МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ государственное БЮДЖЕТНОЕ

образовательное учреждение

высшего образования

«НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

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

Кафедра защиты информации

**

**Лабораторная работа №3**

«Изучение циклических алгоритмов, операторов цикла,

программирование циклического вычислительного процесса»

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

|  |  |
| --- | --- |
| Выполнил:  Студент гр. «АБ-323», «АВТФ»  *Раков Михаил Андреевич*  «30» мая 2024г  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись) | Проверил:  *Ассистент кафедры ЗИ*  *Исаев Глеб Андреевич*  «3» июня 2024г  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (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**: реализовать циклический вычислительный процесс. Самостоятельно решить задачи в соответствии с индивидуальным вариантом.

**Задание № 1 (вариант** 7)

1. Определите и инициализируйте матрицу размерности M × N случайными целыми числами в диапазоне [-50, 50]. Найдите в строках матрицы самую длинную возрастающую последовательность элементов. Инициализируйте новый массив данной последовательностью.
2. Определите и инициализируйте матрицу размерности M × N случайными целыми числами в диапазоне [100, 200]. Найдите и выведите среднее арифметическое чётных столбцов и среднее арифметическое нечётных строк. Если больше арифметическое чётных столбцов, то отсортируйте главную диагональ в порядке убывания, если меньше – в порядке возрастания.
3. Дана матрица символов и строка, содержащая слово. Напишите программу, определяющую, содержится ли заданное слово в матрице или не содержится. Слово может быть составлено из букв последовательно смежных ячеек, где соседние ячейки являются соседними по горизонтали или вертикали. Одна и та же буквенная ячейка не может использоваться более одного раза.

**Код программы (на С++)**

*Файл task1.cpp*

#include <iostream>

#include <vector>

#include <string>

#include <set>

using namespace std;

using Koord = pair<size\_t, size\_t>;

bool find\_word(vector<string> const& m, Koord k, string const& s, set<Koord> st)

{

auto [i, j] = k;

if (i < 0 or j < 0 or i >= m.size() or j >= m[0].size())

return false;

if (st.count({i, j})) return false;

if (s == "") return true;

if (s[0] != m[i][j]) return false;

auto newStr = s.substr(1, s.size() - 1);

st.insert({i, j});

return find\_word(m, {i-1, j}, newStr, st)

or find\_word(m, {i+1, j}, newStr, st)

or find\_word(m, {i, j-1}, newStr, st)

or find\_word(m, {i, j+1}, newStr, st);

}

bool f (vector<string> const& m, string const& s)

{

for (size\_t i = 0; i < m.size(); ++i)

{

for (size\_t j = 0; j < m[0].size(); ++j)

{

if (find\_word(m, {i, j}, s, {}))

return true;

}

}

return false;

}

int main()

{

vector<string> mtx =

{

"QWERH",

"JSPES",

"SFRSF",

"ADOGS",

"SSGFI"

};

cout << (f(mtx, "JSPROGS") ? "true" : "false") << endl;

}

*Файл task2.cpp*

#include <iostream>

#include <vector>

#include <string>

#include <algorithm>

#include <numeric>

#include "matrix\_func.h"

using namespace std;

void sorted\_main\_diagonal(vector<vector<int>> &m, bool rev)

{

size\_t n = min(m.size(), m[0].size());

vector<int> v(n);

for (size\_t i = 0; i < n; ++i)

{

v[i] = m[i][i];

}

sort(v.begin(), v.end());

if (rev)

reverse(v.begin(), v.end());

for (size\_t i = 0; i < n; ++i)

{

m[i][i] = v[i];

}

}

int main()

{

size\_t M, N;

cin >> M >> N;

auto mtx = random\_matrix(M, N, 100, 200);

cout << mtx << endl;

double sumRows = 0;

for (size\_t i = 1; i < M; i += 2)

{

sumRows += accumulate(mtx[i].begin(), mtx[i].end(), 0);

