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»

О Т Ч Е Т

по лабораторной работе № 06

Название ***«Реализация монитора Хоара «Читатели-писатели» под ОС Windows»***

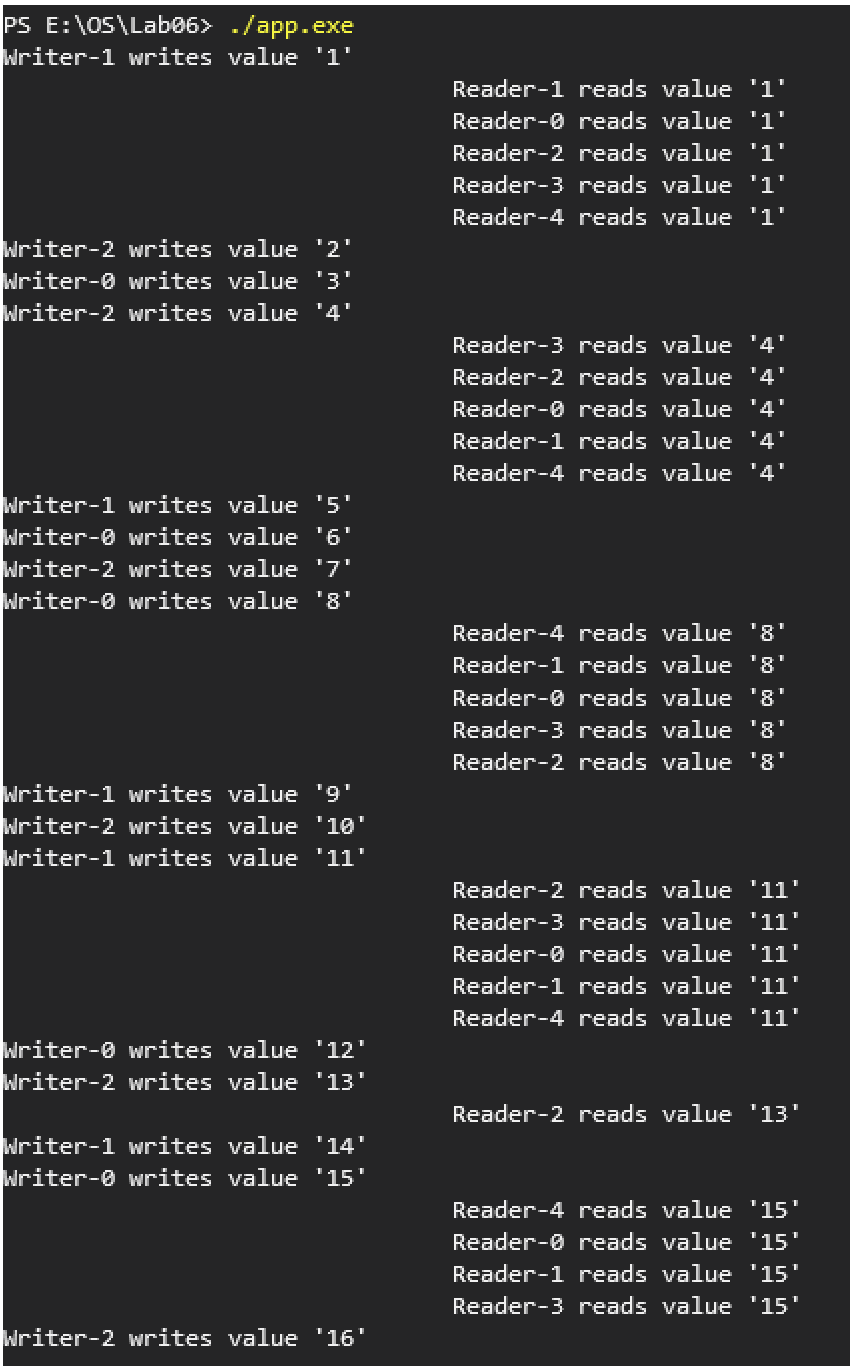
Дисциплина: ***Операционные системы***

Студент	***ИУ7И-56Б***		**Нгуен Ф. С.**
	(Группа)	(Подпись, дата)	(И.О. Фамилия)
Преподаватель			**Рязанова Н. Ю.**
		(Подпись, дата)	(И.О. Фамилия)

