

Міністерство освіти і науки України

Національний технічний університет України

«Київський політехнічний інститут ім. Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

**ЛАБОРАТОРНА РОБОТА №4**

**З ДИСЦИПЛІНИ “ОСНОВИ ПАРАЛЕЛЬНОГО ПРОГРАМУВАННЯ ”**

**НА ТЕМУ: “ПОТОКИ В БІБЛІОТЕЦІ WINAPI”**

**Виконав:**

Студент ІІІ курсу ФІОТ

групи ІО-82

Шендріков Євгеній

Номер у списку - 25

**Перевірив:**

Доцент Корочкін О. В.

м. Київ – 2020 р.

**Завдання**

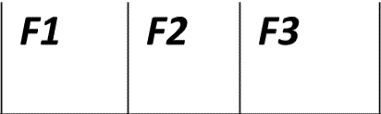
Розробити програму за допомогою засобів бібліотеки WinAPI, яка містить ***паралельні потоки,*** кожен з яких реалізовує відповідну функцію F1, F2, F3 із **Додатку Б** згідно отриманому варіанту.

При створенні задач необхідно:

* вказати ім’я задачі;
* встановити пріоритет задачі;
* задати розмір стека задачі;
* обрати і задати номер процесу (ядра) для виконання кожної задачі.

В тілі задачі задіяти оператор задержки **sleep** при виконанні функцій F1, F2, F3 з невеликим часом затримки.

**Варіант завдання**



1.21 D = SORT(A)+SORT(B)+SORT(C)\*(MA\*ME)

2.27 MF = (MG\*MH)\*TRANS(MK)

3.30 S = (MO\*MP)\*V+t\*MR\*(O+P)

**Виконання роботи**

Під час виконання лабораторної роботи (за допомогою засобів бібліотеки WinAPI) проблемою при тестуванні програмної частини, як і в перших лабораторних, стало введення початкових даних з консолі.

Для вирішення цієї проблеми необхідно блокувати процес, який звертається до спільного ресурсу, який в цей момент вже використовується іншим процесом. А розблокування процесу відбувається одразу ж після звільнення спільного ресурсу.

Для реалізації цього підходу в даній лабораторній роботі було використано мютекси (в попередніх лабораторних використовувались семафори, тому для різноманіття було взято мютекси), щоб організувати взаємодію потоків для рішення задачі взаємного виключення до спільного ресурсу.

В бібліотеці WinAPI мютекси є змінними спеціального типу HANDLE, за якими слідкує сама система. Операція, що створює мютекс виглядає наступним чином:

HANDLE CreateMutex(LPSECURITY\_ATTRIBUTES lpMutexAttributes, BOOL bInitialOwner, LPCTSTR lpName)

* lpMutexAttributes – атрибут безпеки;
* bInitialOwner – початковий стан;
* lpName – ім’я об’єкту.

Кожен потік, який хоче використовувати спільний ресурс, повинен спочатку викликати функцію **WaitForSingleObject()** перед зверненням до ресурсу для отримання блокування. Коли потік завершує роботу з ресурсом, він повинен викликати **ReleaseMutex()** для зняття блокування. По закінченню дескриптор об'єкта mutex потрібно закрити викликавши **CloseHandle()**.

При N > 10 можливість вводити дані з клавіатури не передбачена.

**Висновки**

1. В даній лабораторній роботі було розглянуто принцип побудови паралельних програм за допомогою засобів бібліотеки WinAPI. Для створення процесу в WinAPI використовувалась функція **CreateThread()**. Функція повертає ідентифікатор процесу, який є унікальним і визначає його в системі. Потік залишається в системі, поки він не закінчить роботу і всі його дескриптори не будуть закритими за допомогою функції **CloseHandle()**.

2. Для кожного потоку була створена відповідна потокова функція, встановлений розмір стеку, обрано спеціальне ядро для виконання кожного потоку, було реалізовано запуск потоку з відкладанням.

3. Було використано мютекси для організації взаємодії потоків задля рішення задачі взаємного виключення до спільного ресурсу, яким є клавіатура. В бібліотеці WinAPI мютекси є змінними спеціального типу HANDLE, за якими слідкує сама система.

4. Було зроблено аналіз проблем, які виникли під час виконання, а також запропоновано шляхи їх рішення, перевірено працездатність програми і програмним шляхом оброблено виключні ситуації, наприклад, при N < 0 чи при N > 10 можливість вводити дані з клавіатури не передбачена.

Кінцева мета роботи досягнута.