}

double sumColumns = 0;

for (size\_t i = 0; i < N; i += 2)

{

double sum = 0;

for (size\_t j = 0; j < M; ++j)

{

sum += mtx[j][i];

}

sumColumns += sum;

}

size\_t cntRows = M / 2;

size\_t cntColumns = (N + 1) / 2;

if (sumRows / cntRows > sumColumns / cntColumns)

{

cout << "больше среднее арифметическое нечётных строк" << endl;

sorted\_main\_diagonal(mtx, false);

}

else

{

cout << "больше среднее арифметическое чётных столбцов" << endl;

sorted\_main\_diagonal(mtx, true);

}

cout << mtx;

}

*Файл task3.cpp*

#include <iostream>

#include <vector>

#include <string>

#include <set>

using namespace std;

using Koord = pair<size\_t, size\_t>;

bool find\_word(vector<string> const& m, Koord k, string const& s, set<Koord> st)

{

auto [i, j] = k;

if (i < 0 or j < 0 or i >= m.size() or j >= m[0].size())

return false;

if (st.count({i, j})) return false;

if (s == "") return true;

if (s[0] != m[i][j]) return false;

auto newStr = s.substr(1, s.size() - 1);

st.insert({i, j});

return find\_word(m, {i-1, j}, newStr, st)

or find\_word(m, {i+1, j}, newStr, st)

or find\_word(m, {i, j-1}, newStr, st)

or find\_word(m, {i, j+1}, newStr, st);

}

bool f (vector<string> const& m, string const& s)

{

for (size\_t i = 0; i < m.size(); ++i)

{

for (size\_t j = 0; j < m[0].size(); ++j)

{

if (find\_word(m, {i, j}, s, {}))

return true;

}

}

return false;

}

int main()

{

vector<string> mtx =

{

"QWERH",

"JSPES",

"SFRSF",

"ADOGS",

"SSGFI"

};

cout << (f(mtx, "JSPROGS") ? "true" : "false") << endl;

}

*Файл matrix\_func.h*

#ifndef MATRIX\_FUNC\_H

#define MATRIX\_FUNC\_H

#include <vector>

#include <iostream>

#include <iomanip>

template <class T>

std::ostream & operator<< (std::ostream& out, std::vector<std::vector<T>> const& m)

{

for (auto const& v : m)

{

for (auto el : v)

{

out << std::setw(5) << std::left << el;

}

out << std::endl;

}

return out;

}

template <class T>

std::vector<std::vector<T>>

random\_matrix (size\_t M, size\_t N, T minV, T maxV)

{

std::vector<std::vector<T>> m(M, std::vector<T>(N));

for (auto& v : m)

{

for (auto& el : v)

{

el = minV + (rand() % (maxV - minV));

}

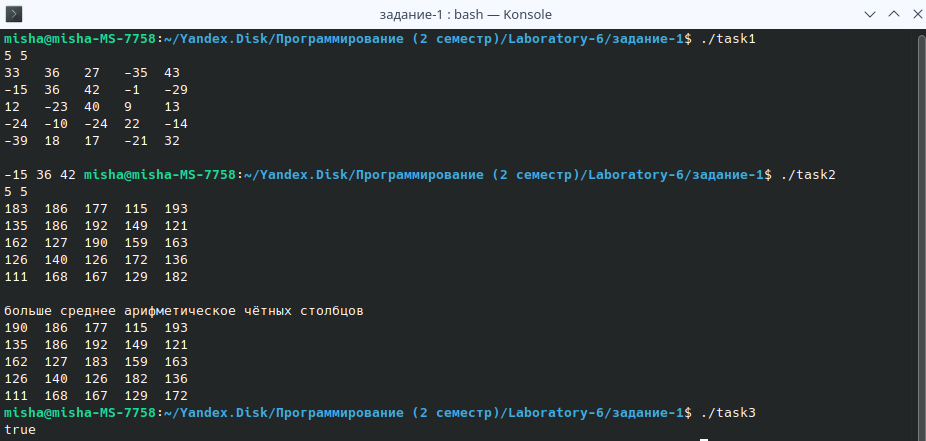
}

return m;

}

#endif

**Результат работы программы**



**Код программы (на Haskell)**

**Задание № 2 (вариант 2)**

Реализовать

1. систему шифрования AES128 (OFB) для преобразования исходного текста;
2. систему шифрования AES128 (OFB) для преобразования

зашифрованного текста в исходный.

