МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

**«КУБАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»**

**(ФГБОУ ВО «КубГУ»)**

**Факультет компьютерных технологий и прикладной математики**

**Кафедра информационных технологий**

**ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ № 3**

**по дисциплине  
 «Параллельное и низкоуровневое программирование»**

Выполнил студент группы 25/2                                       А.А. Козин

Направление подготовки  02.03.03  Математическое обеспечение и администрирование информационных систем

Курс    2

Отчет принял доктор физико-математических наук, профессор                                                                                       А.И. Миков

Краснодар

2022 г.

**Задание 1**: Решение систем линейных уравнений (любым методом).  
**Решение**: для решение этой задачи я выбрал метод Гаусса.  
Метод Гаусса решения СЛАУ включает в себя 2 стадии: прямое исключение и обратная подстановка. Эти 2 стадии я реализовал в функции gauss.  
Сначала вводиться с клавиатуры кол-во уравнений, потом предлагается выбор ввести значения системы вручную или за рандомить их. Для этого я использовал оператор switch, где 1 – ввести вручную (для проверки работы программы), а 2 – за рандомить (чтобы не вписывать миллион значений для распараллеливания).  
Результаты вычисления можно проверить раскоментив 121 и 122 строчку программы.

**Текст программы:**

#include <iostream>

#include <ctime>

#include <thread>

#include <chrono>

using namespace std;

using namespace std::chrono;

double\* gauss(double\*\* a, double\* y, int start, int end){

double\* x, max;

int с = start, index;

const double eps = 0.00001;// точность

x = new double[end];

for (с; с < end; с){

// Поиск строки с максимальным a[i][с]

max = abs(a[с][с]);

index = с;

for (int i = с + 1; i < end; i++){

if (abs(a[i][с]) > max){

max = abs(a[i][с]);

index = i;

}

}

// Перестановка строк

if (max < eps){

// нет ненулевых диагональных элементов

cout << "Solution cannot be obtained due to null column ";

cout << index << " matrices A" << endl;

return 0;

}

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

double swap = a[с][j];

a[с][j] = a[index][j];

a[index][j] = swap;

}

double swap = y[с];

y[с] = y[index];

y[index] = swap;

// Нормализация уравнений

for (int i = с; i < end; i++){

double swap = a[i][с];

if (abs(swap) < eps) continue; // для нулевого коэффициента пропустить

for (int j = 0; j < end; j++)

a[i][j] = a[i][j] / swap;

y[i] = y[i] / swap;

if (i == с) continue; // уравнение не вычитать само из себя

for (int j = 0; j < end; j++)

a[i][j] = a[i][j] - a[с][j];

y[i] = y[i] - y[с];

}

с++;

}

// обратная подстановка

for (с = end - 1; с >= 0; с--){

x[с] = y[с];

for (int i = 0; i < с; i++)

y[i] = y[i] - a[i][с] \* x[с];

}

return x;

}

void main() {

double\*\* a, \* y, \* x;

int n;

cout << "Enter number of equations: ";

cin >> n;

a = new double\* [n];

y = new double[n];

cout << "Do you want to enter values manually or randomly ?" << endl << "1 - manually, 2 - random" << endl;

int input; cin >> input; cout << endl;

switch (input) {

case 1:

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

a[i] = new double[n];

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

cout << "a[" << i << "][" << j << "]= ";

cin >> a[i][j];

}

}

cout << endl;

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

cout << "y[" << i << "]= ";

cin >> y[i];

}

break;

case 2:

srand(time(0));

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

a[i] = new double[n];

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

a[i][j] = rand() % 10;

//cout << "a[" << i << "][" << j << "]= "<< a[i][j]<<"; ";

}

}

cout << endl;

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

y[i] = rand() % 10;

//cout << "y[" << i << "]= "<<y[i];

}

break;

default:

cout << "Error, bad input, quitting." << endl;

break;

