

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 №15

по дисциплине: Основы программирования

тема: «Создание библиотеки для работы с многомерными массивами»

Выполнил: ст. группы ПВ-223
Пахомов Владислав Андреевич

Проверили:
Притчин Иван Сергеевич
Черников Сергей Викторович

Код-ревьюер: ст. группы ПВ-223
Голуцкий Георгий Юрьевич

Белгород 2023 г.

Лабораторная работа № 15

Содержание отчёта:

- Тема лабораторной работы.
- Цель лабораторной работы.
- Решения задач.
 - Текст задания.
 - Исходный код (в т.ч. и тестов).
- Ссылка на открытый репозиторий с решением.
- Скриншот с историей коммитов.
- Вывод по работе.

Тема лабораторной работы: Создание библиотеки для работы с многомерными массивами

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

Решения задач:

- 1) В заголовочном файле `libs\data_structures\matrix\matrix.h` объявите структуру 'матрица' и 'позиция':

```
typedef struct Matrix {  
    int **values; // элементы матрицы  
    int nRows; // количество рядов  
    int nCols; // количество столбцов  
} Matrix;  
  
typedef struct Position {  
    int rowIndex;  
    int colIndex;  
} Position;
```

- 2) В библиотеке `matrix` реализуйте функции для размещения в динамической памяти матриц:

- a) `Matrix getMemMatrix(int nRows, int nCols)` – размещает в динамической памяти матрицу размером `nRows` на `nCols`. Возвращает матрицу.

```
Matrix getMemMatrix(int nRows, int nCols) {  
    int **values = (int **) malloc(sizeof(int *) * nRows);  
    for (int i = 0; i < nRows; i++)  
        values[i] = (int *) malloc(sizeof(int) * nCols);  
    return (Matrix) {values, nRows, nCols};  
}
```

- b) `Matrix *getMemArrayOfMatrices(int nMatrices, int nRows, int nCols)` – размещает в динамической памяти массив из матриц размером `nRows` на `nCols`. Возвращает указатель на нулевую матрицу.

```
Matrix *getMemArrayOfMatrices(int nMatrices, int nRows, int nCols) {  
    Matrix *matrixArray = (Matrix *) malloc(sizeof(Matrix) * nMatrices);  
    for (int i = 0; i < nMatrices; i++)  
        matrixArray[i] = getMemMatrix(nRows, nCols);  
    return matrixArray;  
}
```

- c) `void freeMemMatrix(Matrix m)` – освобождает память, выделенную под хранение матрицы `m`.

```
void freeMemMatrix(Matrix m) {  
    for (int i = 0; i < m.nRows; i++)  
        free(m.values[i]);  
    free(m.values);  
}
```

- d) `void freeMemMatrices(Matrix *ms, int nMatrices)` - освобождает память, выделенную под хранение массива `ms` из `nMatrices` матриц.

```
void freeMemMatrices(Matrix *ms, int nMatrices) {  
    for (int i = 0; i < nMatrices; i++)  
        freeMemMatrix(ms[i]);  
    free(ms);  
}
```

- 3) В библиотеке `matrix` реализуйте функции для ввода и вывода матриц:

- a) `void inputMatrix(Matrix m)` - ввод матрицы `m`.

```
void inputMatrix(Matrix m) {  
    for (int i = 0; i < m.nRows; i++)  
        for (int j = 0; j < m.nCols; j++)  
            scanf("%d", &m.values[i][j]);  
}
```