Для всех вариантов, вывести все сгенерированные ключи, промежуточные результаты State, а также вектор инициализации, добавить генерацию случайного ключа и запись ключа в файл. Реализация должна работать с любым языком, как русским, так и английским.

**Код программы (на С++)**

*Файл main.cpp*

#include <iostream>

#include <fstream>

#include <string>

#include "aes\_func.h"

using namespace std;

Block\_128 random\_key ()

{

Block\_128 key;

for (auto it = byte\_begin(key); it != byte\_end(key); ++it)

{

\*it = rand() % 256;

}

return key;

}

string file\_to\_string (const string& name)

{

string res, s;

ifstream in (name);

while (getline(in, s))

{

res += (s + "\n");

}

in.close();

return res;

}

int main()

{

Block\_128 masterKey = random\_key();

cout << "Основной ключ:" << endl

<< masterKey << endl;

string text = file\_to\_string("text.txt");

string textEncrypt = aes\_encrypt(text, masterKey);

ofstream encrypt ("encrypt.txt");

encrypt << textEncrypt;

encrypt.close();

textEncrypt = file\_to\_string("encrypt.txt");

string textDecrypt = aes\_decrypt(textEncrypt, masterKey);

ofstream decrypt ("decrypt.txt");

decrypt << textDecrypt;

decrypt.close();

}

*Файл en\_de\_crypt.cpp*

#include <vector>

#include "sbox.h"

#include "aes\_func.h"

using namespace std;

vector<Block\_128> key\_expansion (const Block\_128& ikey, int Np, int Nk);

// ------------------------------ для Block\_128 ------------------------------

ostream & operator<<(ostream &out, Block\_128 const& block)

{

for (size\_t i = 0; i < 4; ++i)

{

for (size\_t j = 0; j < 4; ++j)

{

out << hex << (int) block[i][j] << " ";

}

out << endl;

}

return out;

}

pair<vector<Block\_128>, size\_t>

string\_to\_blocks (const string& s)

{

vector<Block\_128> blocks (s.size() / 16);

auto it = s.begin();

for (auto& block : blocks)

{

copy(it, it + 16, byte\_begin(block));

it += 16;

}

blocks.push\_back({});

copy(it, s.end(), byte\_begin(blocks.back()));

return { blocks, 16 - (s.size() % 16) };

}

string blocks\_to\_string (vector<Block\_128> const& blocks)

{

string res;

for (auto const& block : blocks)

{

copy(byte\_begin(block), byte\_end(block), back\_inserter(res));

}

return res;

}

// ------------------------------ общие функции ------------------------------

byte\_t powX\_GF (byte\_t a, int n)

{

while (n--)

{

byte\_t highBit = a & 0x80;

byte\_t shl = (a << 1) & 0xff;

a = (highBit == 0) ? shl : shl ^ 0x1b;

}

return a;

}

byte\_t mul\_GF (byte\_t a, byte\_t b)

{

word\_t res = 0, pwr = 0;

for (; b > 0; b /= 2, ++pwr)

{

if (b % 2 == 1)

res ^= powX\_GF(a, pwr);

}

return res;

}

void mul\_by\_column (Block\_128& block, Block\_128& mtx, size\_t n)

{

byte\_t temp[4] = {};

for (size\_t i = 0; i < 4; ++i)

{

for (size\_t j = 0; j < 4; ++j)

{

temp[i] ^= mul\_GF(mtx[i][j], block[j][n]);

