**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ**

Федеральное ГОСУДАРСТВЕННОЕ бюджетное ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ

«ПОВОЛЖСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет информатики и вычислительной техники

Кафедра ИиСП

**Лабораторная работа**

**по дисциплине «Параллельное программирование»**

Выполнил:

студент группы ПС-31

факультета Информатики

и Вычислительной Техники

специальности «Программная инженерия»

Смиренский А.А.

Проверил:

Филимонов А.

г. Йошкар-Ола

2016

# Постановка задачи

Найти алгебраическое дополнение для каждого элемента матрицы.

Входные данные: произвольная матрица А размерности n х n. Заполнение начальной матрицы реализовать в главном потоке приложения.

1. Реализовать последовательный вариант программы для указанного варианта.
2. Реализовать параллельный вариант программы. Количество потоков выполнения должно являться входным параметром задачи.
3. После завершения программа должна выдавать время своей работы. Подобрать размеры матриц таким образом, чтобы время работы последовательного варианта составляло не менее одной секунды.
4. Посчитать параметры качества вашей параллельной программы и построить в Excel графики для количества потоков = [1;16] и количестве задействованных ядер = [1;4] (используйте функцию SetAffinityMask).
5. Время выполнения
6. Ускорение
7. Эффективность распараллеливания.

# Листинг кода

# .cpp

#include "stdafx.h"

#include "ParamsThread.h"

void FindAlgebraicAddition(const std::vector<std::vector<int>> & matrix, int n, int m)

{

std::vector<std::vector<int>> minor(matrix.size() - 1, std::vector<int>(matrix.size() - 1, 0));

int nShift = 0, mShift = 0;

for (int i = 0; i < matrix.size(); i++)

{

mShift = 0;

if (i == n)

{

nShift = -1;

continue;

}

for (int k = 0; k < matrix[i].size(); k++)

{

if (k == m)

{

mShift = -1;

continue;

}

minor[i + nShift][k + mShift] = matrix[i][k];

}

}

int algebraicAddition = (-1) ^ (n + m);

}

DWORD WINAPI WriteCollumMinor(LPVOID lpParams)

{

Params\* params = (Params\*)lpParams;

for (int i = params->startPos; i < params->endPos; i++)

for (int k = 0; k < (\*(params->matrix))[i].size(); k++)

FindAlgebraicAddition((\*(params->matrix)), k, i);

return 0;

}

void FindAlgebraicAdditionParallel(const std::vector<std::vector<int>> & matrix)

{

HANDLE\* handles = new HANDLE[2];

Params\* params = new Params[2];

params[0].startPos = 0;

params[0].endPos = (int)(matrix.size() / 2);

params[0].matrix = &matrix;

params[1].startPos = params[0].endPos;

params[1].endPos = matrix.size();

params[1].matrix = &matrix;

handles[0] = CreateThread(NULL, 0, (LPTHREAD\_START\_ROUTINE)WriteCollumMinor, (LPVOID)&(params[0]), CREATE\_SUSPENDED, NULL);

handles[1] = CreateThread(NULL, 0, (LPTHREAD\_START\_ROUTINE)WriteCollumMinor, (LPVOID)&(params[1]), CREATE\_SUSPENDED, NULL);

ResumeThread(handles[0]);

ResumeThread(handles[1]);

WaitForMultipleObjects(2, handles, true, INFINITE);

delete(handles);

delete(params);

}

int main()

{

unsigned int startTime = clock();

std::vector<std::vector<int>> matrix(100, std::vector<int>(100, 1));

for (int i = 0; i < matrix.size(); i++)

for (int k = 0; k < matrix[i].size(); k++)

FindAlgebraicAddition(matrix, k, i);

unsigned int endTime = clock();

float searchTime = (endTime - startTime) / 1000.0;

std::cout << "Sequence run time: " << searchTime << std::endl;

startTime = clock();

