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 «Читатели-писатели» под ОС Windows»

Студент Садулаева Т. Р.

Группа ИУ7-54Б

Оценка (баллы) _____

Преподаватель Рязанова Н.Ю.

Москва.
2020 г.

Задание

Написать программу, реализующую задачу «Читатели – писатели» по монитору Хоара с четырьмя функциями: Начать_чтение, Закончить_чтение, Начать_запись, Закончить_запись. В программе всеми процессами разделяется одно единственное значение в разделяемой памяти. Писатели ее только инкрементируют, читатели могут только читать значение.

Код программы

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

#define READERS_COUNT 5
#define WRITERS_COUNT 3
#define READ_ITERS 7
#define WRITE_ITERS 8

HANDLE mutex;
HANDLE can_read;
HANDLE can_write;

LONG waiting_writers = 0;
LONG waiting_readers = 0;
LONG active_readers = 0;

bool active_writer = false;
int val = 0;
```

```
void start_read()
{
    InterlockedIncrement(&waiting_readers);

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

    InterlockedDecrement(&waiting_readers);
    InterlockedIncrement(&active_readers);

    SetEvent(can_read);
    ReleaseMutex(mutex);
}

void stop_read()
{
    InterlockedDecrement(&active_readers);

    if (active_readers == 0)
    {
        ResetEvent(can_read);
        SetEvent(can_write);
    }
}
```

```
void start_write(void)
{
    InterlockedIncrement(&waiting_writers);

    if (active_writer || active_readers > 0)
    {
        WaitForSingleObject(can_write, INFINITE);
    }

    InterlockedDecrement(&waiting_writers);

    active_writer = true;
}

void stop_write(void)
{
    active_writer = false;

    if (waiting_readers)
    {
        SetEvent(can_read);
    }
    else
    {
        SetEvent(can_write);
    }
}
```

```
DWORD WINAPI rr_run(CONST LPVOID lpParams)
{
    int r_id = (int)lpParams;

    for (size_t i = 0; i < READ_ITERS; i++)
    {
        start_read();
        printf("?Reader #%d read: %3d", r_id, val);
        stop_read();
    }

    return 0;
}

DWORD WINAPI wr_run(CONST LPVOID lpParams)
{
    int w_id = (int)lpParams;

    for (size_t i = 0; i < WRITE_ITERS; ++i)
    {
        start_write();
        ++val;
        printf("!Writer #%d wrote: %3d", w_id, val);
        stop_write();
    }
    return 0;
}

int main()
```

```
{
    setbuf(stdout, NULL);

    HANDLE readers_threads[READERS_COUNT];
    HANDLE writers_threads[WRITERS_COUNT];

    if ((mutex = CreateMutex(NULL, FALSE, NULL)) == NULL)
    {
        perror("Failed call of CreateMutex");

        return -1;
    }

    can_read = CreateEvent(NULL, FALSE, FALSE, NULL);
    can_write = CreateEvent(NULL, FALSE, FALSE, NULL);

    if (can_read == NULL || can_write == NULL)
    {
        perror("Failed call of CreateEvent");

        return -1;
    }

    for (size_t i = 0; i < READERS_COUNT; ++i)
    {
        readers_threads[i] = CreateThread(NULL, 0, rr_run, (LPVOID)i, 0, NULL);
        if (readers_threads[i] == NULL)
        {
            perror("Failed call of CreateThread");
```

```
            return -1;
        }
    }

    for (size_t i = 0; i < WRITERS_COUNT; ++i)
    {
        writers_threads[i] = CreateThread(NULL, 0, wr_run, (LPVOID)i, 0, NULL);
        if (writers_threads[i] == NULL)
        {
            perror("Failed call of CreateThread");

            return -1;
        }
    }

    WaitForMultipleObjects(READERS_COUNT, readers_threads, TRUE, INFINITE);
    WaitForMultipleObjects(WRITERS_COUNT, writers_threads, TRUE, INFINITE);

    CloseHandle(mutex);
    CloseHandle(can_read);
    CloseHandle(can_write);

    return 0;
}
```

Пример работы программы

Пример работы первой программы представлен на рисунке 1 ниже.

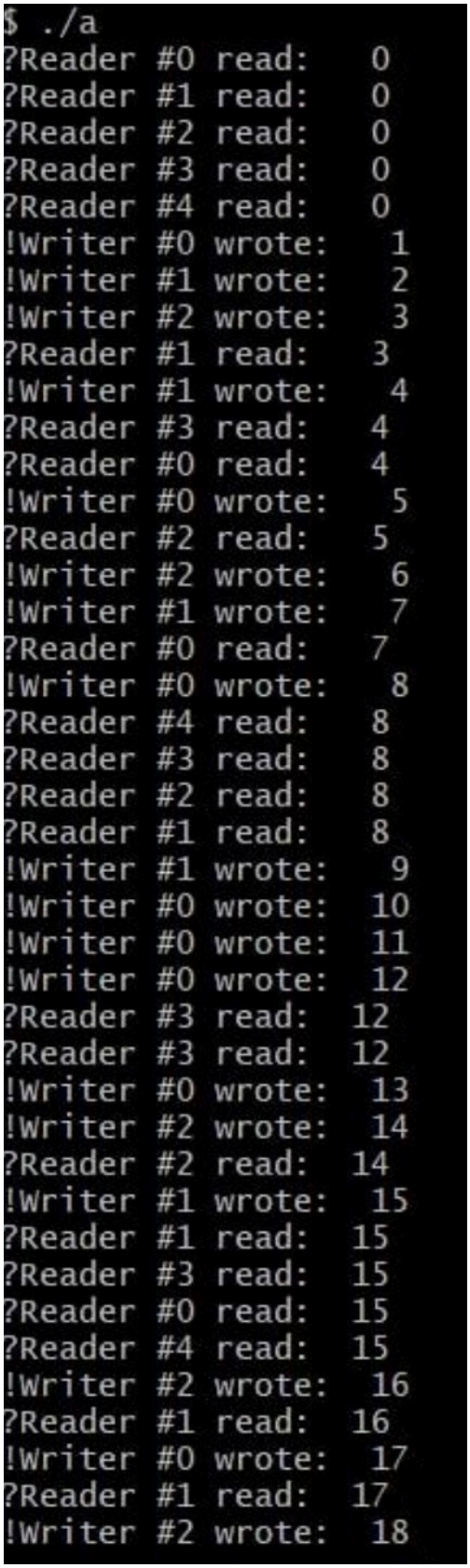

Рисунок 1. – Пример работы программы.