

Министерство науки и высшего образования Российской Федерации  
Федеральное государственное бюджетное образовательное учреждение  
высшего образования  
«Рязанский государственный радиотехнический университет  
имени В. Ф. Уткина»

Кафедра «Вычислительная и прикладная математика»

Отчет  
по лабораторной работе №1  
по дисциплине  
«Объектно-ориентированное программирование»  
на тему  
«Структуры и модули»

Выполнил: студент гр. 143  
Соколов Е.А.  
Проверил: профессор  
Каширин И.Ю.

## Задание (вариант №19)

В данной лабораторной работе необходимо создать модуль, содержащий описание структуры данных, представляющее собой некоторую сущность: вектор, дробь, фигура и т.д. (по вариантам ниже), а также 8 операций (функций) над этой структурой. Основная программа должна запрашивать у пользователя какую операцию необходимо протестировать, далее запрашивать данные для операндов и показывать результат операции с помощью функции модуля «вывод в консоль». В качестве упрощения, задание не подразумевает использование динамической памяти, следовательно, во всех вариантах используются статические массивы (если это обусловлено задачей).

**Важно:** во всех функциях где написано «*результат возвращается как новое значение*» следует создавать и возвращать новую структуру, во всех других случаях следует модифицировать существующую, которая была передана в качестве параметра.

### Вариант 19. «Квадратная матрица 4x4» – Matrix4x4.

Разработать структуру данных Matrix4x4, представляющую квадратную матрицу 4 на 4 в виде массива из 16 её элементов, а также логического значения zero, показывающего является ли данная матрица нулевой. В модуле для работы с матрицами должны быть следующие функции:

- 1) Создание структуры матрицы из массива значений её элементов. Как результат возвращается новая матрица.
- 2) Сложение двух матриц, как результат возвращается новая матрица.
- 3) Умножение двух матриц, как результат возвращается новая матрица.
- 4) Умножение переданной матрицы на скаляр.
- 5) Нахождение тройки значений соответственно минимального, максимального и среднего значений элементов переданной матрицы.
- 6) Нахождение определителя переданной матрицы.
- 7) Преобразование переданной матрицы в обратную.
- 8) Вывод в консоль переданной матрицы в виде:

```
-1.4      3      2.5      3.2
 3.2      1      -0.4     -3
 3.2     4.3      4      1.2
 -4     3.1     -3.4      2
```

[zero]

где zero имеет значение либо «нулевая», либо «не нулевая»

**Цель работы:** научиться работать со структурами и разбивать программу на модули.

**Анализ задания:**

*Входные данные:* Матрицы A, B размером 4 на 4, элементы которых являются действительными числами, скаляр scalar ( $\in R$ ), целочисленная переменная number.

*Промежуточные данные:* целочисленные переменные count, number, i, j, k, p det\_sign; действительные числа s, coefficient, строка str, массив 16-ти действительных чисел array, матрица matrix размером 4 на 4, элементы которой являются действительными числами.

*Выходные данные:* Матрицы A, C размером 4 на 4, элементы которых являются действительными числами, действительные числа det, min, max, aver.

*Математические методы и/или стандартные алгоритмы:* для расчета в формулах используются операции сложения, умножения, вычитания, деления, взятия модуля действительного числа.

*Этапы решения задачи:*

- 1) Ввести номер требуемой операции
- 2) Ввести данные для операции
- 3) Обработать введенные данные при помощи подпрограмм из модуля
- 4) Вывести результаты обработки.

## Описание структуры данных:

В программе используется структура `Matrix4x4`, представляющая собой квадратную матрицу размером 4 на 4, состоящая из следующих полей:

- Двумерный массив действительных чисел размерностью 4 на 4
- Логическое значение `zero`, показывающее, является ли матрица нулевой (`true`) или же нет (`false`).

