**МИНОБРНАУКИ РОССИИ**

Федеральное государственное бюджетное образовательное учреждение высшего образования «Ижевский государственный  
технический университет имени М.Т. Калашникова»

(ФГБОУ ВО «ИжГТУ имени М.Т. Калашникова»)

Кафедра «Вычислительная техника»

Отчет по лабораторной работе №3

по дисциплине «Системное программное обеспечение»

на тему «Синхронизация доступа к ресурсу»

Вариант 14

Выполнил: студент гр. Б06-781-1

Пермитин И.А.

Проверил: к.т.н., доцент кафедры ВТ

Марков Е.М.

Ижевск 2020

**Задание**

Создать многопоточную программу, которая выполняет обмен через буфер, используя один из механизмов синхронизации (семафоры, мониторы и др.). Реализация допускается для любой платформы.

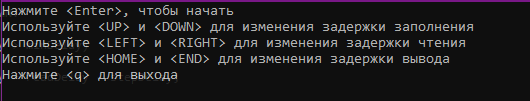
**Описание использованных технологий**

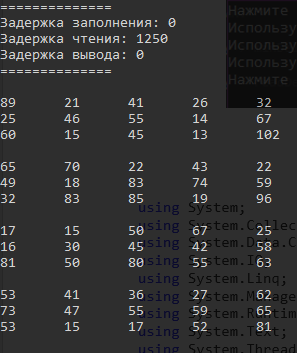
Лабораторная работа выполнялась в IDE «Visual Studio 2019».

Использованный язык программирования – C++

Тип проекта – Консольное приложение.

**Работа приложения**





**Листинг программы**

#include <iostream>

#include <stdio.h>

#include <windows.h>

#include <ctime>

#include <conio.h>

#include <process.h>

using namespace std;

const unsigned int thrMax = 1;

const unsigned int colMatrix = 5;

const unsigned int rowMatrix = 3;

const unsigned int minRand = 11;

const unsigned int maxRand = 95;

const unsigned int stepDelay = 50;

const unsigned int minDelay = 0;

const unsigned int maxDelay = 2000;

HANDLE hSemaphore1;

HANDLE hSemaphore2;

int buffMatrix[rowMatrix][colMatrix];

int matrix[rowMatrix][colMatrix];

unsigned int readDelay = 0;

unsigned int fillDelay = 0;

unsigned int printDelay = 0;

enum delayControl { DEL\_FILL\_UP, DEL\_FILL\_DOWN, DEL\_READ\_UP, DEL\_READ\_DOWN, DEL\_PRINT\_UP, DEL\_PRINT\_DOWN };

void thrFillMatrix(void\* pParams)

{

srand(time(NULL));

while (true)

{

WaitForSingleObject(hSemaphore1, INFINITE);

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

{

for (int j = 0; j < colMatrix; j++)

{

buffMatrix[i][j] = minRand + rand() % maxRand;

}

}

Sleep(fillDelay);

ReleaseSemaphore(hSemaphore1, 1, NULL);

}

}

void thrReadMatrix(void\* pParams)

{

while (true)

{

WaitForSingleObject(hSemaphore1, INFINITE);

WaitForSingleObject(hSemaphore2, INFINITE);

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

{

for (int j = 0; j < colMatrix; j++)

{

matrix[i][j] = buffMatrix[i][j];

}

}

Sleep(readDelay);

ReleaseSemaphore(hSemaphore2, 1, NULL);

ReleaseSemaphore(hSemaphore1, 1, NULL);

}

}

void thrPrintMatrix(void\* pParams)

{

while (true)

{

WaitForSingleObject(hSemaphore2, INFINITE);

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

{

for (int j = 0; j < colMatrix; j++)

{

cout << matrix[i][j] << "\t";

}

cout << "\n";

}

cout << "\n";

Sleep(printDelay);

ReleaseSemaphore(hSemaphore2, 1, NULL);

}

}

void start()

{

hSemaphore1 = CreateSemaphoreA(NULL, thrMax, thrMax, NULL);

hSemaphore2 = CreateSemaphoreA(NULL, thrMax, thrMax, NULL);

\_beginthread(thrFillMatrix, 0, NULL);

\_beginthread(thrReadMatrix, 0, NULL);

\_beginthread(thrPrintMatrix, 0, NULL);

}

void changeDelay(delayControl delCntrl)

{

switch (delCntrl)

{

case DEL\_FILL\_UP:

{

if (fillDelay < maxDelay)

{

fillDelay = fillDelay + stepDelay;

}

break;

}

case DEL\_FILL\_DOWN:

{

if (fillDelay > minDelay)

{

fillDelay = fillDelay - stepDelay;

}

break;

}

case DEL\_READ\_UP:

{

if (readDelay < maxDelay)

{

readDelay = readDelay + stepDelay;

}

break;

}

case DEL\_READ\_DOWN:

{

if (readDelay > minDelay)

{

readDelay = readDelay - stepDelay;

}

break;

}

case DEL\_PRINT\_UP:

{

if (printDelay < maxDelay)

{

printDelay = printDelay + stepDelay;

}

break;

}

case DEL\_PRINT\_DOWN:

{

if (printDelay > minDelay)

{

printDelay = printDelay - stepDelay;

}

break;

}

}

cout << "\n==============\n";

cout << "Задержка заполнения: " << fillDelay << "\n";

cout << "Задержка чтения: " << readDelay << "\n";

cout << "Задержка вывода: " << printDelay << "\n";

cout << "==============\n\n";

}

int main()

{

setlocale(LC\_ALL, "Russian");

bool isStarted = false;

cout << "Нажмите <Enter>, чтобы начать\n";

cout << "Используйте <UP> и <DOWN> для изменения задержки заполнения\n";

cout << "Используйте <LEFT> и <RIGHT> для изменения задержки чтения\n";

cout << "Используйте <HOME> и <END> для изменения задержки вывода\n";

cout << "Нажмите <q> для выхода\n" << endl;

while (true)

{

short keyRetState = GetAsyncKeyState(VK\_RETURN);

short keyUpState = GetAsyncKeyState(VK\_UP);

short keyDownState = GetAsyncKeyState(VK\_DOWN);

short keyLeftState = GetAsyncKeyState(VK\_LEFT);

short keyRightState = GetAsyncKeyState(VK\_RIGHT);

short keyHomeState = GetAsyncKeyState(VK\_HOME);

short keyEndState = GetAsyncKeyState(VK\_END);

short keyQState = GetAsyncKeyState(0x51);

if (keyRetState & 0x8000)

{

if (isStarted == false)

{

start();

isStarted = true;

}

}

if (keyUpState & 0x8000)

{

changeDelay(DEL\_FILL\_UP);

}

if (keyDownState & 0x8000)

{

changeDelay(DEL\_FILL\_DOWN);

}

if (keyLeftState & 0x8000)

{

changeDelay(DEL\_READ\_DOWN);

}

if (keyRightState & 0x8000)

{

changeDelay(DEL\_READ\_UP);

}

if (keyEndState & 0x8000)

{

changeDelay(DEL\_PRINT\_DOWN);

}

if (keyHomeState & 0x8000)

{

changeDelay(DEL\_PRINT\_UP);

}

if (keyQState & 0x8000)

{

break;

}

}

}