Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей Кафедра информатики

Дисциплина: Операционные среды и системное программирование

**ОТЧЕТ**

к лабораторной работе № 6

на тему «Средства синхронизации и взаимного исключения (Windows). Изучение и использование средств синхронизации и взаимного исключения»

Выполнил:

студент гр. 153504

Прескурел Я.Ю.

Проверил:

Гриценко Н.Ю.

Минск 2023

СОДЕРЖАНИЕ

[1 Постановка задачи 3](#_Toc146631498)

[2 Краткие теоретические сведения 4](#_Toc146631499)

[3 Результаты выполнения лабораторной работы 6](#_Toc146631500)

[Выводы 7](#_Toc146631501)

[Список использованных источников 8](#_Toc146631502)

[Приложение А (обязательное) Листинг кода 9](#_Toc146631503)

## 1 ПОСТАНОВКА ЗАДАЧИ

Целью выполнения данной лабораторной работы является создание системы взаимного исключения для обеспечения безопасного доступа к общим ресурсам из нескольких потоков.

## 2 КРАТКИЕ ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

Win32 API (Windows API) представляет собой набор функций и интерфейсов, предоставляемых операционной системой Windows для разработки приложений. Этот мощный набор инструментов обеспечивает доступ к различным функциональным возможностям Windows, включая создание и управление окнами, обработку сообщений, работу с файлами и реестром, а также многие другие операции. Win32 API играет ключевую роль в разработке приложений для Windows и обеспечивает высокую степень контроля над поведением приложений.

Средства синхронизации и взаимного исключения в операционной системе Windows играют критическую роль в обеспечении безопасности и корректности многозадачного выполнения программ. Они позволяют управлять доступом нескольких потоков или процессов к общим ресурсам, таким как общие данные, файлы, устройства или критические секции кода. Средства синхронизации обеспечивают правильную координацию выполнения задач, что предотвращает состояния гонки и гарантирует надежное выполнение программ.

Одним из наиболее распространенных средств синхронизации в Windows являются мьютексы (mutexes), семафоры (semaphores), события (events) и критические секции (critical sections). Мьютексы и семафоры обеспечивают взаимное исключение, позволяя только одному потоку (или процессу) одновременно доступ к общему ресурсу. События позволяют потокам ожидать определенных условий или уведомлений от других потоков. Критические секции предоставляют легкий механизм блокировки для обеспечения безопасности доступа к общим данным.

При разработке многозадачных приложений под Windows, правильное использование средств синхронизации становится критически важным. Это позволяет избегать состояний гонки, повышать производительность, улучшать отзывчивость и обеспечивать надежность программ. Ошибки в синхронизации могут привести к непредсказуемым результатам и сбоям, поэтому понимание и использование этих средств является ключевой компетенцией для разработчиков под Windows.

Для выполнения данной лабораторной работы, были использованы следующие теоретические сведения и концепции:

1. Потоки (Threads): при написании кода были созданы и использованы потоки с помощью функций Windows API, таких как CreateThread и WaitForMultipleObjects. Это позволяет выполнить задачи параллельно, улучшая производительность и эффективность программы.

2. Синхронизация доступа к общим данным: для обеспечения безопасности при работе с общими данными используется критическая секция (CRITICAL\_SECTION). Функции EnterCriticalSection и LeaveCriticalSection используются для захвата и освобождения критической секции. Это предотвращает одновременный доступ нескольких потоков к общим данным и исключает конфликты при изменении.

3. Работа с дескрипторами: программа управляет дескрипторами потоков и критической секцией с помощью функций CloseHandle.

## 3 РЕЗУЛЬТАТЫ ВЫПОЛНЕНИЯ ЛАБОРАТОРНОЙ РАБОТЫ

В ходе выполнения лабораторной работы была создана система взаимного исключения для обеспечения безопасного доступа к общим ресурсам из нескольких потоков. Результат работы программы предоставлен на рисунке 3.1.

