Московский Авиационный Институт

(Национальный Исследовательский Университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

**Лабораторная работа №2 по курсу**

**«Операционные системы»**

**УПРАВЛЕНИЕ ПОТОКАМИ В ОС**

Студент: Басова Татьяна Валентиновна

Группа: М8О–212Б–22

Вариант: 6

Преподаватель: Соколов Андрей Алексеевич

Оценка: \_\_\_\_\_\_\_\_\_\_

Дата: \_\_\_\_\_\_\_\_\_\_

Подпись: \_\_\_\_\_\_\_\_\_\_

Москва, 2023.

**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 максимального количества потоков, работающих в один момент времени, 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 входных данных и количества потоков. Получившиеся результаты необходимо объяснить.

**Вариант 6:** Произвести перемножение 2-ух матриц, содержащих комплексные числа.

**Общие сведения о программе**

Основной файл программы — main.cpp, так же используется файл complex.cpp, содержащий реализацию класса комплексных чисел и файл complex.hpp. Также используется заголовочные файлы: iostream, vector, pthread.h, chrono, string. В программе используются следующие системные вызовы:

1. **pthread\_create** – создание потока.
2. **pthread\_join** – ожидание потока и его завершение.

**Общий метод и алгоритм решения**.

Для реализации поставленной задачи необходимо:

1. Изучить принципы работы pthread\_create, pthread\_join, pthread\_detach, pthread\_mutex, а также библиотеку chrono.
2. Написать класс complex\_double для работы с комплексными числами.
3. Написать файл main.cpp, в котором будет осуществляться перемножение матриц в многопоточном режиме.

**Основные файлы программы**

**main.cpp:**

#include <iostream>

#include <vector>

#include <pthread.h>

#include <string>

#include <chrono>

#include "complex.hpp"

std::vector<std::vector<complex\_double>> matrix1, matrix2, result;

void\* multiply(void \*arg) {

int i = \*((int\*)arg);

int j = \*((int\*)arg + 1);

int step = \*((int\*)arg + 2);

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

result[i][j] = 0;

for (int x = 0; x < matrix1[0].size(); ++x) {

result[i][j] += matrix1[i][x] \* matrix2[x][j];

}

i = (i \* matrix1.size() + j + 1) / matrix1.size();

j = (j + 1) % matrix1.size();

}

return nullptr;

}

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

int number\_of\_threads = atoi(argv[1]);

if (number\_of\_threads <= 0) {

std::cerr << "Заданное число потоков не может быть использовано." << std::endl;

return -1;

}

int n1, m1, n2, m2;

std::cout<<"Введите размер первой матрицы\n";

std::cin >> n1 >> m1;

std::cout<<"Введите первую матрицу\n";

matrix1.resize(n1, std::vector<complex\_double>(m1));

for (int i = 0; i < n1; ++i) {

for (int j = 0; j < m1; ++j) {

std::cin >> matrix1[i][j];

}

}

std::cout<<"Введите размер второй матрицы\n";

std::cin >> n2 >> m2;

std::cout<<"Введите вторую матрицу\n";

matrix2.resize(n2, std::vector<complex\_double>(m2));

for (int i = 0; i < n2; ++i) {

for (int j = 0; j < m2; ++j) {

std::cin >> matrix2[i][j];

}

}

if (m1 != n2) {

std::cerr << "Неверные размеры матриц: невозможно перемножить\n";

return -1;

}

result.resize(n1, std::vector<complex\_double>(m2, complex\_double()));

number\_of\_threads = std::min(number\_of\_threads, n1 \* m2);

std::vector<pthread\_t> threads(number\_of\_threads);

std::vector<int[3]> args(number\_of\_threads);

int nm = n1 \* m2;

int position = 0;

std::chrono::steady\_clock::time\_point start\_time = std::chrono::steady\_clock::now();

for (int \_ = 0; \_ < threads.size(); ++\_) {

int step = nm / number\_of\_threads--;

nm -= step;

int i = position / m2;

int j = position % m2;

args[\_][0] = i; args[\_][1] = j; args[\_][2] = step;

pthread\_create(&threads[\_], NULL, multiply, args[\_]);

position += step;

}

nm = n1 \* m2, number\_of\_threads = threads.size();

position = 0;

for (int \_ = 0; \_ < threads.size(); ++\_) {

pthread\_join(threads[\_], nullptr);