}

}

for (size\_t i = 0; i < 4; ++i)

{

block[i][n] = temp[i];

}

}

void mix\_columns (Block\_128& block, Block\_128& mtx)

{

for (size\_t i = 0; i < 4; ++i)

{

mul\_by\_column (block, mtx, i);

}

}

void add\_round\_key (Block\_128& block, const Block\_128& key)

{

transform (byte\_begin(block), byte\_end(block), byte\_begin(key), byte\_begin(block), bit\_xor<>{});

}

// ------------------------------ зашифровка ------------------------------

void sub\_bytes (Block\_128& block)

{

transform (byte\_begin(block), byte\_end(block), byte\_begin(block), sBox);

}

void shift\_rows (Block\_128& block)

{

rot\_n\_bytes(block.words[1], 3);

rot\_n\_bytes(block.words[2], 2);

rot\_n\_bytes(block.words[3], 1);

}

void encrypt\_block (Block\_128& block, vector<Block\_128> const& keys)

{

Block\_128 mtx = {

0x02, 0x03, 0x01, 0x01,

0x01, 0x02, 0x03, 0x01,

0x01, 0x01, 0x02, 0x03,

0x03, 0x01, 0x01, 0x02

};

add\_round\_key(block, keys[0]);

for (size\_t i = 1; i <= 10 - 1; ++i)

{

sub\_bytes(block);

shift\_rows(block);

mix\_columns(block, mtx);

add\_round\_key(block, keys[i]);

}

sub\_bytes(block);

shift\_rows(block);

add\_round\_key(block, keys[10]);

}

string aes\_encrypt (string const& str, Block\_128 const& key)

{

auto [blocks, cnt] = string\_to\_blocks(str);

vector<Block\_128> keys = key\_expansion(key, 10, 4);

cout << "Промежуточные ключи:" << endl;

for (auto const& k : keys)

{

cout << k << endl;

}

for (auto& block : blocks)

{

encrypt\_block(block, keys);

}

return blocks\_to\_string(blocks);

}

// ------------------------------ расшифровка ------------------------------

void inv\_sub\_bytes (Block\_128& block)

{

transform (byte\_begin(block), byte\_end(block), byte\_begin(block), inv\_sBox);

}

void inv\_shift\_rows (Block\_128& block)

{

rot\_n\_bytes(block.words[1], 1);

rot\_n\_bytes(block.words[2], 2);

rot\_n\_bytes(block.words[3], 3);

}

void decrypt\_block (Block\_128& block, vector<Block\_128> const& keys)

{

Block\_128 mtx = {

0x0e, 0x0b, 0x0d, 0x09,

0x09, 0x0e, 0x0b, 0x0d,

0x0d, 0x09, 0x0e, 0x0b,

0x0b, 0x0d, 0x09, 0x0e,

};

add\_round\_key(block, keys[10]);

inv\_shift\_rows(block);

inv\_sub\_bytes(block);

for (size\_t i = 10-1; i >= 1; --i)

{

add\_round\_key(block, keys[i]);

mix\_columns(block, mtx);

inv\_sub\_bytes(block);

inv\_shift\_rows(block);

}

add\_round\_key(block, keys[0]);

}

string aes\_decrypt (string const& str, Block\_128 const& key)

{

auto [blocks, cnt] = string\_to\_blocks(str);

vector<Block\_128> keys = key\_expansion(key, 10, 4);

for (auto& block : blocks)

{

decrypt\_block(block, keys);

}

auto s = blocks\_to\_string(blocks);

s.erase(s.end() - cnt, s.end());

return s;

}

*Файл key\_expancion.cpp*

#include <vector>

#include "sbox.h"

#include "blocks\_words\_bytes.h"

using namespace std;

word\_t rCon (int round)

{

word\_t w = 0;

byte\_begin(w)[3] = (round == 10) ? 0x36 :

(round == 9) ? 0x1b :

1 << (round -1 );

return w;

}

word\_t rotWord (word\_t w)

