Московский Авиационный Институт

(Национальный Исследовательский Университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

**Лабораторная работа № 3 по курсу**

**«Операционные системы»**

Студент: Меджидли Исмаил Ибрагим оглы

Группа: М8О-201Б-21

Вариант: 10

Преподаватель: Миронов Евгений Сергеевич

Оценка: \_\_\_\_\_\_\_\_\_\_\_

Дата: \_\_\_\_\_\_\_\_\_\_\_

Подпись: \_\_\_\_\_\_\_\_\_\_\_

Москва, 2023

**Содержание**

1. Репозиторий
2. Постановка задачи
3. Общие сведения о программе
4. Общий метод и алгоритм решения
5. Исходный код
6. Демонстрация работы программы
7. Выводы

**Репозиторий**

https://github.com/imedzhidli/Operational-Systems

**Постановка задачи**

Составить программу на языке Си, обрабатывающую данные в многопоточном режиме. При

обработки использовать стандартные средства создания потоков операционной системы

(Windows/Unix). Ограничение потоков должно быть задано ключом запуска вашей программы.

Так же необходимо уметь продемонстрировать количество потоков, используемое вашей

программой с помощью стандартных средств операционной системы.

Наложить K раз медианный фильтр на матрицу, состоящую из целых чисел. Размер окна задается

**Общие сведения о программе**

CMakeLists.txt - описание процесса сборки проекта  
  
main.cpp - считывание и вывод данных  
  
laba3OS.h - заголовочный файл, описаны функции для работы с матрицей  
  
laba3OS.cpp - реализация функций, которые определены в laba3OS.h

utils.h - полезные функции

lab3\_test.cpp - тесты к лабораторной работе

**Общий метод и алгоритм решения**

Программе в качестве Параметров подаётся общее число потоков. Если

оно не задано, то программа запускается в двупоточном режиме. Потом задаётся размер матрицы, размер окна и количество фильтров.

Программа разделяет матрицу на строки и дает обрабатывать одному потоку количество строк так, чтобы каждому потоку досталось одинаковое количество строк (бывают случаи, когда один поток обрабатывает на 1 строку больше чем остальные). Потоки никак не пересекаются, когда обращаются к матрице, ведь программа работает по принципу двух матриц: если нужно наложить 1 раз медианный фильтр, результаты каждой ячейки кладутся во 2-ую матрицу, если же надо 2 раза наложить фильтр, то исходной матрицей для потоков становится матрица 2, а матрица 1 становится той, куда записывается ответ.

