# Программная реализация

#include <cstdio>

#include <cstring>

#include <fstream>

#include <iomanip>

#include <iostream>

#include <vector>

#include <string>

#include <sstream>

#include <cmath>

#define EPS 1E-14

double dabs(double \_) { return \_ > 0 ? \_ : -\_; }

// Матрица в профильном формате

template<typename T>

struct SkylineStorageMatrix{

    std::vector<int> ia;

    std::vector<T> al;

    std::vector<T> au;

    std::vector<T> di;

};

// Считать матрицу в профильном формате

template<typename T>

void readSkyline(std::istream& from, SkylineStorageMatrix<T>& to) {

    int n;

    from >> n;

    to.ia.resize(n + 1);

    for (int i = 0; i < n + 1; i++)

        from >> to.ia[i];

    int k = to.ia[n];

    to.al.resize(k);

    for (int i = 0; i < k; i++)

        from >> to.al[i];

    to.au.resize(k);

    for (int i = 0; i < k; i++)

        from >> to.au[i];

    to.di.resize(n);

    for (int i = 0; i < n; i++)

        from >> to.di[i];

}

// Сконвертировать матрицу в профильном формате в квадратную

template<typename T>

void skylineToSquare(SkylineStorageMatrix<T>& a, std::vector<std::vector<T>>& s) {

    #define elementsInLine(i) (a.ia[i + 1] - a.ia[i])

    s.resize(a.di.size());

    for (unsigned int i = 0; i < a.di.size(); i++)

    {

        s[i].resize(a.di.size());

        for (unsigned int j = 0; j < a.di.size(); j++)

            s[i][j] = (T)0;

    }

    for (unsigned int i = 0; i < a.di.size(); i++)

        s[i][i] = a.di[i];

    for (unsigned int i = 1; i < a.di.size(); i++)

    {

        int offset = i - elementsInLine(i);

        for (int j = 0; j < elementsInLine(i); j++)

        {

            s[i][offset + j] = a.al[a.ia[i] + j];

            s[offset + j][i] = a.au[a.ia[i] + j];

        }

    }

    #undef elementsInLine

}

 // Считать вектор

template<typename T>

void readVector(std::istream& from, std::vector<T>& to) {

    int n;

    from >> n;

    to.resize(n);

    for (int i = 0; i < n; i++)

        from >> to[i];

}

// Распечатать квадратную матрицу

template<typename T>

void printMatrix(const std::vector<std::vector<T>> a) {

    for (unsigned int i = 0; i < a.size(); i++)

    {

        for (unsigned int j = 0; j < a.size(); j++)

            std::cout << std::setw(20) << a[i][j];

        std::cout << std::endl;

    }

}

// Распечатать матрицу в профильном формате

template<typename T>

void printSkyline(SkylineStorageMatrix<T>& a) {

    std::vector<std::vector<T>> s;

    skylineToSquare(a, s);

    printMatrix(s);

}

// LU(sq)-разложение матрицы в профильном формате

template<typename T, typename T2=T>

bool LusqSkyline(SkylineStorageMatrix<T>& a){

    #define \_sqrt(dst, val) { double underSqrt = (double)(val); if (underSqrt <= 0) return false; dst = (T)sqrt(underSqrt); }

    #define elementsInLine(i) (a.ia[i + 1] - a.ia[i]) // Количество элементов в i'ой строке/столбце

    // Первая строка матрицы

    \_sqrt(a.di[0], a.di[0]); // d\_1 = sqrt(a\_11);

    for (unsigned int i = 1; i < a.di.size(); i++) {

        if (elementsInLine(i) == i)

            a.au[a.ia[i]] = a.au[a.ia[i]] / a.di[0]; // U\_1i = a\_1i/q1

    }

    // Обход по строкам матрицы

    for (unsigned int i = 1; i < a.di.size(); i++){

        if (a.al.size()){

            unsigned int alStartIndex = a.ia[i]; // Индекс первого ненулевого элемента i'ой строки в массиве al

            unsigned int alEndIndex = a.ia[i] + elementsInLine(i); // До сюда идти в цикле

            unsigned int j = i - elementsInLine(i);

            // До первого ненулевого элемента не было ненулевых элементов

            a.al[alStartIndex] = a.al[alStartIndex] / a.di[j];

            alStartIndex++;

            j++;

