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»

Отчёт

по лабораторной работе № 6

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

Дисциплина: Операционные системы

Студент	ИУ7-55Б		Д.О. Склифасовский
	(Группа)	(Подпись, дата)	(И.О. Фамилия)
Преподаватель			Н.Ю. Рязанова
		(Подпись, дата)	(И.О. Фамилия)

Москва, 2020

Задание

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».

Программа:

Листинг 1 – Задание

```
#include <stdio.h>
#include <windows.h>
#include <iostream>

#define WRITERS 3
#define READERS 5
#define MAX_VAL 10

#define OK 0
#define ERROR 1

int value = 0;

bool activeWriter = false;
unsigned int activeReaders = 0;

unsigned int waitingReaders = 0;
unsigned int waitingWriters = 0;

HANDLE writers[WRITERS];
HANDLE readers[READERS];

HANDLE canRead;
HANDLE canWrite;
HANDLE mutex;
```

```
26
27 void StartRead()
28 {
29     InterlockedIncrement(&waitingReaders);
30     if (activeWriter || waitingWriters > 0)
31     {
32         WaitForSingleObject(canRead, INFINITE);
33     }
34     WaitForSingleObject(mutex, INFINITE);
35
36     InterlockedDecrement(&waitingReaders);
37     InterlockedIncrement(&activeReaders);
38     SetEvent(canRead);
39
40     ReleaseMutex(mutex);
41 }
42
43 void StopRead()
44 {
45     InterlockedDecrement(&activeReaders);
46     if (activeReaders == 0)
47     {
48         SetEvent(canWrite);
49     }
50 }
51
52 void StartWrite()
53 {
54     InterlockedIncrement(&waitingWriters);
55     if (activeWriter || activeReaders > 0)
56     {
57         WaitForSingleObject(canWrite, INFINITE);
58     }
59     InterlockedDecrement(&waitingWriters);
60     activeWriter = true;
```

```
61 }
62
63 void StopWrite()
64 {
65     ResetEvent(canWrite);
66     activeWriter = false;
67     if (waitingWriters)
68     {
69         SetEvent(canWrite);
70     }
71     else
72     {
73         SetEvent(canRead);
74     }
75 }
76
77 DWORD WINAPI Reader(LPVOID lpParam)
78 {
79     bool isEnd = FALSE;
80     while (!isEnd)
81     {
82         StartRead();
83
84         if (value >= MAX_VAL)
85         {
86             isEnd = TRUE;
87         }
88         else
89         {
90             printf("Reader %d (%d) read %d\n", (int)lpParam,
                    GetCurrentThreadId(), value);
91         }
92
93         StopRead();
94         Sleep(400);
```

```
95      }
96      return OK;
97  }
98
99  DWORD WINAPI Writer(LPVOID lpParam)
100 {
101     bool isEnd = FALSE;
102     while (!isEnd)
103     {
104         StartWrite();
105
106         if (value >= MAX_VAL)
107         {
108             isEnd = TRUE;
109         }
110         else
111         {
112             value++;
113             printf("Writer %d (%d) wrote %d\n", (int)lpParam,
                    GetCurrentThreadId(), value);
114         }
115
116         StopWrite();
117         Sleep(400);
118     }
119     return OK;
120 }
121
122 bool CheckHandle(HANDLE cur, const char* msg)
123 {
124     if (cur == NULL)
125     {
126         CloseHandle(mutex);
127         CloseHandle(canRead);
128         CloseHandle(canWrite);
```

```
129             perror(msg);
130             return false;
131         }
132         return true;
133 }
134
135 int CreateThreads()
136 {
137         for (int i = 0; i < WRITERS; i++)
138         {
139             writers[i] = CreateThread(NULL, 0, &Writer, (LPVOID)i, 0,
                    NULL);
140             if (!CheckHandle(writers[i], "Thread"))
141             {
142                 return ERROR;
143             }
144         }
145
146         for (int i = 0; i < READERS; i++)
147         {
148             readers[i] = CreateThread(NULL, 0, &Reader, (LPVOID)i, 0,
                    NULL);
149             if (!CheckHandle(readers[i], "Thread"))
150             {
151                 return ERROR;
152             }
153         }
154
155         return OK;
156 }
157
158 int InitHandles()
159 {
160         mutex = CreateMutex(NULL, FALSE, NULL);
161         if (mutex == NULL)
```

```
162     {
163         perror("mutex");
164         return ERROR;
165     }
166
167     canRead = CreateEvent(NULL, FALSE, FALSE, TEXT("ReadEvent"));
168     if (canRead == NULL)
169     {
170         CloseHandle(mutex);
171         perror("canRead");
172         return ERROR;
173     }
174
175     canWrite = CreateEvent(NULL, TRUE, FALSE, TEXT("WriteEvent"))
        ;
176     if (canWrite == NULL)
177     {
178         CloseHandle(mutex);
179         CloseHandle(canRead);
180         perror("canWrite");
181         return ERROR;
182     }
183
184     return OK;
185 }
186
187 int main()
188 {
189     if (InitHandles() == ERROR || CreateThreads() == ERROR)
190     {
191         return ERROR;
192     }
193
194     WaitForMultipleObjects(WRITERS, writers, TRUE, INFINITE);
195     WaitForMultipleObjects(READERS, readers, TRUE, INFINITE);
```

```
196
197         CloseHandle(mutex);
198         CloseHandle(canRead);
199         CloseHandle(canWrite);
200
201         return 0;
202 }
```

Результат работы программы:

```
Writer 0 (16848) wrote 1
Writer 1 (4200) wrote 2
Writer 2 (16576) wrote 3
Reader 0 (17168) read 3
Reader 1 (9932) read 3
Reader 2 (1032) read 3
Reader 3 (10156) read 3
Reader 4 (4004) read 3
Reader 3 (10156) read 3
Reader 1 (9932) read 3
Reader 4 (4004) read 3
Reader 0 (17168) read 3
Reader 2 (1032) read 3
Writer 2 (16576) wrote 4
Writer 1 (4200) wrote 5
Writer 0 (16848) wrote 6
Writer 0 (16848) wrote 7
Reader 0 (17168) read 7
Reader 4 (4004) read 7
Reader 1 (9932) read 7
Writer 1 (4200) wrote 8
Reader 2 (1032) read 8
Reader 3 (10156) read 8
Writer 2 (16576) wrote 9
Reader 1 (9932) read 9
Writer 2 (16576) wrote 10
```

Рисунок 1 – Результат работы программы