Задание

1. Написать программу, которая обращает симметричную матрицу методом LDL^T -разложения. Кроме матрицы A^{-1} программа должна выводить матрицу L и главную диагональ матрицы D. Применить программу к следующим ниже входным данным и вывести результат
2. Число обусловленности матрицы из третьего задания в матричной максимум-норме равно $1.0996545413425 \cdot 10^{12}$ Попробуйте вычислить это число по определению с помощью вашей программы. Сколько точных цифр вам удалось получить? Почему?
3. Проведите экспериментальное исследование скорости работы вашей программы в зависимости от размерности матрицы, используя для тестов матрицу со случайными числами. Постройте график зависимости времени работы от размерности. Матрицу какой размерности ваша программа на вашем компьютере может обратить за одну минуту?

Матрица 1:

$$\begin{bmatrix} 9 & 9 & -12 & 12 & 15 \\ 9 & 18 & -27 & 0 & 30 \\ -12 & -27 & 25 & -8 & -57 \\ 12 & 0 & -8 & 19 & -9 \\ 15 & 30 & -57 & -9 & 66 \end{bmatrix}$$

Матрица 2:

$$\begin{bmatrix} 25 & 0 & -5 & 5 & 25 & -10 & -10 & -20 & 5 \\ 0 & 9 & -6 & 3 & -6 & 9 & 3 & 9 & 6 \\ -5 & -6 & 21 & -15 & 19 & -20 & -8 & 18 & -9 \\ 5 & 3 & -15 & 12 & -8 & 12 & 1 & -20 & 7 \\ 25 & -6 & 19 & -8 & 34 & -58 & -38 & -11 & -12 \\ -10 & 9 & -20 & 12 & -58 & -15 & 43 & 10 & -5 \\ -10 & 3 & -8 & 1 & -38 & 43 & -7 & 13 & -10 \\ -20 & 9 & 18 & -20 & -11 & 10 & 13 & 100 & 8 \\ 5 & 6 & -9 & 7 & -12 & -5 & -10 & 8 & -3 \end{bmatrix}$$

Матрица 3:

$$\begin{bmatrix}
81 & -3240 & 41580 & -249480 & 810810 & -1513512 & 162 \\
-3240 & 172800 & -249480 & 15966720 & -54054000 & 103783680 & -113 \\
41580 & -249480 & 38419920 & -256132800 & 891891000 & -1748106360 & 1942 \\
-249480 & 15966720 & -256132800 & 1756339200 & -6243237000 & 12430978560 & -1398 \\
810810 & -54054000 & 891891000 & -6243237000 & 22545022500 & -45450765360 & 51648 \\
-1513512 & 103783680 & -1748106360 & 12430978560 & -45450765360 & 92554285824 & -10605 \\
1621620 & -113513400 & 1942340400 & -13984850880 & 51648597000 & -106051785840 & 12236 \\
-926640 & 65894400 & -1141620480 & 8302694400 & -30918888000 & 63930746880 & -7420 \\
218790 & -15752880 & 275675400 & -2021619600 & 7581073500 & -15768632880 & 18396
\end{bmatrix}$$

Ответы

1. Выводы программы для каждой матрицы из условия
   Для матрицы первой:

output.txt

```
Matrix L :
          1           0           0           0           0
          1           1           0           0           0
   -1.33333    -1.66667           1           0           0
    1.33333    -1.33333        0.75           1           0
    1.66667     1.66667        0.75          -0           1

Diagonal of Matrix D :
          9           9         -16          -4          25

Reversed Matrix A :
     -1.24229   -0.0552083    -0.439792    0.604167        0.01
   -0.0552083     0.276042  -0.00104167   -0.0208333   -0.116667
    -0.439792  -0.00104167    -0.180625       0.1875       -0.03
     0.604167   -0.0208333       0.1875        -0.25           0
         0.01    -0.116667        -0.03            0        0.04

Condition number = 81.9912
```

Для второй матрицы:

```
amdron@Andron-laptop:~/bsu/num-methods/gauss_ldlt$ ./build/gauss_ldlt input2.txt output.txt
terminate called after throwing an instance of 'std::runtime_error'
  what():  determinant of matrix = 0
Aborted (core dumped)
```

