Московский Авиационный Институт

(Национальный Исследовательский Университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

**Лабораторная работа № 3 по курсу**

**«Операционные системы»**

**Процессы и потоки**

Студент: Пирогов М.Д.

Группа: М8О-207Б-21

Вариант: 10

Преподаватель: Миронов Евгений Сергеевич

Оценка: \_\_\_\_\_\_\_\_\_\_\_

Дата: \_\_\_\_\_\_\_\_\_\_\_

Подпись: \_\_\_\_\_\_\_\_\_\_\_

Москва, 2022

**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/pirogovmark/OS-Labs](https://github.com/pirogovmark/OS-Labs/Lab)

**Постановка задачи**

**Цель работы**

Целью является приобретение практических навыков в:

* Управление потоками в ОС
* Обеспечение синхронизации между потоками

**Задание**

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.

В отчете привести исследование зависимости ускорения и эффективности алгоритма от входящих

данных и количества потоков. Получившиеся результаты необходимо объяснить.

**Вариант 10)** Наложить K раз медианный фильтр на матрицу, состоящую из целых чисел. Размер окна задается.

**Общие сведения о программе**

Программа компилируется из файла main.cpp. Также используется заголовочные файлы: unistd.h, ...

**Общий метод и алгоритм решения**

Обработка матрицы будет распределяться по строкам для потоков. Каждую матрицу будем дополнять элементами, которые не будут влиять на подсчет “среднего” значения в области. Алгоритм медианного фильтрования: для каждого элемента в области NxN (N - odd) добавляем элементы из этой области в список, сортируем быстрой сортировкой и берем серединный элемент, который уже и записываем в новую матрицу.

**Исходный код**

#include <iostream>

#include <unistd.h>

#include <time.h>

#include <stdlib.h>

#include <pthread.h>

int const N = 20;

int matrix[20][20], new\_matrix[20][20];

int window, frame;

typedef struct threadArguments{

int numberOfThreads;

int currentThread;

} threadArguments;

void quickSort(int \*array, int low, int high) {

int i = low;

int j = high;

int pivot = array[(i + j) / 2];

int temp;

while (i <= j) {

while (array[i] < pivot) i++;

while (array[j] > pivot) j--;

if (i <= j) {

temp = array[i];

array[i] = array[j];

array[j] = temp;

++i;

--j;

}

}

if (j > low) quickSort(array, low, j);

if (i < high) quickSort(array, i, high);

}

void printAugMatrix(int matrix[N][N]) {

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

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

std::cout << matrix[i][j] << ' ';

}

std::cout << '\n';

}

}

void printMatrix(int matrix[N][N]) {

for (int i = frame; i < N - frame - 1; ++i) {

for (int j = frame; j < N - frame - 1; ++j) {

std::cout << matrix[i][j] << ' ';

}

std::cout << '\n';

}

}

void fillMatrix() {

srand(time(NULL));

// Заполнение матрицы внутри

for (int i = frame; i < N - frame; ++i) {

for (int j = frame; j < N - frame; ++j) {

matrix[i][j] = rand() % 9 + 1;

}

}

// Дополнение матрицы для крайних элементов

for (int i = 1; i < N - 1; ++i) {

for (int j = 0; j < frame; ++j) {

matrix[i][j] = matrix[i][frame];

matrix[i][N - j - 1] = matrix[i][N - frame - 1];

}

}

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

for (int j = 0; j < frame; ++j) {

matrix[j][i] = matrix[frame][i];

matrix[N - j - 1][i] = matrix[N - frame - 1][i];

}

}

}

void resultToMatrix() {

for (int i = frame; i < N - frame; ++i) {

for (int j = frame; j < N - frame; ++j) {

matrix[i][j] = new\_matrix[i][j];

}

}

for (int i = 1; i < N - 1; ++i) {

for (int j = 0; j < frame; ++j) {

matrix[i][j] = matrix[i][frame];

matrix[i][N - j - 1] = matrix[i][N - frame - 1];

}

}

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

for (int j = 0; j < frame; ++j) {

matrix[j][i] = matrix[frame][i];

matrix[N - j - 1][i] = matrix[N - frame - 1][i];

}

}

}

int medianForElem(int i, int j) {

int frame = window / 2;

int sortSize = window \* window - 1;

int numbers[sortSize];

int counter = 0;

for (int l = i - frame; l <= i + frame; ++l) {

for (int k = j - frame; k <= j + frame; ++k) {

numbers[counter] = matrix[l][k];

++counter;

}

}

quickSort(numbers, 0, sortSize - 1);

return numbers[sortSize / 2];

}

void filterString(int stringNumber) {

for (int j = frame; j < N - frame; ++j) {

new\_matrix[stringNumber][j] = medianForElem(stringNumber, j);

}

}

void\* threadFilter(void\* arg) {

threadArguments data = \*((threadArguments\*) arg);

for (int i = data.currentThread; i < N; i += data.numberOfThreads) {

filterString(i);