}

auto t0 = high\_resolution\_clock::now();

x = gauss(a, y, 0, n);

auto t1 = high\_resolution\_clock::now();

cout << endl << "duration\_cast gauss without multithreading: " << duration\_cast<milliseconds>(t1 - t0).count() << " milliseconds" << endl;

thread first(gauss, a, y, 0, n / 2);

thread second(gauss, a, y, n / 2, n);

t0 = high\_resolution\_clock::now();

first.join(); second.join();

t1 = high\_resolution\_clock::now();

cout << "duration\_cast gauss with multithreading: " << duration\_cast<milliseconds>(t1 - t0).count() << " milliseconds" << endl << endl;

//for (int i = 0; i < n; i++) // если нужно проверить ответ

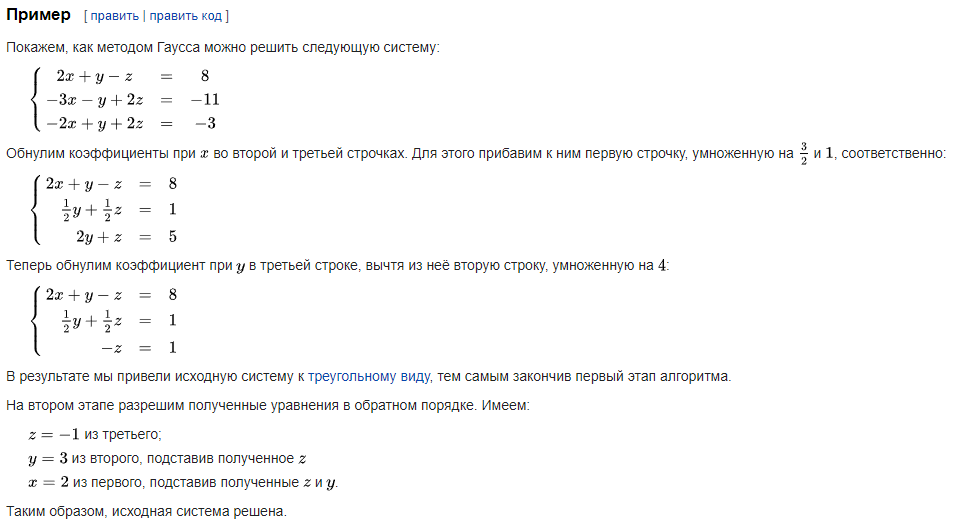
//cout << "x[" << i << "]=" << x[i] << endl;

cin.get(); cin.get();

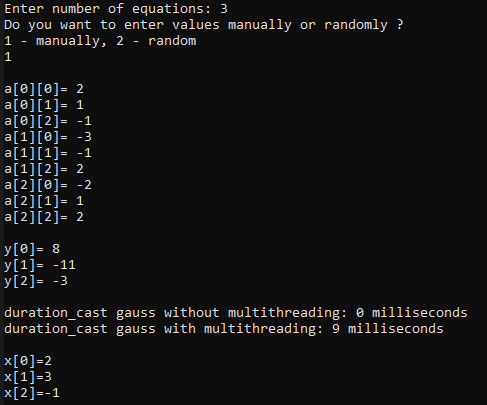
}

**Программа выдает в консоли результаты в следующем виде:**

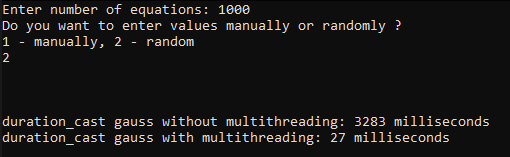
Для проверки верности решения возьму пример из википедии.



Ввожу значения:



Ответ правильный: 2 3 -1.



Видно, из скрина выше, что для 1000 уравнений время работы метода Гаусса без распараллеливания 3283 мсек, а с распараллеливанием 27 мсек.

Из 2-х скриншотов консоли можно сделать вывод, что выгодно в плане времени использовать распараллеливание с большим количеством уравнений, а с маленьким количеством уравнений достаточно использовать обычный алгоритм.