}

std::cout << '\n';

std::cout<<"Результат:\n";

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

for (int j = 0; j < result[i].size(); ++j) {

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

}

std::cout << '\n';

}

std::chrono::steady\_clock::time\_point end\_time = std::chrono::steady\_clock::now();

std::chrono::microseconds duration = std::chrono::duration\_cast<std::chrono::microseconds>(end\_time - start\_time);

std::cout << "Время работы программы: " << duration.count() << " микросекунд. " << "\n";

}

**complex\_double.hpp:**

#pragma once

#include <iostream>

struct complex\_double {

double x;

double i;

complex\_double();

complex\_double(double&);

~complex\_double() = default;

friend std::istream& operator>>(std::istream&, complex\_double&);

friend std::ostream& operator<<(std::ostream&, complex\_double&);

complex\_double operator=(const double&);

complex\_double operator=(const complex\_double&);

complex\_double operator+(const complex\_double&);

complex\_double operator-(const complex\_double&);

complex\_double operator\*(const complex\_double&);

complex\_double operator+=(const complex\_double&);

};

**complex\_double.cpp:**

#include "complex.hpp"

complex\_double::complex\_double() {

x = 0;

i = 0;

}

complex\_double::complex\_double(double& val) {

x = val;

i = 0;

}

std::istream& operator>>(std::istream& in, complex\_double& val) {

std::string s;

in >> s;

int n = 0;

bool sign = true;

for (char c : s) {

if (c <= '9' && c >= '0') {

n = n \* 10 + (c - '0');

}

if (c == 'i') {

if (n == 0) {

if (sign)

val.i = 1;

else

val.i = -1;

}

else {

if (sign)

val.i = n;

else

val.i = -n;

n = 0;

}

}

if (c == '+') {

if (sign)

val.x = n;

else

val.x = -n;

n = 0;

sign = true;

}

if (c == '-') {

if (sign)

val.x = n;

else

val.x = -n;

n = 0;

sign = false;

}

}

if (n > 0){

if (sign)

val.x = n;

else

val.x = -n;

}

return in;

}

std::ostream& operator<<(std::ostream& out, complex\_double& val) {

out << val.x;

if (val.i == 0) {

return out;

}

else if (val.i >= 0) {

out << '+';

}

else {

out << '-';

}

out << std::abs(val.i) << 'i';

return out;

}

complex\_double complex\_double::operator=(const double& val) {

x = val;

i = 0;

return \*this;

}

complex\_double complex\_double::operator=(const complex\_double& other) {

x = other.x;

i = other.i;

return \*this;

}

complex\_double complex\_double::operator+(const complex\_double& other) {

complex\_double \*res = new complex\_double();

res->x = x + other.x;

res->i = i + other.i;

return \*res;

}

complex\_double complex\_double::operator-(const complex\_double& other) {

complex\_double res = complex\_double();

res.x = x - other.x;

res.i = i - other.i;

return res;

}

complex\_double complex\_double::operator\*(const complex\_double& other) {

complex\_double res = complex\_double();

res.x = x \* other.x - i \* other.i;

res.i = i \* other.x + x \* other.i;

return res;

}

complex\_double complex\_double::operator+=(const complex\_double& other) {

x += other.x;

i += other.i;

return \*this;