## Подпрограммы для работы со структурой данных:

```
//Функция по созданию структуры матрицы
//Входной параметр: const float* array – указатель на
//массив из 16-ти действительных чисел
//Функция возвращает структуру матрицы
Matrix4x4 create_matrix(const float* array);

//Функция проверки на нулевую матрицу
//Входной параметр: const Matrix4x4* matrix – указатель на структуру данных
//Возвращает 0, если матрица нулевая, 1 в противном случае.
int check_matrix_zero(const Matrix4x4* matrix);

//Вывод матрицы в консоль
//Входной параметр: const Matrix4x4* matrix – указатель на структуру данных
void output_matrix(const Matrix4x4* matrix);

//Сложение двух матриц
//Входные параметры: Matrix4x4* matrix, Matrix4x4* B – указатели на структуры
//Функция возвращает структуру матрицы
Matrix4x4 summ_matrix(Matrix4x4* A, Matrix4x4* B);

//Умножение двух матриц
//Входные параметры: Matrix4x4* matrix, Matrix4x4* B – указатели на структуры
//Функция возвращает структуру матрицы
Matrix4x4 mult_matrix(Matrix4x4* A, Matrix4x4* B);

//Умножение матрицы на скалярное число
//Входные параметры: Matrix4x4* matrix, Matrix4x4* B – указатели на структуру
//Подпрограмма модифицирует структуру данных по указателю Matrix4x4* matrix
void scalar_mult_matrix(Matrix4x4* matrix, float scalar);

//Поиск определителя матрицы
//Входной параметр: Matrix4x4 matrix – структура данных
//Возвращает определитель переданной матрицы
float determ_matrix(Matrix4x4 matrix);

//Поиск по элементам матрицы
//Входной параметр: Matrix4x4* matrix – указатель на структуру данных, float* min,
//float* max, float* aver – указатели на действительные числа
//Модифицирует данные по указателям float* min, float* max, float* aver
void find_values(const Matrix4x4* matrix, float* min, float* max, float* aver);

//Вспомогательная функция транспонирования матрицы
//Входной параметр: const Matrix4x4* matrix – указатель на структуру данных
//Функция возвращает структуру матрицы
static Matrix4x4 transpose_matrix(const Matrix4x4* matrix)

//Нахождение обратной матрицы
//Входной параметр: Matrix4x4* matrix – указатель на структуру данных
//Подпрограмма модифицирует структуру данных по указателю Matrix4x4* matrix
//Функция возвращает 0, если обратная матрица существует, 1 в противном случае
int inverse_matrix(Matrix4x4* matrix);
```

## Листинг программы:

### Файл matrix4x4.h

```
#ifndef MYLIB_RSREU
#define MYLIB_RSREU

#include <stdlib.h>
#include <stdio.h>
#include <stdbool.h>
#include <math.h>
#include <string.h>

#define ROWS 4
#define COLUMNS 4
#define LENGTH ROWS*COLUMNS
#define EPS 1E-3

typedef struct
{
    float array[4][4]; //array of float;
    bool zero;
} Matrix4x4;

//Создание структуры матрицы
Matrix4x4 create_matrix(const float* array);

//Проверка на нулевую матрицу
int check_matrix_zero(const Matrix4x4* matrix);

//Вывод матрицы в консоль
void output_matrix(const Matrix4x4* matrix);

//Сложение двух матриц
Matrix4x4 summ_matrix(Matrix4x4* A, Matrix4x4* B);

//Умножение двух матриц
Matrix4x4 mult_matrix(Matrix4x4* A, Matrix4x4* B);

//Умножение матрицы на скалярное число
void scalar_mult_matrix(Matrix4x4* matrix, float scalar);

//Поиск определителя матрицы
float determ_matrix(Matrix4x4 matrix);

//Поиск по элементам матрицы
void find_values(const Matrix4x4* matrix, float* min, float* max, float*
aver);

//Нахождение обратной матрицы
int inverse_matrix(Matrix4x4* matrix);

//Вспомогательная функция ввода матрицы
int input_matrix(Matrix4x4* matrix);
#endif
```

## Файл matrix4x4.c

```
#include "matrix4x4.h"

//Проверка на нулевую матрицу
int check_matrix_zero(const Matrix4x4* matrix)
{
    int i;
    for (i = 0; i <= LENGTH; i++)
    {
        if (fabs(matrix->array[0][i]) >= EPS)
            return 1; //не нулевая матрица
    }
    return 0; //нулевая матрица
}

//Создание структуры матрицы
Matrix4x4 create_matrix(const float* array)
{
    Matrix4x4 matrix;
    int i;
    for (i = 0; i <= LENGTH; i++)
    {
        matrix.array[0][i] = *(array+i);
    }
    matrix.zero = (bool)check_matrix_zero(&matrix);
    return matrix;
}

//Вывод матрицы в консоль
void output_matrix(const Matrix4x4* matrix)
{
    int i, j;
    char str[12];
    for (i = 0; i < ROWS; i++)
    {
        for (j = 0; j < COLUMNS; j++)
            printf("%7.2f", matrix->array[i][j]);
        printf("\n");
    }
    if (matrix->zero == 0)
    {
        strcpy(str, "нулевая");
    }
    else
    {
        strcpy(str, "не нулевая");
    }
    printf("[%s]", str);
}

//Сложение двух матриц
Matrix4x4 summ_matrix(Matrix4x4* A, Matrix4x4* B)
{
    Matrix4x4 matrix;
    int i;
    for (i = 0; i <= LENGTH; i++)
    {
```

```

        matrix.array[0][i] = A->array[0][i]+B->array[0][i];
    }
    matrix.zero = (bool)check_matrix_zero(&matrix);
    return matrix;
}

//Умножение двух матриц
Matrix4x4 mult_matrix(Matrix4x4* A, Matrix4x4* B)
{
    Matrix4x4 matrix;
    int i, j, k;
    float s;
    for (i = 0; i < ROWS; i++)
    {
        for (j = 0; j < COLUMNS; j++)
        {
            s = 0;
            for (k = 0; k < ROWS; k++)
            {
                s = s + A->array[i][k] * B->array[k][j];
            }
            matrix.array[i][j] = s;
        }
    }
    matrix.zero = (bool)check_matrix_zero(&matrix);
    return matrix;
}

```

```

//Умножение матрицы на скалярное число
void scalar_mult_matrix(Matrix4x4* matrix, float scalar)
{
    int i, j;
    for (i = 0; i < ROWS; i++)
    {
        for (j = 0; j < COLUMNS; j++)
        {
            matrix->array[i][j] = matrix->array[i][j] * scalar;
        }
    }
    matrix->zero = (bool)check_matrix_zero(matrix);
}

```

```

//Поиск определителя методом Гаусса
float determ_matrix(Matrix4x4 matrix)
{
    int i, j, k, t, max_string;
    float element, j_max_element, coefficient;
    int det_sign = 1;
    for (i = 0; i < ROWS; i++)
    {
        j_max_element = fabs(matrix.array[i][i]);
        max_string = i;
        for (k = i; k < ROWS; k++)
        {
            if (fabs(matrix.array[k][i]) > j_max_element)
            {
                max_string = k;
                j_max_element = fabs(matrix.array[k][i]);
            }
        }
    }
}

```

```

        }
    }
    if (i == max_string)
        det_sign *= -1;
    for (t = 0; t < ROWS; t++)
    {
        element = matrix.array[max_string][t];
        matrix.array[max_string][t] = matrix.array[i][t];
        matrix.array[i][t] = element;
    }
    det_sign *= -1;
    for (j = i + 1; j < ROWS; j++)
    {
        if (fabs(matrix.array[i][i]) <= EPS)
        {
            coefficient = 0;
        }
        else
        {
            coefficient = -1 * (matrix.array[j][i]) /
                (matrix.array[i][i]);
        }
        for (t = 0; t < ROWS; t++)
        {
            (matrix.array[j][t]) += (matrix.array[i][t] )
                * coefficient;
        }
    }
}

float determ = det_sign;
for (i = 0; i < ROWS; i++)
{
    determ *= matrix.array[i][i];
}
return determ;
}