**Исходный код**

**CMakeLists.txt**

**cmake\_minimum\_required(VERSION 3.16.3)**

**project(lab3 LANGUAGES CXX)**

**find\_package(Threads REQUIRED)**

**add\_executable(lab3**

**main.cpp**

**include/laba3OS.h src/laba3OS.cpp)**

**target\_include\_directories(lab3 PRIVATE include)**

**target\_link\_libraries(lab3 PRIVATE Threads::Threads)**

**laba3OS.h**

**#ifndef OS\_LABS\_LAB3\_H**

**#define OS\_LABS\_LAB3\_H**

**#include <vector>**

**#include <iostream>**

**#include <string>**

**#include <pthread.h>**

**#define MAX(a, b) ((a) > (b) ? (a) : (b))**

**#define MIN(a, b) ((a) < (b) ? (a) : (b))**

**using namespace std;**

**using TMatrix = vector<vector<int>>;**

**struct thread\_args{**

**int height;**

**int width;**

**TMatrix \*first\_matrix;**

**TMatrix \*second\_matrix;**

**int area;**

**int lower\_row;**

**int upper\_row;**

**};**

**void ReadMatrix(TMatrix &matrix);**

**void WriteMatrix(TMatrix &matrix);**

**int median(vector <int> &matrix);**

**void median\_filter(int &height, int &width, TMatrix &a, TMatrix &b, int &area, int &lower\_row, int &upper\_row);**

**void First(int &n, int &m, TMatrix &mat1, TMatrix &mat2, int &k, int &window\_size, int &threadCount, TMatrix &res);**

**#endif //OS\_LABS\_LAB3\_H**

**laba3OS.cpp**

**#include "laba3OS.h"**

**#include "utils.h"**

**void ReadMatrix(TMatrix &matrix) {**

**for (int i = 0; i < Isize(matrix); i++) {**

**for (int j = 0; j < Isize(matrix[i]); j++) {**

**cin >> matrix[i][j];**

**}**

**}**

**}**

**void WriteMatrix(TMatrix &matrix) {**

**for (int i = 0; i < Isize(matrix); i++) {**

**for (int j = 0; j < Isize(matrix[i]); j++) {**

**cout << matrix[i][j] << " ";**

**}**

**cout << "\n";**

**}**

**}**

**int median(vector <int> &matrix){**

**int n = matrix.size();**

**for(int i = 0; i < n; i++){**

**for(int j = i; j < n; j++){**

**if(matrix[i] > matrix[j]){**

**int temp = matrix[i];**

**matrix[i] = matrix[j];**

**matrix[j] = temp;**

**}**

**}**

**}**

**return matrix[n / 2];**

**}**

**void median\_filter(int &height, int &width, TMatrix &a, TMatrix &b, int &area, int &lower\_row, int &upper\_row){**

**int radius = (area - 1) / 2;**

**for (int y = lower\_row; y < upper\_row; y++){**

**int top = MAX(y - radius, 0);**

**int bottom = MIN(y + radius, height - 1);**

**for (int x = 0; x < width; x++){**

**int left = MAX(x - radius, 0);**

**int right = MIN(x + radius, width - 1);**

**vector <int> m((bottom - top + 1) \* (right - left + 1));**

**int k = 0;**

**for (int v = top; v <= bottom; v++){**

**for (int u = left; u <= right; u++){**

**m[k] = a[v][u];**

**k++;**

**}**

**}**

**b[y][x] = median(m);**

**}**

**}**

**}**

**void \*lineresation(void \*args){**

**thread\_args \*arg = (thread\_args\*) args;**

**median\_filter(arg->height, arg->width, \*arg->first\_matrix, \*arg->second\_matrix, arg->area, arg->lower\_row, arg->upper\_row);**

**return NULL;**

**}**

**void First(int &n, int &m, TMatrix &mat1, TMatrix &mat2, int &k, int &window\_size, int &threadCount, TMatrix &res){**

**int imedy = n / threadCount;**

**pthread\_t threads[threadCount];**

**pthread\_attr\_t attr;**

**pthread\_attr\_init(&attr);**

**thread\_args p\_args[threadCount];**

**for (int i = 0; i < k; i++){**

**for(int j = 0; j < threadCount; j++){**

**p\_args[j].height = n;**

**p\_args[j].width = m;**

**if (i % 2 == 0){**

**p\_args[j].first\_matrix = &mat1;**

**p\_args[j].second\_matrix = &mat2;**

**}**

**else{**

**p\_args[j].first\_matrix = &mat2;**

**p\_args[j].second\_matrix = &mat1;**

**}**

**p\_args[j].area = window\_size;**

**p\_args[j].lower\_row = j \* imedy;**

**p\_args[j].upper\_row = (j + 1) \* imedy;**

**if (j == threadCount - 1)**

**p\_args[j].upper\_row = n;**

**cout << "Поток "<< j+1 << " работает с [" << p\_args[j].lower\_row <<","<< p\_args[j].upper\_row << ") строкой(ами)" << "\n";**