}

**Пример работы**

**1.**

**tanya@tanya-S431:~/Рабочий стол/prog/build$ ./main2 1**

**Введите размер первой матрицы**

**2 2**

**Введите первую матрицу**

**1 2**

**3 4**

**Введите размер второй матрицы**

**2 2**

**Введите вторую матрицу**

**1 2**

**3 4**

**Результат:**

**7 10**

**15 22**

**Время работы программы: 220 микросекунд.**

**tanya@tanya-S431:~/Рабочий стол/prog/build$ ./main2 2**

**Введите размер первой матрицы**

**2 2**

**Введите первую матрицу**

**1 2**

**3 4**

**Введите размер второй матрицы**

**2 2**

**Введите вторую матрицу**

**1 2**

**3 4**

**Результат:**

**7 10**

**15 22**

**Время работы программы: 267 микросекунд.**

**2.**

**tanya@tanya-S431:~/Рабочий стол/prog/build$ ./main2 1**

**Введите размер первой матрицы**

**5 5**

**Введите первую матрицу**

**1+2i 2+3i 10+7i 5+0i 6+9i**

**2+7i 0+0i 2+9i 3+4i 17+9i**

**12+6i 3+13i 8+9i 2+7i 8+7i**

**1+0i 2+1i 4+8i 14+9i 20+0i**

**5+9i 45+0i 3+9i 1+1i 2+7i**

**Введите размер второй матрицы**

**5 5**

**Введите вторую матрицу**

**1+2i 2+3i 10+7i 5+0i 6+9i**

**2+7i 0+0i 2+9i 3+4i 17+9i**

**12+6i 3+13i 8+9i 2+7i 8+7i**

**1+0i 2+1i 4+8i 14+9i 20+0i**

**5+9i 45+0i 3+9i 1+1i 2+7i**

**Результат:**

**11+273i 202+578i -61+322i 26+186i 74+301i**

**-46+338i 629+506i -153+396i -46+181i -80+383i**

**-71+349i 250+521i -137+478i -70+298i -47+732i**

**92+347i 824+180i -7+468i 92+334i 302+478i**

**-1+517i -42+464i -50+682i 90+304i 640+663i**

**Время работы программы: 389 микросекунд.**

**tanya@tanya-S431:~/Рабочий стол/prog/build$ ./main2 2**

**Введите размер первой матрицы**

**5 5**

**Введите первую матрицу**

**1+2i 2+3i 10+7i 5+0i 6+9i**

**2+7i 0+0i 2+9i 3+4i 17+9i**

**12+6i 3+13i 8+9i 2+7i 8+7i**

**1+0i 2+1i 4+8i 14+9i 20+0i**

**5+9i 45+0i 3+9i 1+1i 2+7i**

**Введите размер второй матрицы**

**5 5**

**Введите вторую матрицу**

**1+2i 2+3i 10+7i 5+0i 6+9i**

**2+7i 0+0i 2+9i 3+4i 17+9i**

**12+6i 3+13i 8+9i 2+7i 8+7i**

**1+0i 2+1i 4+8i 14+9i 20+0i**

**5+9i 45+0i 3+9i 1+1i 2+7i**

**Результат:**

**11+273i 202+578i -61+322i 26+186i 74+301i**

**-46+338i 629+506i -153+396i -46+181i -80+383i**

**-71+349i 250+521i -137+478i -70+298i -47+732i**

**92+347i 824+180i -7+468i 92+334i 302+478i**

**-1+517i -42+464i -50+682i 90+304i 640+663i**

**Время работы программы: 292 микросекунд.**

**3.**

**tanya@tanya-S431:~/Рабочий стол/prog/build$ ./main2 1**

**Введите размер первой матрицы**

**10 10**

**Введите первую матрицу**

**100+i 100+i 100+i 100+i 100+i 100+i 100+i 100+i 100+i 100+i**

**200+2i 200+2i 200+2i 200+2i 200+2i 200+2i 200+2i 200+2i 200+2i 200+2i**

**300+3i 300+3i 300+3i 300+3i 300+3i 300+3i 300+3i 300+3i 300+3i 300+3i**

**400+4i 400+4i 400+4i 400+4i 400+4i 400+4i 400+4i 400+4i 400+4i 400+4i**

**500+5i 500+5i 500+5i 500+5i 500+5i 500+5i 500+5i 500+5i 500+5i 500+5i**

**600+6i 600+6i 600+6i 600+6i 600+6i 600+6i 600+6i 600+6i 600+6i 600+6i**

**700+7i 700+7i 700+7i 700+7i 700+7i 700+7i 700+7i 700+7i 700+7i 700+7i**

**800+8i 