**Лістинг програми**

**Lab4.cpp**

/\*-----------------------------------------------------

| Labwork #4 |

------------------------------------------------------

| Author | Jack (Yevhenii) Shendrikov |

| Group | IO-82 |

| Variant | #25 |

| Date | 15.10.2020 |

------------------------------------------------------

| Function 1 | D = SORT(A)+SORT(B)+SORT(C)\*(MA\*ME) |

| Function 2 | MF = (MG\*MH)\*TRANS(MK) |

| Function 3 | S = (MO\*MP)\*V+t\*MR\*(O+P) |

------------------------------------------------------

\*/

#include "T1.h"

#include "T2.h"

#include "T3.h"

int N;

int main() {

//---------------------------- Input Handler ---------------------------

// header

printf("----------------------------------------------------\n"

"| Function 1 | D = SORT(A)+SORT(B)+SORT(C)\*(MA\*ME) |\n"

"| Function 2 | MF = (MG\*MH)\*TRANS(MK) |\n"

"| Function 3 | S = (MO\*MP)\*V+t\*MR\*(O+P) |\n"

"----------------------------------------------------\n\n"

"!!! Note that if the value of N > 10 -> the result will not be displayed !!!\n"

"!!! If you enter N <= 0 - execution will be terminated !!!\n\n"

"Enter N: ");

cin >> N;

// check for int value of N, else N = 3

if (cin.fail()) {

cout << "\n!!! You should enter data of type int, N will be taken as 3 !!!\n" << endl;

N = 3;

}

// check for positive value of N

if (N <= 0) {

cout << "Restart the program and enter N > 0." << endl;

exit(EXIT\_FAILURE);

}

// if N > 10 - input from the keyboard denied

if (N > 10) {

cout << "If you want to enter a value from the keyboard - enter N <= 10." << endl;

exit(EXIT\_FAILURE);

}

cout << "\n!!! Enter All Values From The Keyboard !!!" << endl;

//------------------------- Main Body ----------------------------------

HANDLE mutex = CreateMutex(NULL, FALSE, NULL);

cout << "\nLab4 started!\n" << endl;

DWORD TidA, TidB, TidC;

HANDLE hThread[3];

T1\* F1 = new T1(N, mutex);

T2\* F2 = new T2(N, mutex);

T3\* F3 = new T3(N, mutex);

hThread[0] = CreateThread(NULL, 20000, T1::startThread, F1, 0, &TidA);

hThread[1] = CreateThread(NULL, 0, T2::startThread, F2, CREATE\_SUSPENDED, &TidB);

hThread[2] = CreateThread(NULL, 0, T3::startThread, F3, 0, &TidC);

SetThreadPriority(hThread[0], THREAD\_PRIORITY\_LOWEST);

SetThreadPriority(hThread[1], THREAD\_PRIORITY\_NORMAL);

SetThreadPriority(hThread[2], THREAD\_PRIORITY\_HIGHEST);

for (int i = 0; i < 3; i++)

SetThreadAffinityMask(hThread[i], 1 << i);

Sleep(10);

ResumeThread(hThread[1]);

WaitForMultipleObjects(3, hThread, true, INFINITE);

for (int i = 0; i < 3; i++) {

CloseHandle(hThread[i]);

}

CloseHandle(mutex);

cout << "\nLab4 finished!\n";

cin.get();

}

**T1.h**

#pragma once

#include <iostream>

#include <Windows.h>

#include "Data.h"

class T1 {

private:

int N;

HANDLE mutex;

public:

T1(int N, HANDLE mutex);

static DWORD WINAPI startThread(void\* param);

DWORD run();

};

**T2.h**

#pragma once

#include <iostream>

#include <Windows.h>

#include "Data.h"

class T2 {

private:

int N;

HANDLE mutex;

public:

T2(int N, HANDLE mutex);

static DWORD WINAPI startThread(void\* param);

DWORD run();

};

**T3.h**

#pragma once

#include <iostream>

#include <Windows.h>

#include "Data.h"

class T3 {

private:

int N;

HANDLE mutex;

public:

T3(int N, HANDLE mutex);

static DWORD WINAPI startThread(void\* param);

DWORD run();

};

**T1.cpp**

#include "T1.h"

T1::T1(int N, HANDLE mutex) {

this->N = N;

this->mutex = mutex;

}

DWORD WINAPI T1::startThread(void\* param) {

T1\* This = (T1\*)param;

return This->run();

}

DWORD T1::run() {

cout << "T1 started." << endl;

Data\* data = new Data(N);

vector<int> A, B, C;

vector<vector<int>> MA, ME;

