Матюков Николай Юрьевич ПС-31

Лабораторная работа №2

**Постановка задачи.**

В качестве примера дано приложение, эмулирующее работу банка. Клиенты периодически изменяют свой баланс методом UpdateClientBalance, в котором между считыванием общего баланса и записью обновленного значения выполняются длительные операции.

Задание: 1. Сделать fork репозитория с примером.

2. Выполнить все TODO. Быстрый поиск TODO в Visual Studio: View -> Task List.

i. TODO в PP2.cpp:

· реализовать создание необходимого количества экземпляров класса CBankClient в зависимости от значения, переданного в параметре командной строки (по умолчанию - 2);

· реализовать условие выхода из программы при вводе с клавиатуры ключевой фразы (quit или exit) и вывести на консоль статистику работы программы (баланс по каждому клиенту, сумму балансов клиентов, баланс банка);

ii. TODO в Bank.cpp:

· в методе SomeLongOperations() реализовать задержку длительностью 1 секунда;

iii. TODO в BankClient.cpp:

· в методе ThreadFunction(LPVOID lpParam) инициализировать генератор псевдослучайных чисел;

· в методе GetSleepDuration(CBankClient \*client) реализовать вычисление случайной продолжительности блокировки потока от 1 до 3 секунд.

3. Изучить код приложения, выяснить причины, почему появляется сообщение об ошибке “! ERROR !”. Отразить это в выводах отчета.

4. Исправить некорректное поведение приложения на основе выводов из пункта 3 (появление сообщения “ERROR” в консоли).

Для реализации использовать различные примитивы синхронизации:

i. Критическая секция

ii. Мьютекс

Использование примитива в runtime должно определяться параметром приложения. Одно из значений ключей должно блокировать использование всех примитивов синхронизации.

5. Приложение должно выдавать справку по своему использованию при запуске с ключом /? .

Каждый пункт задания – как минимум один отдельный коммит (2, 3, 4, 5).

**Выводы.**

Error возникал в методе UpdateClientBalance потому что к totalBalance имели доступ сразу несколько потоков, поэтому когда один поток запрашивает данные о балансе, в это время другой поток его обновляет. Для этого используется критическая секция или mutex. Тем самым при работе программе уже один поток работает с переменной.