```

//Поиск по элементам матрицы

```

void find_values(const Matrix4x4* matrix, float* min, float* max, float* aver)
{
    int i, j;
    *aver = 0;
    *min = *max = matrix->array[0][0];
    for (i = 0; i < ROWS; i++)
    {
        for (j = 0; j < COLUMNS; j++)
        {
            if (matrix->array[i][j] > *max)
            {
                *max = matrix->array[i][j];
            }
            if (matrix->array[i][j] < *min)
            {
                *min = matrix->array[i][j];
            }
            *aver += matrix->array[i][j];
        }
    }
}

```



```

    }
    *aver = *aver / (LENGTH);
}

```

//Вспомогательная функция транспонирования матрицы

```

static Matrix4x4 transpose_matrix(const Matrix4x4* matrix)
{
    int i, j;
    Matrix4x4 result_matrix;
    for (i = 0; i < ROWS; i++)
    {
        for (j = 0; j < COLUMNS; j++)
        {
            result_matrix.array[i][j] = matrix->array[j][i];
        }
    }
    return result_matrix;
}

```

//Нахождение обратной матрицы

```

int inverse_matrix(Matrix4x4* matrix)
{
    int i1, i2, j1, j2, p, det_sign;
    Matrix4x4 result_matrix;
    float temp_matrix[3][3];
    det_sign = 1;
    for (i1 = 0; i1 < ROWS; i1++)
    {
        for (j1 = 0; j1 < COLUMNS; j1++)
        {
            p = 0;
            for (i2 = 0; i2 < ROWS; i2++)
            {
                for (j2 = 0; j2 < COLUMNS; j2++)
                {
                    if (i2 != i1 && j2 != j1)
                    {
                        temp_matrix[0][p] = matrix->array[i2][j2];
                        p += 1;
                    }
                }
            }
            result_matrix.array[i1][j1] = ((temp_matrix[0][0] *
            temp_matrix[1][1] * temp_matrix[2][2] +
            temp_matrix[0][1] *
            temp_matrix[1][2] * temp_matrix[2][0] +
            temp_matrix[0][2] *
            temp_matrix[1][0] * temp_matrix[2][1]) -
            (temp_matrix[0][2] *
            temp_matrix[1][1] * temp_matrix[2][0] +
            temp_matrix[0][0] *
            temp_matrix[1][2] * temp_matrix[2][1] +
            temp_matrix[0][1] *
            temp_matrix[1][0] * temp_matrix[2][2])) * det_sign;
            det_sign *= -1;
        }
    }
}

```

```

        }
        det_sign *= -1;
    }

    result_matrix = transpose_matrix(&result_matrix);

    float det = determ_matrix(*matrix);
    if (fabs(det) <= EPS)
    {
        return 1; //Обратной матрицы нет
    }
    scalar_mult_matrix(&result_matrix, 1 / det);

    *matrix = result_matrix;
    matrix->zero = check_matrix_zero(matrix);
    return 0;
}

//Вспомогательная функция ввода матрицы
int input_matrix(Matrix4x4* matrix)
{
    float array[4][4];
    float *p = (float*)&array;
    int i, count;
    for (i = 0; i < LENGTH; i++)
    {
        count = scanf("%f", (float*)(p + i));
        if (count == 0)
        {
            return 1;
        }
    }
    *matrix = (create_matrix((float*)array));
    return 0;
}