FindAlgebraicAdditionParallel(matrix);

endTime = clock();

searchTime = (endTime - startTime) / 1000.0;

std::cout << "Parallel run time: " << searchTime << std::endl;

return 0;

}

# .h

#pragma once

#include <conio.h>

#include <stdio.h>

#include <windows.h>

#include <vector>

#include <ctime>

#include <iostream>

struct Params

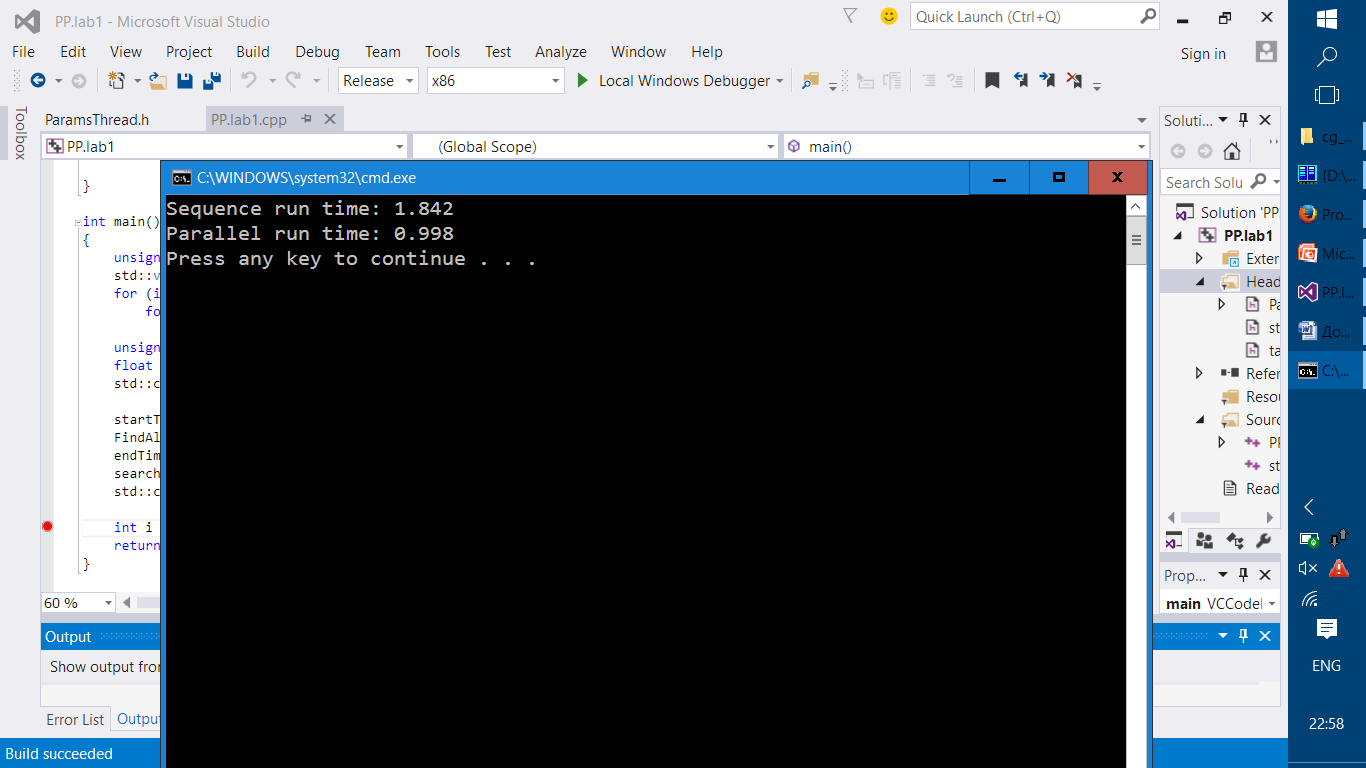
{

int startPos, endPos;

const std::vector<std::vector<int>> \*matrix;

};

# Вывод



При параллельном методе видно что время работы уменьшилось, примерно в 2 раза.