800+8i 800+8i 800+8i 800+8i 800+8i 800+8i 800+8i 800+8i 800+8i**

**900+9i 900+9i 900+9i 900+9i 900+9i 900+9i 900+9i 900+9i 900+9i 900+9i**

**1000+10i 1000+10i 1000+10i 1000+10i 1000+10i 1000+10i 1000+10i 1000+10i 1000+10i 1000+10i**

**Введите размер второй матрицы**

**10 10**

**Введите вторую матрицу**

**100+i 100+i 100+i 100+i 100+i 100+i 100+i 100+i 100+i 100+i**

**200+2i 200+2i 200+2i 200+2i 200+2i 200+2i 200+2i 200+2i 200+2i 200+2i**

**300+3i 300+3i 300+3i 300+3i 300+3i 300+3i 300+3i 300+3i 300+3i 300+3i**

**400+4i 400+4i 400+4i 400+4i 400+4i 400+4i 400+4i 400+4i 400+4i 400+4i**

**500+5i 500+5i 500+5i 500+5i 500+5i 500+5i 500+5i 500+5i 500+5i 500+5i**

**600+6i 600+6i 600+6i 600+6i 600+6i 600+6i 600+6i 600+6i 600+6i 600+6i**

**700+7i 700+7i 700+7i 700+7i 700+7i 700+7i 700+7i 700+7i 700+7i 700+7i**

**800+8i 800+8i 800+8i 800+8i 800+8i 800+8i 800+8i 800+8i 800+8i 800+8i**

**900+9i 900+9i 900+9i 900+9i 900+9i 900+9i 900+9i 900+9i 900+9i 900+9i**

**1000+10i 1000+10i 1000+10i 1000+10i 1000+10i 1000+10i 1000+10i 1000+10i 1000+10i 1000+10i**

**Результат:**

**549945+11000i 549945+11000i 549945+11000i 549945+11000i 549945+11000i 549945+11000i 549945+11000i 549945+11000i 549945+11000i 549945+11000i**

**1.09989e+06+22000i 1.09989e+06+22000i 1.09989e+06+22000i 1.09989e+06+22000i 1.09989e+06+22000i 1.09989e+06+22000i 1.09989e+06+22000i 1.09989e+06+22000i 1.09989e+06+22000i 1.09989e+06+22000i**

**1.64984e+06+33000i 1.64984e+06+33000i 1.64984e+06+33000i 1.64984e+06+33000i 1.64984e+06+33000i 1.64984e+06+33000i 1.64984e+06+33000i 1.64984e+06+33000i 1.64984e+06+33000i 1.64984e+06+33000i**

**2.19978e+06+44000i 2.19978e+06+44000i 2.19978e+06+44000i 2.19978e+06+44000i 2.19978e+06+44000i 2.19978e+06+44000i 2.19978e+06+44000i 2.19978e+06+44000i 2.19978e+06+44000i 2.19978e+06+44000i**

**2.74972e+06+55000i 2.74972e+06+55000i 2.74972e+06+55000i 2.74972e+06+55000i 2.74972e+06+55000i 2.74972e+06+55000i 2.74972e+06+55000i 2.74972e+06+55000i 2.74972e+06+55000i 2.74972e+06+55000i**

**3.29967e+06+66000i 3.29967e+06+66000i 3.29967e+06+66000i 3.29967e+06+66000i 3.29967e+06+66000i 3.29967e+06+66000i 3.29967e+06+66000i 3.29967e+06+66000i 3.29967e+06+66000i 3.29967e+06+66000i**

**3.84962e+06+77000i 3.84962e+06+77000i 3.84962e+06+77000i 3.84962e+06+77000i 3.84962e+06+77000i 3.84962e+06+77000i 3.84962e+06+77000i 3.84962e+06+77000i 3.84962e+06+77000i 3.84962e+06+77000i**

**4.39956e+06+88000i 4.39956e+06+88000i 4.39956e+06+88000i 4.39956e+06+88000i 4.39956e+06+88000i 4.39956e+06+88000i 4.39956e+06+88000i 4.39956e+06+88000i 4.39956e+06+88000i 4.39956e+06+88000i**

**4.9495e+06+99000i 4.9495e+06+99000i 4.9495e+06+99000i 4.9495e+06+99000i 4.9495e+06+99000i 4.9495e+06+99000i 4.9495e+06+99000i 4.9495e+06+99000i 4.9495e+06+99000i 4.9495e+06+99000i**

**5.49945e+06+110000i 5.49945e+06+110000i 5.49945e+06+110000i 5.49945e+06+110000i 5.49945e+06+110000i 5.49945e+06+110000i 5.49945e+06+110000i 5.49945e+06+110000i 5.49945e+06+110000i 5.49945e+06+110000i**

