МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

**«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ
ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ им. В. Г. ШУХОВА»
(БГТУ им. В.Г. Шухова)**

Кафедра программного обеспечения вычислительной техники и автоматизированных систем

Лабораторная работа №13

по дисциплине: Основы программирования
тема: «Создание библиотеки для работы с многомерными массивами»

Выполнил: студент группы ПВ-223
Мелехов Артём Дмитриевич

Проверили:
ст. преп. Притчин Иван Сергеевич
асс. Черников Сергей Викторович

Белгород 2023 г.

Лабораторная работа №13 «Создание библиотеки для работы с многомерными массивами»

Цель работы: закрепление навыков создания библиотек, структур; получение навыков работы с многомерными массивами.

Содержание отчета:

Тема лабораторной работы

Цель лабораторной работы

Решения задач. Для каждой задачи указаны:

– Текст задачи.

– Исходный код (в том числе и тестов).

Вывод.

Список всех импортируемых библиотек:

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <malloc.h>
#include <assert.h>
#include <stdbool.h>
#include <stdlib.h>
```

Задача №1.

В заголовочном файле объявите структуру ’матрица’ и ’позиция’

Код программы:

```
// Структура описывающая матрицу
typedef struct Matrix
{
    long long** values; // значения
    long long n_rows;   // кол-во строк
    long long n_cols;   // кол-во столбцов
} Matrix;

// Структура описывающая позицию элемента в матрице
typedef struct Position
{
    long long row_pos;  // номер строки
    long long col_pos;  // номер столбца
} Position;
```

Задача №2.

В библиотеке matrix реализуйте функции для размещения в динамической памяти матриц:

```
Matrix get_matrix(const long long n_rows, const long long n_cols);

void free_matrix(Matrix a);
Matrix* get_array_of_matrix(const long long k, const long long n, const long long
m);
void free_array_of_matrix(Matrix* a, const long long k)
```

Код программы:

```
// Выделение памяти для матрицы a размера m * n
Matrix get_matrix(const long long n_rows, const long long n_cols)
{
    long long** a = (long long**)malloc(sizeof(long long*) * n_rows);

    for (size_t i = 0; i < n_rows; i++)
        a[i] = (long long*)malloc(sizeof(long long) * n_cols);

    Matrix matrix = {
        a,
        n_rows,
        n_cols
    };
    return matrix;
}
```

```
// Освобождение памяти выделенной под матрицу
void free_matrix(Matrix a)
{
    for (size_t i = 0; i < a.n_rows; i++)
        free(a.values[i]);

    free(a.values);
}


// Выделение памяти для массива матриц a размера k,

// каждвя из которых имеет размер n * m
Matrix* get_array_of_matrix(const long long k,
    const long long n, const long long m)
{
    Matrix* a = (Matrix*)malloc(sizeof(Matrix) * k);

    for (size_t i = 0; i < k; i++)
        a[i] = get_matrix(n, m);

    return a;
}

// Освобождение памяти для массива матриц a размера k,
// каждвя из которых имеет размер n * m
void free_array_of_matrix(Matrix* a, const long long k)
{
    for (size_t i = 0; i < k; i++)
        free_matrix(a[i]);

    free(a);
}
```

Задача №3.

В библиотеке matrix реализуйте функции для ввода и вывода матриц:

```
void input_matrix(Matrix a);

void output_matrix(Matrix a);

void input_array_of_matrix(const Matrix* a, const long long n);

void output_array_of_matrix(const Matrix* a, const long long n)
```

Код программы:

```
// Ввод матрицы a размера n * m
void input_matrix(Matrix a)
{
    for (size_t i = 0; i < a.n_rows; i++)
        for (size_t j = 0; j < a.n_cols; j++)
            scanf("%lld", &a.values[i][j]);
}
```