- b) `void inputMatrices(Matrix *ms, int nMatrices)` - ввод массива из `nMatrices` матриц, хранящейся по адресу `ms`.

```
void inputMatrices(Matrix *ms, int nMatrices) {  
    for (int i = 0; i < nMatrices; i++)  
        inputMatrix(ms[i]);  
}
```

c) `void outputMatrix(Matrix m)` - вывод матрицы `m`.

```
void outputMatrix(Matrix m) {
    for (int i = 0; i < m.nRows; i++) {
        for (int j = 0; j < m.nCols; j++)
            printf("%d ", m.values[i][j]);

        printf("\n");
    }
}
```

d) `void outputMatrices(Matrix *ms, int nMatrices)` - вывод массива из `nMatrices` матриц, хранящейся по адресу `ms`.

```
void outputMatrices(Matrix *ms, int nMatrices) {
    for (int i = 0; i < nMatrices; i++)
        outputMatrix(ms[i]);
}
```

4) В библиотеке `matrix` реализуйте функции для обмена строк и столбцов:

a) `void swapRows(Matrix m, int i1, int i2)` - обмен строк с порядковыми номерами `i1` и `i2` в матрице `m`.

```
void swapRows(Matrix m, int i1, int i2) {
    assert(i1 < m.nRows && i2 < m.nRows && i1 >= 0 && i2 >= 0);

    swap(m.values + i1, m.values + i2, sizeof(m.values[0]));
}
```

```
void testSwapRows() {
    Matrix m = createMatrixFromArray((int[]) {1, 2, 3, 4,
                                              5, 6, 7, 8,
                                              9, 10, 11, 12}, 3, 4);

    swapRows(m, 0, 2);

    Matrix expectedArray = createMatrixFromArray((int[]) {9, 10, 11, 12,
                                                           5, 6, 7, 8,
                                                           1, 2, 3, 4}, 3, 4);

    assert(areTwoMatricesEqual(expectedArray, m));

    freeMemMatrix(m);
    freeMemMatrix(expectedArray);
}
```

- b) void swapColumns(Matrix m, int j1, int j2) - обмен колонок с порядковыми номерами j1 и j2 в матрице m.

```
void swapColumns(Matrix m, int j1, int j2) {
    assert(j1 < m.nCols && j2 < m.nCols && j1 >= 0 && j2 >= 0);

    for (int i = 0; i < m.nRows; i++)
        swap(&m.values[i][j1], &m.values[i][j2], sizeof(m.values[0][0]));
}
```

```
void testSwapColumns() {
    Matrix m = createMatrixFromArray((int[]) {1, 2, 3, 4,
                                              5, 6, 7, 8,
                                              9, 10, 11, 12}, 3, 4);

    swapColumns(m, 0, 2);

    Matrix expectedArray = createMatrixFromArray((int[]) {3, 2, 1, 4,
                                                           7, 6, 5, 8,
                                                           11, 10, 9, 12}, 3, 4);

    assert(areTwoMatricesEqual(expectedArray, m));

    freeMemMatrix(m);
    freeMemMatrix(expectedArray);
}
```

- 5) В библиотеке matrix реализуйте функции для упорядочивания строк и столбцов:

- a) void insertionSortRowsMatrixByRowCriteria(Matrix m, int (*criteria)(int*, int)) - выполняет сортировку вставками строк матрицы m по неубыванию значения функции criteria применяемой для строк.

```
void insertionSortRowsMatrixByRowCriteria(Matrix m,
                                           int (*criteria)(int *, int)) {
    int *keys = (int *) malloc(sizeof(int) * m.nRows);
    for (int i = 0; i < m.nRows; i++)
        keys[i] = criteria(m.values[i], m.nCols);

    for (int i = 1; i < m.nRows; i++) {
        int tKey = keys[i];
        int* tRow = m.values[i];
        int j = i;
        while (j > 0 && keys[j - 1] > tKey) {
            keys[j] = keys[j - 1];
            m.values[j] = m.values[j - 1];
            j--;
        }

        keys[j] = tKey;
        m.values[j] = tRow;
    }

    free(keys);
}
```

```

int getSum(int *array, int arraySize) {
    int result = 0;

    for (int i = 0; i < arraySize; i++)
        result += array[i];

    return result;
}

int getMin(int *array, int arraySize) {
    int result = arraySize == 0 ? 0 : array[0];

    for (int i = 0; i < arraySize; i++)
        if (array[i] < result)
            result = array[i];

    return result;
}

int getMax(int *array, int arraySize) {
    int result = arraySize == 0 ? 0 : array[0];

    for (int i = 0; i < arraySize; i++)
        if (array[i] > result)
            result = array[i];

    return result;
}

void testInsertionSortRowsMatrixByRowCriteria() {
    Matrix m = createMatrixFromArray((int[]) {12, 6, 1, 4, //23
                                              9, 10, 11, 5, //35
                                              7, 2, 3, 8/*20*/}, 3, 4);

    insertionSortRowsMatrixByRowCriteria(m, getSum);

    Matrix expectedArray = createMatrixFromArray((int[]) {7, 2, 3, 8,
                                                           12, 6, 1, 4,
                                                           9, 10, 11, 5}, 3, 4);

    assert(areTwoMatricesEqual(expectedArray, m));

    freeMemMatrix(expectedArray);

    insertionSortRowsMatrixByRowCriteria(m, getMax);

    expectedArray = createMatrixFromArray((int[]) {7, 2, 3, 8,
                                                    9, 10, 11, 5,
                                                    12, 6, 1, 4}, 3, 4);

    assert(areTwoMatricesEqual(expectedArray, m));

    freeMemMatrix(expectedArray);

    insertionSortRowsMatrixByRowCriteria(m, getMin);
}

```

```

expectedArray = createMatrixFromArray((int[]) {12, 6, 1, 4,
                                              7, 2, 3, 8,
                                              9, 10, 11, 5}, 3, 4);

assert(areTwoMatricesEqual(expectedArray, m));

freeMemMatrix(m);
freeMemMatrix(expectedArray);
}

```

- b) `void selectionSortColsMatrixByColCriteria(Matrix m, int (*criteria)(int*, int))` - выполняет сортировку выбором столбцов матрицы `m` по неубыванию значения функции `criteria` применяемой для столбцов

```

void selectionSortColsMatrixByColCriteria(Matrix m,
                                          int (*criteria)(int *, int)) {
    int *keys = (int *) malloc(sizeof(int) * m.nCols);
    for (int i = 0; i < m.nCols; i++) {
        int *column = (int *) malloc(sizeof(int) * m.nRows);

        for (int j = 0; j < m.nRows; j++)
            column[j] = m.values[j][i];

        keys[i] = criteria(column, m.nRows);

        free(column);
    }

    for (int i = 0; i < m.nCols - 1; i++) {
        int minIndex = i;
        for (int j = i + 1; j < m.nCols; j++)
            if (keys[j] < keys[minIndex])
                minIndex = j;

        swapColumns(m, minIndex, i);
        swap(keys + i, keys + minIndex, sizeof(keys[0]));
    }

    free(keys);
}

```

```

void testSelectionSortColsMatrixByColCriteria() {
    Matrix m = createMatrixFromArray((int[]) {12, 6, 1, 4,
                                              9, 10, 11, 5,
                                              7, 2, 3, 8}, 3, 4);

    selectionSortColsMatrixByColCriteria(m, getSum);

    Matrix expectedArray = createMatrixFromArray((int[]) {1, 4, 6, 12,
                                                          11, 5, 10, 9,
                                                          3, 8, 2, 7}, 3, 4);

    assert(areTwoMatricesEqual(expectedArray, m));
}

```

```

freeMemMatrix(expectedArray);

selectionSortColsMatrixByColCriteria(m, getMax);

expectedArray = createMatrixFromArray((int[]) {4, 6, 1, 12,
                                                5, 10, 11, 9,
                                                8, 2, 3, 7}, 3, 4);

assert(areTwoMatricesEqual(expectedArray, m));

freeMemMatrix(expectedArray);

selectionSortColsMatrixByColCriteria(m, getMin);

expectedArray = createMatrixFromArray((int[]) {1, 6, 4, 12,
                                                11, 10, 5, 9,
                                                3, 2, 8, 7}, 3, 4);

assert(areTwoMatricesEqual(expectedArray, m));

freeMemMatrix(m);
freeMemMatrix(expectedArray);
}

