|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **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 по курсу**

**«Операционные системы»**

**Тема**  Реализация Хоара «Читатели-писатели» под ОС Windows

**Студент**  Якуба Д.В.

**Группа**  ИУ7-53Б

**Оценка (баллы)**

**Преподаватель**  Рязанова Н.Ю.

*Москва, 2020 г*

# 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».

Листинг кода 1

|  |
| --- |
| #include <stdio.h>  #include <stdbool.h>  #include <windows.h>  #define WRITERS\_SLEEP\_IN\_MILLI 1000  #define READERS\_SLEEP\_IN\_MILLI 1000  #define WRITERS\_AMOUNT 3  #define READERS\_AMOUNT 5  #define AMOUNT\_OF\_ACTIONS 4  #define SUCCESS 0  #define MUTEX\_ERR 1  #define EVENT\_ERR 2  #define WRITER\_CRT\_ERR 3  #define READER\_CRT\_ERR 4  HANDLE mutex, canRead, canWrite;  HANDLE writersThreads[WRITERS\_AMOUNT];  HANDLE readersThreads[READERS\_AMOUNT];  bool writing = false;  LONG writersInQueue = 0;  LONG readersInQueue = 0;  LONG readingMembers = 0;  int sharedMemory = 0;  void **startWrite**()  {  InterlockedIncrement(&writersInQueue);  if (writing || readingMembers > 0)  WaitForSingleObject(canWrite, INFINITE);  InterlockedDecrement(&writersInQueue);  writing = true;  }  void **stopWrite**()  {  writing = false;  if (readersInQueue > 0)  SetEvent(canRead);  else  SetEvent(canWrite);  }  DWORD WINAPI **writer**(LPVOID lpParams)  {  for (int i = 0; i < AMOUNT\_OF\_ACTIONS; i++)  {  startWrite();  sharedMemory++;  printf("<<---Writer[ID = %d]: write value %d\n", lpParams, sharedMemory);  stopWrite();  Sleep(WRITERS\_SLEEP\_IN\_MILLI);  }  return SUCCESS;  }  void **startRead**()  {  WaitForSingleObject(mutex, INFINITE);  InterlockedIncrement(&readersInQueue);  if (writing || writersInQueue > 0)  WaitForSingleObject(canRead, INFINITE);  InterlockedIncrement(&readingMembers);  SetEvent(canRead);  ReleaseMutex(mutex);  }  void **stopRead**()  {  InterlockedDecrement(&readingMembers);  if (readingMembers == 0)  SetEvent(canWrite);  }  DWORD WINAPI **reader**(LPVOID lpParams)  {  while (sharedMemory < WRITERS\_AMOUNT \* AMOUNT\_OF\_ACTIONS)  {  startRead();  printf("->>Reader[ID = %d]: read value %d\n", lpParams, sharedMemory);  stopRead();  Sleep(READERS\_SLEEP\_IN\_MILLI);  }  return SUCCESS;  }  int **main**()  {  if ((mutex = CreateMutex(NULL, FALSE, NULL)) == NULL)  {  perror("Mutex error");  exit(MUTEX\_ERR);  }  if ((canRead = CreateEvent(NULL, FALSE, TRUE, NULL)) == NULL)  {  perror("CanRead event error");  exit(EVENT\_ERR);  }  if ((canWrite = CreateEvent(NULL, FALSE, TRUE, NULL)) == NULL)  {  perror("CanWrite event error");  exit(EVENT\_ERR);  }  for (int i = 0; i < WRITERS\_AMOUNT; i++)  if ((writersThreads[i] = CreateThread(NULL, 0, writer, i, 0, NULL)) == NULL)  {  perror("Writer thread creation error");  exit(WRITER\_CRT\_ERR);  }  for (int i = 0; i < READERS\_AMOUNT; i++)  if ((readersThreads[i] = CreateThread(NULL, 0, reader, i, 0, NULL)) == NULL)  {  perror("Reader thread creation error");  exit(READER\_CRT\_ERR);  }  WaitForMultipleObjects(WRITERS\_AMOUNT, writersThreads, TRUE, INFINITE);  WaitForMultipleObjects(READERS\_AMOUNT, readersThreads, TRUE, INFINITE);  CloseHandle(mutex);  CloseHandle(canRead);  CloseHandle(canWrite);  return SUCCESS;  } |

Демонстрация работы программы предоставлена на следующих страницах.

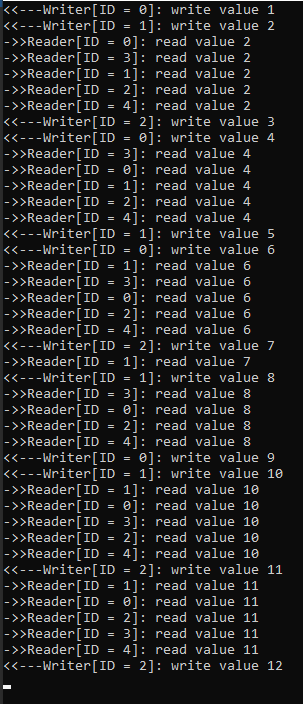


Рисунок 1, Демонстрация работы программы (задержка писателей 1 секунда, задержка читателей 1 секунда)

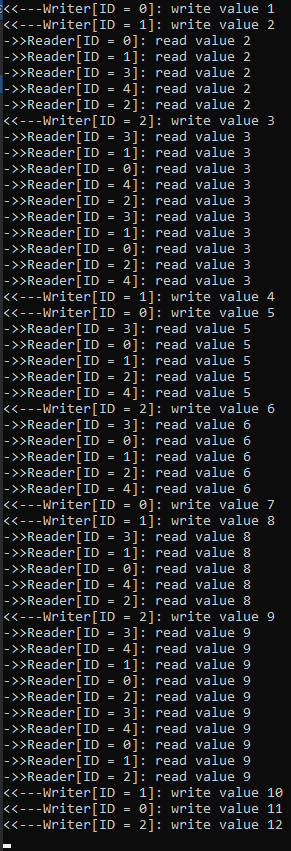


Рисунок 2, Демонстрация работы программы (задержка писателей 2.5 секунды, задержка читателей 1 секунда)

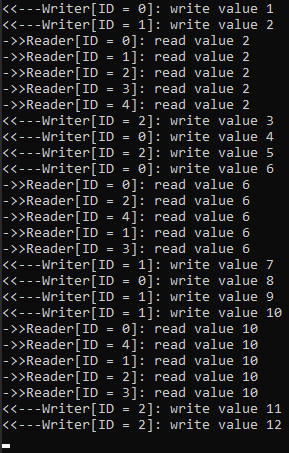


Рисунок 3, Демонстрация работы программы (задержка писателей 1 секунда, задержка читателей 2.5 секунды)