{

rot\_n\_bytes(w, 3);

return w;

}

word\_t subWord(word\_t w)

{

transform (byte\_begin(w), byte\_end(w), byte\_begin(w), sBox);

return w;

}

vector<Block\_128> key\_expansion (const Block\_128& ikey, int Np, int Nk)

{

size\_t wordsCnt = (Np + 1) \* Nk;

vector<Block\_128> keys(Np + 1);

keys[0] = ikey;

auto get\_word = [&keys, Nk] (size\_t i) -> word\_t &

{

return keys[i / Nk].words[i % Nk];

};

for (size\_t i = Nk; i < wordsCnt; ++i)

{

auto temp = (i % Nk != 0)

? get\_word(i - 1)

: subWord(rotWord(get\_word(i - 1))) ^ rCon(i / Nk);

get\_word(i) = temp ^ get\_word(i - Nk);

}

return keys;

}

*Файл block\_words\_bytes.cpp*

#ifndef BYTE\_ITERATOR\_H

#define BYTE\_ITERATOR\_H

#include <cstdint>

#include <algorithm>

using word\_t = uint32\_t;

using byte\_t = uint8\_t;

template <class T> byte\_t \* byte\_begin (T& obj)

{

return reinterpret\_cast<byte\_t \*> (&obj);

}

template <class T> byte\_t const\* byte\_begin (T const& obj)

{

return reinterpret\_cast<byte\_t const\*> (&obj);

}

template <class T> auto byte\_end (T& obj)

{

return byte\_begin(obj) + sizeof(T);

}

template <class T> void rot\_n\_bytes (T& obj, int n)

{

std::rotate (byte\_begin(obj), byte\_begin(obj) + n, byte\_end(obj));

}

struct Block\_128

{

Block\_128 () = default;

Block\_128 (std::initializer\_list<byte\_t> il)

{

std::copy(il.begin(), il.end(), reinterpret\_cast<byte\_t \*> (words));

}

byte\_t \* operator[] (size\_t i)

{

return reinterpret\_cast<byte\_t \*> (&words[i]);

}

byte\_t const\* operator[] (size\_t i) const

{

return reinterpret\_cast<byte\_t const\*> (&words[i]);

}

word\_t words [4] = {};

};

#endif

*Файл aes\_func.cpp*

#ifndef AES\_FUNC\_H

#define AES\_FUNC\_H

#include <iostream>

#include <string>

#include "blocks\_words\_bytes.h"