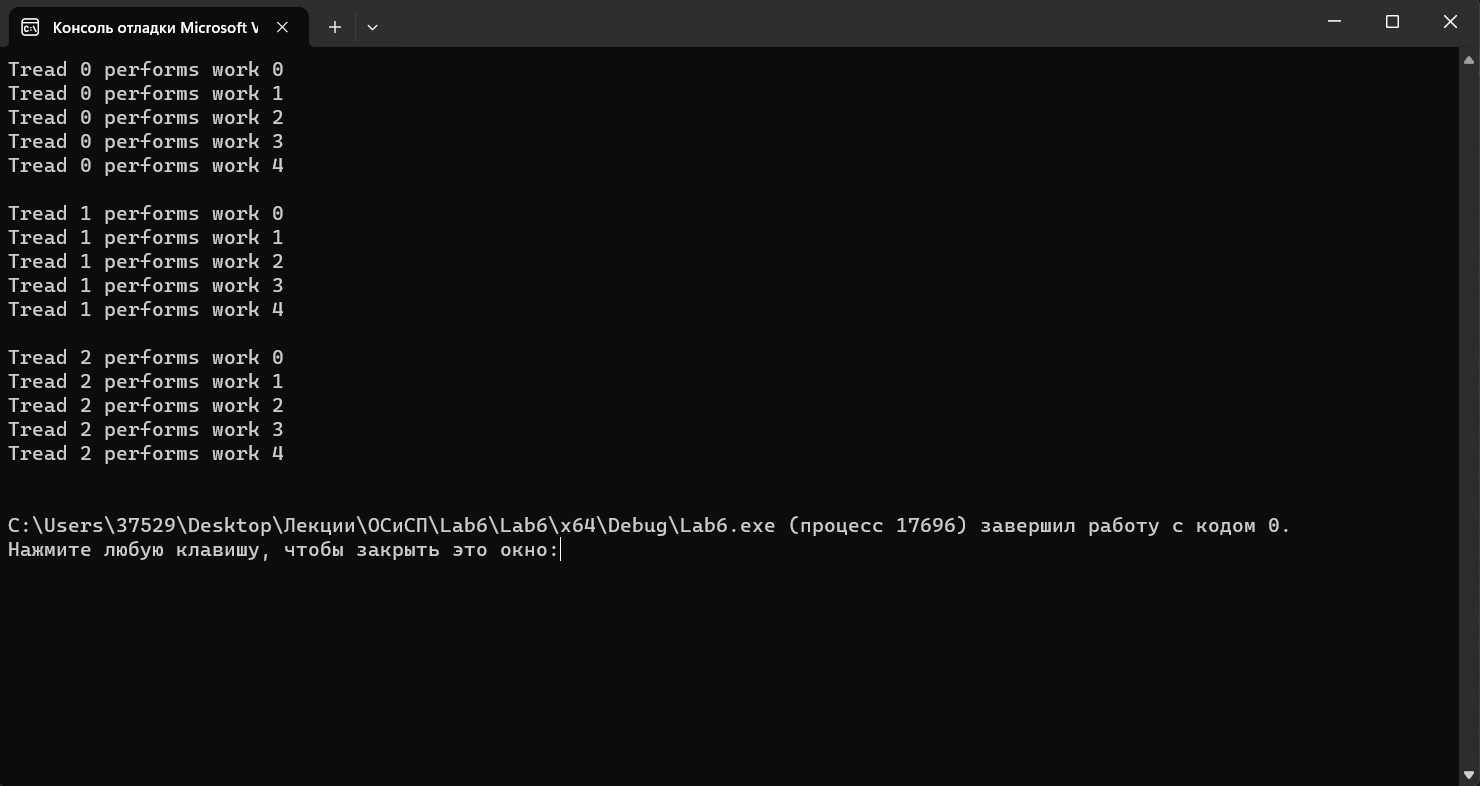


Рисунок 3.1 – Результат работы программы

## ВЫВОДЫ

В ходе выполнения данной лабораторной работы была создана простая система взаимного исключения для обеспечения безопасного доступа к общим ресурсам из нескольких потоков.

## СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Щупак Ю. Win32 API. Разработка приложений для Windows. – СПб: Питер, 2008. – 592 с.: ип.
2. [Электронный ресурс]. – Режим доступа: [https://learn.microsoft.com/ru-ru/windows/win32/procthread/process-and-thread-functions – Дата доступа 24.10.2023](https://learn.microsoft.com/ru-ru/windows/win32/procthread/process-and-thread-functions%20–%20Дата%20доступа%2024.10.2023)

**ПРИЛОЖЕНИЕ А**

## (обязательное)

## Листинг кода

**Lab6.cpp**

#include <iostream>

#include <windows.h>

CRITICAL\_SECTION cs; // Критическая секция для взаимного исключения

void worker(int id)

{

for (int i = 0; i < 5; i++)

{

EnterCriticalSection(&cs); // Захватываю критическую секцию

std::cout << "Thread " << id << " performs work " << i << std::endl; // Критическая секция

LeaveCriticalSection(&cs); // Освобождаю критическую секцию

}

std::cout << "\n";

}

int main()

{

const int num\_threads = 3;

HANDLE threads[num\_threads];

InitializeCriticalSection(&cs); // Инициализация критической секции

for (int i = 0; i < num\_threads; i++)

{

threads[i] = CreateThread(NULL, 0, (LPTHREAD\_START\_ROUTINE)worker, (LPVOID)i, 0, NULL); // Создаю потоки

}

WaitForMultipleObjects(num\_threads, threads, TRUE, INFINITE); // Жду завершения всех потоков

for (int i = 0; i < num\_threads; i++)

{

CloseHandle(threads[i]); // Закрываю дескрипторы потоков

}

DeleteCriticalSection(&cs); // Удаляю критическую секцию

return 0;

}