**Время работы программы: 1063 микросекунд.**

**tanya@tanya-S431:~/Рабочий стол/prog/build$ ./main2 2**

**Введите размер первой матрицы**

**10 10**

**Введите первую матрицу**

**100+i 100+i 100+i 100+i 100+i 100+i 100+i 100+i 100+i 100+i**

**200+2i 200+2i 200+2i 200+2i 200+2i 200+2i 200+2i 200+2i 200+2i 200+2i**

**300+3i 300+3i 300+3i 300+3i 300+3i 300+3i 300+3i 300+3i 300+3i 300+3i**

**400+4i 400+4i 400+4i 400+4i 400+4i 400+4i 400+4i 400+4i 400+4i 400+4i**

**500+5i 500+5i 500+5i 500+5i 500+5i 500+5i 500+5i 500+5i 500+5i 500+5i**

**600+6i 600+6i 600+6i 600+6i 600+6i 600+6i 600+6i 600+6i 600+6i 600+6i**

**700+7i 700+7i 700+7i 700+7i 700+7i 700+7i 700+7i 700+7i 700+7i 700+7i**

**800+8i 800+8i 800+8i 800+8i 800+8i 800+8i 800+8i 800+8i 800+8i 800+8i**

**900+9i 900+9i 900+9i 900+9i 900+9i 900+9i 900+9i 900+9i 900+9i 900+9i**

**1000+10i 1000+10i 1000+10i 1000+10i 1000+10i 1000+10i 1000+10i 1000+10i 1000+10i 1000+10i**

**Введите размер второй матрицы**

**10 10**

**Введите вторую матрицу**

**100+i 100+i 100+i 100+i 100+i 100+i 100+i 100+i 100+i 100+i**

**200+2i 200+2i 200+2i 200+2i 200+2i 200+2i 200+2i 200+2i 200+2i 200+2i**

**300+3i 300+3i 300+3i 300+3i 300+3i 300+3i 300+3i 300+3i 300+3i 300+3i**

**400+4i 400+4i 400+4i 400+4i 400+4i 400+4i 400+4i 400+4i 400+4i 400+4i**

**500+5i 500+5i 500+5i 500+5i 500+5i 500+5i 500+5i 500+5i 500+5i 500+5i**

**600+6i 600+6i 600+6i 600+6i 600+6i 600+6i 600+6i 600+6i 600+6i 600+6i**

**700+7i 700+7i 700+7i 700+7i 700+7i 700+7i 700+7i 700+7i 700+7i 700+7i**

**800+8i 800+8i 800+8i 800+8i 800+8i 800+8i 800+8i 800+8i 800+8i 800+8i**

**900+9i 900+9i 900+9i 900+9i 900+9i 900+9i 900+9i 900+9i 900+9i 900+9i**

**1000+10i 1000+10i 1000+10i 1000+10i 1000+10i 1000+10i 1000+10i 1000+10i 1000+10i 1000+10i**

**Результат:**

**549945+11000i 549945+11000i 549945+11000i 549945+11000i 549945+11000i 549945+11000i 549945+11000i 549945+11000i 549945+11000i 549945+11000i**

**1.09989e+06+22000i 1.09989e+06+22000i 1.09989e+06+22000i 1.09989e+06+22000i 1.09989e+06+22000i 1.09989e+06+22000i 1.09989e+06+22000i 1.09989e+06+22000i 1.09989e+06+22000i 1.09989e+06+22000i**

**1.64984e+06+33000i 1.64984e+06+33000i 1.64984e+06+33000i 1.64984e+06+33000i 1.64984e+06+33000i 1.64984e+06+33000i 1.64984e+06+33000i 1.64984e+06+33000i 1.64984e+06+33000i 1.64984e+06+33000i**

**2.19978e+06+44000i 2.19978e+06+44000i 2.19978e+06+44000i 2.19978e+06+44000i 2.19978e+06+44000i 2.19978e+06+44000i 2.19978e+06+44000i 2.19978e+06+44000i 2.19978e+06+44000i 2.19978e+06+44000i**

**2.74972e+06+55000i 2.74972e+06+55000i 2.74972e+06+55000i 2.74972e+06+55000i 2.74972e+06+55000i 2.74972e+06+55000i 2.74972e+06+55000i 2.74972e+06+55000i 2.74972e+06+55000i 2.74972e+06+55000i**

**3.29967e+06+66000i 3.29967e+06+66000i 3.29967e+06+66000i 3.29967e+06+66000i 3.29967e+06+66000i 3.29967e+06+66000i 3.29967e+06+66000i 3.29967e+06+66000i 3.29967e+06+66000i 3.29967e+06+66000i**

