МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ

ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ

ВЫСШЕГО ОБРАЗОВАНИЯ

«НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Кафедра защиты информации



**ЛАБОРАТОРНАЯ РАБОТА №6**

**«Обработка двумерных массивов и контейнеров»**

**по дисциплине: «*Программирование*»**

|  |  |
| --- | --- |
| Выполнил:  Студент гр. «АБс-322», «АВТФ»  *Аврамчук Илья Вячеславович*  «21» июня 2024 г.  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись) | Проверил:  *Ассистент кафедры ЗИ*  *Исаев Г. А.*  «14» июня 2024 г.  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись) |

Новосибирск 2024

**Цели и задачи работы:** изучение алгоритмов формирования и обработки двумерных массивов, программирование и отладка программ формирования и обработки массивов (статических и динамических) и контейнеров STL.

**Задание к работе**: Самостоятельно решить задачи в соответствии с индивидуальным вариантом.

**Методика выполнения работы**:

1. Разработать алгоритм решения задачи по индивидуальному заданию.

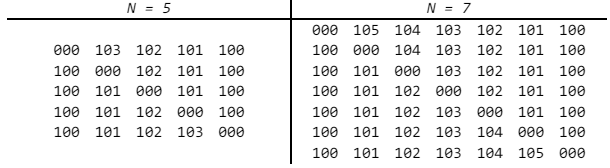
2. Написать и отладить программу решения задачи на двух языках (С++ и второй, по выбору).

3. Протестировать работу программы на различных исходных данных.

**Задание №6.1 Обработка двумерных массивов.**

**Вариант 1**

1. Определите и инициализируйте матрицу размерности M × N случайными целыми числами в диапазоне [0, 50]. Определите, какое простое число встречается в матрице наибольшее количество раз.
2. Напишите программу, инициализирующую матрицу, как в примере. Размерность матрицы задаётся с клавиатуры. Модифицируйте вывод таким образом, чтобы вместо “0” в консоль выводилось “000”.



1. В двумерном массиве размерностью M × N, все элементы которого различны, требуется найти такие элементы, которые одновременно являются минимальными в своей строке и максимальными в своем столбце.

**Язык программирования : С++**

**Код программы :**

#include <iostream>

#include <vector>

#include <cstdlib>

#include <ctime>

#include <map>

#include <iomanip>

#include <climits>

// Функция для проверки простого числа

bool isPrime(int num) {

if (num <= 1) return false;

for (int i = 2; i \* i <= num; ++i) {

if (num % i == 0) return false;

}

return true;

}

// Задача 1: Поиск наиболее часто встречающегося простого числа в случайной матрице

void task1() {

srand(time(0));

int M, N;

std::cout << "Введите размеры матрицы M и N: ";

std::cin >> M >> N;

std::vector<std::vector<int>> matrix(M, std::vector<int>(N));

std::map<int, int> primeCount;

// Заполнение матрицы случайными числами и подсчет простых чисел

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

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

matrix[i][j] = rand() % 51;

if (isPrime(matrix[i][j])) {

primeCount[matrix[i][j]]++;

}

}

}

// Поиск простого числа, которое встречается наибольшее количество раз

int mostFrequentPrime = -1;

int maxCount = 0;

for (const auto& pair : primeCount) {

if (pair.second > maxCount) {

maxCount = pair.second;

mostFrequentPrime = pair.first;

}

}

// Вывод матрицы и результата

std::cout << "Матрица:\n";

for (const auto& row : matrix) {

for (int num : row) {

std::cout << num << " ";

}

std::cout << "\n";

}

if (mostFrequentPrime != -1) {

std::cout << "Простое число, которое встречается наибольшее количество раз: " << mostFrequentPrime << "\n";

std::cout << "Количество вхождений: " << maxCount << "\n";

} else {

std::cout << "Простых чисел в матрице нет.\n";

}

}

// Задача 2: Инициализация и вывод матрицы с определенными правилами

void task2() {

int N;

std::cout << "Enter the size of the matrix: ";

std::cin >> N;

int\*\* matrix = new int\*[N];

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

matrix[i] = new int[N];

}

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

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

if (i == j) {

matrix[i][j] = 0;

} else if (j == 0 || j == N - 1) {

matrix[i][j] = 100;

} else if (i == 0) {

matrix[i][j] = 100 + N - 1 - j;

} else if (i == N - 1) {

matrix[i][j] = 100 + j;

} else if (i < j) {

matrix[i][j] = 100 + j - i;

} else {

matrix[i][j] = 100 + i - j;

}

}

}

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

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

if (matrix[i][j] == 0) {

std::cout << "000 ";

} else {

std::cout << std::setw(3) << std::setfill('0') << matrix[i][j] << " ";

}

}

std::cout << std::endl;

}

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

delete[] matrix[i];

}

delete[] matrix;

}

// Задача 3: Поиск элементов, которые одновременно минимальны в строке и максимальны в столбце

void task3() {

srand(time(0)); // Инициализация генератора случайных чисел

int M, N;

std::cout << "Введите размеры матрицы M и N: ";

std::cin >> M >> N;

std::vector<std::vector<int>> matrix(M, std::vector<int>(N));

// Заполнение матрицы случайными числами

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

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

matrix[i][j] = rand() % 100; // Случайные числа от 0 до 99

}

}

std::vector<int> minInRow(M, INT\_MAX);

std::vector<int> maxInCol(N, INT\_MIN);

// Поиск минимальных элементов в строках и максимальных в столбцах

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

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

if (matrix[i][j] < minInRow[i]) {

minInRow[i] = matrix[i][j];

}

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

maxInCol[j] = matrix[i][j];

}

}

}

std::cout << "Матрица:\n";

for (const auto& row : matrix) {

for (int num : row) {

std::cout << num << " ";

}

std::cout << "\n";

}

std::cout << "Элементы, которые одновременно являются минимальными в своей строке и максимальными в своем столбце:\n";

bool found = false;

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

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

if (matrix[i][j] == minInRow[i] && matrix[i][j] == maxInCol[j]) {

std::cout << "matrix[" << i << "][" << j << "] = " << matrix[i][j] << "\n";

found = true;

}

}

}

if (!found) {

std::cout << "Таких элементов нет.\n";

}

}

int main() {

int choice;

do {

std::cout << "Выберите задачу:\n";

std::cout << "1. Поиск наиболее часто встречающегося простого числа в случайной матрице\n";

std::cout << "2. Инициализация и вывод матрицы с определенными правилами\n";

std::cout << "3. Поиск элементов, которые одновременно минимальны в строке и максимальны в столбце\n";

std::cout << "0. Выход\n";

std::cin >> choice;

switch (choice) {

case 1:

task1();

break;

case 2:

task2();

break;

case 3:

task3();

break;

case 0:

std::cout << "Выход...\n";

break;

default:

std::cout << "Неверный выбор. Пожалуйста, попробуйте снова.\n";

break;

