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

```
61      activeWriter = false;
62      ResetEvent(canWrite);
63      if (waitingWriters)
64      {
65          SetEvent(canWrite);
66      }
67      else
68      {
69          SetEvent(canRead);
70      }
71  }
72
73  DWORD WINAPI Reader(LPVOID lpParam)
74  {
75      bool isEnd = FALSE;
76      while (!isEnd)
77      {
78          StartRead();
79
80          if (value >= MAX_VAL)
81          {
82              isEnd = TRUE;
83          }
84          else
85          {
86              printf("Reader %d (%d) read %d\n", (int)lpParam,
                    GetCurrentThreadId(), value);
87          }
88
89          StopRead();
90          Sleep(200);
91      }
92      return OK;
93  }
94
```

```
95 DWORD WINAPI Writer(LPVOID lpParam)
96 {
97     bool isEnd = FALSE;
98     while (!isEnd)
99     {
100         StartWrite();
101         WaitForSingleObject(mutex, INFINITE);
102 
103         if (value >= MAX_VAL)
104         {
105             isEnd = TRUE;
106         }
107         else
108         {
109             value++;
110             printf("Writer %d (%d) wrote %d\n", (int)lpParam,
                    GetCurrentThreadId(), value);
111         }
112 
113         ReleaseMutex(mutex);
114         StopWrite();
115         Sleep(200);
116     }
117     return OK;
118 }
119 
120 bool CheckHandle(HANDLE cur, const char* msg)
121 {
122     if (cur == NULL)
123     {
124         CloseHandle(mutex);
125         CloseHandle(canRead);
126         CloseHandle(canWrite);
127         perror(msg);
128         return false;
```

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

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

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

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