```
// Вывод матрицы a размера n * m
void output_matrix(Matrix a)
{
    for (size_t i = 0; i < a.n_rows; i++)
    {
        for (size_t j = 0; j < a.n_cols; j++)
            printf("%d ", a.values[i][j]);

        printf("\n");
    }
}


// Ввод массива матриц a размера n

void input_array_of_matrix(const Matrix* a, const long long n)
{
    for (size_t i = 0; i < n; i++)
        input_matrix(a[i]);
}

// Вывод массива матриц a размера k, каждая из которых n * m
void output_array_of_matrix(const Matrix* a, const long long n)
{
    for (size_t i = 0; i < n; i++)
        output_matrix(a[i]);
}
```

Задача №4.

В библиотеке matrix реализуйте функции для обмена строк и столбцов:

```
void swap_rows(Matrix a, const long long i1, const long long i2);

void swap_cols(Matrix a, long long j1, long long j2)
```

Код программы:

```
void swap_pointers(long long** a, long long** b)
{
    long long* t = *a;
    *a = *b;
    *b = t;
}

// Обмен 2-ух строк в матрице
void swap_rows(Matrix a, const long long i1, const long long i2)
{
    assert(0 <= i1 && i1 <= a.n_rows);
    assert(0 <= i2 && i2 <= a.n_rows);

    swap_pointers(&a.values[i1], &a.values[i2]);
}

void swap(long long* a, long long* b)
{
    long long t = *a;
    *a = *b;
    *b = t;
}
```

```
// Обмен 2-ух столбцов в матрице
void swap_cols(Matrix a, long long j1, long long j2)
{
    assert(0 <= j1 && j1 <= a.n_cols);
    assert(0 <= j2 && j2 <= a.n_cols);

    for (size_t i = 0; i < a.n_rows; i++)
        swap(&a.values[i][j1], &a.values[i][j2]);
}



void test_swap_rows()
{
    long long m_[] = {
            1, 2, 3, 4,
            5, 6, 7, 8,
            9, 10, 11, 12
    };
    Matrix m = create_matrix_from_array(m_, 3, 4);
    free(m_);

    swap_rows(m, 0, 2);

    long long expected_array_[] = {
                    9, 10, 11, 12,
                    5, 6, 7, 8,
                    1, 2, 3, 4
    };
    Matrix expected_array = create_matrix_from_array(expected_array_, 3, 4);
    free(expected_array_);

    assert(are_two_matrices_equal(expected_array, m));

    free_matrix(m);
    free_matrix(expected_array);
}

void test_swap_cols()
{
    long long m_[] = {
            1, 2, 3, 4,
            5, 6, 7, 8,
            9, 10, 11, 12
    };
    Matrix m = create_matrix_from_array(m_, 3, 4);
    free(m_);

    swap_cols(m, 0, 2);

    long long expected_array_[] = {
                        3, 2, 1, 4,
                        7, 6, 5, 8,
                        11, 10, 9, 12
    };
    Matrix expected_array = create_matrix_from_array(expected_array_, 3, 4);
    free(expected_array_);

    assert(are_two_matrices_equal(expected_array, m));

    free_matrix(m);
    free_matrix(expected_array);
}
```

Задача №5.

В библиотеке matrix реализуйте функции для упорядочивания строк и столбцов:

```
void sort_rows_by_criteria(Matrix a, long long (*criteria)(const long long*, long long));
void sort_cols_by_criteria(Matrix a, long long (*criteria)(const long long*, long long))
```

Код программы:

```
// Сортировка строк матрицы по критерию
void sort_rows_by_criteria(Matrix a,
    long long (*criteria)(const long long*, long long))
{
    long long* criteria_values = (long long*)malloc(sizeof(long long) * a.n_rows);

    for (size_t i = 0; i < a.n_rows; i++)
        criteria_values[i] = criteria(a.values[i], a.n_cols);

    for (size_t i = 0; i < a.n_rows; i++)
    {
        long long min_index = i;

        for (size_t j = i + 1; j < a.n_rows; j++)
            if (criteria_values[j] < criteria_values[min_index])
                min_index = j;

        if (i != min_index)
        {
            swap(&criteria_values[i], &criteria_values[min_index]);
            swap_rows(a, i, min_index);
        }
    }

    free(criteria_values);
}
```

```c
// Сортировка столбцов матрицы по критерию
void sort_cols_by_criteria(Matrix a,
    long long (*criteria)(const long long*, long long))
{
    long long* criteria_values = (long long*)malloc(sizeof(long long) * a.n_cols);
    long long* column = (long long*)malloc(sizeof(long long) * a.n_rows);

    for (size_t j = 0; j < a.n_cols; j++)
    {
        for (size_t i = 0; i < a.n_rows; i++)
            column[i] = a.values[i][j];

        criteria_values[j] = criteria(a.values[j], a.n_cols);
    }

    for (size_t i = 0; i < a.n_cols; i++)
    {
        long long min_index = i;

        for (size_t j = i + 1; j < a.n_rows; j++)
            if (criteria_values[j] < criteria_values[min_index])
                min_index = j;

        if (i != min_index)
        {
            swap(&criteria_values[i], &criteria_values[min_index]);
            swap_cols(a, i, min_index);
        }
    }

    free(criteria_values);
    free(column);
}



// Поиск суммы элементов в строке массива
long long get_sum(const long long* a, const long long n)
{
    long long sum = 0;

    for (size_t i = 0; i < n; i++)
        sum += a[i];

    return sum;
}
```

```
void test_sort_rows_by_criteria()
{
    long long m_[] = {
            5, 6, 7, 8,
            1, 2, 3, 4,
            9, 10, 11, 12
    };
    Matrix m = create_matrix_from_array(m_, 3, 4);
    free(m_);

    sort_rows_by_criteria(m, get_sum);

    long long expected_array_[] = {
                        1, 2, 3, 4,
                        5, 6, 7, 8,
                        9, 10, 11, 12
    };
    Matrix expected_array = create_matrix_from_array(expected_array_, 3, 4);
    free(expected_array_);

    assert(are_two_matrices_equal(expected_array, m));

    free_matrix(m);
    free_matrix(expected_array);
}

void test_sort_cols_by_criteria()
{
    long long m_[] = {
            6, 5, 7, 8,
            2, 1, 3, 4,
            10, 9, 11, 12
    };
    Matrix m = create_matrix_from_array(m_, 3, 4);
    free(m_);

    sort_cols_by_criteria(m, get_sum);

    long long expected_array_[] = {
                        5, 6, 7, 8,
                        1, 2, 3, 4,
                        9, 10, 11, 12
    };
    Matrix expected_array = create_matrix_from_array(expected_array_, 3, 4);
    free(expected_array_);

    assert(are_two_matrices_equal(expected_array, m));

    free_matrix(m);
    free_matrix(expected_array);
}
```

Задача №6.

В библиотеке matrix реализуйте следующие функции-предикаты:

```
bool is_square_matrix(Matrix m);

bool are_two_matrices_equal(Matrix m1, Matrix m2);

bool is_e_matrix(Matrix m);

bool is_symmetric_matrix(Matrix m)
```

Код программы:

```
// Возвращает значение 'истина', если матрица m квадратная
bool is_square_matrix(Matrix m)
{
    return m.n_cols == m.n_rows;
}

// Возвращает 'истина', если матрицы m1 и m2 равны
bool are_two_matrices_equal(Matrix m1, Matrix m2)
{
    if (m1.n_cols != m2.n_cols || m1.n_rows != m2.n_rows)
        return false;

    for (size_t i = 0; i < m1.n_rows; i++)
        for (size_t j = 0; j < m1.n_cols; j++)
            if (m1.values[i][j] != m2.values[i][j])
                return false;

    return true;
}

// Возвращает 'истина', если матрица является единичной
bool is_e_matrix(Matrix m)
{
    if (m.n_cols == m.n_rows)
        return false;

    long long n = m.n_rows;

    for (size_t i = 0; i < n; i++)
        for (size_t j = 0; j < n; j++)
            if ((i == j) != m.values[i][j])
                return false;
}