**3.84962e+06+77000i 3.84962e+06+77000i 3.84962e+06+77000i 3.84962e+06+77000i 3.84962e+06+77000i 3.84962e+06+77000i 3.84962e+06+77000i 3.84962e+06+77000i 3.84962e+06+77000i 3.84962e+06+77000i**

**4.39956e+06+88000i 4.39956e+06+88000i 4.39956e+06+88000i 4.39956e+06+88000i 4.39956e+06+88000i 4.39956e+06+88000i 4.39956e+06+88000i 4.39956e+06+88000i 4.39956e+06+88000i 4.39956e+06+88000i**

**4.9495e+06+99000i 4.9495e+06+99000i 4.9495e+06+99000i 4.9495e+06+99000i 4.9495e+06+99000i 4.9495e+06+99000i 4.9495e+06+99000i 4.9495e+06+99000i 4.9495e+06+99000i 4.9495e+06+99000i**

**5.49945e+06+110000i 5.49945e+06+110000i 5.49945e+06+110000i 5.49945e+06+110000i 5.49945e+06+110000i 5.49945e+06+110000i 5.49945e+06+110000i 5.49945e+06+110000i 5.49945e+06+110000i 5.49945e+06+110000i**

**Время работы программы: 758 микросекунд.**

**tanya@tanya-S431:~/Рабочий стол/prog/build$ ./main2 3**

**Введите размер первой матрицы**

**10 10**

**Введите первую матрицу**

**100+i 100+i 100+i 100+i 100+i 100+i 100+i 100+i 100+i 100+i**

**200+2i 200+2i 200+2i 200+2i 200+2i 200+2i 200+2i 200+2i 200+2i 200+2i**

**300+3i 300+3i 300+3i 300+3i 300+3i 300+3i 300+3i 300+3i 300+3i 300+3i**

**400+4i 400+4i 400+4i 400+4i 400+4i 400+4i 400+4i 400+4i 400+4i 400+4i**

**500+5i 500+5i 500+5i 500+5i 500+5i 500+5i 500+5i 500+5i 500+5i 500+5i**

**600+6i 600+6i 600+6i 600+6i 600+6i 600+6i 600+6i 600+6i 600+6i 600+6i**

**700+7i 700+7i 700+7i 700+7i 700+7i 700+7i 700+7i 700+7i 700+7i 700+7i**

**800+8i 800+8i 800+8i 800+8i 800+8i 800+8i 800+8i 800+8i 800+8i 800+8i**

**900+9i 900+9i 900+9i 900+9i 900+9i 900+9i 900+9i 900+9i 900+9i 900+9i**

**1000+10i 1000+10i 1000+10i 1000+10i 1000+10i 1000+10i 1000+10i 1000+10i 1000+10i 1000+10i**

**Введите размер второй матрицы**

**10 10**

**Введите вторую матрицу**

**100+i 100+i 100+i 100+i 100+i 100+i 100+i 100+i 100+i 100+i**

**200+2i 200+2i 200+2i 200+2i 200+2i 200+2i 200+2i 200+2i 200+2i 200+2i**

**300+3i 300+3i 300+3i 300+3i 300+3i 300+3i 300+3i 300+3i 300+3i 300+3i**

**400+4i 400+4i 400+4i 400+4i 400+4i 400+4i 400+4i 400+4i 400+4i 400+4i**

**500+5i 500+5i 500+5i 500+5i 500+5i 500+5i 500+5i 500+5i 500+5i 500+5i**

**600+6i 600+6i 600+6i 600+6i 600+6i 600+6i 600+6i 600+6i 600+6i 600+6i**

**700+7i 700+7i 700+7i 700+7i 700+7i 700+7i 700+7i 700+7i 700+7i 700+7i**

**800+8i 800+8i 800+8i 800+8i 800+8i 800+8i 800+8i 800+8i 800+8i 800+8i**

**900+9i 900+9i 900+9i 900+9i 900+9i 900+9i 900+9i 900+9i 900+9i 900+9i**

**1000+10i 1000+10i 1000+10i 1000+10i 1000+10i 1000+10i 1000+10i 1000+10i 1000+10i 1000+10i**

**Результат:**

**549945+11000i 549945+11000i 549945+11000i 549945+11000i 549945+11000i 549945+11000i 549945+11000i 549945+11000i 549945+11000i 549945+11000i**