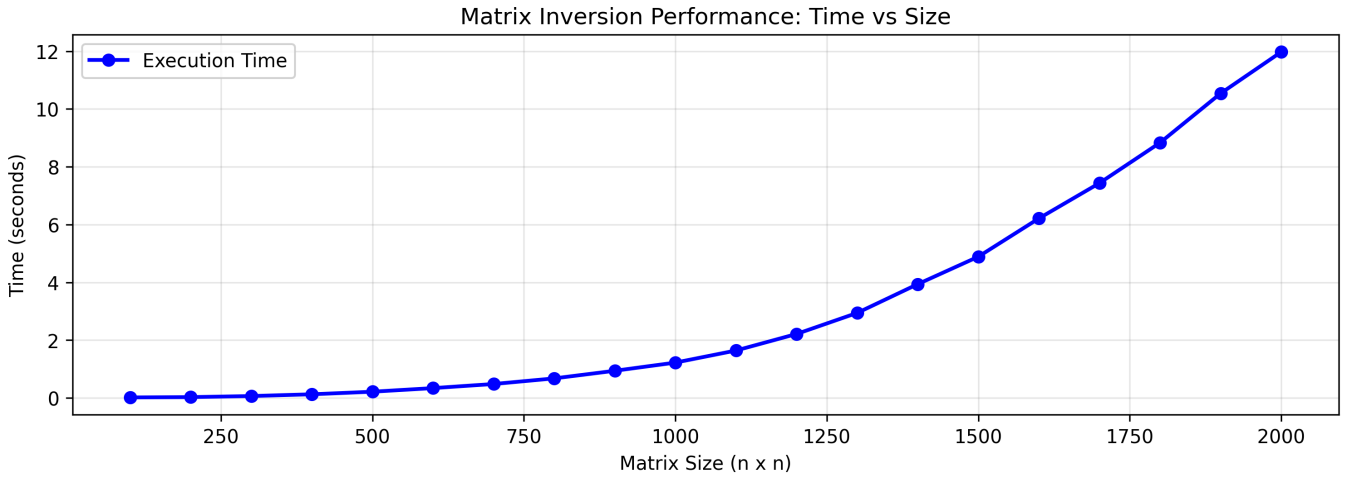
Третья матрица:

output.txt

```
Matrix L :
           1            0            0            0            0            0            0            0            0
         -40            1            0            0            0            0            0            0            0
     513.333       32.725            1            0            0            0            0            0            0
       -3080        138.6      11.1005            1            0            0            0            0            0
       10010       -500.5      -40.538     -3.69768            1            0            0            0            0
    -18685.3         1001       81.755      7.52786     -3.72733            1            0            0            0
       20020     -1126.12     -92.5686     -8.58662      5.87424     -2.89225            1            0            0
      -11440      667.333      55.1371      5.14492     -4.3471      2.96592     -1.90469            1            0
     2701.11     -162.067     -13.4464     -1.26084      1.23974     -1.04791     0.942234     -0.92757            1

Diagonal of Matrix D :
          81        43200  -2.91885e+07   3.75474e+09   2.35499e+08   3.19174e+07   1.85952e+06       34569.7       116.62

Reversed Matrix A :
      0.111099  -1.48467e-06   1.78156e-06   0.00555852     0.0095272     0.0119083      0.013231     0.0138922     0.0141446
  -1.48467e-06  -1.58693e-11   4.45383e-07    5.9385e-07    6.3627e-07   6.36271e-07   6.18597e-07    5.93854e-07   5.66861e-07
   1.78156e-06   4.45383e-07  -9.52156e-12  -1.78166e-07  -2.54518e-07  -2.86331e-07  -2.96935e-07   -2.96934e-07  -2.91535e-07
    0.00555852    5.9385e-07  -1.78166e-07   0.000634445    0.00118988    0.00158666    0.00185121     0.00201957    0.0021206
     0.0095272    6.3627e-07  -2.54518e-07    0.00118988    0.00226685    0.00306045    0.00360672     0.00396749    0.00419506
     0.0119083   6.36271e-07  -2.86331e-07    0.00158666    0.00306045    0.00417358    0.00495949      0.0054937    0.00584337
      0.013231   6.18597e-07  -2.96935e-07    0.00185121    0.00360672    0.00495949     0.0059346     0.00661295    0.00706992
     0.0138922   5.93854e-07  -2.96934e-07    0.00201957    0.00396749     0.0054937    0.00661295     0.00740661    0.00795378
     0.0141446   5.66861e-07  -2.91535e-07     0.0021206    0.00419506    0.00584337    0.00706992     0.00795378    0.00857485

Condition number = 1.35949e+10
```

код:
https://github.com/Amdronm/num-methods/tree/main/gauss_ldlt

2. В третьем задании число обусловленности по матричной максимум норме (максимальная сумма модулей элементов строки) получилась равной $69721112136.237548828 \quad = (6.9721112136 \cdot 10^{10})$, что сильно отличается от данного даже порядком
3. За минуту получилось обратить матрицу размерности 3400