            // Основной цикл

            for (unsigned int alIndex = alStartIndex; alIndex < alEndIndex; alIndex++, j++){

                T sum = (T)0;

                unsigned int elementsInLineI = elementsInLine(i);

                unsigned int elementsInColJ = elementsInLine(j);

                for (unsigned int k = 0; k < j; k++){

                    T l\_ik = (k < i - elementsInLineI) ? (T)0 : a.al[a.ia[i] + (k - (i - elementsInLineI))];

                    T u\_kj = (k < j - elementsInColJ) ? (T)0 : a.au[a.ia[j] + (k - (j - elementsInColJ))];

                    sum += l\_ik \* u\_kj;

                }

                a.al[alIndex] = (a.al[alIndex] - sum) / a.di[j];

            }

        }

        // d\_i = sqrt(d\_i - \sum\_{k=1}{i-1} l\_ik \* u\_ki)

        {

            T2 sum = (T2)0;

            unsigned int elementsInLineI = elementsInLine(i);

            for (unsigned int k = 0; k < i; k++){

                T l\_ik = (k < i - elementsInLineI) ? (T)0 : a.al[a.ia[i] + (k - (i - elementsInLineI))];

                T l\_ki = (k < i - elementsInLineI) ? (T)0 : a.au[a.ia[i] + (k - (i - elementsInLineI))];

                sum += (T2)l\_ik \* (T2)l\_ki;

            }

            \_sqrt(a.di[i], a.di[i] - sum);

        }

        if (a.au.size()){

            for (unsigned int j = i + 1; j < a.di.size(); j++){

                if (i >= j - elementsInLine(j)){

                    T2 sum = (T2)0;

                    unsigned int elementsInLineI = elementsInLine(i);

                    unsigned int elementsInColJ = elementsInLine(j);

                    for (unsigned int k = 0; k < i; k++){

                        T l\_ik = (k < i - elementsInLineI) ? (T)0 : a.al[a.ia[i] + (k - (i - elementsInLineI))];

                        T u\_kj = (k < j - elementsInColJ) ? (T)0 : a.au[a.ia[j] + (k - (j - elementsInColJ))];

                        sum += (T2)l\_ik \* (T2)u\_kj;

                    }

                    unsigned int auIndex = a.ia[j] + (i - (j - elementsInLine(j)));

                    a.au[auIndex] = (a.au[auIndex] - sum) / a.di[i];

                }

            }

        }

    }

    #undef elementsInLine

    #undef \_sqrt

    return true;

}

// Решить уравнение Ly=b

template<typename T, typename T2=T>

bool findY(std::vector<T>& y, SkylineStorageMatrix<T>& a, std::vector<T>& b){

    #define \_div(dst, a, b) { if (((b) > (T)0 ? (b) : -(b)) < (T)EPS) return false; dst = (a) / (b); }

    #define elementsInLine(i) (a.ia[i + 1] - a.ia[i])

    for (unsigned int i = 0; i < a.di.size(); i++){

        T2 sum = (T2)0;

        unsigned int elementsInLineI = elementsInLine(i);

        for (unsigned int alIndex = a.ia[i], bIndex = i - elementsInLineI, counter = 0; counter < elementsInLineI; alIndex++, bIndex++, counter++)

            sum += (T2)a.al[alIndex] \* (T2)y[bIndex];

        \_div(y[i], b[i] - sum, a.di[i]);

    }

    #undef \_div

    #undef elementsInLine

    return true;

}

// Решить уравнение Ux=y

template<typename T, typename T2=T>

bool findX(std::vector<T>& x, SkylineStorageMatrix<T>& a, std::vector<T>& y){

    #define \_div(dst, a, b) { if (((b) > (T)0 ? (b) : -(b)) < (T)EPS) return false; dst = (a) / (b); }

    #define elementsInLine(i) (a.ia[i + 1] - a.ia[i])

    for (unsigned int i = a.di.size() - 1; true; i--){

        T2 sum = (T2)0;

        for (unsigned int j = i + 1; j < a.di.size(); j++){

            unsigned int elementsInColJ = elementsInLine(j);

            sum += (i < j - elementsInColJ) ? (T2)0 : (T2)a.au[a.ia[j] + (i - (j - elementsInColJ))] \* x[j];