```

## Файл main.c

```
#include "matrix4x4.h"

//Вспомогательная функция, не включённая в модуль matrix4x4
void check_errors(int count, int error_code)
{
    if (count != 0)
    {
        puts("Ошибка при вводе элемента матрицы.");
        exit(error_code);
    }
}

int main()
{
    int number;
    system("chcp 1251");
    Matrix4x4 A;
    Matrix4x4 B;
    Matrix4x4 C;
    puts("1) Создание структуры матрицы");
    puts("2) Сложение двух матриц");
    puts("3) Умножение двух матриц");
    puts("4) Умножение матрицы на скаляр");
    puts("5) Нахождение мин., макс., среднего значения элементов матрицы");
    puts("6) Нахождение определителя матрицы");
    puts("7) Преобразование матрицы в обратную");
    puts("8) Вывод матрицы в консоль");
    puts("Введите номер операции:");
    int count = scanf("%d", &number);
    if (!count || number < 1 || number > 8)
        printf("Неверно выбран номер операции\n");
    switch (number) {
        case 2: //Сложение двух матриц
        {
            puts("Введите элементы матрицы A");
            count = input_matrix(&A);
            check_errors(count, 2);
            puts("Введите элементы матрицы B");
            count = input_matrix(&B);
            check_errors(count, 3);
            puts("Вывод матрицы C = A + B");
            C = summ_matrix(&A, &B);
            output_matrix(&C);
            break;
        }
        case 3: //Умножение двух матриц
        {
            puts("Введите элементы матрицы A");
            count = input_matrix(&A);
            check_errors(count, 4);
            puts("Введите элементы матрицы B");
            count = input_matrix(&B);
            check_errors(count, 5);
            puts("Вывод матрицы C = A * B");
            C = mult_matrix(&A, &B);
        }
    }
}
```

```

        output_matrix(&C);
        break;
    }
    case 4: //Умножение матрицы на число
    {
        puts("Введите элементы матрицы A");
        count = input_matrix(&A);
        check_errors(count, 6);
        float scalar;
        puts("Введите скаляр");
        count = scanf("%f", &scalar);
        if (count == 0)
        {
            puts("Неправильно введен скаляр");
            return 7;
        }
        puts("Вывод матрицы A, умноженной на скаляр");
        scalar_mult_matrix(&A, scalar);
        output_matrix(&A);
        break;
    }
    case 5: //Нахождение мин., макс., среднего значения элементов
    {
        puts("Введите элементы матрицы A");
        count = input_matrix(&A);
        check_errors(count, 8);
        float min, max, aver;
        find_values(&A, &min, &max, &aver);
        printf("Мин. значение = %.2f\n", min);
        printf("Макс. значение = %.2f\n", max);
        printf("Среднее значение = %.2f\n", aver);
        break;
    }
    case 6: //Нахождение определителя матрицы
    {
        puts("Введите элементы матрицы A");
        count = input_matrix(&A);
        check_errors(count, 9);
        float det = determ_matrix(A);
        printf("det A = %.2f\n", det);
        break;
    }
    case 7: //Преобразование матрицы в обратную
    {
        puts("Введите элементы матрицы A");
        count = input_matrix(&A);
        check_errors(count, 10);
        count = inverse_matrix(&A);
        if (count != 0)
        {
            puts("Обратной матрицы не существует");
            return 11;
        }
        puts("Вывод матрицы, обратной матрице A");
        output_matrix(&A);
        break;
    }
}

```

```

        case 8: //Вывод матрицы в консоль
        {
            //Фактически равносильно case 1:
        }
        case 1: //Создание структуры матрицы
        {
            puts("Введите элементы матрицы A");
            count = input_matrix(&A);
            check_errors(count, 12);
            puts("Вывод матрицы A:");
            output_matrix(&A);
            break;
        }
    }
    return 0;
}

```

## Результаты работы программы:

### 1) Создание структуры матрицы:

```

Введите номер операции:|
1
Введите элементы матрицы A
-3 4 1.4 8
2 1.1 2 0
-1 2 3 0.3
9 7 -7.1 5
Вывод матрицы A:
-3.00  4.00  1.40  8.00
 2.00  1.10  2.00  0.00
-1.00  2.00  3.00  0.30
 9.00  7.00 -7.10  5.00
[не нулевая]

```

### 2) Сложение матриц:

```

Введите номер операции:
2
Введите элементы матрицы A
3.5  7  -2  4
5  -2.7 8  1.31
-4  3  9  6
5  2  0.5 2.8
Введите элементы матрицы B
-1  4.3  1  0.37
4  4  2  -4.9
9  5  3  -5
2  5.2 2.77 4
Вывод матрицы C = A + B
 2.50 11.30 -1.00  4.37
 9.00  1.30 10.00 -3.59
 5.00  8.00 12.00  1.00
 7.00  7.20  3.27  6.80
[не нулевая]