// Generate Input Values

Sleep(50);

cout << "T1 is waiting for a permit." << endl;

WaitForSingleObject(mutex, INFINITE);

Sleep(100);

cout << "\nT1 gets a permit.\n" << endl;

A = data->VectorInput('A');

B = data->VectorInput('B');

C = data->VectorInput('C');

MA = data->MatrixInput("MA");

ME = data->MatrixInput("ME");

cout << "\nT1 releases the permit." << endl;

ReleaseMutex(mutex);

cout << "\nT1 is waiting for a permit." << endl;

// Calculate The Result

vector<int> result = data->Func1(A, B, C, MA, ME);

Sleep(100);

// Output

WaitForSingleObject(mutex, INFINITE);

cout << "T1 gets a permit.\n" << endl;

cout << "T1 result:\n";

data->VectorOutput(result, 'D');

cout << "T1 releases the permit." << endl;

ReleaseMutex(mutex);

cout << "T1 finished.\n" << endl;

delete data;

return 0;

}

**T2.cpp**

#include "T2.h"

T2::T2(int N, HANDLE mutex) {

this->N = N;

this->mutex = mutex;

}

DWORD WINAPI T2::startThread(void\* param) {

T2\* This = (T2\*)param;

return This->run();

}

DWORD T2::run() {

cout << "T2 started." << endl;

Data\* data = new Data(N);

vector<vector<int>> MG, MH, MK;

// Generate Input Values

Sleep(50);

cout << "T2 is waiting for a permit." << endl;

WaitForSingleObject(mutex, INFINITE);

Sleep(100);

cout << "\nT2 gets a permit.\n" << endl;

MG = data->MatrixInput("MG");

MH = data->MatrixInput("MH");

MK = data->MatrixInput("MK");

cout << "\nT2 releases the permit." << endl;

ReleaseMutex(mutex);

cout << "\nT2 is waiting for a permit." << endl;

// Calculate The Result

vector<vector<int>> result = data->Func2(MG, MH, MK);

Sleep(100);

// Output

WaitForSingleObject(mutex, INFINITE);

cout << "T2 gets a permit.\n" << endl;

cout << "T2 result:\n";

data->MatrixOutput(result, "MF");

cout << "T2 releases the permit." << endl;

ReleaseMutex(mutex);

cout << "T2 finished.\n" << endl;

delete data;

return 0;

}

**T3.cpp**

#include "T3.h"

T3::T3(int N, HANDLE mutex) {

this->N = N;

this->mutex = mutex;

}

DWORD WINAPI T3::startThread(void\* param) {

T3\* This = (T3\*)param;

return This->run();

}

DWORD T3::run() {

cout << "T3 started." << endl;

Data\* data = new Data(N);

int t;

vector<int> V, O, P;

vector<vector<int>> MO, MP, MR;

// Generate Input Values

Sleep(50);

cout << "T3 is waiting for a permit." << endl;

WaitForSingleObject(mutex, INFINITE);

Sleep(100);

cout << "\nT3 gets a permit.\n" << endl;

t = data->NumInput('t');

V = data->VectorInput('V');

O = data->VectorInput('O');

P = data->VectorInput('P');

MO = data->MatrixInput("MO");

MP = data->MatrixInput("MP");

MR = data->MatrixInput("MR");

cout << "\nT3 releases the permit." << endl;

ReleaseMutex(mutex);

cout << "\nT3 is waiting for a permit." << endl;

// Calculate The Result

vector<int> result = data->Func3(t, V, O, P, MO, MP, MR);

Sleep(100);

// Output

WaitForSingleObject(mutex, INFINITE);

cout << "T3 gets a permit.\n" << endl;

cout << "T3 result:\n";

data->VectorOutput(result, 'S');

cout << "T3 releases the permit." << endl;

ReleaseMutex(mutex);

cout << "T3 finished.\n" << endl;

delete data;

return 0;

}

**Data.h**

#pragma once

#include <random>

#include <ctime>

#include <string>

using namespace std;

class Data {

private:

int N;

public:

Data(int N);

// ------------------- Fill Matrix/Vector With Specific Number -------------------

vector<vector<int>> FillMatrixWithNumber(int number);

vector<int> FillVectorWithNumber(int number);

// ---------- Data Entry Handler For Matrices, Vectors And Numbers ---------

vector<vector<int>> MatrixInput(string name);

vector<int> VectorInput(char name);

int NumInput(char name);

// ------------- Print Matrix, Vector And Number Into Console --------------

void MatrixOutput(vector<vector<int>> MA, string name);

void VectorOutput(vector<int> A, char name);

void NumOutput(int a, char name);