}

} while (choice != 0);

return 0;

}

**Язык программирования : Java**

**Код программы :**

import java.util.\*;

public class Main {

// Функция для проверки простого числа

public static boolean isPrime(int num) {

if (num <= 1) return false;

for (int i = 2; i \* i <= num; ++i) {

if (num % i == 0) return false;

}

return true;

}

// Задача 1: Поиск наиболее часто встречающегося простого числа в случайной матрице

public static void task1() {

Random rand = new Random();

Scanner scanner = new Scanner(System.in);

System.out.print("Введите размеры матрицы M и N: ");

int M = scanner.nextInt();

int N = scanner.nextInt();

int[][] matrix = new int[M][N];

Map<Integer, Integer> primeCount = new HashMap<>();

// Заполнение матрицы случайными числами и подсчет простых чисел

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

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

matrix[i][j] = rand.nextInt(51);

if (isPrime(matrix[i][j])) {

primeCount.put(matrix[i][j], primeCount.getOrDefault(matrix[i][j], 0) + 1);

}

}

}

// Поиск простого числа, которое встречается наибольшее количество раз

int mostFrequentPrime = -1;

int maxCount = 0;

for (Map.Entry<Integer, Integer> entry : primeCount.entrySet()) {

if (entry.getValue() > maxCount) {

maxCount = entry.getValue();

mostFrequentPrime = entry.getKey();

}

}

// Вывод матрицы и результата

System.out.println("Матрица:");

for (int[] row : matrix) {

for (int num : row) {

System.out.print(num + " ");

}

System.out.println();

}

if (mostFrequentPrime != -1) {

System.out.println("Простое число, которое встречается наибольшее количество раз: " + mostFrequentPrime);

System.out.println("Количество вхождений: " + maxCount);

} else {

System.out.println("Простых чисел в матрице нет.");

}

}

// Задача 2: Инициализация и вывод матрицы с определенными правилами

public static void task2() {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter the size of the matrix: ");

int N = scanner.nextInt();

int[][] matrix = new int[N][N];

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

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

if (i == j) {

matrix[i][j] = 0;

} else if (j == 0 || j == N - 1) {

matrix[i][j] = 100;

} else if (i == 0) {

matrix[i][j] = 100 + N - 1 - j;

} else if (i == N - 1) {

matrix[i][j] = 100 + j;

} else if (i < j) {

matrix[i][j] = 100 + j - i;

} else {

matrix[i][j] = 100 + i - j;

}

}

}

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

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

if (matrix[i][j] == 0) {

System.out.print("000 ");

} else {

System.out.printf("%03d ", matrix[i][j]);

}

}

System.out.println();

}

}

// Задача 3: Поиск элементов, которые одновременно минимальны в строке и максимальны в столбце

public static void task3() {

Random rand = new Random();

Scanner scanner = new Scanner(System.in);

System.out.print("Введите размеры матрицы M и N: ");

int M = scanner.nextInt();

int N = scanner.nextInt();

int[][] matrix = new int[M][N];

// Заполнение матрицы случайными числами

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

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

matrix[i][j] = rand.nextInt(100); // Случайные числа от 0 до 99

}

}

int[] minInRow = new int[M];

int[] maxInCol = new int[N];

Arrays.fill(minInRow, Integer.MAX\_VALUE);

Arrays.fill(maxInCol, Integer.MIN\_VALUE);

// Поиск минимальных элементов в строках и максимальных в столбцах

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

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

if (matrix[i][j] < minInRow[i]) {

minInRow[i] = matrix[i][j];

}

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

maxInCol[j] = matrix[i][j];

}

}

}

System.out.println("Матрица:");

for (int[] row : matrix) {

for (int num : row) {

System.out.print(num + " ");

}

System.out.println();

}

System.out.println("Элементы, которые одновременно являются минимальными в своей строке и максимальными в своем столбце:");

boolean found = false;

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

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

if (matrix[i][j] == minInRow[i] && matrix[i][j] == maxInCol[j]) {

System.out.println("matrix[" + i + "][" + j + "] = " + matrix[i][j]);

found = true;

}

}

}

if (!found) {

System.out.println("Таких элементов нет.");

}

}

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

int choice;

do {

System.out.println("Выберите задачу:");

System.out.println("1. Поиск наиболее часто встречающегося простого числа в случайной матрице");

System.out.println("2. Инициализация и вывод матрицы с определенными правилами");

System.out.println("3. Поиск элементов, которые одновременно минимальны в строке и максимальны в столбце");

System.out.println("0. Выход");

choice = scanner.nextInt();

switch (choice) {

case 1:

task1();

break;

case 2:

task2();

break;

case 3:

task3();

break;

case 0:

System.out.println("Выход...");

break;

default:

System.out.println("Неверный выбор. Пожалуйста, попробуйте снова.");

break;

}

} while (choice != 0);

scanner.close();

}

// Остальные методы (task1, task2, task3) остаются без изменений

}

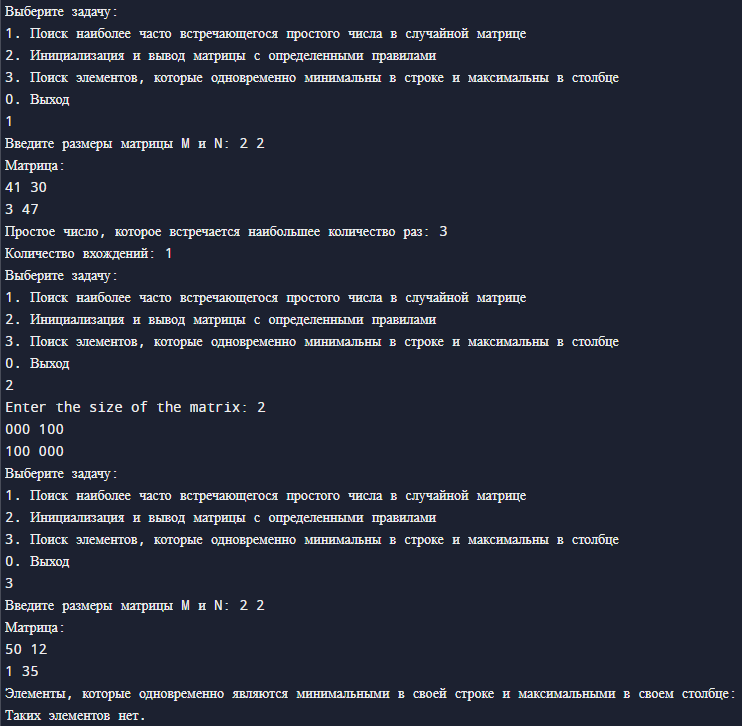


Рисунок 1 – Пример работы программы

**Задание №6.2 Многоалфавитное шифрование с использованием алгоритма AES. Дешифрование текста.**

**Вариант 1**

а) систему шифрования AES128 (CFB) для преобразования исходного текста;

б) систему шифрования AES128 (CFB) для преобразования зашифрованного текста в исходный.