```

6) В библиотеке `matrix` реализуйте следующие функции-предикаты:

- a) `bool isSquareMatrix(Matrix m)` - возвращает значение 'истина', если матрица `m` является квадратной, ложь – в противном случае.

```

bool isSquareMatrix(Matrix m) {
    return m.nRows == m.nCols;
}

```

```

void testIsSquareMatrix() {
    // Why go through all this trouble for such a small function...
    Matrix m = createMatrixFromArray((int[]) {12, 6, 1, 4,
                                                9, 10, 11, 5,
                                                7, 2, 3, 8}, 3, 4);

    assert(!isSquareMatrix(m));

    freeMemMatrix(m);

    m = createMatrixFromArray((int[]) {12, 6, 1, 4,
                                        9, 10, 11, 5,
                                        7, 2, 3, 8,
                                        1, 2, 3, 4}, 4, 4);

    assert(isSquareMatrix(m));

    freeMemMatrix(m);
}

```


b) `bool areTwoMatricesEqual(Matrix m1, Matrix m2)` - возвращает значение 'истина', если матрицы `m1` и `m2` равны, ложь – в противном случае

```
bool areTwoMatricesEqual(Matrix m1, Matrix m2) {
    if (m1.nRows != m2.nRows || m1.nCols != m2.nCols)
        return false;

    for (int i = 0; i < m1.nRows; i++)
        if(memcmp(m1.values[i], m2.values[i], sizeof(int) * m1.nCols))
            return false;

    return true;
}
```

```
void testAreTwoMatricesEqual() {
    // Why go through all this trouble for such a small function...
    Matrix m1 = createMatrixFromArray((int[]) {12, 6, 1, 4,
                                                9, 10, 11, 5,
                                                7, 2, 3, 8}, 3, 4);

    Matrix m2 = createMatrixFromArray((int[]) {12, 6, 1, 4,
                                                9, 10, 11, 5,
                                                7, 2, 3, 8}, 3, 4);

    assert(areTwoMatricesEqual(m1, m2));

    freeMemMatrix(m1);
    freeMemMatrix(m2);

    m1 = createMatrixFromArray((int[]) {12, 6, 1, 4,
                                        9, 16, 11, 5,
                                        7, 2, 3, 8}, 3, 4);
    m2 = createMatrixFromArray((int[]) {12, 6, 1, 4,
                                        9, 10, 11, 5,
                                        7, 2, 3, 8}, 3, 4);

    assert(!areTwoMatricesEqual(m1, m2));

    freeMemMatrix(m1);
    freeMemMatrix(m2);

    m1 = createMatrixFromArray((int[]) {12, 6,
                                        9, 16
                                        }, 2, 2);
    m2 = createMatrixFromArray((int[]) {12, 6, 1, 4,
                                        9, 10, 11, 5,
                                        7, 2, 3, 8}, 3, 4);

    assert(!areTwoMatricesEqual(m1, m2));

    freeMemMatrix(m1);
    freeMemMatrix(m2);
}
```

- c) `bool isEMatrix(Matrix m)` – возвращает значение 'истина', если матрица `m` является единичной, ложь – в противном случае.

```
bool isEMatrix(Matrix m) {
    if (!isSquareMatrix(m))
        return false;

    for (int i = 0; i < m.nRows; i++)
        for (int j = 0; j < m.nCols; j++)
            if ((i == j) != m.values[i][j])
                return false;

    return true;
}
```

```
void testIsEMatrix() {
    Matrix m1 = createMatrixFromArray((int[]) {12, 6, 1, 4,
                                                9, 10, 11, 5,
                                                7, 2, 3, 8}, 3, 4);

    assert(!isEMatrix(m1));

    freeMemMatrix(m1);

    m1 = createMatrixFromArray((int[]) {12, 6, 1, 4,
                                        9, 10, 11, 5,
                                        7, 2, 3, 8,
                                        1, 1, 1, 1}, 4, 4);

    assert(!isEMatrix(m1));

    freeMemMatrix(m1);

    m1 = createMatrixFromArray((int[]) {1, 0, 0, 0,
                                        0, 1, 0, 0,
                                        0, 0, 1, 0,
                                        0, 0, 0, 1}, 4, 4);

    assert(isEMatrix(m1));

    freeMemMatrix(m1);

    m1 = createMatrixFromArray((int[]) {1, 0, 0, 0,
                                        0, 1, 0, 1,
                                        0, 0, 1, 0,
                                        0, 0, 0, 1}, 4, 4);

    assert(!isEMatrix(m1));

    freeMemMatrix(m1);
}
```

d) `bool isSymmetricMatrix(Matrix m)` - возвращает значение 'истина', если матрица `m` является симметричной, ложь – в противном случае.

```
bool isSymmetricMatrix(Matrix m) {
    if (!isSquareMatrix(m))
        return false;

    for (int i = 0; i < m.nRows; i++)
        for (int j = i + 1; j < m.nCols; j++)
            if (m.values[i][j] != m.values[j][i])
                return false;

    return true;
}
```

```
void testIsSymmetricMatrix() {
    Matrix m = createMatrixFromArray((int[]) {12, 6, 1, 4,
                                                9, 10, 11, 5,
                                                7, 2, 3, 8}, 3, 4);

    assert(!isSymmetricMatrix(m));

    freeMemMatrix(m);

    m = createMatrixFromArray((int[]) {1, 3, 0,
                                        3, 2, 6,
                                        0, 6, 5}, 3, 3);

    assert(isSymmetricMatrix(m));

    freeMemMatrix(m);

    m = createMatrixFromArray((int[]) {1, 0, 0,
                                        0, 1, 0,
                                        0, 0, 1}, 3, 3);

    assert(isSymmetricMatrix(m));

    freeMemMatrix(m);

    m = createMatrixFromArray((int[]) {1, 5,
                                        5, 7}, 2, 2);

    assert(isSymmetricMatrix(m));

    freeMemMatrix(m);

    m = createMatrixFromArray((int[]) {2}, 1, 1);

    assert(isSymmetricMatrix(m));

    freeMemMatrix(m);
}
```

- 7) В библиотеке `matrix` реализуйте следующие функции преобразования матриц:
- a) `void transposeSquareMatrix(Matrix m)` - транспонирует квадратную матрицу `m`.

```
void transposeSquareMatrix(Matrix m) {
    assert(isSquareMatrix(m));

    for (int i = 0; i < m.nRows; i++)
        for (int j = i + 1; j < m.nCols; j++) {
            int t = m.values[i][j];
            m.values[i][j] = m.values[j][i];
            m.values[j][i] = t;
        }
}
```

```
void testTransposeSquareMatrix() {
    Matrix m = createMatrixFromArray((int[]) {1, 2, 3,
                                                4, 5, 6,
                                                7, 8, 9}, 3, 3);

    transposeSquareMatrix(m);

    Matrix expected = createMatrixFromArray((int[]) {1, 4, 7,
                                                        2, 5, 8,
                                                        3, 6, 9}, 3, 3);

    assert(areTwoMatricesEqual(m, expected));

    freeMemMatrix(m);
    freeMemMatrix(expected);
}
```

