МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ

Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования «Белгородский государственный технологический университет им. В.Г. Шухова»

**ИНСТИТУТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ И УПРАВЛЯЮЩИХ СИСТЕМ Кафедра программного обеспечения вычислительной техники и автоматизированных систем**

Лабораторная работа №1

по дисциплине: Вычислительная математика:

Метод Гаусса

Работу выполнил:

Студент группы ПВ-21:

Донцов Александр Алексеевич

Проверил:

Бондаренко Татьяна Владимировна

Белгород – 2018

**Лабораторная работа № 1 Метод Гаусса**

**Цель работы:** изучить прямой и обратный ход метода Гаусса для решения систем линейных алгебраических уравнений; изучить схему единственного деления с выбором максимального по модулю элемента; изучить применение метода Гаусса для вычисления определителя матрицы и обратной матрицы; получить практические навыки программной реализации метода Гаусса и решения поставленных задач методом Гаусса с помощью ЭВМ.

***Задания к работе***

1. Выполнить вручную действия над матрицами А и В из пункта 3 задания соответствующего варианта.

2. Выполнить следующие действия, не используя метод Гаусса:

- решить вручную систему линейных алгебраических уравнений с тремя неизвестными из пункта 1 задания соответствующего варианта;

- найти вручную определитель матрицы А из пункта 2 задания соответствующего варианта;

- найти вручную матрицу А-1 обратную матрице А из пункта 2 задания соответствующего варианта.

3. Решить вручную методом Гаусса систему линейных алгебраических уравнений с тремя неизвестными из пункта 1 задания соответствующего варианта.

4. Найти вручную с помощью метода Гаусса определитель матрицы А из пункта 2 задания соответствующего варианта.

5. Найти вручную с помощью метода Гаусса матрицу А-1 обратную матрице А из пункта 2 задания соответствующего варианта.

Выполнить проверку полученной матрицы на соответствие условию:

А·А-1 = Е, где Е ― единичная матрица.

6. Создать модуль для работы с матрицами произвольного порядка, содержащий подпрограммы для умножения двух матриц, умножения числа на матрицу, сложения матриц, вычитания матриц, транспонирования матрицы, умножения матрицы на вектор, ввода и вывода матрицы.

7. Создать модуль, содержащий подпрограммы, реализующие прямой и обратный ход метода Гаусса для схемы единственного деления с выбором максимального по модулю элемента.

8. Создать программу для решения следующих задач:

- нахождение методом Гаусса решения системы линейных алгебраических уравнений с одной и той же матрицей коэффициентов при неизвестных и произвольным числом столбцах свободных членов.

*Замечание.* В прямом ходе метода Гаусса выполняется приведение расширенной матрицы (коэффициенты при неизвестных и свободные члены) к треугольному виду, и одновременно изменяются все столбцы свободных членов. На этапе обратного хода выполняется вычисление решения системы для каждого столбца свободных членов, составляется матрица решений.

- вычисление определителя заданной матрицы методом Гаусса;

- нахождение для заданной матрицы обратной матрицы методом Гаусса.

9. Решить все задания соответствующего варианта с помощью составленной программы. Выполнить проверку правильности найденного решения системы линейных уравнений и матрицы обратной к заданной матрице с помощью составленной программы, сравнить значения, полученные при решении заданий с помощью метода Гаусса и с использованием произвольного метода решения.

**Задание 6**

//модуль, реализующий матрицу Matr.java

#include "Matr.h"

#include "malloc.h"

#include "stdio.h"

//создание матрицы

double \*\*creatMatr(int str, int tab) {

double \*\*matr = (double \*\*)malloc(str \* sizeof(double \*));

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

matr[i] = (double \*)malloc(tab \* sizeof(double));

return matr;

}

//удаление матрицы

void delMatr(double \*\*m, int str, int tab) {

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

free(m[i]);

free(m);

}

//ввод матрицы

void input(double \*\*m, int str, int tab) {

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

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

scanf("%d", &m[i][j]);

}

//вывод матрицы

void output(double \*\*m, int str, int tab) {

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

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

printf("%d ", m[i][j]);

printf("\n");

}

}

//функция конкатинации двух квадратных матриц

void swapElemsMatr(double \*\*m, int i1, int j1, int i2, int j2) {

double t = m[i1][j1];

m[i1][j1] = m[i2][j2];

m[i2][j2] = t;

}

//обмен двумя элементами в матрице

void swapStrMatr(double \*\*m, int tab, int str1, int str2) {

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

swapElemsMatr(m, str1, i, str2, i);

}

}

//нахождение максимального элемента в столбце мартицы

int findMaxElemTabMatr(double \*\*m, int strMatr, int str, int tab) {

double max = m[str][tab], current;

int pos\_max = str;

for (int i = str + 1; i < strMatr; i++) {

current = m[i][tab];

if (max < current) {

max = current;

pos\_max = i;

}

}

return pos\_max;

}

//функция конкатинации двух квадратных матриц

void konkatMatrs(double \*\*a, double \*\*b, double \*\*res, int str, int tab) {

int left = tab;

int right = tab + tab;

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

//элементы из первой матрицы

for (int j1 = 0; j1 < left; j1++)

res[i][j1] = a[i][j1];