// Возвращает 'истина', если матрица m симметричная
bool is_symmetric_matrix(Matrix m)
{
    if (m.n_cols != m.n_rows)
        return false;

    Matrix mt = get_matrix(m.n_rows, m.n_cols);

    for (size_t i = 0; i < m.n_rows; i++)
        for (size_t j = 0; j < m.n_cols; j++)
            mt.values[i][j] = m.values[i][j];

    transpose_square_matrix(&mt);

    return are_two_matrices_equal(m, mt);
}
```

```
void test_are_two_matrices_equal()
{
    long long m_[] = {
            1, 2, 3, 4,
            5, 6, 7, 8,
            9, 10, 11, 12
    };
    Matrix m = create_matrix_from_array(m_, 3, 4);
    free(m_);

    long long m1_[] = {
                        1, 2, 3, 4,
                        5, 6, 7, 8,
                        9, 10, 11, 12
    };
    Matrix m1 = create_matrix_from_array(m1_, 3, 4);
    free(m1_);

    assert(are_two_matrices_equal(m, m1));

    free_matrix(m);
    free_matrix(m1);
}



void test_is_square_matrix()
{
    long long m_[] = {
            6, 5, 7,
            2, 1, 3,
            10, 9, 11
    };
    Matrix m = create_matrix_from_array(m_, 3, 3);
    free(m_);

    assert(is_square_matrix(m));

    free_matrix(m);
}

void test_is_e_matrix()
{
    long long m_[] = {
                1, 0, 0,
                0, 1, 0,
                0, 0, 1
    };
    Matrix m = create_matrix_from_array(m_, 3, 3);
    free(m_);

    assert(is_e_matrix(m));

    free_matrix(m);
}
```

```
void test_is_symmetric_matrix()
{
    long long m_[] = {
                1, 0, 0,
                0, 1, 0,
                0, 0, 1
    };
    Matrix m = create_matrix_from_array(m_, 3, 3);
    free(m_);

    assert(is_symmetric_matrix(m));

    free_matrix(m);
}
```

Задача №7.

В библиотеке matrix реализуйте следующие функции преобразования матриц:

```
void transpose_square_matrix(Matrix* m);

void transpose_matrix(Matrix* m)
```

Код программы:

```
// Транспонирует квадратную матрицу m
void transpose_square_matrix(Matrix* m)
{
    assert(m->n_cols == m->n_rows);

    for (size_t i = 0; i < m->n_rows; i++)
        for (size_t j = i + 1; j < m->n_cols; j++)
            swap(&m->values[i][j], &m->values[j][i]);
}

// Транспонирует матрицу m
void transpose_matrix(Matrix* m)
{
    Matrix t = get_matrix(m->n_cols, m->n_rows);

    for (size_t i = 0; i < m->n_rows; i++)
        for (size_t j = 0; j < m->n_cols; j++)
            t.values[j][i] = m->values[i][j];

    free_matrix(*m);
    *m = t;
}
```

```
void test_transpose_square_matrix()
{
    long long m_[] = {
                6, 5, 7,
                2, 1, 3,
                10, 9, 11
    };
    Matrix m = create_matrix_from_array(m_, 3, 3);
    free(m_);

    transpose_square_matrix(&m);

    long long expected_array_[] = {
                        6, 2, 10,
                        5, 1, 9,
                        7, 3, 11
    };
    Matrix expected_array = create_matrix_from_array(expected_array_, 3, 3);
    free(expected_array_);

    assert(are_two_matrices_equal(expected_array, m));

    free_matrix(m);
    free_matrix(expected_array);
}

void test_transpose_matrix()
{
    long long m_[] = {
            6, 5, 7, 8,
            2, 1, 3, 4,
            10, 9, 11, 12
    };
    Matrix m = create_matrix_from_array(m_, 3, 4);
    free(m_);

