**Липецкий государственный технический университет**

Факультет автоматизации и информатики

Кафедра автоматизированных систем управления

ЛАБОРАТОРНАЯ РАБОТА №1

по дисциплине

«Операционные системы»

«Основы работы с потоками»

Вариант 2

Студент

Группа АИ-20-1 \_\_\_\_\_\_\_\_\_\_ Глубоков Г.В.

Подпись, дата

Руководитель

к.т.н.,доц. Батищев Р. В.

Подпись, дата

Липецк 2022г.

# Задание кафедры:

Написать программу параллельной многопоточной генерации и обработки элементов строк двумерной матрицы (каждый поток генерирует одну строку и осуществляет ее обработку) с выводом в дополнительном потоке результата обработки (вид операции и размерность матрицы выбираются из табл. 3 прил.). В каждый поток, выполняющий генерацию и обработку, следует передавать в качестве параметра указатель на структуру, поля которой соответствуют строке матрицы и результату обработки.

# Теоретические сведения:

Для создания нового потока используется функция **pthread\_create()**.

Новый поток начинает выполнение с вызова функции, указанной в виде значения start и принимающей аргумент arg (то есть start(arg)). Поток, который вызвал **pthread\_ create()**, продолжает работу, выполняя инструкцию, следующую за данным вызовом

Функция **pthread\_join()** ждет завершения потока, обозначенного аргументом thread (если поток уже завершился, она сразу же возвращается). Эта операция называется присоединением.

# Ход работы:

## Код программы на C++:

#include <pthread.h>

#include <stdlib.h>

#include <stdio.h>

#include <iostream>

//размеры матриц

#define N 5

#define M 5

//специальная структура для данных потока

typedef struct{

int rowN; //номер обрабатываемой строки

int rowSize; //размер строки

int\*\* resArr;

} pthrData;

void\* threadFunc(void\* thread\_data){

//получаем структуру с данными

pthrData \*data = (pthrData\*) thread\_data;

for(int i = 0; i < data->rowSize; i++){

data->resArr[data->rowN][i] = rand()%10;

}

return NULL;

}

void\* MaxElement(void\* thread\_data){

int\*\* data = (int\*\*) thread\_data;

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

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

if(data[i][j]>=data[0][0]){

data[0][0]=data[i][j];

}

}

}

std::cout<<'\n'<<data[0][0];

return NULL;

}

int main(){

//выделяем память под двумерные массивы

int\*\* resultMatrix = (int\*\*) malloc(N \* sizeof(int\*));

//выделяем память под элементы матриц

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

resultMatrix[i] = (int\*) malloc(M \* sizeof(int));

}

//выделяем память под массив идентификаторов потоков

pthread\_t\* threads = (pthread\_t\*) malloc(N+1 \* sizeof(pthread\_t));

//сколько потоков - столько и структур с потоковых данных

pthrData\* threadData = (pthrData\*) malloc(N \* sizeof(pthrData));

//инициализируем структуры потоков

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

threadData[i].rowN = i;

threadData[i].rowSize = M;

threadData[i].resArr = resultMatrix;

//запускаем поток

pthread\_create(&(threads[i]), NULL, threadFunc, &threadData[i]);

}

//ожидаем выполнение всех потоков

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

pthread\_join(threads[i], NULL);

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

std::cout<<'\n';

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

std::cout<<resultMatrix[i][j];

}

}

pthread\_t t1;

pthread\_create(&t1, NULL, MaxElement, resultMatrix);

pthread\_join(t1, NULL);

//освобождаем память

free(threads);

free(threadData);

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

free(resultMatrix[i]);

}

free(resultMatrix);

return 0;

## }Пример выполнения:

На рисунке 1 показано выполнение программы.

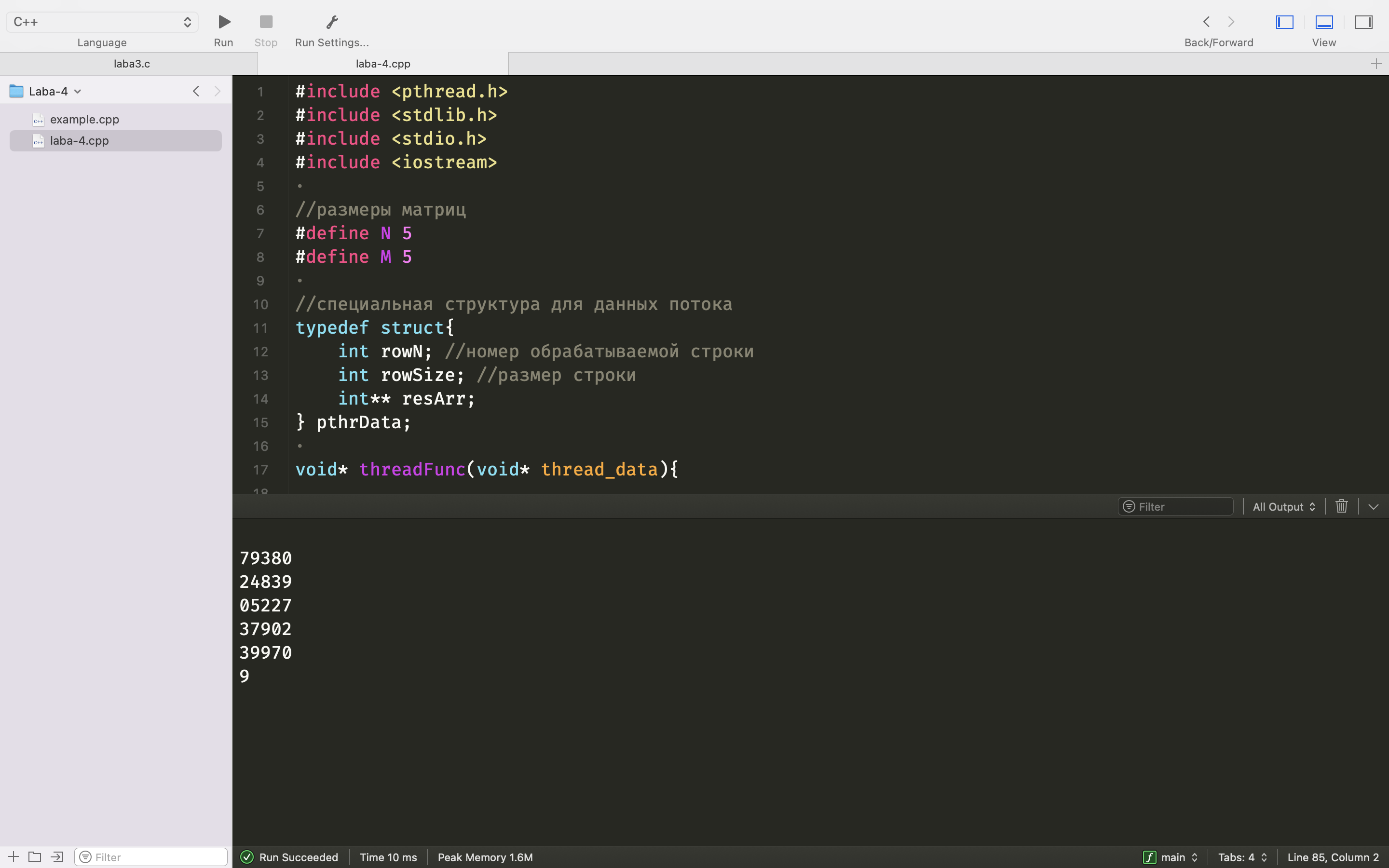


Рисунок 1 – Генерация матрицы и поиск максимума