//элементы из второй матрицы

for (int j2 = left; j2 < right; j2++)

res[i][j2] = a[i][j2 - left];

}

}

//модуль, реализующий алгебраические операции над матрицами MathOfMatr.java

#include "MathMatr.h"

//копирование мартицы 1 в матрицу 2

void copy(double \*\*a, double \*\*b, int str, int tab) {

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

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

b[i][j] = a[i][j];

}

//умножение

void multiplication(double \*\*a, double \*\*b, double \*\*res, int str, int tab) {

int sum;

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

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

sum = 0;

for (int k = 0; k < str; k++)

sum += a[i][k] \* b[k][j];

res[i][j] = sum;

}

}

}

//умножение на число

void multiplicationOnNumb(double \*\*a, double \*\*res, int value, int str, int tab) {

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

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

res[i][j] = a[i][j] \* value;

}

//сложение

void addition(double \*\*a, double \*\*b, double \*\*res, int str, int tab) {

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

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

res[i][j] = a[i][j] + b[i][j];

}

//вычитание

void subtraction(double \*\*a, double \*\*b, double \*\*res, int str, int tab) {

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

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

res[i][j] = a[i][j] - b[i][j];

}

//транспонирование

void transposition(double \*\*a, double \*\*res, int str, int tab) {

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

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

res[j][i] = a[i][j];

}

//умножение на вектор

void multiplicationOnVector(double \*\*a, double \*vect, double \*res, int str, int tab) {

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

res[i] = 0;

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

res[i] += a[i][j] \* vect[j];

}

}

**Задание 7**

//модуль, реализующий метод Гауса MethodGaus.java

#include "Gaus.h"

#include "Matr.h"

#include "MathMatr.h"

//решение системы линейных уравнений методом гауса

void SLAE(double \*\*a, int str, int tab, double \*vect) {

double \*\*matr = creatMatr(str, tab);

copy(a, matr, str, tab);

double m, t;

int k, j, i;

//прямой ход

for (k = 1; k < str; k++) {

for (j = k; j < str; j++) {

swapStrMatr(a, tab, findMaxElemTabMatr(a, str, k - 1, k - 1), k - 1);

m = a[j][k - 1] / a[k - 1][ k - 1];

for (i = 0; i < tab; i++)

a[j][i] = a[j][i] - (m \* a[k - 1][i]);

}

}

//обратный ход

for (i = str - 1; i >= 0; i--) {

vect[i] = a[i][str] / a[i][i];

for (j = str - 1; j > i; j--)

vect[i] = vect[i] - a[i][j] \* vect[j] / a[i][i];

}

}

// приведение матрицы к тругольному виду методом гауса

void toTriangularMatrices(double \*\*matr, double \*\*a, int str, int tab) {

copy(matr, a, str, tab);

double m, t;

int k, j, i;

for (k = 1; k < str; k++) {

for (j = k; j < str; j++) {

// найти максимальный элемент в столбце

swapStrMatr(a, tab, findMaxElemTabMatr(a, str, k - 1, k - 1), k - 1);

m = a[j][k - 1] / a[k - 1][k - 1];

for (i = 0; i < tab; i++)

a[j][i] = a[j][i] - (m \* a[k - 1][i]);

}

}

}

//нахождение определителя матрицы любого порядка

double determinant(double \*\*a, int str, int tab) {

double res = 1;

double \*\*matr = creatMatr(str, tab);

toTriangularMatrices(a,matr, str, tab);

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

res \*= matr[i][i];

return res;

}

//создание единичной матрицы нужного порядка

void getUnitMatrix(double \*\*unit, int str, int tab) {

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

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

if (i == j)

unit[i][j] = 1;

else

unit[i][j] = 0;

}

//находим обратную матрицу

void inverseMatrix(double \*\*a, double \*\*res, int str, int tab) {

// создаём матрицу, необходимых размеров для склеивания еденичной и a

double \*\*matrPlusOne = creatMatr(str, tab + tab);

//создаём еденичную матрицу

double \*\*unit = creatMatr(str, str);

konkatMatrs(a, unit, matrPlusOne, str, tab);

for (int k = str - 1; k >= 0; k--) {

for (int i = str - 1; i >= 0; i--) {

res[k][i] = a[i][str] / a[i][i];

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

res[k][i] = res[k][i] - a[i][j] \* res[k][j] / a[i][j];

}

}

}

**Задание 8**



int main() {

double sol[3];

double \*\*m = creatMatr(3, 4);

input(m,3,4);

SLAE(m, 3, 4, sol);

for (int i = 0; i < 3; i++)

printf("%d ", sol[i]);

return 0;

}

Результат программы:



**Задание 9**



Программа для нахождения определителя:

void main() {

double matr = creatMatr(3, 3);

input(matr);

double rez = determinant(matr);

println(“\n%d”, rez);

}

Результат программы:





Программа для решения:

void main() {

double a = creatMatr(3, 3);

double b = creatMatr(3, 3);

input(a);

input(b);

double rez1 = creatMatr(3,3);

double rez2 = creatMatr(3,3);

addition(a,b,rez1, 3, 3)

multiplicationOnNumb(rez2, rez2, 2 , 3, 3);

subtraction(a, rez2, rez2, 3, 3)

multiplication(rez1, rez2, a, 3, 3);

output(a, 3, 3);

}

Результат программы:

