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 №5

З ДИСЦИПЛІНИ “ОСНОВИ ПАРАЛЕЛЬНОГО ПРОГРАМУВАННЯ ”

НА ТЕМУ: “ПОТОКИ В БІБЛІОТЕЦІ OPENMP”

Виконав:
Студент III курсу ФІОТ
групи ІО-82
Шендріков Євгеній
Номер у списку - 25

Перевірив:
Доцент
Корочкін О. В.

м. Київ – 2020 р.

Завдання

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

2. Програма повинна складатися із пакету *Data* і основної програми — процедури (класу) *Lab5*. Пакет реалізовує ресурси, необхідні для обчислення функцій *F1, F2, F3* через підпрограми *T1, T2, T3*.

3. Кожен потік має здійснювати дії, необхідні для паралельного обчислення відповідної функції, а саме: введення відповідних даних, обчислення функції F1, F2, F3, виведення результату.

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

Варіант завдання

		F1	F2	F3	
25	Шендріков Євгеній Олександрович	1.21	2.27	3.30	

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 OpenMP на базі мови програмування C++ було розроблено відповідну програму, що містить паралельні потоки, кожен з яких реалізовує функцію згідно варіанту. Для кожного класу було створено заголовковий файл для опису використаних у класі методів та змінних.

Для роботи з векторами та матрицями було створено спеціальний пакет *Data*, який містить функціонал для реалізації функцій F1, F2, F3.

Також було зроблено спробу реалізувати введення даних з консолі, як і в попередніх лабораторних. 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 була спроба використати механізм критичних секцій (оскільки семафорів та мютексів в OpenMP нема, то вибір впав на них), щоб організувати взаємодію потоків для рішення задачі взаємного виключення до спільного ресурсу.

Однак, через вагання щодо правильності та коректності реалізації, а також через брак інформації (позаяк на лекціях ми це ще не проходили) дана ідея була відкинута та реалізовано лише «забиття» всіх елементів для T1 – як 1, для T2 – як 2, для T3 – як 3.

При N > 10 можливість виводити результат в консоль не передбачена.

Висновки

1. На основі засобів бібліотеки OpenMP на C++ було розроблено програму, яка містить паралельні потоки, кожен з яких реалізовує відповідну функцію *F1*, *F2*, *F3*. OpenMP дозволяє взяти послідовну програму, проаналізувати її, відокремити паралельні ділянки і отримати паралельну програму (шляхом вставки директив компілятора в потрібні місця вхідного коду послідовної програми). Також було розроблено пакет *Data*, який містить функціонал для роботи з матрицями та векторами.

2. За допомогою прагм *omp sections* та *omp section* тіло *omp parallel* (базова директива для визначення паралельних ділянок програми і створення потоків) було розподілено на секції, які будуть виконуватись в окремих потоках. Для створення точної кількості потоків, які необхідно створити, було задіяно директиву *omp set_num_threads()*.

3. Проблема роботи зі спільними ресурсами для потоків вирішувалась встановлення всіх значень для T1 – як 1, для T2 – як 2, для T3 – як 3. Була зроблена спроба використати механізм критичних секцій, та згодом ця ідея була відкинута через вагання щодо правильності та коректності реалізації.

4. Було зроблено аналіз проблем, які виникли під час виконання, перевірено працездатність програми і програмним шляхом оброблено виключні ситуації, наприклад, при N < 0 чи при N > 10 можливість виводити результат не передбачена також виводиться помилка при введенні даних відмінних від int.

Лістинг програми

Lab5.cpp

```
/*-------------------------------------------------------
 |                      Labwork #5                      |
 -------------------------------------------------------
 |  Author  |       Jack (Yevhenii) Shendrikov          |
 |  Group   |                IO-82                      |
 |  Variant |                 #25                       |
 |  Date    |             01.11.2020                    |
 -------------------------------------------------------
 | 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 <omp.h>
#include "F1.h"
#include "F2.h"
#include "F3.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);
	}

	//------------------------ Main Body --------------------------------
	cout << "\nLab5 started!\n" << endl;

	omp_set_num_threads(3);
	#pragma omp parallel
	{
		#pragma omp sections
		{
			#pragma omp section
			{
				F1 T1 = F1(N);
				T1.run();
			}
			#pragma omp section
			{
```

```
                        F2 T2 = F2(N);
                        T2.run();
                }
                #pragma omp section
                {
                        F3 T3 = F3(N);
                        T3.run();
                }
        }
    }

    cout << "\nLab5 finished!\n";
    cin.get();
}
```

T1.h

```
#pragma once
#include <omp.h>
#include <iostream>
#include <Windows.h>
#include "Data.h"

class F1 {
private:
    int N;
public:
    F1(int N);
    DWORD run();
};
```

T2.h

```
#pragma once
#include <omp.h>
#include <iostream>
#include <Windows.h>
#include "Data.h"

class F2 {
private:
    int N;
public:
    F2(int N);
    DWORD run();
};
```

T3.h

```
#pragma once
#include <omp.h>
#include <iostream>
#include <Windows.h>
#include "Data.h"

class F3 {
private:
    int N;
public:
    F3(int N);
    DWORD run();
};
```

T1.cpp

```
#include "F1.h"

F1::F1(int N) {
	this->N = N;
}

DWORD F1::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 << "\nT1 gets a permit.\n" << endl;

	A = data->FillVectorWithNumber(1);
	B = data->FillVectorWithNumber(1);
	C = data->FillVectorWithNumber(1);
	MA = data->FillMatrixWithNumber(1);
	ME = data->FillMatrixWithNumber(1);

	// Calculate The Result
	vector<int> result = data->Func1(A, B, C, MA, ME);
	Sleep(100);

	// Output
	if (N <= 10) {
		cout << "T1 result:\n";
		data->VectorOutput(result, 'D');
	}

	cout << "T1 releases the permit." << endl;
	cout << "T1 finished.\n" << endl;

	delete data;

	return 0;
}
```

T2.cpp

```
#include "F2.h"

F2::F2(int N) {
	this->N = N;
}

DWORD F2::run() {
	cout << "T2 started." << endl;
	Data* data = new Data(N);

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

	// Generate Input Values
	Sleep(100);
	cout << "\nT2 gets a permit.\n" << endl;

	MG = data->FillMatrixWithNumber(2);
	MH = data->FillMatrixWithNumber(2);
	MK = data->FillMatrixWithNumber(2);

	// Calculate The Result
	vector<vector<int>> result = data->Func2(MG, MH, MK);
```

```
	Sleep(100);

	// Output
	if (N <= 10) {
	cout << "T2 result:\n";
		data->MatrixOutput(result, "MF");
	}

	cout << "T2 releases the permit." << endl;
	cout << "T2 finished.\n" << endl;

	delete data;

	return 0;
}
```

T3.cpp

```
#include "F3.h"

F3::F3(int N) {
	this->N = N;
}

DWORD F3::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
	cout << "T3 gets a permit." << endl;

	t = 3;
	V = data->FillVectorWithNumber(3);
	O = data->FillVectorWithNumber(3);
	P = data->FillVectorWithNumber(3);
	MO = data->FillMatrixWithNumber(3);
	MP = data->FillMatrixWithNumber(3);
	MR = data->FillMatrixWithNumber(3);

	// Calculate The Result
	vector<int> result = data->Func3(t, V, O, P, MO, MP, MR);
	Sleep(100);

	// Output
	if (N <= 10) {
		cout << "T3 result:\n";
		data->VectorOutput(result, 'S');
	}

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

```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

Lab5 started!

T1 started.
T1 gets a permit.
T1 result:
        Vector D: 6 6
T1 releases the permit.
T1 finished.

T2 started.
T2 gets a permit.
T2 result:
        Matrix MF:
                32 32
                32 32
T2 releases the permit.
T2 finished.

T3 started.
T3 gets a permit.
T3 result:
        Vector S: 216 216
T3 releases the permit.
T3 finished.

Lab5 finished!

D:\C++\Lab5\Debug\Lab5.exe exited with code 0.
Press any key to close this window . . .
```