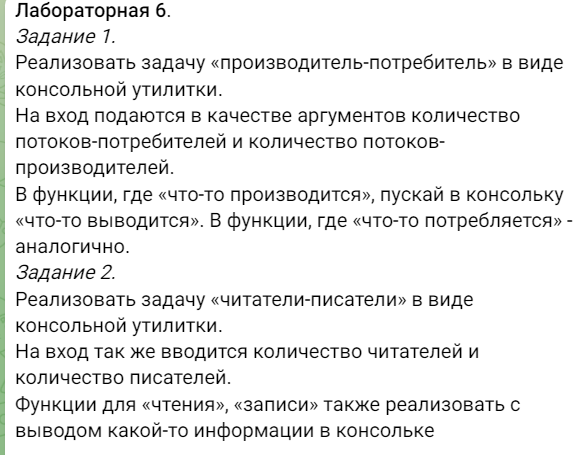
**Лабораторная работа 6**

**Студент:** Смирнов Константин ПКС-303

**Учитель:** Хусточка Алёна Витальевна



1)

#include <windows.h>

#include <processthreadsapi.h>

#include <stdio.h>

#include <conio.h>

#include <clocale>

/\*

Семафор

состояния:

0 - свободен

1 - занят

2 - в ожидании

\*/

struct Semaphore

{

int condition = 0;

};

#define ISFREE 0

#define ISBUSY 1

#define ISWAITING 2

int FreeSem(Semaphore\* sem)

{

sem->condition = 0;

return sem->condition;

}

int BusySem(Semaphore\* sem)

{

sem->condition = 1;

return sem->condition;

}

int WaitSem(Semaphore\* sem)

{

sem->condition = 2;

return sem->condition;

}

bool\* Product;

int SizeM, SizeC;

bool isWorking = true;

Semaphore sem;

DWORD WINAPI ManufacturerHandler(int index)

{

while (isWorking)

{

if (sem.condition == ISWAITING) FreeSem(&sem);

if (sem.condition == ISFREE)

{

BusySem(&sem);

for (int i = 0; i < SizeC; i++)

{

if (Product[i] == false)

{

Product[i] = true;

printf("Производитель №%d: произвёл продукт №%d\n", index, i);

break;

}

}

WaitSem(&sem);

}

Sleep(100);

}

return 0;

}

DWORD ConsumerHandler(int index)

{

while (isWorking)

{

if (sem.condition == ISWAITING) FreeSem(&sem); //если семофор готов к осово

if (sem.condition == ISFREE)

{

BusySem(&sem);

for (int i = 0; i < SizeC; i++)

{

if (Product[i] == true)

{

Product[i] = false;

printf("\t\tПотребитель №%d: купил продукт №%d\n", index, i);

break;

}

}

WaitSem(&sem);

}

Sleep(100);

}

return 0;

}

int main()

{

setlocale(0, "ru");

printf\_s("Введите кол-во производителей: ");

scanf\_s("%d", &SizeM);

printf\_s("Введите кол-во потребителей: ");

scanf\_s("%d", &SizeC);

HANDLE\* ThreadsManufacturer = (HANDLE\*)malloc(sizeof(HANDLE) \* SizeM);

if (!ThreadsManufacturer) return 1;

HANDLE\* ThreadsConsumer = (HANDLE\*)malloc(sizeof(HANDLE) \* SizeC);

if (!ThreadsConsumer) return 1;

BusySem(&sem);

for (int i = 0; i < SizeM; i++)

ThreadsManufacturer[i] = CreateThread(NULL, NULL, (LPTHREAD\_START\_ROUTINE)(ManufacturerHandler), (LPVOID)i, NULL, NULL);

for (int i = 0; i < SizeC; i++)

ThreadsConsumer[i] = CreateThread(NULL, NULL, (LPTHREAD\_START\_ROUTINE)(ConsumerHandler), (LPVOID)i, NULL, NULL);

WaitForMultipleObjects(SizeM, ThreadsManufacturer, true, 1000);

WaitForMultipleObjects(SizeC, ThreadsConsumer, true, 1000);

WaitSem(&sem);

Product = (bool\*)calloc(true, sizeof(bool) \* SizeC);

\_getch();

isWorking = false;

for (int i = 0; i < SizeM; i++)

CloseHandle(ThreadsManufacturer[i]);

for (int i = 0; i < SizeC; i++)

CloseHandle(ThreadsConsumer[i]);

}

2 задание

#include <windows.h>

#include <processthreadsapi.h>

#include <stdio.h>

#include <conio.h>

#include <clocale>

/\*

Семафор

состояния:

0 - свободен

1 - занят

2 - в ожидании

\*/

struct Semaphore

{

int condition = 0;

};

#define ISFREE 0

#define ISBUSY 1

#define ISWAITING 2

int FreeSem(Semaphore\* sem)

{

sem->condition = 0;

return sem->condition;

}

int BusySem(Semaphore\* sem)

{

sem->condition = 1;

return sem->condition;

}

int WaitSem(Semaphore\* sem)

{

sem->condition = 2;

return sem->condition;

}

int sizeW, sizeR;

bool isWorking = true;

int ReadersCount = 0;

Semaphore sem;

DWORD WINAPI WriterHandler(int index)

{

while (isWorking)

{

if (sem.condition == ISWAITING) FreeSem(&sem);

if (sem.condition == ISFREE)

{

BusySem(&sem);

printf\_s("Писатель №%d пишет документ!\n", index);

printf\_s("Писатель №%d перестал писать документ!\n", index);

WaitSem(&sem);

}

Sleep(1000);

}

return 0;

}

DWORD ReaderHandler(int index)

{

while (isWorking)

{

//if (sem.condition == ISWAITING) FreeSem(&sem); //если семофор готов к осово

if (sem.condition == ISWAITING)

{

BusySem(&sem);

printf\_s("\t\tЧитатель №%d читает документ!\n", index);

ReadersCount++;

printf\_s("\t\tЧитатель №%d перестал читать документ!\n", index);

WaitSem(&sem);

}

ReadersCount--;

if (ReadersCount == 0) FreeSem(&sem);

Sleep(1000);

}

return 0;

}

int main()

{

setlocale(0, "ru");

printf\_s("Введите кол-во писателей: ");

scanf\_s("%d", &sizeW);

printf\_s("Введите кол-во читателей: ");

scanf\_s("%d", &sizeR);

HANDLE\* ThreadsWriter = (HANDLE\*)malloc(sizeof(HANDLE) \* sizeW);

if (!ThreadsWriter) return 1;

HANDLE\* ThreadsReader = (HANDLE\*)malloc(sizeof(HANDLE) \* sizeR);

if (!ThreadsReader) return 1;

BusySem(&sem);

for (int i = 0; i < sizeW; i++)

ThreadsWriter[i] = CreateThread(NULL, NULL, (LPTHREAD\_START\_ROUTINE)(WriterHandler), (LPVOID)i, NULL, NULL);

for (int i = 0; i < sizeR; i++)

ThreadsReader[i] = CreateThread(NULL, NULL, (LPTHREAD\_START\_ROUTINE)(ReaderHandler), (LPVOID)i, NULL, NULL);

WaitForMultipleObjects(sizeW, ThreadsWriter, true, 100);

WaitForMultipleObjects(sizeR, ThreadsReader, true, 100);

WaitSem(&sem);

\_getch();

isWorking = false;

for (int i = 0; i < sizeW; i++)

CloseHandle(ThreadsWriter[i]);

for (int i = 0; i < sizeR; i++)

CloseHandle(ThreadsReader[i]);

}