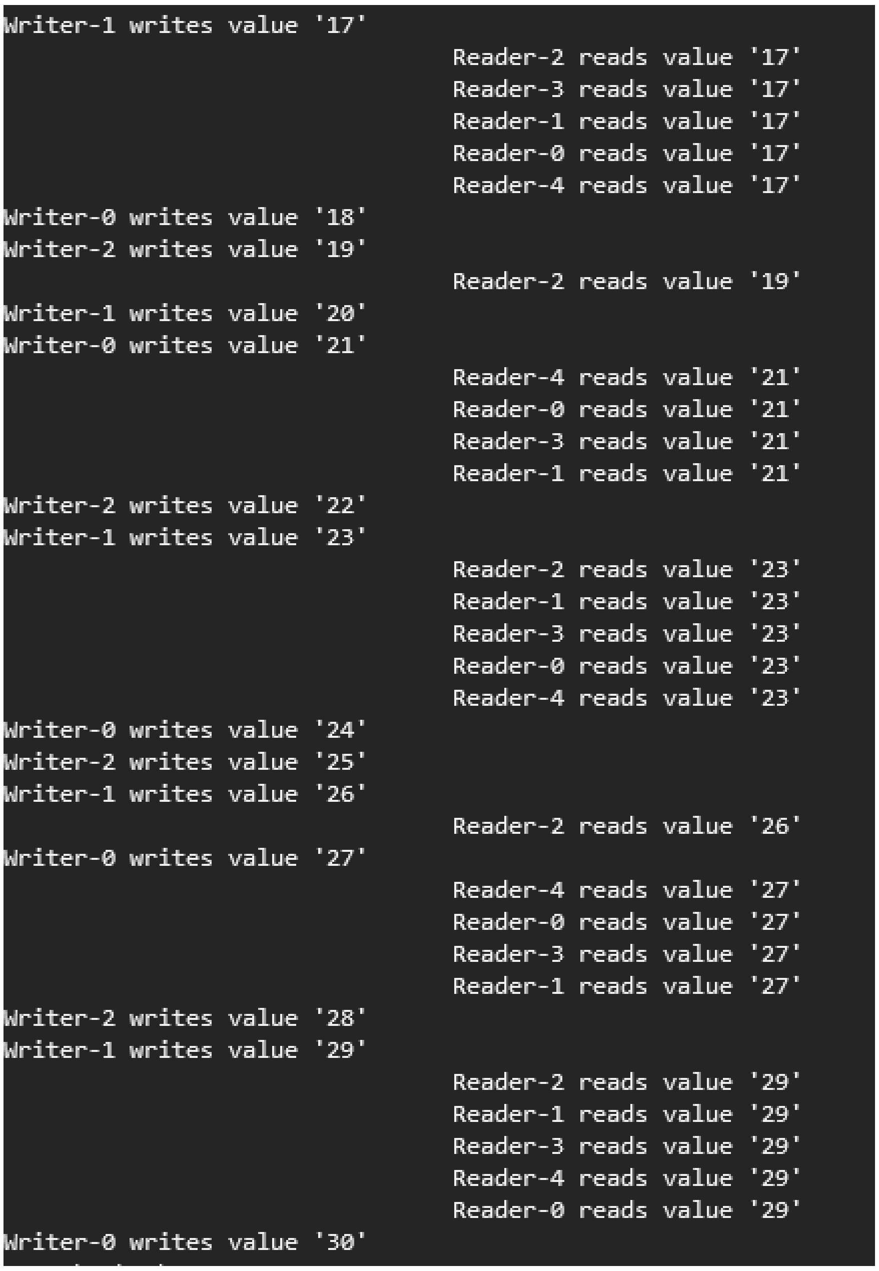
Москва, 2020

❖ **Задание:**

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».



```
PS E:\OS\Lab06> ./app.exe
Writer-1 writes value '1'
                                  Reader-1 reads value '1'
                                  Reader-0 reads value '1'
                                  Reader-2 reads value '1'
                                  Reader-3 reads value '1'
                                  Reader-4 reads value '1'
Writer-2 writes value '2'
Writer-0 writes value '3'
Writer-2 writes value '4'
                                  Reader-3 reads value '4'
                                  Reader-2 reads value '4'
                                  Reader-0 reads value '4'
                                  Reader-1 reads value '4'
                                  Reader-4 reads value '4'
Writer-1 writes value '5'
Writer-0 writes value '6'
Writer-2 writes value '7'
Writer-0 writes value '8'
                                  Reader-4 reads value '8'
                                  Reader-1 reads value '8'
                                  Reader-0 reads value '8'
                                  Reader-3 reads value '8'
                                  Reader-2 reads value '8'
Writer-1 writes value '9'
Writer-2 writes value '10'
Writer-1 writes value '11'
                                  Reader-2 reads value '11'
                                  Reader-3 reads value '11'
                                  Reader-0 reads value '11'
                                  Reader-1 reads value '11'
                                  Reader-4 reads value '11'
Writer-0 writes value '12'
Writer-2 writes value '13'
                                  Reader-2 reads value '13'
Writer-1 writes value '14'
Writer-0 writes value '15'
                                  Reader-4 reads value '15'
                                  Reader-0 reads value '15'
                                  Reader-1 reads value '15'
                                  Reader-3 reads value '15'
Writer-2 writes value '16'
```



```
Writer-1 writes value '17'
                                Reader-2 reads value '17'
