Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

ДИСЦИПЛИНА «Операционные системы»

Лабораторная работа № 6

Тема «Реализация монитора Хоара»

Студент Шиленков А. А.

Группа ИУ7-55Б

Оценка (баллы) _____

Преподаватель Рязанова Н.Ю.

Москва.
2020 г.

Цели и задачи

В лабораторной работе необходимо разработать многопоточное приложение, используя API ОС Windows такие как, потоки, события (event) и мьютексы (mutex). Потоки разделяют единственную глобальную переменную. Приложение реализует монитор Хоара «Читатели-писатели».

Код программы:

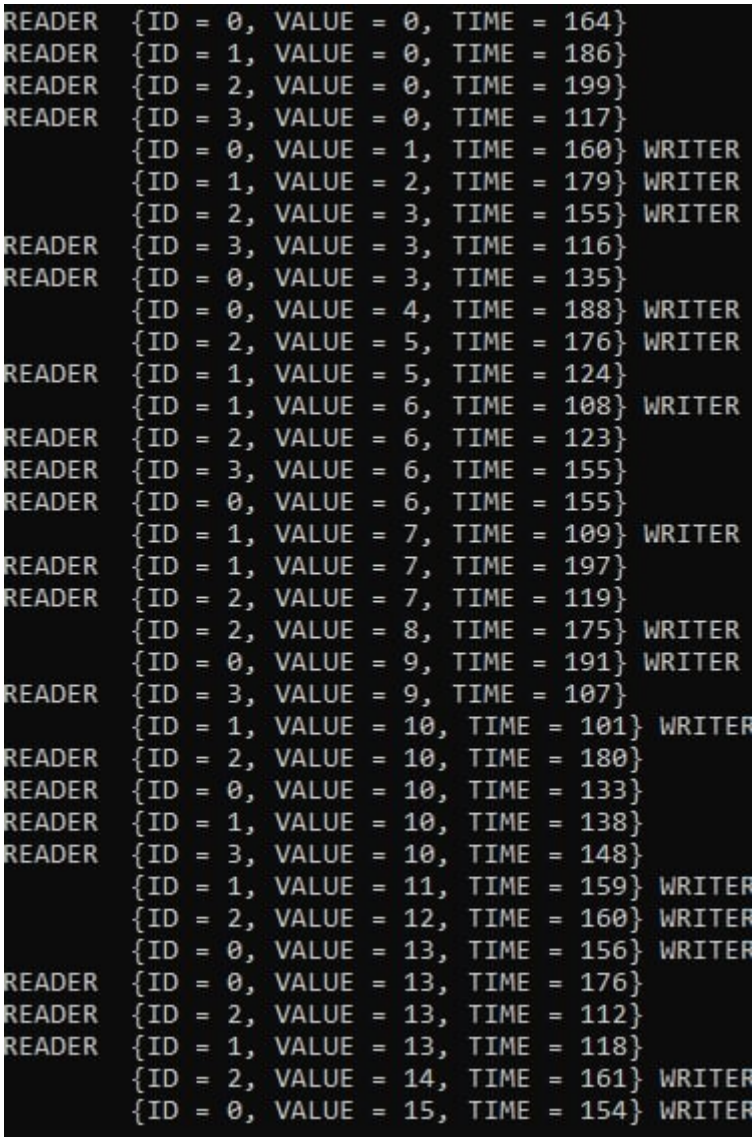
```
// Читатели писатели
#include <windows.h>
#include <stdbool.h>
#include <stdio.h>
#include <time.h>
#include <stdbool.h>

enum counts {
    CNT_ITERS = 5,
    READERS_NUMBER = 4,
    WRITERS_NUMBER = 3
};

enum errors {
    OK = 0,
    CREATE_MUTEX_ERROR =-4,
    CREATE_EVENT_ERROR,
    CREATE_READER_THREAD_ERROR,
    CREATE_WRITER_THREAD_ERROR
};

enum delays {
    MINIMUM_READER_DELAY = 100,
    MINIMUM_WRITER_DELAY = 100,
    MAXIMUM_READER_DELAY = 200,
    MAXIMUM_WRITER_DELAY = 200
};

HANDLE canRead;
HANDLE canWrite;
HANDLE mutex;
HANDLE readerThreads[READERS_NUMBER];
HANDLE writerThreads[WRITERS_NUMBER];


int readersID[READERS_NUMBER];
int writersID[WRITERS_NUMBER];

int readersRand[READERS_NUMBER * CNT_ITERS];
int writersRand[READERS_NUMBER * CNT_ITERS];

int value = 0;

LONG WWCount = 0;
LONG WRCount = 0;
LONG ARCount = 0;
bool writing = false;

void StartRead()
{
```

```
    InterlockedIncrement(&WRCount);  // WRCount++;
    // Читатель может начать, если нет активного писателя и писателей в очереди
    if (writing || WaitForSingleObject(canWrite, 0) == WAIT_OBJECT_0)
        WaitForSingleObject(canRead, INFINITE);
    WaitForSingleObject(mutex, INFINITE);
    InterlockedDecrement(&WRCount); // WRCount--;
    InterlockedIncrement(&ARCount); // ARCount++;
    SetEvent(canRead);
    ReleaseMutex(mutex);
}

void StopRead()
{
    InterlockedDecrement(&ARCount); // ARCount--;
    if (!ARCount)
        SetEvent(canWrite);
}

void StartWrite()
{
    InterlockedIncrement(&WWCount); // WW++;
    // Писатель может начать, если нет читающик читателей и нет активного писателя
    if (ARCount > 0 || writing)
        WaitForSingleObject(canWrite, INFINITE);
    InterlockedDecrement(&WWCount);  // WW--;
    writing = true;
}

void StopWrite()
{
    writing = false;
    if (WRCount)
        SetEvent(canRead);
    else
        SetEvent(canWrite);
}


DWORD WINAPI Reader(CONST LPVOID param)
{
    int id = *(int *)param;
    int sleepTime;
    int begin = id * CNT_ITERS;
    for (int i = 0; i < CNT_ITERS; i++)