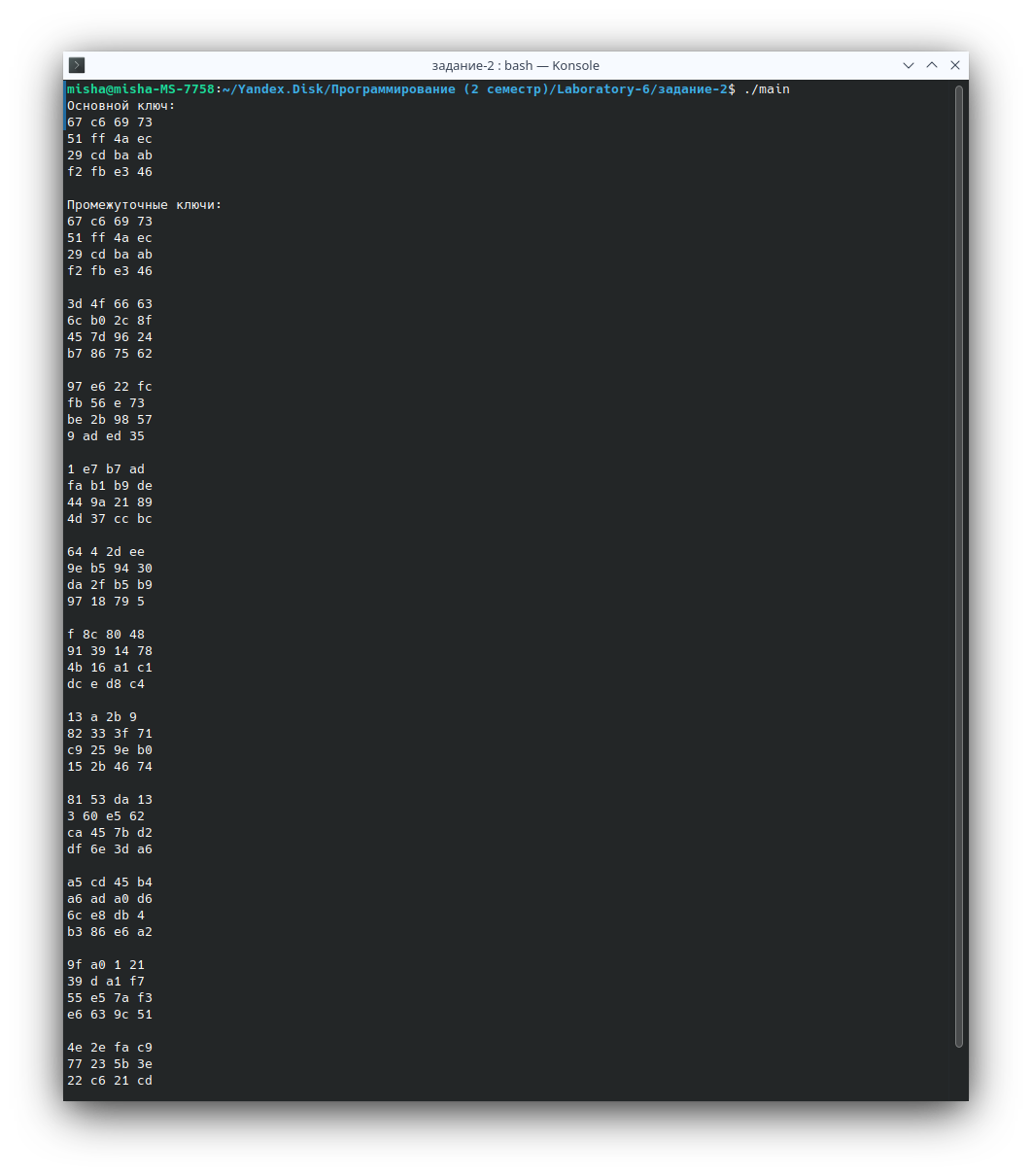
std::string aes\_encrypt (std::string const& str, Block\_128 const& key);

std::string aes\_decrypt (std::string const& str, Block\_128 const& key);

std::ostream & operator<< (std::ostream &out, Block\_128 const& block);

#endif

**Результаты работы программы**



**Код программы (на Haskell)**

**Задание № 3 (вариант 2)**

Из табл.6.3.1 выбрать данные для системы линейных уравнений. Найти решение этой системы прямым и приближенным методами с точностью до ε=10-3. Варианты нечетные решить систему уравнений методом Гаусса с выбором главного элемента, четные варианты – методом LU-разложения.

**Код программы (на С++)**

*Файл main.cpp*

#include "file.h"

using namespace std;

int main()

{

system("chcp 65001");

Matrix A =

{

{-1.21, -0.04, 0.21, -0.18},

{ 0.25, -1.23, 0.2, -0.09},

{-0.21, 0.2, 0.8, -0.13},

{ 0.15, -1.31, 0.06, 0.88}

};

Vector b = {-1.24, 0.88, 2.56, 0.1};

cout << "Метод LU-разложения: " << endl << endl;

Vector x = compute\_x(A, b);

cout << fixed << setprecision(3)

<< "x1 = " << x[0] << endl

<< "x2 = " << x[1] << endl

<< "x3 = " << x[2] << endl

<< "x4 = " << x[3] << endl

<< endl;

x = simple\_iterations(A, b, 0.01, 10);

cout << fixed << setprecision(3)