**Язык программирования: C++  
Код программы:**

#include <iostream>

#include <vector>

#include <string>

#include <windows.h>

#include <random>

#include <locale>

#include <iomanip> // Добавляем заголовочный файл для использования setw, setfill и hex

using namespace std;

void GENERATE\_MASTER\_KEY(vector<unsigned char>& MASTER\_KEY, mt19937\_64& mt) {

uniform\_int\_distribution<int> letters('a', 'z'); //кодировка ASCII

uniform\_int\_distribution<int> numbers(0, 9); //кодировка ASCII

for (int i = 1; i <= 8; i++) {

MASTER\_KEY.push\_back(letters(mt));

MASTER\_KEY.push\_back('0' + numbers(mt));

}

}

void ShiftRows(std::vector<unsigned char>& line) {

vector<unsigned char> v(line.size());

for (int i = 1; i < line.size(); i++) {

v[i - 1] = line[i]; //сдвиг элементов влево

}

v[line.size() - 1] = line[0]; //последний элемент становится первым

line = v;

}

void SubBytes(vector<unsigned char>& line) { //замена на соответствующие байты из S-BOX

vector<unsigned char> Sbox = {

0x63, 0x7c, 0x77, 0x7b, 0xf2, 0x6b, 0x6f, 0xc5, 0x30, 0x01, 0x67, 0x2b, 0xfe, 0xd7, 0xab, 0x76,

0xca, 0x82, 0xc9, 0x7d, 0xfa, 0x59, 0x47, 0xf0, 0xad, 0xd4, 0xa2, 0xaf, 0x9c, 0xa4, 0x72, 0xc0,

0xb7, 0xfd, 0x93, 0x26, 0x36, 0x3f, 0xf7, 0xcc, 0x34, 0xa5, 0xe5, 0xf1, 0x71, 0xd8, 0x31, 0x15,

0x04, 0xc7, 0x23, 0xc3, 0x18, 0x96, 0x05, 0x9a, 0x07, 0x12, 0x80, 0xe2, 0xeb, 0x27, 0xb2, 0x75,

0x09, 0x83, 0x2c, 0x1a, 0x1b, 0x6e, 0x5a, 0xa0, 0x52, 0x3b, 0xd6, 0xb3, 0x29, 0xe3, 0x2f, 0x84,

0x53, 0xd1, 0x00, 0xed, 0x20, 0xfc, 0xb1, 0x5b, 0x6a, 0xcb, 0xbe, 0x39, 0x4a, 0x4c, 0x58, 0xcf,

0xd0, 0xef, 0xaa, 0xfb, 0x43, 0x4d, 0x33, 0x85, 0x45, 0xf9, 0x02, 0x7f, 0x50, 0x3c, 0x9f, 0xa8,

0x51, 0xa3, 0x40, 0x8f, 0x92, 0x9d, 0x38, 0xf5, 0xbc, 0xb6, 0xda, 0x21, 0x10, 0xff, 0xf3, 0xd2,

0xcd, 0x0c, 0x13, 0xec, 0x5f, 0x97, 0x44, 0x17, 0xc4, 0xa7, 0x7e, 0x3d, 0x64, 0x5d, 0x19, 0x73,

0x60, 0x81, 0x4f, 0xdc, 0x22, 0x2a, 0x90, 0x88, 0x46, 0xee, 0xb8, 0x14, 0xde, 0x5e, 0x0b, 0xdb,

0xe0, 0x32, 0x3a, 0x0a, 0x49, 0x06, 0x24, 0x5c, 0xc2, 0xd3, 0xac, 0x62, 0x91, 0x95, 0xe4, 0x79,

0xe7, 0xc8, 0x37, 0x6d, 0x8d, 0xd5, 0x4e, 0xa9, 0x6c, 0x56, 0xf4, 0xea, 0x65, 0x7a, 0xae, 0x08,

0xba, 0x78, 0x25, 0x2e, 0x1c, 0xa6, 0xb4, 0xc6, 0xe8, 0xdd, 0x74, 0x1f, 0x4b, 0xbd, 0x8b, 0x8a,

0x70, 0x3e, 0xb5, 0x66, 0x48, 0x03, 0xf6, 0x0e, 0x61, 0x35, 0x57, 0xb9, 0x86, 0xc1, 0x1d, 0x9e,

0xe1, 0xf8, 0x98, 0x11, 0x69, 0xd9, 0x8e, 0x94, 0x9b, 0x1e, 0x87, 0xe9, 0xce, 0x55, 0x28, 0xdf,

0x8c, 0xa1, 0x89, 0x0d, 0xbf, 0xe6, 0x42, 0x68, 0x41, 0x99, 0x2d, 0x0f, 0xb0, 0x54, 0xbb, 0x16

};

vector<unsigned char> v;

for (auto i : line) {

v.push\_back(Sbox[i]);

}

line = v;

}

vector<unsigned char> AddRoundKey(vector<unsigned char>& FIRST, vector<unsigned char>& SECOND) {

vector<unsigned char> v;

for (int i = 0; i < 4; ++i) {

v.push\_back(FIRST[i] ^ SECOND[i]);

}

return v;

}

void KeyExpansion(vector<unsigned char>& MASTER\_KEY, vector<vector<unsigned char>>& ROUND\_KEYS) {

const vector<unsigned char> Rcon = {

0x00, 0x00, 0x00, 0x00,

0x01, 0x00, 0x00, 0x00,

0x02, 0x00, 0x00, 0x00,

0x04, 0x00, 0x00, 0x00,

0x08, 0x00, 0x00, 0x00,

0x10, 0x00, 0x00, 0x00,

0x20, 0x00, 0x00, 0x00,

0x40, 0x00, 0x00, 0x00,

0x80, 0x00, 0x00, 0x00,

0x1b, 0x00, 0x00, 0x00,

0x36, 0x00, 0x00, 0x00

};

vector<unsigned char> v;

int i = 0;

int Nk = 4;

int Nb = 4;

int Nr = 10;