}

}

int main() {

int numberOfThreads;

std::cout << "Enter the number of threads: ";

std::cin >> numberOfThreads;

std::cout << "Enter window size: ";

std::cin >> window;

frame = window / 2;

int overlays;

std::cout << "Enter the number of overlays: ";

std::cin >> overlays;

fillMatrix();

printMatrix(matrix);

std::cout << '\n';

double start = clock();

for (int k = 0; k < overlays; ++k) {

pthread\_t threads[numberOfThreads];

threadArguments\* data = (threadArguments\*)malloc(sizeof(threadArguments) \* numberOfThreads);

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

data[i].currentThread = i;

data[i].numberOfThreads = numberOfThreads;

}

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

if (pthread\_create(&threads[i], NULL, &threadFilter, &data[i]) != 0) {

std::cout << "Failed to create thread\n";

return 1;

}

}

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

if (pthread\_join(threads[i], NULL) != 0) {

std::cout << "Failed to join thread\n";

return 1;

}

}

resultToMatrix();

}

std::cout << "\nNew matrix:\n";

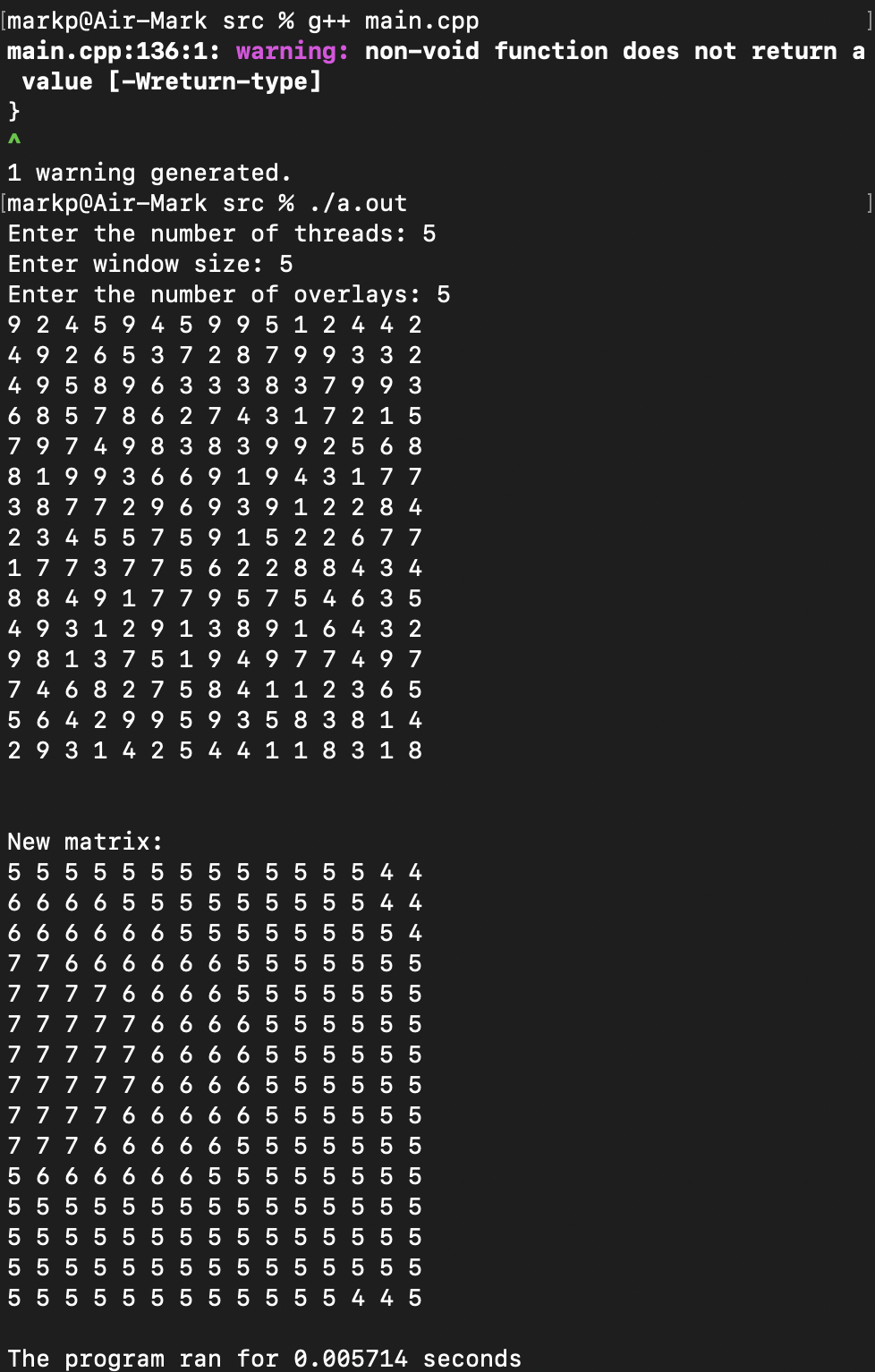
printMatrix(matrix);

std::cout << "\nThe program ran for " << (clock() - start) / (CLOCKS\_PER\_SEC) << " seconds\n";

return 0;

}

**Демонстрация работы программы**



**Выводы**

В результате выполнения этой лабораторной работы я познакомился с потоками в ОС, управлять ими и создавать многопоточные программы.