Running program

```
mkdir build
cd build
cmake ..
./build/gauss_ldlt <input.txt> <output.txt>
```

Where input.txt and output.txt are optional filenames for data
By default program takes input.txt is stdin and stdout

`create_matrix.py` - Python script for generating random input and bencmarking program
`collect_stats.py` - Python script that runs some benches and saves plot of performance in `./pictures/benches.png`

`matrix.h`

```
#pragma once

#include <cstddef>
#include <cstdlib>
#include <iomanip>
#include <ostream>
#include <stdexcept>
#include <utility>
#include <vector>

constexpr double kEps = 1e-14;
constexpr size_t kWidth = 15;

class Matrix;

Matrix Identity(size_t dim);
Matrix Transpose(const Matrix& mat);

class Matrix {
public:
    Matrix(size_t rows, size_t cols) {
        data_ = std::vector<std::vector<double>>(rows,
                                                 std::vector<double>(cols,
0));
    }

    Matrix(size_t dim) {
        data_ =
            std::vector<std::vector<double>>(dim, std::vector<double>(dim,
0));
    }

    Matrix(const std::vector<std::vector<double>>& data) : data_(data) {}

    Matrix(const std::vector<double>& diag) {
        Matrix mat = Matrix(diag.size());
        for (size_t i = 0; i < diag.size(); ++i) {
```

```
            mat[i, i] = diag[i];
        }
        *this = mat;
    }

    size_t Rows() const { return data_.size(); }

    size_t Columns() const { return data_.front().size(); }

    bool operator==(const Matrix& other) const = default;

    double& operator[](size_t row, size_t col) { return data_[row][col]; }

    double operator[](size_t row, size_t col) const { return data_[row][col];
}

    std::vector<double>& operator[](size_t row) { return data_[row]; }

    Matrix operator*(const Matrix& other) const {
        Matrix res = Matrix(this->Rows(), other.Columns());
        for (size_t i = 0; i < res.Rows(); ++i) {
            for (size_t j = 0; j < res.Columns(); ++j) {
                double elem = 0.;
                for (size_t k = 0; k < this->Columns(); ++k) {
                    elem += (*this)[i, k] * other[k, j];
                }
                res[i, j] = elem;
            }
        }
        return res;
    }

    Matrix& operator*=(const Matrix& other) {
        Matrix mul = (*this) * other;
        *this = mul;
        return *this;
    }

    std::vector<double> GetDiagonal() const {
        std::vector<double> res(data_.size());
        for (size_t i = 0; i < data_.size(); ++i) {
            res[i] = data_[i][i];
        }
        return res;
    }
```

```
    friend std::ostream& operator<<(std::ostream& out, const Matrix& mat) {
        for (const auto& row : mat.data_) {
            for (const auto& elem : row) {
                out << std::setw(kWidth) << elem << " ";
            }
            out << "\n";
        }
        return out;
    }

    void MultiplyAddRow(size_t bear_row, double val, size_t row) {
        for (size_t j = 0; j < this->Columns(); ++j) {
            data_[row][j] += data_[bear_row][j] * val;
        }
    }

    double FindMaxElemInColumn(size_t row, size_t col) {
        size_t idx = row;
        for (size_t i = row; i < this->Rows(); ++i) {
            if (data_[i][col] > data_[idx][col]) {
                idx = i;
            }
        }
        std::swap(data_[idx], data_[row]);
        if (std::abs(data_[row][col]) < kEps) {
            throw std::runtime_error("determinant of matrix = 0");
        }
        return data_[row][col];
    }

    Matrix FindLUDecomp() {
        Matrix res = Identity(this->Rows());
        for (size_t j = 0; j < this->Columns(); ++j) {
            double op_elem = data_[j][j];
            if (std::abs(op_elem) < kEps) {
                op_elem = this->FindMaxElemInColumn(j, j);
            }
            for (size_t i = j + 1; i < this->Rows(); ++i) {
                double rev = -data_[i][j] / op_elem;
                this->MultiplyAddRow(j, rev, i);
                res[i, j] = -rev;
            }
        }
        return res;
    }
```

```
private:
    std::vector<std::vector<double>> data_;
};

Matrix Transpose(const Matrix& mat) {
    Matrix res = Matrix(mat.Columns(), mat.Rows());
    for (size_t i = 0; i < mat.Rows(); ++i) {
        for (size_t j = 0; j < mat.Columns(); ++j) {
            res[j, i] = mat[i, j];
        }
    }
    return res;
}

Matrix Identity(size_t dim) {
    Matrix res = Matrix(dim);
    for (size_t i = 0; i < res.Rows(); ++i) {
        res[i, i] = 1;
    }
    return res;
}
```

