ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»
Факультет компьютерных наук
Департамент программной инженерии

НАХОЖДЕНИЕ РЕШЕНИЯ СИСТЕМЫ ЛИНЕЙНЫХ УРАВНЕНИЙ С ПОМОЩЬЮ ФОРМУЛ КРАМЕРА ИСПОЛЬЗУЯ OPENMP (вариант 8)

Отчёт

Дисциплина: «Архитектура вычислительных систем»

Исполнитель:
студент группы БПИ198
Гудзикевич М. С.

Москва 2020

СОДЕРЖАНИЕ

1. ТЕКСТ ЗАДАНИЯ

Изучить применение OpenMP для разработки многопоточных приложений. Используя формулы Крамера, найти решение системы линейных уравнений.

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + a_{14}x_4 = b_1 \\ a_{21}x_1 + a_{22}x_2 + a_{23}x_3 + a_{24}x_4 = b_2 \\ a_{31}x_1 + a_{32}x_2 + a_{33}x_3 + a_{34}x_4 = b_3 \\ a_{41}x_1 + a_{42}x_2 + a_{43}x_3 + a_{44}x_4 = b_4 \end{cases}$$

Предусмотреть возможность деления на ноль. Входные данные:
коэффициенты системы. Оптимальное количество потоков выбрать самостоятельно.

2. ПРИМЕНЯЕМЫЕ РАСЧЕТНЫЕ МЕТОДЫ

Программа запускается из консоли, обязательный единственный входной параметр – путь до файла, содержащего коэффициенты системы.

Формат строки с путём до файла test: path-to\test (например, C:\Users\max_g\Desktop\ABC\test1.txt).

Чтобы программа работала корректно – необходимо соблюдать формат входного файла. Формат – четыре строки, в каждой пять действительных чисел через пробел. Во избежание непредвиденного поведения не рекомендуется вводить большие числа (большие 10000 по модулю).

Программа работает по парадигме взаимодействующих данных [2]. Такая модель была выбрана в связи с тем, что в данной задаче мы имеем фиксированное количество подзадач – мы должны посчитать 5 различных определителей квадратных матриц размера 4. Логично распараллелить задачу, чтобы не тратить много времени на последовательное вычисление определителей. Для распараллеливания используется стандарт OpenMP – в частности объявление директивы #pragma omp for для параллельного запуска потоков в цикле.

Программа работает следующим образом: она считывает данные из файла, преобразовывая их в матрицу (для этого специально написан класс Matrix). Затем, мы запускаем пять потоков – i-ый для вычисления определителя матрицы СЛАУ без столбца i. Определитель – статическая функция, считается рекурсивно методом разложения по первой строке. Далее проверяется определитель матрицы A на равенство 0, после чего в консоль выводится решение системы с помощью правила Крамера (или программа сообщает о вырожденности матрицы A).

3. ТЕСТИРОВАНИЕ ПРОГРАММЫ

```
C:\Users\maxma\Desktop\Ass\task04>task4 C:\Users\maxma\Desktop\Ass\task04\tests\test1.txt
Input coeffs:
1       2       3       4       5
6       7       8       6       4
1       2       2       7       1
3       2       4       5       5

x1 = -1.85366
x2 = -2.09756
x3 = 3.78049
x4 = -0.0731707
```

Рис. 1 – Программа даёт верный ответ на простой пример

```
C:\Users\maxma\Desktop\Ass\task04>task4 C:\Users\maxma\Desktop\Ass\task04\tests\test3.txt
Input coeffs:
10.1    12.2    3.09    -7.15   4
121.01  -2      3       43.2    1
1       3       3       2       1
-10     20      5       6       1

x1 = 0.0720713
x2 = 0.0478991
x3 = 0.397491
x4 = -0.204121
```

Рис. 2 – Программа даёт верный ответ на пример посложнее (отриц. и нецелые числа)

```
C:\Users\maxma\Desktop\Ass\task04>task4 C:\Users\maxma\Desktop\Ass\task04\tests\test2.txt
Input coeffs:
1       2       0       3       1
3       2       0       0       2
2       0       0       1       -4
0       0       0       -1      -4

Det(A) = 0! Kramer's method is useless here (the system has 0 or +oo number of solutions)
```

Рис. 3 – Программа сообщает о делении на 0 в формуле Крамера

```
C:\Users\maxma\Desktop\Ass\task04>task4 badpath
Can't find the file in a directory: badpath

C:\Users\maxma\Desktop\Ass\task04>task4 arg1 arg2
Your input is incorrect, only one argument (input data path) expected.
```

