Лабораторная работа №5.

Примитивы синхронизации. CRITICAL SECTION.

Цели и задачи

Познакомиться с работой примитива синхронизации CRITICAL SECTION, MUTEX.

Постановка задачи

Написать утилиту, которая могла бы запускать одновременно два инстанса приложения. (тут можно по разному решить вопрос, можно батник написать, можно каким то образом синхронизировать начало выполнения программ).

Необходимо найти файл LAB\_5.cpp в папке с лабораторной и разобраться в ходе ее работы.

Запустить приложение в одном и двух экземплярах и увидеть, что программа работает с логическими ошибками, баланс пользователя не может быть в минусе, но это происходит.

Разобраться для чего необходим “CRITICAL\_SECTION FileLockingCriticalSection” и к чему может привести его отсутствие.

Задачи:

1. Синхронизировать при помощи CRITICAL\_SECTION участки кода, которые приведут к логически верному результату. Запустить приложение в единственном экземпляре, посмотреть на вывод. Запустить приложение в двух экземплярах, посмотреть на вывод.
2. Синхронизировать при помощи MUTEX участки кода, которые приведут к логически верному результату. Запустить приложение в единственном экземпляре, посмотреть на вывод. Запустить приложение в двух экземплярах, посмотреть на вывод.

Требования

1. Программа должна быть реализована в виде консольного приложения и использовать Windows SDK, язык программирования – С++.
2. Сохранить текстовые файлы результатов выполнения программы без синхронизаций и с синхронизациями для удобства защиты лабораторной. Можно через код, можно просто копируя с консоли результат.
3. Рассказать каким подходом пользовались при выполнении задания для обнаружения участков, которые необходимо синхронизировать.
4. Объяснить разницу между CRITICAL\_SECTION и MUTEX. Рассказать в каких ситуациях они должны использоваться.
5. Объяснить правильно ли используется FileLockingCriticalSection, нужно ли в этом случае использовать Mutex или можно обойтись CRITICAL\_SECTION.