ROUND\_KEYS.resize(Nb \* (Nr + 1), vector<unsigned char>(4)); //размер 44 по 4 строчки

while (i < Nk) {

v = { MASTER\_KEY[4 \* i], MASTER\_KEY[4 \* i + 1], MASTER\_KEY[4 \* i + 2], MASTER\_KEY[4 \* i + 3] }; //генерация раундового ключа по алгоритму

ROUND\_KEYS[i] = v;

i++;

}

i = Nk;

while (i < (Nb \* (Nr + 1))) { //обработка раундовых ключей по алгоритму

v = ROUND\_KEYS[i - 1];

if (i % Nk == 4) {

ShiftRows(v);

SubBytes(v);

for (int k = 0; k < v.size(); ++k) {

v[k] = v[k] ^ Rcon[i / Nk];

}

}

else if (Nk > 6 && (i % Nk == 4)) {

SubBytes(v);

}

for (int j = 0; j < v.size(); ++j) {

ROUND\_KEYS[i][j] = ROUND\_KEYS[i - Nk][j] ^ v[j];

}

i++;

}

}

unsigned char galois\_multiply(unsigned char a, unsigned char b) {

unsigned char result = 0; // Инициализация переменной для хранения результата умножения

unsigned char carry; // Переменная для хранения переноса

for (int i = 0; i < 8; ++i) { // Цикл по битам переменной b (предполагается, что char содержит 8 бит)

if (b & 1) { // Если младший бит b равен 1

result ^= a; // Выполняем XOR текущего результата с переменной a

}

carry = a & 0x80; // Проверяем старший бит переменной a перед сдвигом

a <<= 1; // Сдвигаем переменную a влево на 1 бит

if (carry) { // Если старший бит a был 1

a ^= 0x1b; // Выполняем XOR переменной a с модулем AES

}

b >>= 1; // Сдвигаем переменную b вправо на 1 бит

}

return result; // Возвращаем результат умножения

}

vector<unsigned char> generate\_mult\_by\_2() {

vector<unsigned char> mult\_by\_2(256);

for (int i = 0; i < 256; ++i) {

mult\_by\_2[i] = galois\_multiply(i, 2);

}

return mult\_by\_2;

}

// Генерация таблицы mult\_by\_3

vector<unsigned char> generate\_mult\_by\_3() {

vector<unsigned char> mult\_by\_3(256);

for (int i = 0; i < 256; ++i) {

mult\_by\_3[i] = galois\_multiply(i, 3);

}

return mult\_by\_3;

}

void mix\_column(vector<unsigned char>& line) {

vector<unsigned char> mult\_by\_2 = generate\_mult\_by\_2(); //операции умножения на 2 и 3 к столбцам матрицы данных

vector<unsigned char> mult\_by\_3 = generate\_mult\_by\_3();

vector<unsigned char> v;

v.push\_back(mult\_by\_2[line[0]] ^ mult\_by\_3[line[1]] ^ line[2] ^ line[3]);

v.push\_back(mult\_by\_2[line[1]] ^ mult\_by\_3[line[2]] ^ line[0] ^ line[3]);

v.push\_back(mult\_by\_2[line[2]] ^ mult\_by\_3[line[3]] ^ line[0] ^ line[1]);

v.push\_back(mult\_by\_2[line[3]] ^ mult\_by\_3[line[0]] ^ line[1] ^ line[2]);

line = v;

}

vector<vector<unsigned char>> XOR\_blocks(const vector<vector<unsigned char>>& a, const vector<vector<unsigned char>>& b) {

vector<vector<unsigned char>> res(4, vector<unsigned char>(4, 0));

for (int i = 0; i < 4; ++i) {

//res.push\_back(matrixRow);

for (int j = 0; j < 4; ++j) {

res[i][j] = (a[i][j] ^ b[i][j]);

}

}

return res;

}

void BLOCK\_GENERATE(string& text, vector<vector<vector<unsigned char>>>& BLOCK) {

while (text.size() % 16 != 0) { // добавление пробелов

text += ' ';

}

vector<vector<vector<unsigned char>>> v; // временный BLOCK

int count = 0;

vector<vector<unsigned char>> sixteen(4, vector<unsigned char>(4)); // матрица 4

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

int a = (i % 16) % 4; // по блокам выбирается строка

int b = (i % 16) / 4;// по блокам выбирается столбец

sixteen[a][b] = text[i];

if ((i + 1) % 16 == 0) { //если блок заполнился

v.push\_back(sixteen);

sixteen = vector<vector<unsigned char>>(4, vector<unsigned char>(4));

}

}

BLOCK = v;

}

vector<vector<unsigned char>> CIPHER(vector<vector<unsigned char>>& BLOCK, vector<vector<unsigned char>>& ROUND\_KEYS, vector<unsigned char> MASTER\_KEY) {

vector<vector<unsigned char>> v(4, vector<unsigned char>(4, 0)); //первый addroundkey

for (int i = 0; i < 4; ++i) {

v[i] = AddRoundKey(BLOCK[i], ROUND\_KEYS[i]);

}

//ROUNDS\_KEYS[0] - забрали

for (int i = 1; i <= 9; i++) { // REPEAT 10 - 1

for (int j = 0; j <= 3; j++) { //обрабатываем каждую строку BLOCK

SubBytes(v[j]);

ShiftRows(v[j]);

mix\_column(v[j]);

v[j] = AddRoundKey(v[j], ROUND\_KEYS[i]);

}

}

//ROUNDS\_KEYS[0] - ROUNDS\_KEYS[9] - забрали

for (int j = 0; j <= 3; j++) {

SubBytes(v[j]);

ShiftRows(v[j]);

v[j] = AddRoundKey(v[j], ROUND\_KEYS[10]);

}//ROUNDS\_KEYS[10] - забрали

return v;

}

int main()

{

mt19937\_64 mt(random\_device{}());

vector<vector<unsigned char>> ROUND\_KEYS;

vector<unsigned char> MASTER\_KEY;

vector<vector<vector<unsigned char>>> BLOCK;

string text;

cout << "enter text for encryption ==>> ";

getline(cin, text);

BLOCK\_GENERATE(text, BLOCK);

cout << "----------------------------------------------";

cout << "\nencryption block looks like this: \n";

for (auto i : BLOCK) {

for (auto j : i) {

for (auto k : j) {

cout << setw(4) << k << " ";

}

cout << endl;

}

cout << "\n------";

}

cout << "\n----------------------------------------------" << endl;

GENERATE\_MASTER\_KEY(MASTER\_KEY, mt);

cout << "\nmaster key 128 bit: ";

for (auto i : MASTER\_KEY) {

cout << i;