// Transposing Matrix

vector<vector<int>> MatrixTransp(vector<vector<int>> MA);

// Multiply 2 Matrices

vector<vector<int>> MatrixMult(vector<vector<int>> MA, vector<vector<int>> MB);

// Multiply Matrix And Vector

vector<int> VectorMatrixMult(vector<int> A, vector<vector<int>> MA);

// Calculates Sum Of 2 Vectors

vector<int> SumVectors(vector<int> A, vector<int> B);

// Multiply Integer And Matrix

vector<int> IntVectorMult(int a, vector<int> A);

// Sort Vector

vector<int> SortVector(vector<int> A);

// F1 -> D = SORT(A)+SORT(B)+SORT(C)\*(MA\*ME)

vector<int> Func1(vector<int> A, vector<int> B, vector<int> C, vector<vector<int>> MA, vector<vector<int>> ME);

// F2 -> MF = (MG\*MH)\*TRANS(MK)

vector<vector<int>> Func2(vector<vector<int>> MG, vector<vector<int>> MH, vector<vector<int>> MK);

// F3 -> S = (MO\*MP)\*V+t\*MR\*(O+P)

vector<int> Func3(int t, vector<int> V, vector<int> O, vector<int> P, vector<vector<int>> MO, vector<vector<int>> MP, vector<vector<int>> MR);

};

**Data.cpp**

#include "Data.h"

#include <iostream>

Data::Data(int N)

{

this->N = N;

}

// ------------------- Fill Matrix/Vector With Specific Number -------------------

vector<vector<int>> Data::FillMatrixWithNumber(int number)

{

vector<vector<int>> MA(N, vector <int>(N));

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

{

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

{

MA[i][j] = number;

}

}

return MA;

}

vector<int> Data::FillVectorWithNumber(int number)

{

std::vector<int> A(N);

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

{

A[i] = number;

}

return A;

}

// ---------- Data Entry Handler For Matrices, Vectors And Numbers ---------

vector<vector<int>> Data::MatrixInput(string name)

{

cout << "Enter the " << N \* N << " elements of the Matrix " << name << ":" << endl;

vector<vector<int>> MA(N, vector <int>(N));

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

{

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

{

cout << name << "[" << i << "][" << j << "] = ";

cin >> MA[i][j];

}

}

return MA;

}

vector<int> Data::VectorInput(char name)

{

cout << "Enter the " << N << " elements of the Vector " << name << ":" << endl;

vector<int> A(N);

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

{

cout << name << "[" << i << "] = ";

cin >> A[i];

}

return A;

}

int Data::NumInput(char name)

{

int a;

cout << "Enter number " << name << " = ";

cin >> a;

return a;

}

// ------------- Print Matrix, Vector And Number Into Console --------------

void Data::MatrixOutput(vector<vector<int>> MA, string name)

{

cout << "\tMatrix " << name << ":" << endl;

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

{

cout << "\t\t";

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

{

cout << MA[i][j] << " ";

}

cout << endl;

}

}

void Data::VectorOutput(vector<int> A, char name)

{

cout << "\tVector " << name << ": ";

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

{

cout << A[i] << " ";

}

cout << endl;

}

void Data::NumOutput(int a, char name)

{

cout << "\tNumber " << name << ": " << a << "\n";

}

// Transposing Matrix

vector<vector<int>> Data::MatrixTransp(vector<vector<int>> MA)

{

int buf;

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

{

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

{

buf = MA[i][j];

MA[i][j] = MA[j][i];

MA[j][i] = buf;

}

}

return MA;

}

// Multiply 2 Matrices

vector<vector<int>> Data::MatrixMult(vector<vector<int>> MA, vector<vector<int>> MB)

{

vector<vector<int>> MC(N, vector <int>(N));

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

{

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

{

for (int k = 0; k < N; k++)

{

MC[i][j] += MA[i][k] \* MB[k][j];

}

}

}

return MC;

}

// Multiply Matrix And Vector

vector<int> Data::VectorMatrixMult(vector<int> A, vector<vector<int>> MA)

{

vector<int> B(N);

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

{

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

{

B[i] += A[j] \* MA[i][j];

}

}

return B;

}

// Calculates Sum Of 2 Vectors

vector<int> Data::SumVectors(vector<int> A, vector<int> B)

{

vector<int> C(N);

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

{

C[i] = A[i] + B[i];

}

return C;

}

// Multiply Integer And Matrix

vector<int> Data::IntVectorMult(int a, vector<int> A)

