

ГУАП
КАФЕДРА №43

ОТЧЁТ
ЗАЩИЩЕН С ОЦЕНКОЙ
ПРЕПОДАВАТЕЛЬ

К.Т.Н., доц..
должность, уч. степень, звание

подпись, дата

А.В. Туманова
инициалы, фамилия

ОТЧЁТ О ЛАБОРАТОРНОЙ РАБОТЕ №6

ОБРАБОТКА ЧИСЛОВЫХ МАТРИЦ

по дисциплине: ОСНОВЫ ПРОГРАММИРОВАНИЯ

РАБОТУ ВЫПОЛНИЛ

СТУДЕНТ ГР. Z7431

 01.12.2018
подпись, дата

 М.Д.Семочкин
инициалы, фамилия

Санкт-Петербург
2018

1. Цель работы

Целью работы является изучение структуры данных двумерный массив.

2. Задание

Согласно варианту №14,

Осуществить циклический сдвиг элементов квадратной матрицы размерности $M \times N$ вправо на k элементов таким образом: элементов 1-й строки сдвигаются в последний столбец сверху вниз, из него – в последнюю строку справа налево, из неё – в первый столбец снизу вверх, из него – в первую строку; для остальных элементов аналогично.

При написании программ можно использовать как динамические, так и нединамические массивы. Размерность последних задаётся именованной константой.

Примечание: массивы, созданные с помощью классов (таких, как например, `array` или `vector` из библиотеки STL) использовать запрещается.

Примечание: если использовались операторы динамического выделения памяти, то следует вставить дополнительный код, обнаруживающий утечки памяти. См. раздел «Как обнаружить утечки памяти?». Все утечки памяти должны быть устранены.

3. Описание созданных функций

Для реализации задания нам потребуются следующие функции:

Имя: `fillMatrixWithRandomNumbers`

Назначение: Заполняет матрицу случайными числами от `min` до `max`

Входные данные:

- `matrix[][MATRIX_SIZE]` – двумерный массив
- `min` – минимальное значение случайного числа, которым может быть заполнен элемент матрицы
- `max` – максимальное значение случайного числа, которым может быть заполнен элемент матрицы

Выходные данные: нет

Побочный эффект: переданный в качестве аргумента двумерный массив заполняется случайными числами от `min` до `max`

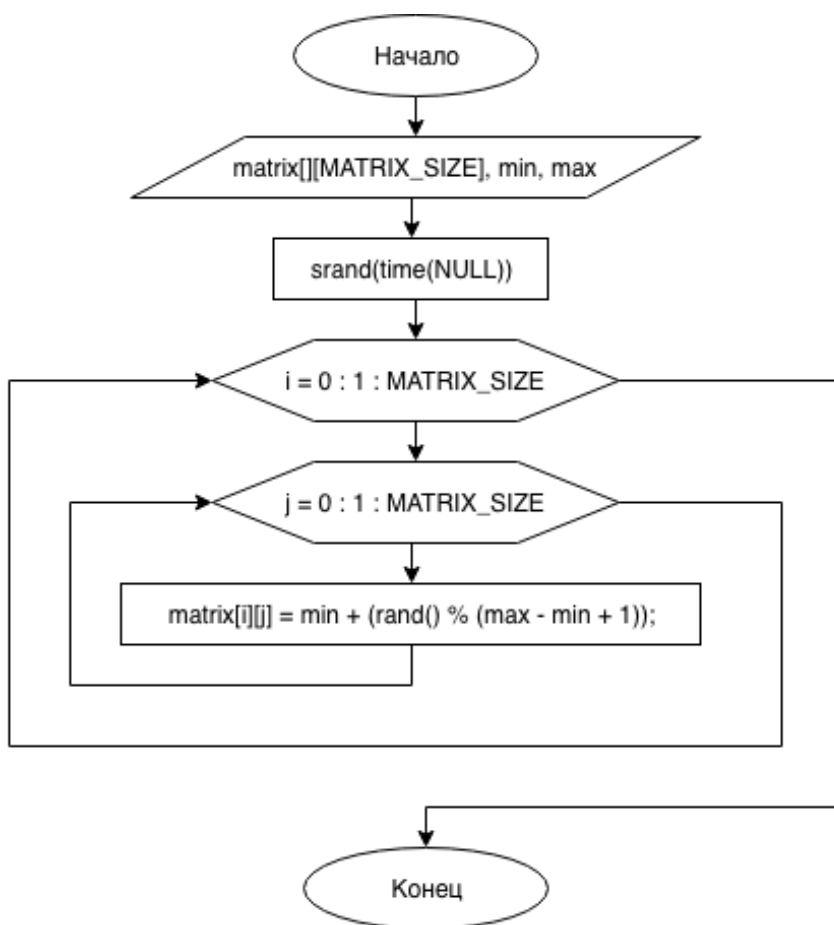
Тестовые данные:

matrix[][MATRIX_SIZE] до выполнения функции	min	max	matrix[][MATRIX_SIZE] после выполнения функции
<пустая матрица>	10	99	<заполненная случайными числами от min до max матрица>