<< endl

<< "x1 = " << x[0] << endl

<< "x2 = " << x[1] << endl

<< "x3 = " << x[2] << endl

<< "x4 = " << x[3] << endl;

}

*Файл simple\_iterations.cpp*

#include "file.h"

using namespace std;

double matrix\_norm (Matrix const& m)

{

double mx = 0;

auto absAdd = [] (double prev, double d) { return prev + fabs(d); };

for (auto const& v : m)

{

double absSum = accumulate(v.begin(), v.end(), 0, absAdd);

mx = max(mx, absSum);

}

return mx;

}

double max\_diff (Vector const& v, Vector const& v2)

{

double mx = 0;

for (size\_t i = 0; i < v.size(); ++i)

{

mx = max(mx, fabs(v[i] - v2[i]));

}

return mx;

}

Vector operator\* (Matrix const& m, Vector const& v)

{

Vector res (v.size(), 0);

for (size\_t i = 0; i < m.size(); ++i)

{

for (size\_t j = 0; j < m.size(); ++j)

{

res[i] += m[i][j] \* v[j];

}

}

return res;

}

Vector operator+ (Vector v, Vector const& v2)

{

for (size\_t i = 0; i < v.size(); ++i)

{

v[i] += v2[i];

}

return v;

}

pair<Matrix, Vector>

canonical (Matrix const& A, Vector const& b)

{

size\_t n = A.size();

Matrix C (n, Vector (n));

Vector f (n);

for (size\_t i = 0; i < n; ++i)

{

f[i] = b[i] / A[i][i];

for (size\_t j = 0; j < n; ++j)

{

C[i][j] = (i != j)

? -A[i][j] / A[i][i]

: 0;

}

}

return {C, f};

}

Vector simple\_iterations (Matrix const& A, Vector const& b, double eps, size\_t maxIters)

{

Vector x (A.size(), 0);

auto [C, f] = canonical(A, b);

cout << "Метод простых итераций ("

<< (matrix\_norm(C) > 1 ? "не " : "") << "сходится): "

<< endl;

static auto titles = {"N", "x1", "x2", "x3", "x4", "eps"};

copy(titles.begin(), titles.end(), ostream\_iterator<string> (cout, "\t"));

cout << endl;

for (size\_t k = 0; k < maxIters; ++k)

{

Vector nextX = C \* x + f;

double maxDiff = max\_diff(nextX, x);

x = nextX;

if (maxDiff < eps)

break;

cout << fixed << setprecision(3) << (k + 1) << "\t";

copy(x.begin(), x.end(), ostream\_iterator<double> (cout, "\t"));

cout << maxDiff << endl;

}

return x;

}

*Файл lu\_decomposition.cpp*

#include "file.h"

using namespace std;

double & at (Matrix& m, size\_t i, size\_t j)

{

return m[i-1][j-1];

}

double & at (Vector& v, size\_t i)

{

return v[i-1];

}

double compute\_u (Matrix& a, Matrix& L, Matrix& U, size\_t i, size\_t j)

{

if (i == 1)

return at(a, 1, j) / at(L, 1, 1);

double sum = 0;

for (size\_t k = 1; k <= i-1; ++k)

{

sum += (at(L, i, k) \* at(U, k, j));

}

return (at(a, i, j) - sum) / at(L, i, i);

}

double compute\_l (Matrix& a, Matrix& L, Matrix& U, size\_t i, size\_t j)

{

if (j == 1)

return at(a, i, 1);

double sum = 0;

for (size\_t k = 1; k <= j-1; ++k)

{

sum += (at(L, i, k) \* at(U, k, j));

}

return at(a, i, j) - sum;

}

pair<Matrix, Matrix>

func\_LU (Matrix& m)

{

size\_t n = m.size();

Matrix l (n, Vector (n));

Matrix u (n, Vector (n));

for (size\_t i = 1; i <= n; ++i)

{

for (size\_t j = 1; j <= n; ++j)

{

if (i >= j)

at(l, i, j) = compute\_l(m, l, u, i, j);

if (j >= i)

at(u, i, j) = compute\_u(m, l, u, i, j);

}

}

return {l, u};

}

ostream & operator<< (ostream& out, Matrix const& m)

{

cout << fixed << setprecision(3);

for (auto& v : m)

{

for (auto el : v)

{

cout << left << setw(10) << el;

}

cout << endl;

}

return out;

}

Vector compute\_x (Matrix& m, Vector B)

{

auto [l, u] = func\_LU(m);

size\_t n = m.size();

cout << "Матрица L:\n" << l << endl

<< "Матрица U:\n" << u << endl;

Vector y (n);

at(y, 1) = at(B, 1) / at(l, 1, 1);

for (size\_t i = 2; i <= n; ++i)

{

double sum = 0;

for (size\_t k = 1; k <= i-1; ++k)

{

sum += (at(l, i, k) \* at(y, k));

}

at(y, i) = (at(B, i) - sum) / at(l, i, i);

}

Vector x (n);

at(x, n) = at(y, n);

for (size\_t i = n - 1; i >= 1; --i)

{

double sum = 0;

for (size\_t k = i + 1; k <= n; ++k)

{

sum += (at(u, i, k) \* at(x, k));

}

at(x, i) = at(y, i) - sum;

}

return x;

}

*Файл file.*h

#ifndef FILE\_H

#define File\_H

#include <iostream>

#include <vector>

#include <cmath>

#include <iomanip>

#include <algorithm>

#include <iterator>

#include <numeric>

using Vector = std::vector<double>;

using Matrix = std::vector<Vector>;

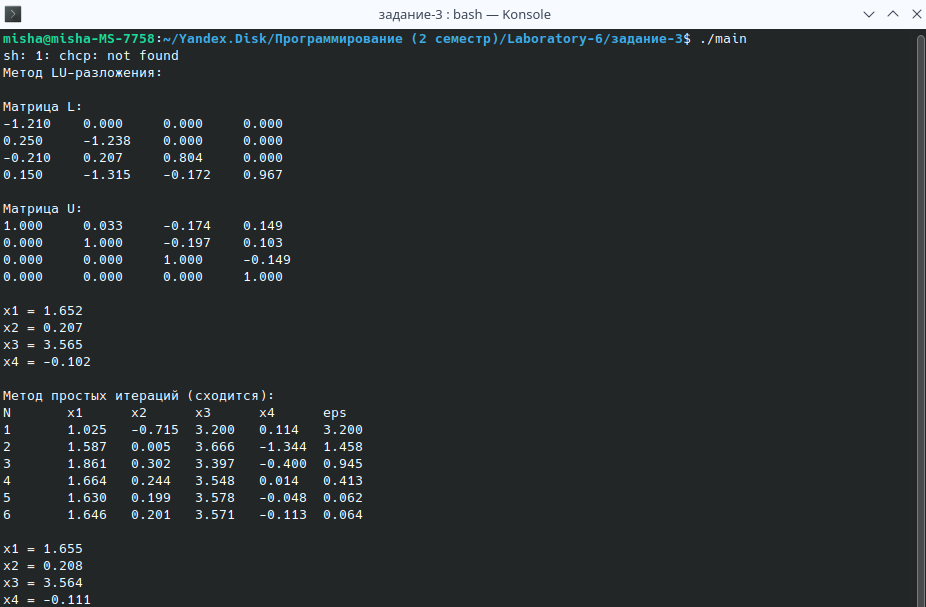
Vector compute\_x (Matrix&, Vector);

std::ostream & operator<< (std::ostream&, Matrix const&);

Vector simple\_iterations (Matrix const&, Vector const&, double, size\_t);

#endif

**Результаты работы программы**



**Код программы (на Haskell)**

**Вывод**

В соответствии с заданием по индивидуальному варианту была реализована обработка двумерных массивов и контейнеров. Также был реализован шифр aes.