}

cout << "\n\n----------------------------------------------\n";

KeyExpansion(MASTER\_KEY, ROUND\_KEYS); // ГЕНЕРАЦИЯ КЛЮЧЕЙ

cout << "generated keys \n----------------------------------------------\n";

for (auto i : ROUND\_KEYS) {

for (auto j : i) {

cout << hex << setw(4) << static\_cast<int>(j) << " ";

}

cout << endl;

}

cout << "\n----------------------------------------------\n" << endl;

vector<vector<unsigned char>> PREV(4, vector<unsigned char>(4, 0));

vector<vector<unsigned char>> DEFOLT = PREV;

vector<vector<unsigned char>> TEK;

vector<vector<vector<unsigned char>>> ECRYPT;

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

vector<vector<unsigned char>> res(4, vector<unsigned char>(4, 0));

vector<vector<unsigned char>> B = BLOCK[i];

TEK = CIPHER(PREV, ROUND\_KEYS, MASTER\_KEY);

for (int i = 0; i < 4; ++i) {

for (int j = 0; j < 4; ++j) {

res[i][j] = (TEK[i][j] ^ B[i][j]);

}

}

ECRYPT.push\_back(res);

PREV = TEK;

}

cout << "final cipher after encryption:\n" << endl;

for (auto t : ECRYPT) {

for (auto i : t) {

for (auto j : i) {

cout << setw(4) << j << " ";

}

cout << endl;

}

}

cout << "\n\n----------------------------------------------\n" << endl;

PREV = DEFOLT;

vector<vector<vector<unsigned char>>> DECRYPT;

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

vector<vector<unsigned char>> res(4, vector<unsigned char>(4, 0));

vector<vector<unsigned char>> B = ECRYPT[i];

TEK = CIPHER(PREV, ROUND\_KEYS, MASTER\_KEY);

for (int i = 0; i < 4; ++i) {

for (int j = 0; j < 4; ++j) {

res[i][j] = (B[i][j] ^ TEK[i][j]);

}

}

DECRYPT.push\_back(res);

PREV = TEK;

}

cout << "received message after decryption:\n" << endl;

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

vector<vector<unsigned char>>& decrypt = DECRYPT[i];

for (int j = 0; j < decrypt.size(); j++) {

for (int k = 0; k < decrypt[j].size(); k++) {

cout << decrypt[k][j];

}

}

}

cout << "\n\n----------------------------------------------\n" << endl;

}

**Язык программирования: Java  
Код программы:**  
import java.util.\*;

public class AESExample {

private static final int[] Rcon = {

0x00, 0x00, 0x00, 0x00,

0x01, 0x00, 0x00, 0x00,

0x02, 0x00, 0x00, 0x00,

0x04, 0x00, 0x00, 0x00,

0x08, 0x00, 0x00, 0x00,

0x10, 0x00, 0x00, 0x00,

0x20, 0x00, 0x00, 0x00,

0x40, 0x00, 0x00, 0x00,

0x80, 0x00, 0x00, 0x00,

0x1b, 0x00, 0x00, 0x00,

0x36, 0x00, 0x00, 0x00

};

private static final byte[] Sbox = {

// S-box values here

};

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter text for encryption ==>> ");

String text = scanner.nextLine();

List<List<List<Byte>>> block = new ArrayList<>();

blockGenerate(text, block);

System.out.println("Encryption block looks like this:");

for (List<List<Byte>> b : block) {

for (List<Byte> row : b) {

for (Byte val : row) {

System.out.printf("%4d ", val);

}

System.out.println();

}

System.out.println("------");

}

Random rand = new Random();

List<Byte> masterKey = new ArrayList<>();

generateMasterKey(masterKey, rand);

System.out.print("\nMaster key 128 bit: ");

for (Byte b : masterKey) {

System.out.print((char) (byte) b);

}

List<List<Byte>> roundKeys = new ArrayList<>();

keyExpansion(masterKey, roundKeys);

System.out.println("\nGenerated keys:");

for (List<Byte> key : roundKeys) {

for (Byte b : key) {

System.out.printf("%4x ", b);

}

System.out.println();

}

List<List<Byte>> prev = new ArrayList<>();

for (int i = 0; i < 4; i++) {

prev.add(new ArrayList<>(Collections.nCopies(4, (byte) 0)));

}

List<List<List<Byte>>> encrypted = new ArrayList<>();

for (List<List<Byte>> b : block) {

List<List<Byte>> tek = cipher(prev, roundKeys, masterKey);

List<List<Byte>> res = xorBlocks(tek, b);

encrypted.add(res);

prev = tek;

}

System.out.println("Final cipher after encryption:");

for (List<List<Byte>> t : encrypted) {

for (List<Byte> row : t) {

for (Byte val : row) {

System.out.printf("%4d ", val);

}

System.out.println();

}

}

prev.clear();

for (int i = 0; i < 4; i++) {

prev.add(new ArrayList<>(Collections.nCopies(4, (byte) 0)));

}

List<List<List<Byte>>> decrypted = new ArrayList<>();

for (List<List<Byte>> e : encrypted) {

List<List<Byte>> tek = cipher(prev, roundKeys, masterKey);

List<List<Byte>> res = xorBlocks(e, tek);

decrypted.add(res);

prev = tek;

}

System.out.println("Received message after decryption:");

for (List<List<Byte>> d : decrypted) {

for (List<Byte> row : d) {

for (Byte val : row) {

System.out.print((char) (byte) val);

}

}

}

}

private static void generateMasterKey(List<Byte> masterKey, Random rand) {

for (int i = 0; i < 16; i++) {

if (i % 2 == 0) {

masterKey.add((byte) (rand.nextInt(26) + 'a'));

} else {

masterKey.add((byte) (rand.nextInt(10) + '0'));

}

}

}

private static void blockGenerate(String text, List<List<List<Byte>>> block) {

while (text.length() % 16 != 0) {

text += ' ';

}

List<List<List<Byte>>> blocks = new ArrayList<>();

for (int i = 0; i < text.length(); i += 16) {

List<List<Byte>> sixteen = new ArrayList<>();

for (int j = 0; j < 4; j++) {

List<Byte> row = new ArrayList<>();

for (int k = 0; k < 4; k++) {

row.add((byte) text.charAt(i + j \* 4 + k));

}

sixteen.add(row);

}

blocks.add(sixteen);

}

block.addAll(blocks);

}

private static void keyExpansion(List<Byte> masterKey, List<List<Byte>> roundKeys) {

int nk = 4;

int nb = 4;

int nr = 10;

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

List<Byte> key = new ArrayList<>();

for (int j = 0; j < 4; j++) {

key.add(masterKey.get(i \* 4 + j));