        }

        \_div(x[i], y[i] - sum, a.di[i]);

        if (i == 0) break;

    }

    #undef \_div

    #undef elementsInLine

    return true;

}

// Решить уравнение Ax=B

template<typename T, typename T2=T>

bool solve(std::vector<T>& x, SkylineStorageMatrix<T>& a, std::vector<T>& b){

    std::vector<T> y(b.size());

    if (!LusqSkyline<T,T2>(a)){

        std::cerr << "Matrix is not LU(sq)-decomposeable" << std::endl;

        return false;

    }

    if (!findY<T,T2>(y, a, b) || !findX<T,T2>(x, a, y)){

        std::cerr << "System is inconsistent" << std::endl;

        return false;

    }

    return true;

}

// Записать матрицу в профильном формате

template<typename T>

void writeSkyline(SkylineStorageMatrix<T>& from, std::ostream& to){

    to << from.di.size() << std::endl;

    for (unsigned int i = 0; i < from.ia.size(); i++)

        to << from.ia[i] << " ";

    to << std::endl;

    for (unsigned int i = 0; i < from.al.size(); i++)

        to << std::fixed << std::setprecision(14) << from.al[i] << " ";

    to << std::endl;

    for (unsigned int i = 0; i < from.au.size(); i++)

        to << std::fixed << std::setprecision(14) << from.au[i] << " ";

    to << std::endl;

    for (unsigned int i = 0; i < from.di.size(); i++)

        to << std::fixed << std::setprecision(14) << from.di[i] << " ";

    to << std::endl;

}

// Записать вектор

template<typename T>

void writeVector(std::vector<T>& from, std::ostream& to){

    to << from.size() << std::endl;

    for (unsigned int i = 0; i < from.size(); i++)

        to << std::fixed << std::setprecision(14) << from[i] << std::endl;

    to << std::endl;

}

// Провести серию тестов на число обусловленности

void conditionNumberTestSeries(int tests){

    int counter = 1;

    std::ofstream outFile("condition\_method.txt");

    if (!outFile.is\_open()) {

        std::cerr << "Не удалось открыть файл для записи." << std::endl;

        return;

    }

    outFile << std::left

              << std::setw(5)  << "k"

              << std::setw(25) << "x^k (float)"

              << std::setw(20) << "|x\* - x^k| (float)"

              << std::setw(25) << "x^k (double)"

              << std::setw(20) << "|x\* - x^k| (double)"

              << std::setw(25) << "x^k (mixed)"

              << std::setw(20) << "|x\* - x^k| (mixed)"

              << std::endl;

    outFile << std::string(140, '-') << std::endl;

    for (int test = 0; test < tests; test++)

    {

        //float

        std::fstream f\_ifile("test6.txt");

        struct SkylineStorageMatrix<float> f\_a;

        readSkyline(f\_ifile, f\_a);

        int n = f\_a.di.size();

        std::vector<float> f\_b;

        readVector(f\_ifile, f\_b);

        f\_a.di[0] += (float)pow(10.0, -test);

        f\_b[0] += (float)pow(10.0, -test);

        std::vector<float> f\_x(n);

        bool ok1 = solve(f\_x, f\_a, f\_b);

        //double

        std::fstream d\_ifile("test6.txt");

        struct SkylineStorageMatrix<double> d\_a;

        readSkyline(d\_ifile, d\_a);

        std::vector<double> d\_b;

        readVector(d\_ifile, d\_b);

        d\_a.di[0] += (double)pow(10.0, -test);

        d\_b[0] += (double)pow(10.0, -test);

        std::vector<double> d\_x(n);

        bool ok2 = solve(d\_x, d\_a, d\_b);

        //mixed

        std::fstream m\_ifile("test6.txt");

        struct SkylineStorageMatrix<float> m\_a;

        readSkyline(m\_ifile, m\_a);

        std::vector<float> m\_b;

        readVector(m\_ifile, m\_b);

        m\_a.di[0] += (float)pow(10.0, -test);

        m\_b[0] += (float)pow(10.0, -test);

        std::vector<float> m\_x(n);

        bool ok3 = solve<float, double>(m\_x, m\_a, m\_b);

        for (int i = 0; i < n; i++){

            outFile << std::left << std::setw(5) << (counter%10 == 0? std::to\_string(counter/10) : " ")