b) `void transposeMatrix(Matrix *m)` - транспонирует матрицу `m`.

```
void transposeMatrix(Matrix *m) {
    Matrix newMatrix = getMemMatrix(m->nCols, m->nRows);

    for (int i = 0; i < m->nRows; i++)
        for (int j = 0; j < m->nCols; j++)
            newMatrix.values[j][i] = m->values[i][j];

    freeMemMatrix(*m);
    *m = newMatrix;
}
```

```
void testTransposeMatrix() {
    Matrix m = createMatrixFromArray((int[]) {1, 2, 3,
                                              4, 5, 6}, 2, 3);

    transposeMatrix(&m);

    Matrix expected = createMatrixFromArray((int[]) {1, 4,
                                                      2, 5,
                                                      3, 6}, 3, 2);

    assert(areTwoMatricesEqual(m, expected));

    freeMemMatrix(m);
    freeMemMatrix(expected);
}
```

8) В библиотеке `matrix` реализуйте функции для поиска минимального и максимального элемента матрицы:

a) `Position getMinValuePos(Matrix m)` - возвращает позицию минимального элемента матрицы `m`.

```
Position getMinValuePos(Matrix m) {
    assert(m.nRows >= 1 && m.nCols >= 1);
    Position minPosition = {0, 0};

    for (int i = 0; i < m.nRows; i++)
        for (int j = 0; j < m.nCols; j++)
            if (m.values[i][j] <
                m.values[minPosition.rowIndex][minPosition.colIndex])
                minPosition = (Position) {i, j};

    return minPosition;
}
```

```
void testGetMinValuePos() {
    Matrix m = createMatrixFromArray((int[]) {19, 2, 3,
                                              4, 1, 6}, 2, 3);

    Position resultMinPos = getMinValuePos(m);

    assert(resultMinPos.rowIndex == 1 && resultMinPos.colIndex == 1);

    freeMemMatrix(m);
}
```

- b) `Position getMaxValuePos(Matrix m)` - возвращает позицию максимального элемента матрицы `m`.

```
Position getMaxValuePos(Matrix m) {
    assert(m.nRows >= 1 && m.nCols >= 1);
    Position maxPosition = {0, 0};

    for (int i = 0; i < m.nRows; i++)
        for (int j = 0; j < m.nCols; j++)
            if (m.values[i][j] >
                m.values[maxPosition.rowIndex][maxPosition.colIndex])
                maxPosition = (Position) {i, j};

    return maxPosition;
}
```

```
void testGetMaxValuePos() {
    Matrix m = createMatrixFromArray((int[]) {3, 2, 19,
                                              4, 1, 6}, 2, 3);

    Position resultMinPos = getMaxValuePos(m);

    assert(resultMinPos.rowIndex == 0 && resultMinPos.colIndex == 2);

    freeMemMatrix(m);
}
```

- 9) Дополните библиотеку функциями для тестирования:

- a) `Matrix createMatrixFromArray(const int *a, int nRows, int nCols)` - возвращает матрицу размера `nRows` на `nCols`, построенную из элементов массива `a`:

```
Matrix createMatrixFromArray(const int *a, int nRows,
                             int nCols) {
    Matrix newMatrix = getMemMatrix(nRows, nCols);

    int k = 0;
    for (int i = 0; i < nRows; i++)
        for (int j = 0; j < nCols; j++)
            newMatrix.values[i][j] = a[k++];

    return newMatrix;
}
```

```

void testCreateMatrixFromArray() {
    Matrix m = createMatrixFromArray(
        (int[]) {
            1, 1, 0,
            0, 0, 0,
            0, 0, 1,
            0, 0, 0,
            0, 1, 1,
        }, 5, 3);

    assert(m.values[0][0] && m.values[0][1] && !m.values[0][2] &&
        !m.values[1][0] && !m.values[1][1] && !m.values[1][2] &&
        !m.values[2][0] && !m.values[2][1] && m.values[2][2] &&
        !m.values[3][0] && !m.values[3][1] && !m.values[3][2] &&
        !m.values[4][0] && m.values[4][1] && m.values[4][2]);

    freeMemMatrix(m);
}

