Практическая работа №3

по дисциплине «Вычислительная математика»

**«Вычисление обратной матрицы методом Гаусса»**

студента группы M3207 Бойцова Виталия

## 

//The greatest enemy of knowledge is not ignorance...

//...it is the illusion of knowledge.

// (C) Stephen Hawking

#include <vector>

#include <iostream>

using namespace *std*;

class Matrix {

public:

Matrix(int n, int m) {

data = *vector*<*vector*<double>>(n, *vector*<double>(m));

}

*size\_t* height() {

return data.*size*();

}

*size\_t* width() {

return height() ? data[0].*size*() : 0;

}

*vector*<double> & operator [] (*size\_t* i) {

return data[i];

}

//Возвращает единичную матрицу

static Matrix identity(*size\_t* n) {

Matrix ans(n, n);

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

ans[i][i] = 1;

return ans;

}

//соединяет две матрицы

Matrix operator & (Matrix b) {

Matrix a = Matrix(data);

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

a[i].*insert*(a[i].*end*(), b[i].*begin*(), b[i].*end*());

return a;

}

//добавляет к строке i2 строку i1 умноженную на multy

void add\_string\_to\_string(int i1, double multy, int i2) {

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

data[i2][j] += data[i1][j] \* multy;

}

//умножает строку на число

void multy\_string(int i, double multy) {

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

data[i][j] \*= multy;

}

//приводит левую часть матрицы к единичной с помощью элементарных преобразований

void gauss() {

for (*size\_t* j = 0; j < height(); j++) {

multy\_string(j, 1 / data[j][j]);

for (*size\_t* i = j + 1; i < height(); i++)

add\_string\_to\_string(j, -data[i][j] / data[j][j], i);

}

for (int j = height() - 1; j > 0; j--)

for (int i = j - 1; i >= 0; i--)

add\_string\_to\_string(j, -data[i][j] / data[j][j], i);

}

//возвращает правую часть матрицы

Matrix get\_right\_part() {

Matrix ans(height(), width() - height());

for (int i = 0; i < height(); i++)

for (int j = 0; j < width() - height(); j++)

ans[i][j] = data[i][height() + j];

return ans;

}

void print() {

for (*size\_t* i = 0; i < height(); i++, *cout* << '\n')

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

*printf*("%lf ", data[i][j]);

}

private:

*vector*<*vector*<double>> data;

};

int main() {

int n;

*cin* >> n;

Matrix a(n, n);

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

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

*cin* >> a[i][j];

Matrix m = a & Matrix::identity(n);

m.gauss();

m.get\_right\_part().print();

}