}

roundKeys.add(key);

}

for (int i = nk; i < nb \* (nr + 1); i++) {

List<Byte> temp = new ArrayList<>(roundKeys.get(i - 1));

if (i % nk == 0) {

shiftRows(temp);

subBytes(temp);

for (int j = 0; j < temp.size(); j++) {

temp.set(j, (byte) (temp.get(j) ^ Rcon[i / nk]));

}

} else if (nk > 6 && i % nk == 4) {

subBytes(temp);

}

List<Byte> newKey = new ArrayList<>();

for (int j = 0; j < 4; j++) {

newKey.add((byte) (roundKeys.get(i - nk).get(j) ^ temp.get(j)));

}

roundKeys.add(newKey);

}

}

private static void subBytes(List<Byte> line) {

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

line.set(i, Sbox[line.get(i) & 0xFF]);

}

}

private static void shiftRows(List<Byte> line) {

Collections.rotate(line, -1);

}

private static List<Byte> addRoundKey(List<Byte> first, List<Byte> second) {

List<Byte> result = new ArrayList<>();

for (int i = 0; i < 4; i++) {

result.add((byte) (first.get(i) ^ second.get(i)));

}

return result;

}

private static void mixColumn(List<Byte> line) {

byte[] multBy2 = generateMultBy2();

byte[] multBy3 = generateMultBy3();

List<Byte> result = new ArrayList<>();

result.add((byte) (multBy2[line.get(0) & 0xFF] ^ multBy3[line.get(1) & 0xFF] ^ line.get(2) ^ line.get(3)));

result.add((byte) (multBy2[line.get(1) & 0xFF] ^ multBy3[line.get(2) & 0xFF] ^ line.get(0) ^ line.get(3)));

result.add((byte) (multBy2[line.get(2) & 0xFF] ^ multBy3[line.get(3) & 0xFF] ^ line.get(0) ^ line.get(1)));

result.add((byte) (multBy2[line.get(3) & 0xFF] ^ multBy3[line.get(0) & 0xFF] ^ line.get(1) ^ line.get(2)));

line.clear();

line.addAll(result);

}

private static byte[] generateMultBy2() {

byte[] multBy2 = new byte[256];

for (int i = 0; i < 256; i++) {

multBy2[i] = galoisMultiply((byte) i, (byte) 2);

}

return multBy2;

}

private static byte[] generateMultBy3() {

byte[] multBy3 = new byte[256];

for (int i = 0; i < 256; i++) {

multBy3[i] = galoisMultiply((byte) i, (byte) 3);

}

return multBy3;

}

private static byte galoisMultiply(byte a, byte b) {

byte p = 0;

byte hiBitSet;

for (int i = 0; i < 8; i++) {

if ((b & 1) != 0) {

p ^= a;

}

hiBitSet = (byte) (a & 0x80);

a <<= 1;

if (hiBitSet != 0) {

a ^= 0x1b;

}

b >>= 1;

}

return p;

}

private static List<List<Byte>> xorBlocks(List<List<Byte>> a, List<List<Byte>> b) {

List<List<Byte>> result = new ArrayList<>();

for (int i = 0; i < 4; i++) {

List<Byte> row = new ArrayList<>();

for (int j = 0; j < 4; j++) {

row.add((byte) (a.get(i).get(j) ^ b.get(i).get(j)));

}

result.add(row);

}

return result;

}

private static List<List<Byte>> cipher(List<List<Byte>> block, List<List<Byte>> roundKeys, List<Byte> masterKey) {

List<List<Byte>> result = new ArrayList<>();

for (int i = 0; i < 4; i++) {

result.add(addRoundKey(block.get(i), roundKeys.get(i)));

}

for (int round = 1; round <= 9; round++) {

for (int i = 0; i < 4; i++) {

subBytes(result.get(i));

shiftRows(result.get(i));

mixColumn(result.get(i));

result.set(i, addRoundKey(result.get(i), roundKeys.get(round \* 4 + i)));

}

}

for (int i = 0; i < 4; i++) {

subBytes(result.get(i));

shiftRows(result.get(i));

result.set(i, addRoundKey(result.get(i), roundKeys.get(10 \* 4 + i)));

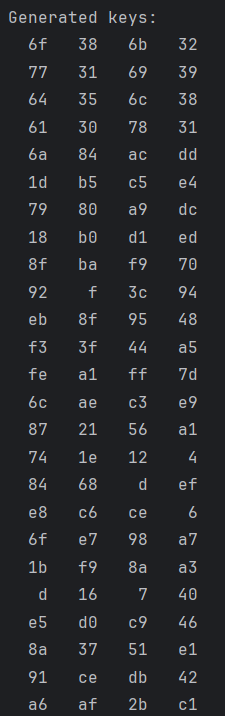
}

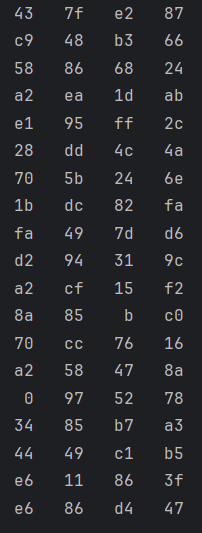
return result;

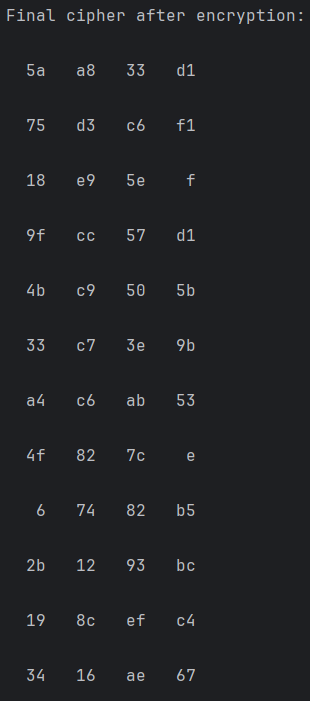
}

}  
**Результат работы программы:**









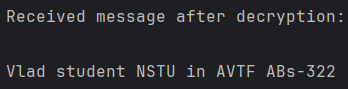
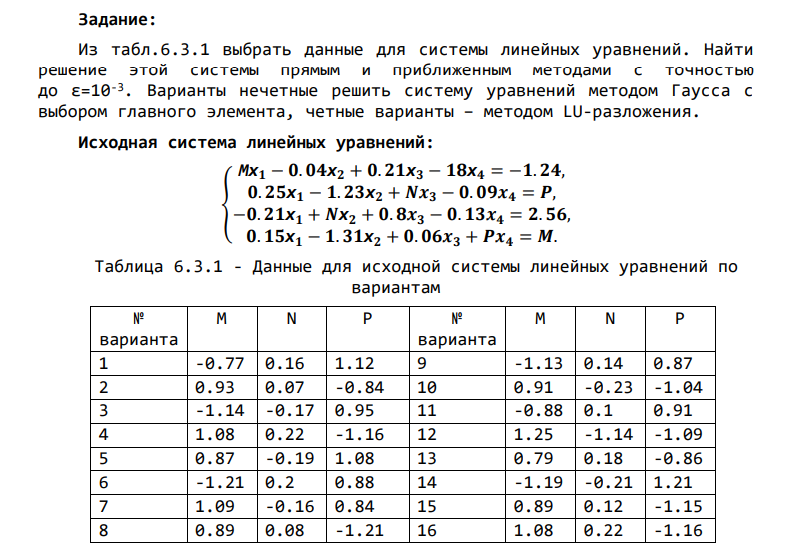