                                Reader-3 reads value '17'
                                Reader-1 reads value '17'
                                Reader-0 reads value '17'
                                Reader-4 reads value '17'
Writer-0 writes value '18'
Writer-2 writes value '19'
                                Reader-2 reads value '19'
Writer-1 writes value '20'
Writer-0 writes value '21'
                                Reader-4 reads value '21'
                                Reader-0 reads value '21'
                                Reader-3 reads value '21'
                                Reader-1 reads value '21'
Writer-2 writes value '22'
Writer-1 writes value '23'
                                Reader-2 reads value '23'
                                Reader-1 reads value '23'
                                Reader-3 reads value '23'
                                Reader-0 reads value '23'
                                Reader-4 reads value '23'
Writer-0 writes value '24'
Writer-2 writes value '25'
Writer-1 writes value '26'
                                Reader-2 reads value '26'
Writer-0 writes value '27'
                                Reader-4 reads value '27'
                                Reader-0 reads value '27'
                                Reader-3 reads value '27'
                                Reader-1 reads value '27'
Writer-2 writes value '28'
Writer-1 writes value '29'
                                Reader-2 reads value '29'
                                Reader-1 reads value '29'
                                Reader-3 reads value '29'
                                Reader-4 reads value '29'
                                Reader-0 reads value '29'
Writer-0 writes value '30'
```

```c
#include <stdio.h>
#include <stdbool.h>
#include <windows.h>

#define WRITERS_COUNT 3
#define READERS_COUNT 5

#define ITERATIONS_NUMBER 10

#define PAUSE 200 /* ms*/

HANDLE mutex;
HANDLE can_read;
HANDLE can_write;

HANDLE writers[WRITERS_COUNT];
HANDLE readers[READERS_COUNT];

volatile LONG active_readers_count = 0;
bool active_writer = false;

volatile LONG waiting_writers_count = 0;
volatile LONG waiting_readers_count = 0;

int value = 0;