**}**

**for (int i = 0; i < threadCount; i++){**

**int res = pthread\_create(&threads[i], &attr, lineresation, &p\_args[i]);**

**if (res != 0){**

**cout << "Ошибка с созданием потока!";**

**exit(-2);**

**}**

**}**

**for (int i = 0; i < threadCount; i++){**

**int res = pthread\_join(threads[i], NULL);**

**if (res != 0){**

**cout << "Ошибка с ожиданием выхода!";**

**exit(-3);**

**}**

**}**

**printf("-------------------------------------------------\n");**

**}**

**if (k % 2 == 1){**

**res = mat2;}**

**else {**

**res = mat1;}**

**}**

**utils.h**

**#ifndef OS\_LABS\_UTILS\_H**

**#define OS\_LABS\_UTILS\_H**

**template <typename Container>**

**inline int Isize(const Container& cont) {**

**return static\_cast<int>(cont.size());**

**}**

**#endif //OS\_LABS\_UTILS\_H**

**main.cpp**

**#include "laba3OS.h"**

**int main(int argc, char\* argv[]){**

**int n, m, threadCount;**

**cout << "Введите кол-во потоков: ";**

**cin >> threadCount;**

**cout << "Введите размеры матрицы: ";**

**cin >> n >> m;**

**TMatrix mat1(n, vector<int>(m));**

**TMatrix mat2(n, vector<int>(m));**

**if (threadCount > n) threadCount = n;**

**int window\_size;**

**printf("Введите размер окна (от 3 и более нечетное число): ");**

**cin >> window\_size;**

**printf("Введите кол-во медианных фильтров: ");**

**int k;**

**cin >> k;**

**printf("Введите вашу матрицу:\n");**

**ReadMatrix(mat1);**

**TMatrix res;**

**First(n, m, mat1, mat2, k, window\_size, threadCount, res);**

**WriteMatrix(res);**

**return 0;**

**}**

**lab3\_test.cpp**

**#include <gtest/gtest.h>**

**#include "laba3OS.h"**

**#include "utils.h"**

**#include <chrono>**

**namespace {**

**TMatrix GenerateMatrix(int n, int m) {**

**TMatrix result(n, vector<int>(m));**

**srand(time(nullptr));**

**for(int i = 0; i < n; ++i) {**

**for(int j = 0; j < m; ++j) {**

**result[i][j] = rand() % 100;**

**}**

**}**

**return result;**

**}**

**}**

**bool operator==(const TMatrix& lhs, const TMatrix& rhs) {**

**if(lhs.size() != rhs.size()) {**

**return false;**

**}**

**for(int i = 0; i < Isize(lhs); ++i) {**

**if(lhs[i].size() != rhs[i].size()) {**

**return false;**

**}**

**for(int j = 0; j < Isize(lhs); ++j) {**

**if(lhs[i][j] != rhs[i][j]) {**

**return false;**

**}**

**}**

**}**

**return true;**

**}**

**TEST(Lab3Test, MedianTest) {**

**int expectedres = 3;**

**int expectedres2 = 2;**

**vector <int> beforefunc = {0, 0, 1, 20, 34, 3, 50};**

**vector <int> beforefunc2 = {0, 2, 1, 3};**

**int res = median(beforefunc);**

**int res2 = median(beforefunc2);**

**EXPECT\_EQ(expectedres, res);**

**EXPECT\_EQ(expectedres2, res2);**

**}**

**TEST(Lab3Test, FirstTest) {**

**srand(time(NULL));**

**auto getAvgTime = [](int thread\_count) {**

**int n = 3; int m = 3;**

**int window\_size = 3;**

**// int thread\_count = 2;**

**int k = 2;**

**TMatrix expectedMatrix{**

**{1, 1, 2},**

**{1, 1, 2},**

**{1, 1, 3}**

**};**

**TMatrix mat1 = {**

**{1, 2, 3},**

**{1, 1, 0},**

**{1, 4, 5}**

**};**

**TMatrix mat2(n, vector<int>(m));**

**TMatrix res;**

**double avg = 0;**

**auto begin = chrono::high\_resolution\_clock::now();**

**First(n, m, mat1, mat2, k, window\_size, thread\_count, res);**

**EXPECT\_EQ(res, expectedMatrix);**

**auto end = chrono::high\_resolution\_clock::now();**

**avg += chrono::duration\_cast<chrono::milliseconds>(end - begin).count();**

**return avg / 10;**

**};**

**auto singleThread = getAvgTime(1);**

**auto multiThread = getAvgTime(3);**

**cout << "Avg time for 1 thread: " << singleThread << '\n';**

**cout << "Avg time for 3 threads: " << multiThread << '\n';**

**EXPECT\_LE(singleThread, multiThread);**

**}**

**Демонстрация работы программы**

imedzhidli@imedzhidli:~/Desktop/OS/3Laba/build$ /home/imedzhidli/Desktop/OS/3Laba/build/lab3

Введите кол-во потоков: 3

Введите размеры матрицы: 3 3

Введите размер окна (от 3 и более нечетное число): 3

Введите кол-во медианных фильтров: 2

Введите вашу матрицу:

1 2 3

1 1 0

1 4 5

Поток 1 работает с [0,1) строкой(ами)

Поток 2 работает с [1,2) строкой(ами)

Поток 3 работает с [2,3) строкой(ами)

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

Поток 1 работает с [0,1) строкой(ами)

Поток 2 работает с [1,2) строкой(ами)

Поток 3 работает с [2,3) строкой(ами)

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

1 1 2

1 1 2

1 1 3

imedzhidli@imedzhidli:~/Desktop/OS/3Laba/build$

**Выводы**Я приобрел практические навыки в:

1. Управлении потоками в ОС
2. Обеспечении синхронизации между потоками