```

3) Умножение матриц:

```
Введите номер операции:
3
Введите элементы матрицы A
3.5  7  -2  4
5  -2.7 8  1.31
-4  3  9  6
5  2  0.5 2.8
Введите элементы матрицы B
-1  4.3  1  0.37
4  4  2  -4.9
9  5  3  -5
2  5.2 2.77  4
Вывод матрицы C = A * B
14.50  53.85  22.58  -7.00
58.82  57.51  27.23  -19.68
109.00  71.00  45.62  -37.18
13.10  46.56  18.26  0.75
[не нулевая]
```

4) Умножение матрицы на скаляр

```
Введите номер операции:
4
Введите элементы матрицы A
3.5  7  -2  4
5  -2.7 8  1.31
-4  3  9  6
5  2  0.5 2.8
Введите скаляр
4.44
Вывод матрицы A, умноженной на скаляр
15.54  31.08  -8.88  17.76
22.20 -11.99  35.52  5.82
-17.76 13.32  39.96  26.64
22.20  8.88  2.22  12.43
[не нулевая]
```

5) Нахождение тройки минимального, максимального и среднего значений элементов переданной матрицы:

```
Введите номер операции:
5
Введите элементы матрицы A
3.5  7  -2  4
5  -2.7 8  1.31
-4  3  9  6
5  2  0.5 2.8
Мин. значение = -4.00
Макс. значение = 9.00
Среднее значение = 3.03
```

6) Нахождение определителя матрицы:

```
Введите номер операции:
6
Введите элементы матрицы A
3.5  7  -2  4
5  -2.7 8  1.31
-4  3  9  6
5  2  0.5 2.8
det A = -945.51
```

## 7) Преобразование матрицы в обратную:

```
Введите номер операции:
7
Введите элементы матрицы A
3.5 7 -2 4
5 -2.7 8 1.31
-4 3 9 6
5 2 0.5 2.8
Вывод матрицы, обратной матрице A
0.06 0.09 -0.07 0.02
0.40 0.20 -0.06 -0.53
0.17 0.19 -0.01 -0.32
-0.42 -0.34 0.16 0.76
[не нулевая]
```

## 8) Вывод в консоль переданной матрицы:

```
Введите номер операции:|
1
Введите элементы матрицы A
-3 4 1.4 8
2 1.1 2 0
-1 2 3 0.3
9 7 -7.1 5
Вывод матрицы A:
-3.00 4.00 1.40 8.00
2.00 1.10 2.00 0.00
-1.00 2.00 3.00 0.30
9.00 7.00 -7.10 5.00
[не нулевая]
```

```
Введите номер операции:
1
Введите элементы матрицы A
0 0 0 0
0 0 0 0
0 0 0 0
0 0 0 0
Вывод матрицы A:
0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00
[нулевая]
```

## Проверка работы программы:

### 1) Создание структуры матрицы:

Матрица состоит из значений, введённых пользователем построчно.

### 2) Сложение матриц:

Результаты, выведенные программой, совпадают с таковыми, полученными в системе Mathcad.

$$A := \begin{pmatrix} 3.5 & 7 & -2 & 4 \\ 5 & -2.7 & 8 & 1.31 \\ -4 & 3 & 9 & 6 \\ 5 & 2 & 0.5 & 2.8 \end{pmatrix} \quad B := \begin{pmatrix} -1 & 4.3 & 1 & 0.37 \\ 4 & 4 & 2 & -4.9 \\ 9 & 5 & 3 & -5 \\ 2 & 5.2 & 2.77 & 4 \end{pmatrix}$$

$$C := A + B = \begin{pmatrix} 2.50 & 11.30 & -1.00 & 4.37 \\ 9.00 & 1.30 & 10.00 & -3.59 \\ 5.00 & 8.00 & 12.00 & 1.00 \\ 7.00 & 7.20 & 3.27 & 6.80 \end{pmatrix}$$