{

vector<int> B(N);

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

{

B[i] = a \* A[i];

}

return B;

}

// Sort Vector

vector<int> Data::SortVector(vector<int> A)

{

sort(A.begin(), A.end());

return A;

}

// F1 -> D = SORT(A)+SORT(B)+SORT(C)\*(MA\*ME)

vector<int> Data::Func1(vector<int> A, vector<int> B, vector<int> C, vector<vector<int>> MA, vector<vector<int>> ME)

{

return SumVectors(SumVectors(SortVector(A), SortVector(B)), VectorMatrixMult(SortVector(C), MatrixMult(MA, ME)));

}

// F2 -> MF = (MG\*MH)\*TRANS(MK)

vector<vector<int>> Data::Func2(vector<vector<int>> MG, vector<vector<int>> MH, vector<vector<int>> MK)

{

return MatrixMult(MatrixMult(MG, MH), MatrixTransp(MK));

}

// F3 -> S = (MO\*MP)\*V+t\*MR\*(O+P)

vector<int> Data::Func3(int t, vector<int> V, vector<int> O, vector<int> P, vector<vector<int>> MO, vector<vector<int>> MP, vector<vector<int>> MR)

{

return SumVectors((VectorMatrixMult(V, MatrixMult(MO, MP))), IntVectorMult(t, VectorMatrixMult(SumVectors(O, P), MR)));

}

**Приклад роботи програми**

----------------------------------------------------

| Function 1 | D = SORT(A)+SORT(B)+SORT(C)\*(MA\*ME) |

| Function 2 | MF = (MG\*MH)\*TRANS(MK) |

| Function 3 | S = (MO\*MP)\*V+t\*MR\*(O+P) |

----------------------------------------------------

!!! Note that if the value of N > 10 -> the result will not be displayed !!!

!!! If you enter N <= 0 - execution will be terminated !!!

Enter N: 2

!!! Enter All Values From The Keyboard !!!

Lab4 started!

T3 started.

T1 started.

T2 started.

T1 is waiting for a permit.

T3 is waiting for a permit.

T2 is waiting for a permit.

T1 gets a permit.

Enter the 2 elements of the Vector A:

A[0] = 1

A[1] = 1

Enter the 2 elements of the Vector B:

B[0] = 1

B[1] = 1

Enter the 2 elements of the Vector C:

C[0] = 1

C[1] = 1

Enter the 4 elements of the Matrix MA:

MA[0][0] = 1

MA[0][1] = 1

MA[1][0] = 1

MA[1][1] = 1

Enter the 4 elements of the Matrix ME:

ME[0][0] = 1

ME[0][1] = 1

ME[1][0] = 1

ME[1][1] = 1

T1 releases the permit.

T1 is waiting for a permit.

T3 gets a permit.

Enter number t = 3

Enter the 2 elements of the Vector V:

V[0] = 3

V[1] = 3

Enter the 2 elements of the Vector O:

O[0] = 3

O[1] = 3

Enter the 2 elements of the Vector P:

P[0] = 3

P[1] = 3

Enter the 4 elements of the Matrix MO:

MO[0][0] = 3

MO[0][1] = 3

MO[1][0] = 3

MO[1][1] = 3

Enter the 4 elements of the Matrix MP:

MP[0][0] = 3

MP[0][1] = 3

MP[1][0] = 3

MP[1][1] = 3

Enter the 4 elements of the Matrix MR:

MR[0][0] = 3

MR[0][1] = 3

MR[1][0] = 3

MR[1][1] = 3

T3 releases the permit.

T3 is waiting for a permit.

T2 gets a permit.

Enter the 4 elements of the Matrix MG:

MG[0][0] = 2

MG[0][1] = 2

MG[1][0] = 2

MG[1][1] = 2

Enter the 4 elements of the Matrix MH:

MH[0][0] = 2

MH[0][1] = 2

MH[1][0] = 2

MH[1][1] = 2

Enter the 4 elements of the Matrix MK:

MK[0][0] = 2

MK[0][1] = 2

MK[1][0] = 2

MK[1][1] = 2

T2 releases the permit.

T2 is waiting for a permit.

T1 gets a permit.

T1 result:

Vector D: 6 6

T1 releases the permit.

T1 finished.

T3 gets a permit.

T3 result:

Vector S: 216 216

T3 releases the permit.

T3 finished.

T2 gets a permit.

T2 result:

Matrix MF:

32 32

32 32

T2 releases the permit.

T2 finished.

Lab4 finished!

Press any key to close this window . . .