    {
        sleepTime = readersRand[begin + i];
        StartRead();
        printf("READER  {ID = %d, VALUE = %d, TIME = %d}\n", id, value, sleepTime);
        StopRead();
        Sleep(sleepTime);
    }
}

DWORD WINAPI Writer(CONST LPVOID param)
{
    int id = *(int *)param;
    int sleepTime;
    int begin = id * CNT_ITERS;
    for (int i = 0; i < CNT_ITERS; i++)
    {
        sleepTime = writersRand[begin + i];

        StartWrite();
        ++value;
        printf("\t{ID = %d, VALUE = %d, TIME = %d} WRITER\n", id, value, sleepTime);
        StopWrite();
        Sleep(sleepTime);
```

```
    }
}


int main(void)
{
    setbuf(stdout, NULL);
    srand(time(NULL));

    for (int i = 0; i < READERS_NUMBER * CNT_ITERS; i++)
        readersRand[i] = rand() % (MAXIMUM_READER_DELAY - MINIMUM_READER_DELAY) +
MINIMUM_READER_DELAY;

    for (int i = 0; i < WRITERS_NUMBER * CNT_ITERS; i++)
        writersRand[i] = rand() % (MAXIMUM_WRITER_DELAY - MINIMUM_WRITER_DELAY) +
MINIMUM_WRITER_DELAY;

    if ((mutex = CreateMutex(NULL, FALSE, NULL)) == NULL)
    {
        perror("Can't create mutex!\n");
        return CREATE_MUTEX_ERROR;
    }
    if ((canRead = CreateEvent(NULL, FALSE, FALSE, NULL)) == NULL)
    {
        perror("Can't create event (canRead)!\n");
        return CREATE_EVENT_ERROR;
    }
    if ((canWrite = CreateEvent(NULL, FALSE, FALSE, NULL)) == NULL)
    {
        perror("Can't create event (canWrite)!\n");
        return CREATE_EVENT_ERROR;
    }

    // Создаем потоки
    DWORD id = 0;
    for (int i = 0; i < READERS_NUMBER; i++)
    {
        readersID[i] = i;
        if ((readerThreads[i] = CreateThread(NULL, 0, &Reader, readersID + i, 0, &id))
== NULL)
        {
            perror("Can't CreateThread (reader)");
            return CREATE_READER_THREAD_ERROR;
        }
    }

    for (int i = 0; i < WRITERS_NUMBER; i++)
    {
        writersID[i] = i;
        if ((writerThreads[i] = CreateThread(NULL, 0, &Writer, writersID + i, 0, &id))
== NULL)
        {
            perror("Can't CreateThread (writer)");
            return CREATE_WRITER_THREAD_ERROR;
        }
    }

    WaitForMultipleObjects(READERS_NUMBER, readerThreads, TRUE, INFINITE);
    WaitForMultipleObjects(WRITERS_NUMBER, writerThreads, TRUE, INFINITE);

    // Закрываем
    for (int i = 0; i < READERS_NUMBER; i++)
        CloseHandle(readerThreads[i]);

    for (int i = 0; i < WRITERS_NUMBER; i++)
        CloseHandle(writerThreads[i]);
```

```
    CloseHandle(canRead);
    CloseHandle(canWrite);
    CloseHandle(mutex);
    return OK;
}
```

Пример работы программы:



Изображение 1. – Работа программы.