**1.09989e+06+22000i 1.09989e+06+22000i 1.09989e+06+22000i 1.09989e+06+22000i 1.09989e+06+22000i 1.09989e+06+22000i 1.09989e+06+22000i 1.09989e+06+22000i 1.09989e+06+22000i 1.09989e+06+22000i**

**1.64984e+06+33000i 1.64984e+06+33000i 1.64984e+06+33000i 1.64984e+06+33000i 1.64984e+06+33000i 1.64984e+06+33000i 1.64984e+06+33000i 1.64984e+06+33000i 1.64984e+06+33000i 1.64984e+06+33000i**

**2.19978e+06+44000i 2.19978e+06+44000i 2.19978e+06+44000i 2.19978e+06+44000i 2.19978e+06+44000i 2.19978e+06+44000i 2.19978e+06+44000i 2.19978e+06+44000i 2.19978e+06+44000i 2.19978e+06+44000i**

**2.74972e+06+55000i 2.74972e+06+55000i 2.74972e+06+55000i 2.74972e+06+55000i 2.74972e+06+55000i 2.74972e+06+55000i 2.74972e+06+55000i 2.74972e+06+55000i 2.74972e+06+55000i 2.74972e+06+55000i**

**3.29967e+06+66000i 3.29967e+06+66000i 3.29967e+06+66000i 3.29967e+06+66000i 3.29967e+06+66000i 3.29967e+06+66000i 3.29967e+06+66000i 3.29967e+06+66000i 3.29967e+06+66000i 3.29967e+06+66000i**

**3.84962e+06+77000i 3.84962e+06+77000i 3.84962e+06+77000i 3.84962e+06+77000i 3.84962e+06+77000i 3.84962e+06+77000i 3.84962e+06+77000i 3.84962e+06+77000i 3.84962e+06+77000i 3.84962e+06+77000i**

**4.39956e+06+88000i 4.39956e+06+88000i 4.39956e+06+88000i 4.39956e+06+88000i 4.39956e+06+88000i 4.39956e+06+88000i 4.39956e+06+88000i 4.39956e+06+88000i 4.39956e+06+88000i 4.39956e+06+88000i**

**4.9495e+06+99000i 4.9495e+06+99000i 4.9495e+06+99000i 4.9495e+06+99000i 4.9495e+06+99000i 4.9495e+06+99000i 4.9495e+06+99000i 4.9495e+06+99000i 4.9495e+06+99000i 4.9495e+06+99000i**

**5.49945e+06+110000i 5.49945e+06+110000i 5.49945e+06+110000i 5.49945e+06+110000i 5.49945e+06+110000i 5.49945e+06+110000i 5.49945e+06+110000i 5.49945e+06+110000i 5.49945e+06+110000i 5.49945e+06+110000i**

**Время работы программы: 806 микросекунд.**

|  |  |  |
| --- | --- | --- |
| Количество потоков | Размеры матриц | Время работы программы, мкс |
| 1 | 2\*2, 2\*2 | 220 |
| 2 | 2\*2, 2\*2 | 267 |
| 1 | 5\*5, 5\*5 | 389 |
| 2 | 5\*5, 5\*5 | 292 |
| 1 | 10\*10, 10\*10 | 1063 |
| 2 | 10\*10, 10\*10 | 758 |
| 3 | 10\*10, 10\*10 | 806 |
| 4 | 10\*10, 10\*10 | 836 |
| 5 | 10\*10, 10\*10 | 1239 |
| 1 | 10\*40, 40\*10 | 1237 |
| 2 | 10\*40, 40\*10 | 804 |
| 3 | 10\*40, 40\*10 | 717 |
| 4 | 10\*40, 40\*10 | 765 |
| 5 | 10\*40, 40\*10 | 2403 |

Как видно из таблицы, многопоточность позволяет увеличить скорость работы программы, что более заметно на больших данных. Чрезмерное количество потоков, однако, может замедлить работу: для решения моей задачи, оптимальное количество потоков — не более четырех.

**Вывод**

Проделав работу, я успешно реализовала разделение вычислений на несколько потоков, что позволило эффективно использовать ресурсы и ускорить процесс перемножения матриц содержащих комплексные числа. Я получила практические навыки работы с многопоточностью в ОС UNIX.

Многопоточность позволяет улучшить производительность программы в задачах, которые могут быть разделены на отдельные задачи.