Рис. 4 – Если программе ввести неверный путь или несколько аргументов – она сообщит об этом

4. ИСТОЧНИКИ

1. SoftCraft, сайт по учебной дисциплине. [Электронный ресурс] http://softcraft.ru/ (дата обращения: 25.11.2020).
2. Парадигмы параллельного программирования, Блог Программиста. [Электронный ресурс] https://pro-prof.com/forums/topic/parallel-programming-paradigms (дата обращения: 26.11.2020).
3. Что такое OpenMP? Parallel.ru [Электронный ресурс] https://parallel.ru/tech/tech_dev/openmp.html (дата обращения: 26.11.2020).

ПРИЛОЖЕНИЕ 1

КОД ПРОГРАММЫ

```
#include <iostream>
#include <fstream>
#include <vector>
#include <omp.h>

using namespace std;

class Matrix // Класс матриц
{
private:
	int n, m; // n columns and m rows
	vector<double> data;

public:
	Matrix(int cols, int rows)
	{
		n = cols;
		m = rows;
		data = vector<double>(n * m);
	}

	double& operator()(int i, int j)
	{
		return data[i * n + j];
	}

	double operator()(int i, int j) const
	{
		return data[i * n + j];
	}

	void print() const
	{
		for (int i = 0; i < m; ++i) {
			for (int j = 0; j < n; ++j)
				cout << this->operator()(i, j) << "\t";
			cout << endl;
		}
	}

	bool isSquare() const { return n == m && n > 0; }
	int getN() const { return n; }
	int getM() const { return m; }

	Matrix excludeCol(int q) const // returns original matrix without column q
	{
		Matrix temp(getN() - 1, getM());
		int i = 0, j = 0; // here we save indexes of elements in temp

		if (q == -1)
			q = n - 1;

		for (int row = 0; row < m; ++row)
		{
			for (int col = 0; col < n; ++col)
			{
				if (col != q)
				{
					temp(i, j) = this->operator()(row, col);
					j++;
```

```
                                if (j == n - 1)
                                {
                                        j = 0;
                                        i++;
                                }
                        }
                }
        }
        return temp;
}

Matrix cofactor(int p, int q) const
{
        Matrix temp(getN() - 1, getM() - 1);
        int i = 0, j = 0; // here we save indexes of elements in temp

        for (int row = 0; row < m; ++row)
        {
                for (int col = 0; col < n; ++col)
                {
                        if (row != p && col != q)
                        {
                                temp(i, j) = this->operator()(row, col);
                                j++;

                                if (j == n - 1)
                                {
                                        j = 0;
                                        i++;
                                }
                        }
                }
        }
        return temp;
}

static double det(const Matrix& m)
{
        if (!m.isSquare())
                throw new exception();

        double D = 0;
        int n = m.getN();

        if (n == 1)
                return m(0, 0);

        Matrix temp(n - 1, n - 1);
        int sign = 1;

        for (int j = 0; j < n; j++)
        {
                temp = m.cofactor(0, j);
                D += sign * m(0, j) * Matrix::det(temp);
                sign = -sign;
        }

        return D;
}
};

void solve(Matrix& m)
{
```

```
	double x[5];

	#pragma omp parallel num_threads(5)
	{
		#pragma omp for
		for (int i = 0; i < 5; ++i)
		{
			Matrix temp = m.excludeCol(i);
			x[i] = Matrix::det(temp);
		}

	}

	if (x[4] == 0)
		cout << "Det(A) = 0! Kramer's method is useless here (the system has 0 or
+oo number of solutions)" << endl;
	else
	{
		cout << "x1 = " << -x[0] / x[4] << endl;
		cout << "x2 = " << x[1] / x[4] << endl;
		cout << "x3 = " << -x[2] / x[4] << endl;
		cout << "x4 = " << x[3] / x[4] << endl;
	}

	return;
}

int main(int argc, char** argv)
{
	if (argc != 2) {
		cout << "Your input is incorrect, only one argument (input data path)
expected." << endl;
		return -1;
	}

	ifstream in(argv[1]);
	Matrix m(5, 4);

	if (!in.is_open()) {
		cout << "Can't find the file in a directory: " << argv[1] << endl;
		return -1;
	}

	for (int i = 0; i < 4; ++i)
		for (int j = 0; j < 5; ++j)
			in >> m(i, j);

	cout << "Input coeffs:\n";
	m.print();
	cout << endl;

	solve(m);

	return 0;
}
```