```

```c
void start_read(void)
{
    WaitForSingleObject(mutex, INFINITE);

    InterlockedIncrement(&waiting_readers_count);

    if (active_writer || WaitForSingleObject(can_write, 0) == WAIT_OBJECT_0)
    {
        WaitForSingleObject(can_read, INFINITE);
    }

    InterlockedDecrement(&waiting_readers_count);
    InterlockedIncrement(&active_readers_count);
    SetEvent(can_read);
    ReleaseMutex(mutex);
}

void stop_read(void)
{
    InterlockedDecrement(&active_readers_count);
    if (active_readers_count == 0)
    {
        SetEvent(can_write);
    }
}

void start_write(void)
{
    InterlockedIncrement(&waiting_writers_count);
    if (active_writer || active_readers_count > 0)
    {
        WaitForSingleObject(can_write, INFINITE);
    }

    InterlockedDecrement(&waiting_writers_count);
    active_writer = true;
    ResetEvent(can_write);
}

void stop_write(void)
{
    active_writer = false;

    if (waiting_readers_count > 0)
    {
        SetEvent(can_read);
    }
    else
    {
        SetEvent(can_write);
    }
}

DWORD WINAPI writer(LPVOID lpParams)
{
    for (int i = 0; i < ITERATIONS_NUMBER; ++i)
    {
        start_write();

        value++;
        printf("Writer-%d writes value '%d'\n", (int)lpParams, value);

        stop_write();
        Sleep(PAUSE);
    }

    return EXIT_SUCCESS;
}

DWORD WINAPI reader(LPVOID lpParams)
{
    while (value < WRITERS_COUNT * ITERATIONS_NUMBER)
    {
        start_read();

        printf("\t\t\t\tReader-%d reads value '%d'\n", (int)lpParams, value);

        stop_read();
        Sleep(PAUSE);
    }

    return EXIT_SUCCESS;
}

int createHandles(void)
{
    if ((mutex = CreateMutex(NULL, FALSE, NULL)) == NULL)
    {
        perror("CreateMutex");
        return EXIT_FAILURE;
    }

    if ((can_read = CreateEvent(NULL, FALSE, TRUE, NULL)) == NULL)
    {
        perror("CreateEvent");
        return EXIT_FAILURE;
    }

    if ((can_write = CreateEvent(NULL, TRUE, TRUE, NULL)) == NULL)
    {
        perror("CreateEvent");
        return EXIT_FAILURE;
    }

    return EXIT_SUCCESS;
}

int createThreads(HANDLE* threads, int threads_count, DWORD(*fn_on_thread)(LPVOID))
{
    for (int i = 0; i < threads_count; ++i)
    {
        if ((threads[i] = CreateThread(NULL, 0, fn_on_thread, (LPVOID)i, 0, NULL)) == NULL)
        {
            perror("CreateThread");
            return EXIT_FAILURE;
        }
    }

    return EXIT_SUCCESS;
}

void closeHandleThreads(HANDLE* threads, int threads_count)
{
    for (int i = 0; i < threads_count; i++)
    {
        CloseHandle(threads[i]);
    }
}

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

    int rc = EXIT_SUCCESS;

    if ((rc = createHandles()) != EXIT_SUCCESS || (rc = createThreads(writers, WRITERS_COUNT, writer)) != EXIT_SUCCESS || (rc = createThreads(readers, READERS_COUNT, reader)) != EXIT_SUCCESS)
    {
        return rc;
    }

    WaitForMultipleObjects(WRITERS_COUNT, writers, TRUE, INFINITE);
    WaitForMultipleObjects(READERS_COUNT, readers, TRUE, INFINITE);

    closeHandleThreads(writers, WRITERS_COUNT);
    closeHandleThreads(readers, READERS_COUNT);
    CloseHandle(mutex);
    CloseHandle(can_read);
    CloseHandle(can_write);

    return rc;
}
```