```

- b) `Matrix *createArrayOfMatrixFromArray(const int *values, size_t nMatrices, int nRows, int nCols)` - возвращает указатель на нулевую матрицу массива из `nMatrices` матриц, размещенных в динамической памяти, построенных из элементов массива `a`.

```

Matrix *createArrayOfMatrixFromArray(const int *values,
                                     size_t nMatrices, int nRows, int nCols) {
    Matrix *matrixArray = (Matrix *) malloc(sizeof(Matrix) * nMatrices);
    int k = 0;
    for (int i = 0; i < nMatrices; i++)
        matrixArray[i] = createMatrixFromArray(values + (k++) * nRows * nCols,
        nRows, nCols);
    return matrixArray;
}

```

```

void testCreateArrayOfMatrixFromArray() {
    Matrix *mArray = createArrayOfMatrixFromArray(
        (int[]) {
            1, 1, 0,
            0, 0, 0,
            0, 0, 1,
            0, 0, 0,
            0, 1, 1,
            1, 0, 1
        }, 3, 2, 3);

    assert(mArray[0].values[0][0] && mArray[0].values[0][1] &&
!mArray[0].values[0][2] &&
        !mArray[0].values[1][0] && !mArray[0].values[1][1] &&
!mArray[0].values[1][2]);

    assert(!mArray[1].values[0][0] && !mArray[1].values[0][1] &&
mArray[1].values[0][2] &&
        !mArray[1].values[1][0] && !mArray[1].values[1][1] &&
!mArray[1].values[1][2]);

    assert(!mArray[2].values[0][0] && mArray[2].values[0][1] &&
mArray[2].values[0][2] &&
        mArray[2].values[1][0] && !mArray[2].values[1][1] &&
mArray[2].values[1][2]);

    freeMemMatrices(mArray, 3);
}

```

Ссылка на репозиторий: <https://github.com/IAmProgrammist/programming-and-algorithmization-basics>

Скриншоты с коммитами (репозиторий вёлся с сентября, в него выкладывалось решение каждой из лаб):