                      << std::setw(25) << (ok1 ? std::to\_string(f\_x[i]) : "N/A")

                      << std::setw(20) << (ok1 ? std::to\_string(fabs((i + 1) - f\_x[i])) : "N/A")

                      << std::setw(25) << (ok2 ? std::to\_string(d\_x[i]) : "N/A")

                      << std::setw(20) << (ok2 ? std::to\_string(fabs((i + 1) - d\_x[i])) : "N/A")

                      << std::setw(25) << (ok3 ? std::to\_string(m\_x[i]) : "N/A")

                      << std::setw(20) << (ok3 ? std::to\_string(fabs((i + 1) - m\_x[i])) : "N/A")

                      << std::endl;

            counter++;

        }

        if (test != tests - 1) {

            outFile << std::string(140, '-') << std::endl;

        }

    }

    outFile.close();

}

// Решить систему Ax=B с плотной матрицей методом Гаусса с выбором ведущего элемента

template<typename T>

bool solveGauss(std::vector<std::vector<T>> A, std::vector<T>& x, std::vector<T> B){

    unsigned int n = A.size();

    if (n == 0 || A[0].size() != n || B.size() != n)

        return false; // Неверные размеры

    for (unsigned int k = 0; k < n - 1; k++){

        // Постолбцовый выбор главного элемента

        int m = k;

        for (int i = k + 1; i < n; i++){

            if (std::abs(A[i][k]) > std::abs(A[m][k]))

                m = i;

        }

        if (std::abs(A[m][k]) < 1E-5)

            return false; // Система вырождена

        if (m != k) { // Обмен строк

            std::swap(A[k], A[m]);

            std::swap(B[k], B[m]);

        }

        // Прямой ход

        for (int i = k + 1; i < n; i++){

            T factor = A[i][k] / A[k][k];

            B[i] -= factor \* B[k];

            for (int j = k; j < n; j++)

                A[i][j] -= factor \* A[k][j];

        }

    }

    // Обратный ход

    for (int k = n - 1; k >= 0; k--){

        T sum = 0.0;

        for (int j = k + 1; j < n; j++)

            sum += A[k][j] \* x[j];

        x[k] = (B[k] - sum) / A[k][k];

    }

    return true; // Успешное решение

}

// Сравнить Гаусса и разложение тестами на числом обусловленности

void compareGaussLusq(int tests){

    int counter = 1;

    std::ofstream outFile("сomparison\_methods.txt");

    if (!outFile.is\_open()) {

        std::cerr << "Не удалось открыть файл для записи." << std::endl;

        return;

    }

    // Заголовок таблицы

    outFile << std::left

              << std::setw(5)  << "k"

              << std::setw(25) << "x^k (LU(sq))"

              << std::setw(20) << "|x\* - x^k| (LU(sq))"

              << std::setw(25) << "x^k (Gauss)"

              << std::setw(20) << "|x\* - x^k| (Gauss)"

              << std::endl;

    outFile << std::string(120, '-') << std::endl;

    for (int test = 0; test < tests; test++){

        std::fstream d\_ifile("test6.txt");

        struct SkylineStorageMatrix<double> d\_a;

        readSkyline(d\_ifile, d\_a);

        int n = d\_a.di.size();

        std::vector<double> d\_b;

        readVector(d\_ifile, d\_b);

        d\_a.di[0] += (double)pow(10.0, -test);

        d\_b[0] += (double)(pow(10.0, -test));

        std::vector<double> d\_x(n);

        bool ok1 = solve(d\_x, d\_a, d\_b);

        std::fstream g\_ifile("test6.txt");

        struct SkylineStorageMatrix<double> g\_a;

        readSkyline(g\_ifile, g\_a);

        std::vector<double> g\_b;

        readVector(g\_ifile, g\_b);

        g\_a.di[0] += (double)pow(10.0, -test);

        g\_b[0] += (double)(pow(10.0, -test));

        std::vector<std::vector<double>> A;

        skylineToSquare(g\_a, A);

        std::vector<double> g\_x(n);

        bool ok2 = solveGauss(A, g\_x, g\_b);

