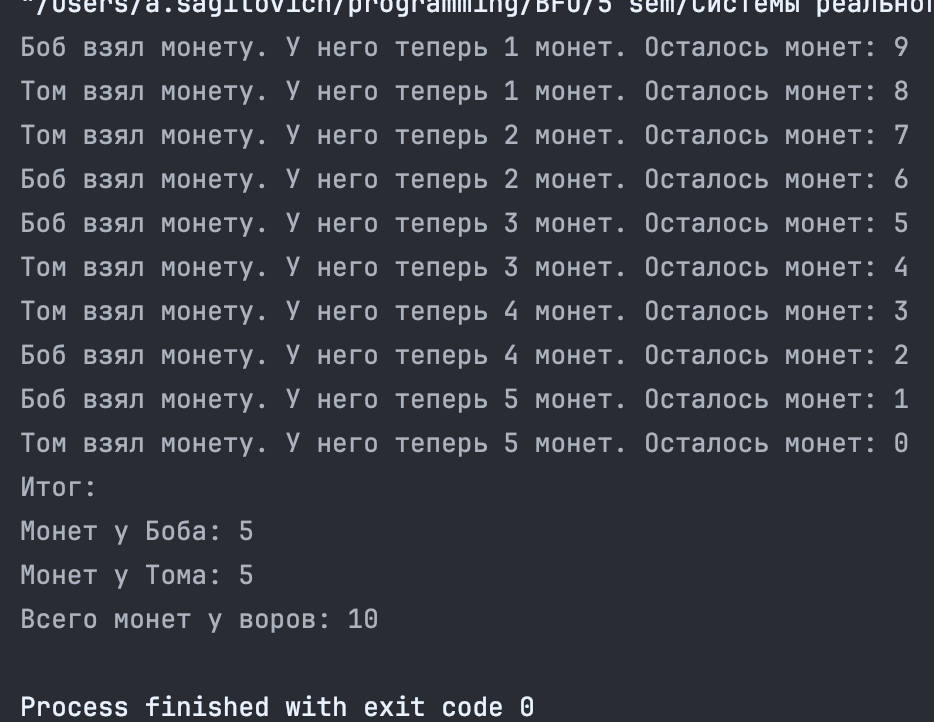
*Сагитов Александр Александрович*

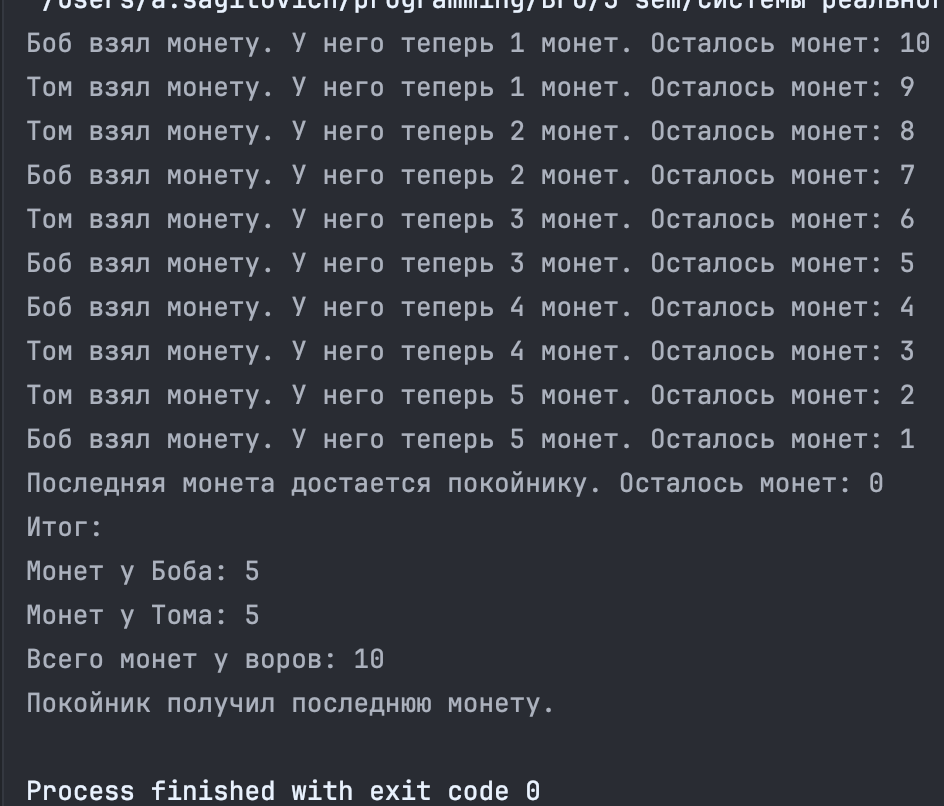
После выполнения лабораторной работы №7 был успешно реализован самодельный класс mutex.

Для наглядной демонстрации его работы был видоизменён код программы для выполнения лабораторной работы №5 «1001 монета» (Приложение 1).

Ниже приведён результат выполнения программы.

P.S.: для более компактного вывода первоначальное количество монет: 10 и 11 – для первого и второго скриншота соответственно.





**Приложение 1**

*main.cpp*

#include <iostream>  
#include <thread>  
#include <atomic>  
#include <string>  
  
class MyFavoriteMutex {  
  
 std::atomic<bool> locked{false};  
  
 public:  
 void lock() {  
 bool flag = false;  
 while (!locked.compare\_exchange\_strong(flag, true, std::memory\_order\_acquire)) {  
 flag = false;  
 }  
 }  
  
 void unlock() {  
 locked.store(false, std::memory\_order\_release);  
 }  
};  
  
int coins = 11; // общее количество монет  
bool flag = false;  
int Bob\_coins = 0; // монеты Боба  
int Tom\_coins = 0; // монеты Тома  
MyFavoriteMutex mudex; // заменяем std::mutex на CustomMutex  
  
void coin\_sharing(const std::string& name, int& thief\_coins, const int& companion\_coins) {  
 while (true) {  
 mudex.lock();  
  
 if (coins == 1 && Bob\_coins == Tom\_coins) {  
 coins--;  
 std::cout << "Последняя монета достается покойнику. Осталось монет: " << coins << std::endl;  
 mudex.unlock();  
 break;  
 }  
  
 if (coins > 0 && thief\_coins <= companion\_coins) {  
 thief\_coins++;  
 coins--;  
 std::cout << name << " взял монету. У него теперь "  
 << thief\_coins << " монет. Осталось монет: "  
 << coins << std::endl;  
 mudex.unlock();  
 } else {  
 mudex.unlock();  
 if (coins == 0) {  
 break;  
 }  
 }  
 }  
}  
  
int main() {  
 if (coins % 2 != 0) {  
 flag = true;  
 }  
  
 std::thread bob\_thread(coin\_sharing, "Боб", std::ref(Bob\_coins), std::ref(Tom\_coins));  
 std::thread tom\_thread(coin\_sharing, "Том", std::ref(Tom\_coins), std::ref(Bob\_coins));  
  
 bob\_thread.join();  
 tom\_thread.join();  
  
 std::cout << "Итог:\n"  
 << "Монет у Боба: " << Bob\_coins << "\n"  
 << "Монет у Тома: " << Tom\_coins << "\n"  
 << "Всего монет у воров: " << (Bob\_coins + Tom\_coins) << "\n";  
 if (flag) {  
 std::cout << "Покойник получил последнюю монету." << std::endl;  
 }  
  
 return 0;  
}