Рисунок 4 — Результат работы программы(AES)**Задание №6.3 Методы решения систем линейных алгебраических уравнений**

**Вариант 1**

 **Язык программирования : С++**

**Задание 6.3**

**Код программы :**

#include <iostream> // Подключение библиотеки для ввода-вывода

#include <vector> // Подключение библиотеки для работы с векторами

#include <cmath> // Подключение библиотеки для математических функций

#include <iomanip> // Подключение библиотеки для форматирования вывода

using namespace std; // Использование стандартного пространства имен

// Функция для решения системы линейных уравнений методом Гаусса с выбором главного элемента

void gaussElimination(vector<vector<double>>& A, vector<double>& b) {

int n = A.size(); // Получение размера матрицы A

// Прямой ход

for (int i = 0; i < n; i++) { // Цикл по строкам матрицы

// Поиск главного элемента в текущем столбце

double maxEl = abs(A[i][i]); // Инициализация максимального элемента текущим диагональным элементом

int maxRow = i; // Инициализация индекса строки с максимальным элементом

for (int k = i + 1; k < n; k++) { // Цикл по строкам ниже текущей

if (abs(A[k][i]) > maxEl) { // Если текущий элемент больше текущего максимального

maxEl = abs(A[k][i]); // Обновление максимального элемента

maxRow = k; // Обновление индекса строки с максимальным элементом

}

}

// Поменять местами текущую строку и строку с главным элементом

for (int k = i; k < n; k++) { // Цикл по столбцам текущей строки

swap(A[maxRow][k], A[i][k]); // Обмен элементов в строках maxRow и i

}

swap(b[maxRow], b[i]); // Обмен элементов в векторе свободных членов

// Обнуление элементов ниже главного элемента

for (int k = i + 1; k < n; k++) { // Цикл по строкам ниже текущей

double c = -A[k][i] / A[i][i]; // Вычисление коэффициента для обнуления элемента

for (int j = i; j < n; j++) { // Цикл по столбцам

if (i == j) { // Если диагональный элемент

A[k][j] = 0; // Обнуление элемента

} else {

A[k][j] += c \* A[i][j]; // Обновление элемента

}

}

b[k] += c \* b[i]; // Обновление элемента вектора свободных членов

}

}

// Обратный ход

vector<double> x(n); // Вектор для хранения решения

for (int i = n - 1; i >= 0; i--) { // Цикл по строкам матрицы снизу вверх

x[i] = b[i] / A[i][i]; // Нахождение значения переменной

for (int k = i - 1; k >= 0; k--) { // Цикл по строкам выше текущей

b[k] -= A[k][i] \* x[i]; // Подстановка найденного значения в верхнюю строку

}

}

// Вывод решения

cout << "Метод Гаусса:" << endl; // Вывод заголовка

for (int i = 0; i < n; i++) { // Цикл по найденным значениям переменных

cout << "x" << i + 1 << " = " << x[i] << endl; // Вывод значения переменной

}

}

// Функция для решения системы линейных уравнений методом простых итераций (метод Якоби)

void simpleIterations(const vector<vector<double>>& A, const vector<double>& b, double epsilon) {

int n = A.size(); // Получение размера матрицы A

vector<double> x(n, 0.0); // Начальное приближение

vector<double> x\_new(n, 0.0); // Вектор для нового приближения

int iterations = 0; // Счетчик итераций

double error; // Переменная для хранения ошибки

// Заголовок таблицы

cout << left << setw(5) << "N" << setw(15) << "x1" << setw(15) << "x2" << setw(15) << "x3" << setw(15) << "x4" << setw(15) << "Точность" << endl;

cout << string(65, '-') << endl; // Разделительная линия

do {

iterations++; // Увеличение счетчика итераций

error = 0.0; // Обнуление ошибки

for (int i = 0; i < n; i++) { // Цикл по строкам матрицы

double sum = 0.0; // Переменная для суммы

for (int j = 0; j < n; j++) { // Цикл по столбцам матрицы

if (i != j) { // Если не диагональный элемент

sum += A[i][j] \* x[j]; // Суммирование произведения элемента матрицы на соответствующий элемент вектора

}

}

x\_new[i] = (b[i] - sum) / A[i][i]; // Вычисление нового приближения

}

for (int i = 0; i < n; i++) { // Цикл по строкам матрицы

error = max(error, abs(x\_new[i] - x[i])); // Вычисление максимальной ошибки

x[i] = x\_new[i]; // Обновление приближения

}

// Вывод промежуточных значений в таблицу

cout << left << setw(5) << iterations; // Вывод номера итерации

for (int i = 0; i < n; i++) { // Цикл по строкам матрицы

cout << setw(15) << fixed << setprecision(6) << x[i]; // Вывод значений переменных

}

cout << setw(15) << fixed << setprecision(6) << error << endl; // Вывод ошибки

} while (error > epsilon); // Проверка условия остановки

// Вывод окончательного решения

cout << "Метод простых итераций:" << endl; // Вывод заголовка

for (int i = 0; i < n; i++) { // Цикл по найденным значениям переменных

cout << "x" << i + 1 << " = " << x[i] << endl; // Вывод значения переменной

}

}

int main() {

// Матрица коэффициентов и вектор свободных членов

vector<vector<double>> A = { // Инициализация матрицы коэффициентов

{-0.77, -0.04, 0.21, -18},

{0.25, -1.23, 0.16, -0.09},

{-0.21, 0.16, 0.8, -0.13},

{0.15, -1.31, 0.06, 1.12}

};

vector<double> b = {-1.24, 1.12, 2.56, -0.77}; // Инициализация вектора свободных членов

gaussElimination(A, b); // Вызов функции для решения системы уравнений методом Гаусса