Commits on Feb 19, 2023

Lab 15 done

 IAmProgrammist committed 3 minutes ago ✓

b61bc19



Lab15 code done

 IAmProgrammist committed 3 hours ago ✗

4b73483



Commits on Feb 12, 2023

20 badly done

 IAmProgrammist committed last week ✓

d2dd696



19 done

 IAmProgrammist committed last week ✓

b7155c7



Little cleanups, attempted to make 19

 IAmProgrammist committed last week ✓

547ee7b



Commits on Feb 11, 2023

lab10 mid-done

 IAmProgrammist committed last week ✓

0680d12



Commits on Dec 8, 2022

lab9 done

 IAmProgrammist committed on Dec 8, 2022 ✓

4bb5859



Commits on Dec 4, 2022

Linux compability, cleanup, added writing files, bug fixes


 IAmProgrammist committed on Dec 4, 2022 ✗

9356e3e



Commits on Dec 2, 2022


Lab8 mid-done, small project development started

 IAmProgrammist committed on Dec 2, 2022 ✗

be8ba87




Lab8 mid-done, small project development started

 IAmProgrammist committed on Dec 2, 2022

07a5bcd




Lab8 mid-done, small project development started

 IAmProgrammist committed on Dec 2, 2022

cc3bda8



Lab8 mid-done, small project development started

 IAmProgrammist committed on Dec 2, 2022

b3d1301



Commits on Nov 24, 2022


Lab7 comments fix

 IAmProgrammist committed on Nov 24, 2022 ✓

efcbae5



Lab6 comments fix

 IAmProgrammist committed on Nov 24, 2022 ✓

6268e62




Lab6 comments fix

 IAmProgrammist committed on Nov 24, 2022

16ea594



Lab6 comments fix

 IAmProgrammist committed on Nov 24, 2022

17dd136



Commits on Nov 23, 2022

Another lab7 fixes

 IAmProgrammist committed on Nov 23, 2022 ✓

4e45af8



Lab7 fixes

 IAmProgrammist committed on Nov 23, 2022 ✓

191a94d



Lab7 is done

 IAmProgrammist committed on Nov 23, 2022 ✓

4554f94



Commits on Nov 16, 2022

CMakeLists.txt fixed

IAmProgrammist committed on Nov 16, 2022 ✓



7a9a1d5



lab6 code review fixes

IAmProgrammist committed on Nov 16, 2022 ✗



10b8d08



Commits on Nov 15, 2022

lab6 done

IAmProgrammist committed on Nov 15, 2022 ✓



2651992



Mid done 6 lab

IAmProgrammist committed on Nov 15, 2022 ✓



4e68e10



Commits on Nov 8, 2022

Code review for lab 5 fixup

IAmProgrammist committed on Nov 8, 2022 ✓



36b2b4a



Commits on Nov 4, 2022

lab5 is done

IAmProgrammist committed on Nov 4, 2022 ✓



adf7f25



Commits on Oct 25, 2022

Code cleanup in lab4

IAmProgrammist committed on Oct 25, 2022 ✓



494c212



Commits on Oct 24, 2022

Tests fixed

IAmProgrammist committed on Oct 24, 2022 ✓



840f497



Tests fixed

IAmProgrammist committed on Oct 24, 2022 ✗



212c22a



Tests fixed

IAmProgrammist committed on Oct 24, 2022



4fcdf76



Fixed 15 task in lab 4

IAmProgrammist committed on Oct 24, 2022 ✗



5ebcf8e



Optimized 16 task in lab 4

IAmProgrammist committed on Oct 24, 2022 ✓



f656b5b



Some small fixes

IAmProgrammist committed on Oct 24, 2022 ✓



327bb91



Code cleanup and optimization after code review

IAmProgrammist committed on Oct 24, 2022 ✓



30ea64f



Code cleanup

IAmProgrammist committed on Oct 24, 2022 ✓



bc5b4b7



Commits on Oct 22, 2022

Added badge

IAmProgrammist committed on Oct 22, 2022 ✓



de09697



Newer

Older

Commits on Oct 22, 2022

Actions fix

IAmProgrammist committed on Oct 22, 2022 ✓

d671d1c

<>

Lab 4 done, testing system added, github actions to run tests added

IAmProgrammist committed on Oct 22, 2022 ✗

fc5f6da

<>

Commits on Oct 12, 2022

Lab3 done, committed without code-review.

IAmProgrammist committed on Oct 12, 2022

debtcf4

<>

Commits on Oct 5, 2022

Line breakers fixed and some performance improvements

IAmProgrammist committed on Oct 5, 2022

289323e

<>

Commits on Oct 4, 2022

A man who hates spaghetti code and having a big sharp axe is hunting ...

IAmProgrammist committed on Oct 4, 2022

32364e9

<>

Lab 2 done

IAmProgrammist committed on Oct 4, 2022

3e9c014

<>

Commits on Sep 28, 2022

Lab requirements improvements in 746A abd 1269A

IAmProgrammist committed on Sep 28, 2022

0fcd1f1

<>

Commits on Sep 23, 2022

Format 1064A

IAmProgrammist committed on Sep 23, 2022

cc2f60a

<>

Optimized 1064A and 1358A, formatted code (2 spaces -> 4 spaces), add...

IAmProgrammist committed on Sep 23, 2022

0814447

<>

Commits on Sep 21, 2022

Const values moved to define

IAmProgrammist committed on Sep 21, 2022

e3ac6e3

<>

Commits on Sep 18, 2022

Performance improvements

IAmProgrammist committed on Sep 18, 2022

f0bf579

<>

Reformat code and performance improvements

IAmProgrammist committed on Sep 18, 2022

1a15fe6

<>

Initial commit, added files for 1 lab

IAmProgrammist committed on Sep 18, 2022

0225bfe

<>

Newer

Older

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