`main.cpp`

```
#include <cmath>
#include <cstddef>
#include <cstring>
#include <fstream>
#include <iomanip>
#include <ios>
#include <iostream>
#include <istream>
#include <ostream>
#include <vector>

#include "matrix.h"

Matrix ReadMatrix(std::istream& fin) {
    if (fin.bad()) {
        std::cerr << "Wrong file" << std::endl;
        return 0;
    }
    size_t n = 0;
    fin >> n;

    Matrix mat_a = Matrix(n);
```

```
    for (size_t i = 0; i < n; ++i) {
        for (size_t j = 0; j < n; ++j) {
            fin >> mat_a[i, j];
        }
    }
    return mat_a;
}

std::vector<double> SolveBotTriangle(const Matrix& mat,
                                     const std::vector<double>& vec) {
    std::vector<double> sol;
    sol.reserve(vec.size());
    sol.push_back(vec.front());
    for (size_t i = 1; i < vec.size(); ++i) {
        double x = vec[i];
        for (size_t j = 0; j < i; ++j) {
            x -= mat[i, j] * sol[j];
        }
        sol.push_back(x);
    }
    return sol;
}

std::vector<double> SolveTopTriangle(const Matrix& mat,
                                     const std::vector<double>& vec) {
    std::vector<double> sol(vec.size());
    sol.back() = vec.back();
    for (int i = vec.size() - 2; i >= 0; --i) {
        double x = vec[i];
        for (size_t j = i; j < vec.size(); ++j) {
            x -= mat[i, j] * sol[j];
        }
        sol[i] = x;
    }
    return sol;
}

void MulVector(std::vector<double>* vec1, const std::vector<double>& vec_by) {
    for (size_t i = 0; i < vec1->size(); ++i) {
        (*vec1)[i] *= vec_by[i];
    }
}

template <class T>
std::ostream& operator<<(std::ostream& out, const std::vector<T>& other) {
```

```
    for (const auto& elem : other) {
        out << std::setw(kWidth) << elem << " ";
    }
    out << "\n";
    return out;
}

double FindMaxNorm(const Matrix& mat) {
    double res = 0.;
    for (size_t i = 0; i < mat.Rows(); ++i) {
        double norm = 0.;
        for (size_t j = 0; j < mat.Columns(); ++j) {
            norm += std::fabs(mat[i, j]);
        }
        res = std::max(norm, res);
    }
    return res;
}

void SolveFast(std::istream& in, std::ostream& out);

int main(int argc, char* argv[]) {
    std::ios_base::sync_with_stdio(false);

    std::ifstream fin;
    std::ofstream fout;
    std::istream* in_ptr;
    std::ostream* out_ptr;

    if (argc != 3) {
        fin.open("input.txt");
        in_ptr = &fin;
        out_ptr = &std::cout;
    } else {
        fin.open(argv[1]);
        in_ptr = &fin;

        fout.open(argv[2]);
        out_ptr = &fout;

        if (!fin.is_open() || !fout.is_open()) {
            std::cerr << "wrong filename(s)" << std::endl;
            return 1;
        }
    }
    auto& in = *in_ptr;
```

```
    auto& out = *out_ptr;

    SolveFast(in, out);

    return 0;
}

void SolveFast(std::istream& in, std::ostream& out) {
    Matrix mat_a = ReadMatrix(in);
    double norm_a = FindMaxNorm(mat_a);

    Matrix mat_l = mat_a.FindLUDecomp();
    auto diag_d = mat_a.GetDiagonal();

    out << "Matrix L :\n" << mat_l << "\nDiagonal of Matrix D : \n" << diag_d;
    auto diag_d_rev = diag_d;
    for (auto& elem : diag_d_rev) {
        elem = 1. / elem;
    }

    size_t dim = diag_d.size();
    Matrix rev_a(dim);
    auto mat_lt = Transpose(mat_l);
    for (size_t i = 0; i < dim; ++i) {
        std::vector<double> vec_e(dim, 0.);
        vec_e[i] = 1.;
        auto res = SolveBotTriangle(mat_l, vec_e);
        MulVector(&res, diag_d_rev);
        res = SolveTopTriangle(mat_lt, res);
        rev_a[i] = res;
    }

    out << "\nReversed Matrix A : \n" << rev_a;

    out << "\nCondition number = " << std::setprecision(20)
        << norm_a * FindMaxNorm(rev_a) << std::endl;
}
```