    transpose_matrix(&m);

    long long expected_array_[] = {
                        6, 2, 10,
                        5, 1, 9,
                        7, 3, 11,
                        8, 4, 12
    };
    Matrix expected_array = create_matrix_from_array(expected_array_, 4, 3);
    free(expected_array_);

    assert(are_two_matrices_equal(expected_array, m));

    free_matrix(m);
    free_matrix(expected_array);
}
```

Задача №8.

В библиотеке matrix реализуйте функции для поиска минимального и макси□мального элемента матрицы:

```
 Position get_min_value_pos(Matrix a);

Position get_max_value_pos(Matrix a)
```

Код программы:

```
// Позиция минимума в матрице
Position get_min_value_pos(Matrix a)
{
    long long i_min = 0;
    long long j_min = 0;

    for (size_t i = 0; i < a.n_rows; i++)
        for (size_t j = 0; j < a.n_cols; j++)
            if (a.values[i][j] < a.values[i_min][j_min])
            {
                i_min = i;
                j_min = j;
            }

    Position result = { i_min, j_min };
    return result;
}

// Позиция максимума в матрице
Position get_max_value_pos(Matrix a)
{
    long long i_max = 0;
    long long j_max = 0;

    for (size_t i = 0; i < a.n_rows; i++)
        for (size_t j = 0; j < a.n_cols; j++)
            if (a.values[i][j] > a.values[i_max][j_max])
            {
                i_max = i;
                j_max = j;
            }

    Position result = { i_max, j_max };
    return result;
}
```

```
void test_get_min_value_pos()
{
    long long m_[] = {
                6, 2, 10,
                5, 1, 9,
                7, 3, 11,
                8, 4, 12
    };
    Matrix m = create_matrix_from_array(m_, 4, 3);
    free(m_);

    Position min_pos = get_min_value_pos(m);

    assert(min_pos.row_pos == 1 && min_pos.col_pos == 1);

    free_matrix(m);
}

void test_get_max_value_pos()
{
    long long m_[] = {
                6, 2, 10,
                5, 1, 9,
                7, 3, 11,
                8, 4, 12
    };
    Matrix m = create_matrix_from_array(m_, 4, 3);
    free(m_);

    Position min_pos = get_max_value_pos(m);

    assert(min_pos.row_pos == 3 && min_pos.col_pos == 2);

    free_matrix(m);
}
```

Задача №9.

Дополните библиотеку функциями для тестирования:

```
Matrix create_matrix_from_array(const long long* a, long long n_rows, long long
n_cols);

Matrix* create_array_of_matrix_from_array(const long long* values, size_t
n_matrices, size_t n_rows, size_t n_cols)
```

Код программы:

```
// Преобразовывает одномерный массив в двумерный (матрицу)
Matrix create_matrix_from_array(const long long* a, long long n_rows, long long
n_cols)
{
    Matrix m = get_matrix(n_rows, n_cols);

    long long k = 0;

    for (size_t i = 0; i < n_rows; i++)
        for (size_t j = 0; j < n_cols; j++)
            m.values[i][j] = a[k++];

    return m;
}

// Возвращает указатель на нулевую матрицу массива из
// n_matrices матриц, размещённых в динамической памяти,
// построенных из массива a
Matrix* create_array_of_matrix_from_array(
    const long long* values, size_t n_matrices,
    size_t n_rows, size_t n_cols)
{
    Matrix* ms = get_array_of_matrix(n_matrices, n_rows, n_cols);
    long long l = 0;

    for (size_t k = 0; k < n_matrices; k++)
        for (size_t i = 0; i < n_rows; i++)
            for (size_t j = 0; j < n_cols; j++)
                ms[k].values[i][j] = values[l++];

    return ms;
}
```

Вывод: в ходе выполнения работы закреплены навыки создания библиотек, структур; получены навыки работы с многомерными массивами.