// Для метода простых итераций используем те же коэффициенты и вектор

// Задаем точность

double epsilon = 0.001; // Установка точности

simpleIterations(A, b, epsilon); // Вызов функции для решения системы уравнений методом простых итераций

return 0; // Завершение программы

}

**Язык программирования : Java**

**Задание 6.3**

**Код программы :**

import java.util.ArrayList;

import java.util.List;

public class LinearEquationsSolver {

// Функция для решения системы линейных уравнений методом Гаусса с выбором главного элемента

public static void gaussElimination(List<List<Double>> A, List<Double> b) {

int n = A.size(); // Получение размера матрицы A

// Прямой ход

for (int i = 0; i < n; i++) { // Цикл по строкам матрицы

// Поиск главного элемента в текущем столбце

double maxEl = Math.abs(A.get(i).get(i)); // Инициализация максимального элемента текущим диагональным элементом

int maxRow = i; // Инициализация индекса строки с максимальным элементом

for (int k = i + 1; k < n; k++) { // Цикл по строкам ниже текущей

if (Math.abs(A.get(k).get(i)) > maxEl) { // Если текущий элемент больше текущего максимального

maxEl = Math.abs(A.get(k).get(i)); // Обновление максимального элемента

maxRow = k; // Обновление индекса строки с максимальным элементом

}

}

// Поменять местами текущую строку и строку с главным элементом

for (int k = i; k < n; k++) { // Цикл по столбцам текущей строки

double tmp = A.get(maxRow).get(k);

A.get(maxRow).set(k, A.get(i).get(k));

A.get(i).set(k, tmp);

}

double tmp = b.get(maxRow);

b.set(maxRow, b.get(i));

b.set(i, tmp);

// Обнуление элементов ниже главного элемента

for (int k = i + 1; k < n; k++) { // Цикл по строкам ниже текущей

double c = -A.get(k).get(i) / A.get(i).get(i); // Вычисление коэффициента для обнуления элемента

for (int j = i; j < n; j++) { // Цикл по столбцам

if (i == j) { // Если диагональный элемент

A.get(k).set(j, 0.0); // Обнуление элемента

} else {

A.get(k).set(j, A.get(k).get(j) + c \* A.get(i).get(j)); // Обновление элемента

}

}

b.set(k, b.get(k) + c \* b.get(i)); // Обновление элемента вектора свободных членов

}

}

// Обратный ход

List<Double> x = new ArrayList<>(n); // Вектор для хранения решения

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

x.add(0.0);

}

for (int i = n - 1; i >= 0; i--) { // Цикл по строкам матрицы снизу вверх

x.set(i, b.get(i) / A.get(i).get(i)); // Нахождение значения переменной

for (int k = i - 1; k >= 0; k--) { // Цикл по строкам выше текущей

b.set(k, b.get(k) - A.get(k).get(i) \* x.get(i)); // Подстановка найденного значения в верхнюю строку

}

}

// Вывод решения

System.out.println("Метод Гаусса:"); // Вывод заголовка

for (int i = 0; i < n; i++) { // Цикл по найденным значениям переменных

System.out.println("x" + (i + 1) + " = " + x.get(i)); // Вывод значения переменной

}

}

// Функция для решения системы линейных уравнений методом простых итераций (метод Якоби)

public static void simpleIterations(List<List<Double>> A, List<Double> b, double epsilon) {

int n = A.size(); // Получение размера матрицы A

List<Double> x = new ArrayList<>(n); // Начальное приближение

List<Double> x\_new = new ArrayList<>(n); // Вектор для нового приближения

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

x.add(0.0);

x\_new.add(0.0);

}

int iterations = 0; // Счетчик итераций

double error; // Переменная для хранения ошибки

// Заголовок таблицы

System.out.printf("%-5s %-15s %-15s %-15s %-15s %-15s%n", "N", "x1", "x2", "x3", "x4", "Точность");

System.out.println(String.join("", Collections.nCopies(65, "-"))); // Разделительная линия

do {

iterations++; // Увеличение счетчика итераций

error = 0.0; // Обнуление ошибки

for (int i = 0; i < n; i++) { // Цикл по строкам матрицы

double sum = 0.0; // Переменная для суммы

for (int j = 0; j < n; j++) { // Цикл по столбцам матрицы

if (i != j) { // Если не диагональный элемент

sum += A.get(i).get(j) \* x.get(j); // Суммирование произведения элемента матрицы на соответствующий элемент вектора

}

}

x\_new.set(i, (b.get(i) - sum) / A.get(i).get(i)); // Вычисление нового приближения

}

for (int i = 0; i < n; i++) { // Цикл по строкам матрицы

error = Math.max(error, Math.abs(x\_new.get(i) - x.get(i))); // Вычисление максимальной ошибки

x.set(i, x\_new.get(i)); // Обновление приближения

}

// Вывод промежуточных значений в таблицу

System.out.printf("%-5d", iterations); // Вывод номера итерации

for (int i = 0; i < n; i++) { // Цикл по строкам матрицы

System.out.printf("%-15.6f", x.get(i)); // Вывод значений переменных

}

System.out.printf("%-15.6f%n", error); // Вывод ошибки

} while (error > epsilon); // Проверка условия остановки

// Вывод окончательного решения

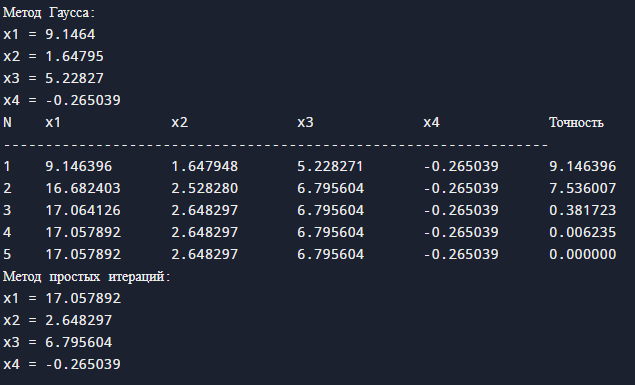
System.out.println("Метод простых итераций:"); // Вывод заголовка

for (int i = 0; i < n; i++) { // Цикл по найденным значениям переменных

System.out.println("x" + (i + 1) + " = " + x.get(i)); // Вывод значения переменной

}

Рисунок 5 – Результат работы программы



**Вывод**

В ходе лабораторной работы были выполнены все поставленные задачи, которые были направлены на изучение алгоритмов формирования и обработки двумерных массивов, программирование и отладка программ формирования и обработки массивов (статических и динамических) и контейнеров STL, на двух языках программирования: C++ и Java.