### 3) Умножение матриц:

Результаты, выведенные программой, совпадают с таковыми, полученными в системе Mathcad.

$$A := \begin{pmatrix} 3.5 & 7 & -2 & 4 \\ 5 & -2.7 & 8 & 1.31 \\ -4 & 3 & 9 & 6 \\ 5 & 2 & 0.5 & 2.8 \end{pmatrix} \quad B := \begin{pmatrix} -1 & 4.3 & 1 & 0.37 \\ 4 & 4 & 2 & -4.9 \\ 9 & 5 & 3 & -5 \\ 2 & 5.2 & 2.77 & 4 \end{pmatrix}$$

$$C := A \cdot B = \begin{pmatrix} 14.50 & 53.85 & 22.58 & -7.01 \\ 58.82 & 57.51 & 27.23 & -19.68 \\ 109.00 & 71.00 & 45.62 & -37.18 \\ 13.10 & 46.56 & 18.26 & 0.75 \end{pmatrix}$$

### 4) Умножение матрицы на скаляр

Результаты, выведенные программой, совпадают с таковыми, полученными в

системе Mathcad.

$$A := \begin{pmatrix} 3.5 & 7 & -2 & 4 \\ 5 & -2.7 & 8 & 1.31 \\ -4 & 3 & 9 & 6 \\ 5 & 2 & 0.5 & 2.8 \end{pmatrix} \quad B := \begin{pmatrix} -1 & 4.3 & 1 & 0.37 \\ 4 & 4 & 2 & -4.9 \\ 9 & 5 & 3 & -5 \\ 2 & 5.2 & 2.77 & 4 \end{pmatrix}$$

$$C := A \cdot 4.44 = \begin{pmatrix} 15.54 & 31.08 & -8.88 & 17.76 \\ 22.20 & -11.99 & 35.52 & 5.82 \\ -17.76 & 13.32 & 39.96 & 26.64 \\ 22.20 & 8.88 & 2.22 & 12.43 \end{pmatrix}$$

- 5) Нахождение тройки минимального, максимального и среднего значений элементов переданной матрицы:

Результаты, выведенные программой, совпадают с таковыми, полученными в Microsoft Excel.

<b>A =</b>	3,50	7,00	-2,00	4,00
	5,00	-2,70	8,00	1,31
	-4,00	3,00	9,00	6,00
	5,00	2,00	0,50	2,80
<b>мин.</b>	<b>-4,00</b>		МИН(МАТР)	
<b>средн.</b>	<b>3,03</b>		СРЗНАЧ(МАТР)	
<b>макс.</b>	<b>9,00</b>		МАКС(МАТР)	

- 6) Нахождение определителя матрицы:

Результаты, выведенные программой, совпадают с таковыми, полученными в системе Mathcad.

$$A := \begin{pmatrix} 3.5 & 7 & -2 & 4 \\ 5 & -2.7 & 8 & 1.31 \\ -4 & 3 & 9 & 6 \\ 5 & 2 & 0.5 & 2.8 \end{pmatrix}$$

$$|A| = -945.512$$

- 7) Преобразование матрицы в обратную:

Результаты, выведенные программой, совпадают с таковыми, полученными в системе Mathcad

$$A := \begin{pmatrix} 3.5 & 7 & -2 & 4 \\ 5 & -2.7 & 8 & 1.31 \\ -4 & 3 & 9 & 6 \\ 5 & 2 & 0.5 & 2.8 \end{pmatrix}$$

$$A^{-1} = \begin{pmatrix} 0.06 & 0.09 & -0.07 & 0.02 \\ 0.40 & 0.20 & -0.06 & -0.53 \\ 0.17 & 0.19 & -0.01 & -0.32 \\ -0.42 & -0.34 & 0.16 & 0.76 \end{pmatrix}$$

- 8) Вывод в консоль переданной матрицы:

Матрица состоит из значений, введенных пользователем построчно, правильно выводится информация о том, введена ли нулевая матрица (справа) или нет (слева).