        // Печать строк таблицы

        for (int i = 0; i < n; i++){

                outFile << std::left << std::setw(5) << (counter%10 == 0? std::to\_string(counter/10) : " ")

            << std::setw(25) << (ok1 ? std::to\_string(d\_x[i]) : "N/A")

            << std::setw(20) << (ok1 ? std::to\_string(fabs((i + 1) - d\_x[i])) : "N/A")

            << std::setw(25) << (ok2 ? std::to\_string(g\_x[i]) : "N/A")

            << std::setw(20) << (ok2 ? std::to\_string(fabs((i + 1) - g\_x[i])) : "N/A")

            << std::endl;

            counter++;

        }

        if (test != tests - 1)

            outFile << std::string(120, '-') << std::endl;

    }

}

// Сгенерировать матрицу Гильберта

template<typename T>

void generateHilbert(SkylineStorageMatrix<T>& dst, std::vector<T>& dstr, int n){

    std::vector<std::vector<T>> square(n); // Для удобства параллельно профильной матрице

    for (int i = 0; i < n; i++) // заполним квадратную

        square[i].resize(n);

    // Матрица

    dst.ia.resize(n + 1);

    dst.ia[0] = 0;

    dst.ia[1] = 0;

    for (int i = 2; i < n + 1; i++)

        dst.ia[i] = dst.ia[i - 1] + (i - 1);

    int ii, jj;

    T sumij = (T)0;

    dst.al.resize(dst.ia[n]);

    ii = 1, jj = 0;

    for (int i = 0; i < dst.ia[n]; i++){

        dst.al[i] = (T)((T)1 / (T)(ii + jj + 1));

        sumij += dst.al[i];

        square[ii][jj] = dst.al[i];

        if (++jj == ii)

            ii++, jj = 0;

    }

    dst.au.resize(dst.ia[n]);

    ii = 1, jj = 0;

    for (int i = 0; i < dst.ia[n]; i++){

        dst.au[i] = (T)((T)1 / (T)(ii + jj + 1));

        sumij += dst.au[i];

        square[jj][ii] = dst.al[i];

        if (++jj == ii)

            ii++, jj = 0;

    }

    dst.di.resize(n);

    for (int i = 0; i < n; i++){

        dst.di[i] = (T)((T)1 / (T)(i + i + 1));;

        square[i][i] = dst.di[i];

    }

    // Вектор

    dstr.resize(n);

    for (int i = 0; i < n; i++){

        dstr[i] = (T)0;

        for (int j = 0; j < n; j++)

            dstr[i] += square[i][j] \* (T)(j + 1);

    }

}

// Провести серию тестов с матрицами Гильберта размерности от n1 до n2

void hilbertTestSeries(int n1, int n2){

    // Открываем файл для записи

    std::ofstream outFile("hilbert\_method.txt");

    if (!outFile.is\_open()) {

        std::cerr << "Не удалось открыть файл для записи." << std::endl;

        return;

    }

    // Заголовок таблицы

    outFile << std::left << std::setw(5) << "k"

            << std::setw(25) << "x^k (float)"

            << std::setw(25) << "|x\* - x^k| (float)"

            << std::setw(25) << "x^k (double)"

            << std::setw(25) << "|x\* - x^k| (double)" << std::endl;

    outFile << std::string(105, '-') << std::endl;

    for (int n = n1; n <= n2; n++){

        struct SkylineStorageMatrix<float> f\_a;

        std::vector<float> f\_b;

        generateHilbert(f\_a, f\_b, n);

        std::vector<float> f\_x(n);

        bool ok1 = solve(f\_x, f\_a, f\_b);

        struct SkylineStorageMatrix<double> d\_a;

        std::vector<double> d\_b;

        generateHilbert(d\_a, d\_b, n);

        std::vector<double> d\_x(n);

        bool ok2 = solve(d\_x, d\_a, d\_b);

        for (int i = 0; i < n; i++){

                       outFile << std::left << std::setw(5) << (i == (n - 1) ? std::to\_string(i + 1) : " ")

                    << std::setw(25) << (ok1 ? std::to\_string(f\_x[i]) : "N/A")

                    << std::setw(25) << (ok1 ? std::to\_string(fabs((i + 1) - f\_x[i])) : "N/A")

                    << std::setw(25) << (ok2 ? std::to\_string(d\_x[i]) : "N/A")

                    << std::setw(25) << (ok2 ? std::to\_string(fabs((i + 1) - d\_x[i])) : "N/A")

                    << std::endl;

        }

        if (n != n2)

            outFile << std::string(105, '-') << std::endl;

    }

    outFile.close();

}

void showMenu() {

    std::cout << "Выберите команду:" << std::endl;

    std::cout << "1. Решить систему уравнений (double)" << std::endl;

    std::cout << "2. Решить систему уравнений (float)" << std::endl;

    std::cout << "3. Решить квадратную систему уравнений (double)" << std::endl;

    std::cout << "4. Тест серии Гильберта" << std::endl;

    std::cout << "5. Тест числа обусловленности" << std::endl;

    std::cout << "6. Сравнение алгоритмов Гаусса и LU-разложения" << std::endl;

    std::cout << "7. Выход" << std::endl;

}

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

{

    int memory\_num\_first, memory\_num\_end;

    int choice = -1;

    std::string inputFile, outputFile;

    inputFile = "test2.txt";

    while (choice != 7) {

        showMenu();

        std::cout << "Введите номер команды: ";

        std::cin >> choice;

        switch (choice) {

            case 1: {

                std::fstream ifile(inputFile);

                struct SkylineStorageMatrix<double> a;

                readSkyline(ifile, a);

                std::vector<double> b;

                readVector(ifile, b);

                outputFile = "solve\_double.txt";

                std::fstream ofile(outputFile, std::ios::out);

                if (!ofile.is\_open())

                    std::cerr << "Ошибка: не удалось открыть файл " << outputFile << " для записи." << std::endl;

                std::vector<double> x(b.size());

                solve(x, a, b);

                writeVector(x, ofile);

                break;

            }

            case 2: {

                std::fstream ifile(inputFile, std::ios::in);

                struct SkylineStorageMatrix<float> a;

                readSkyline(ifile, a);

                std::vector<float> b;

                readVector(ifile, b);

                outputFile = "solve\_float.txt";

                std::fstream ofile(outputFile, std::ios::out);

                if (!ofile.is\_open())

                    std::cerr << "Ошибка: не удалось открыть файл " << outputFile << " для записи." << std::endl;

                std::vector<float> x(b.size());

                solve(x, a, b);

                writeVector(x, ofile);

                break;

            }

            case 3: {

                std::fstream ifile(inputFile, std::ios::in);

                struct SkylineStorageMatrix<double> a;

                readSkyline(ifile, a);

                std::vector<double> b;

                readVector(ifile, b);

                std::cout << "Отладка: Исходный файл прочитан. " << std::endl;

                std::vector<std::vector<double>> A;

                skylineToSquare(a, A);

                std::cout << "Отладка: Перевод в квадрат переведен. " << std::endl;

                outputFile = "solve\_square.txt";

                std::fstream ofile(outputFile, std::ios::out);

                if (!ofile.is\_open())

                    std::cerr << "Ошибка: не удалось открыть файл " << outputFile << " для записи." << std::endl;

                std::vector<double> x(b.size());

                solveGauss(A, x, b);

                std::cout << "Отладка: Гаусс посчитан. " << std::endl;

                writeVector(x, ofile);

                std::cout << "Отладка: Запись результата. " << std::endl;

                break;

            }

            case 4: {

                std::cout << "Введите начальное значение: " << std::endl;

                std::cin >> memory\_num\_first;

                std::cout << "Введите конечное значение: " << std::endl;

                std::cin >> memory\_num\_end;

                hilbertTestSeries(memory\_num\_first, memory\_num\_end);

                break;

            }

            case 5: {

                std::cout << "Введите размерность матрицы: " << std::endl;

                std::cin >> memory\_num\_first;

                conditionNumberTestSeries(memory\_num\_first);

                break;

            }

            case 6: {

                std::cout << "Введите размерность матрицы: " << std::endl;

                std::cin >> memory\_num\_first;

                compareGaussLusq(memory\_num\_first);

                break;

            }

            case 7: {

                std::cout << "Выход из программы." << std::endl;

                break;

            }

            default: {

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

                break;

            }

        }

    }

    return 0;

}