Прототип: void fillMatrixWithRandomNumbers(int matrix[][MATRIX_SIZE], int min, int max)

Алгоритм:

- псевдокод
для каждого элемента двумерного массива
присвоить текущему элементу случайное значение
- блок-схема



Имя: printMatrix

Назначение: Вывести матрицу на экран

Входные данные:

- matrix[][MATRIX_SIZE] – двумерный массив

Выходные данные: нет

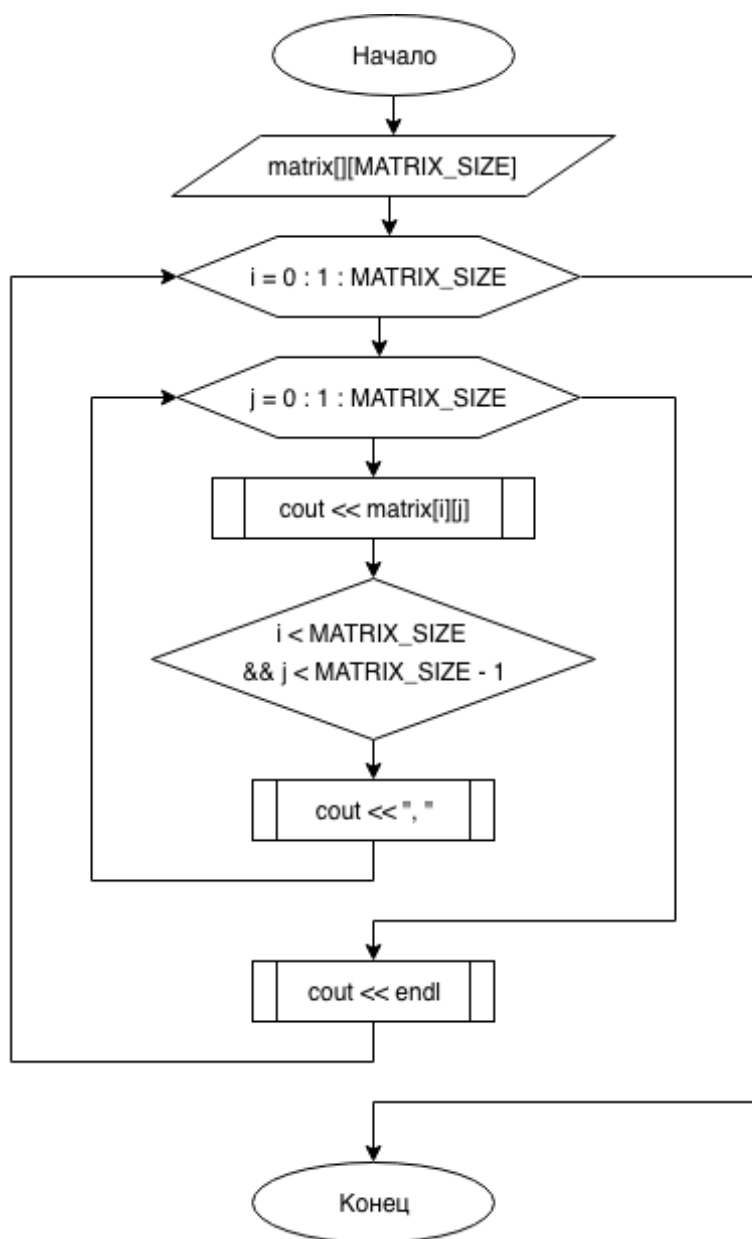
Побочный эффект: на экран выводятся все элементы матрицы

Прототип: void printMatrix(int matrix[][MATRIX_SIZE])

Алгоритм:

- псевдокод
для каждого элемента двумерного массива
 вывести текущий элемент на экран
 если это не последний элемент
 вывести “,”

- блок-схема



Имя: shiftMatrix

Назначение: Выполнить циклический сдвиг элементов в матрице как указано в задании

Входные данные:

- matrix[][MATRIX_SIZE] – двумерный массив
- nShifts – количество сдвигов

Выходные данные: нет

Побочный эффект: в исходном массиве происходит циклический сдвиг элементов

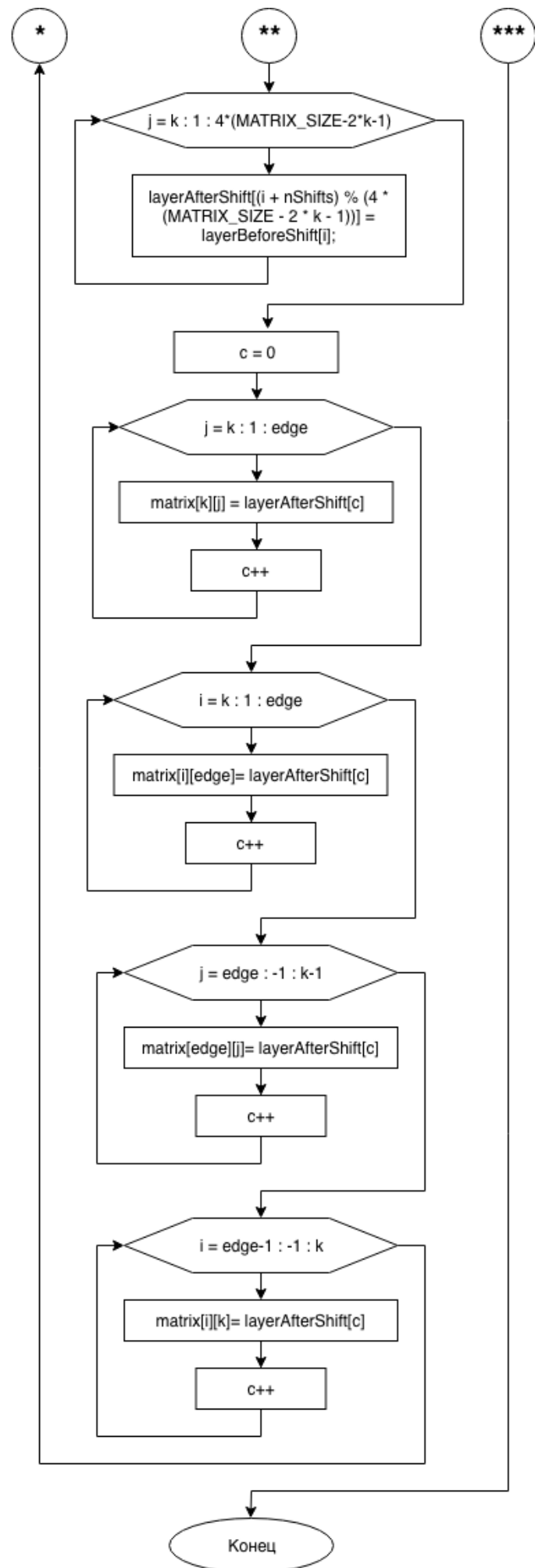
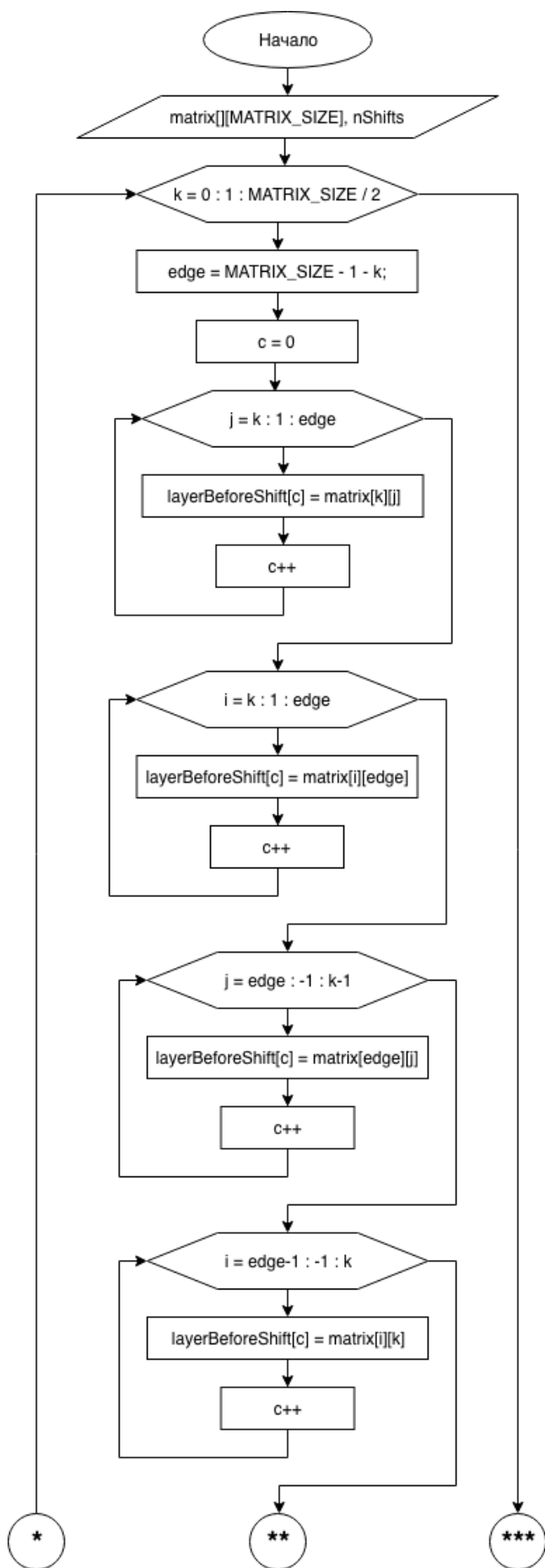
Тестовые данные:

массив	nShifts	результат
[41, 66, 14, 44, 46,] [77, 20, 71, 84, 64,] [72, 69, 94, 82, 19,] [76, 85, 48, 48, 64,] [64, 42, 43, 33, 45]	1	[77, 41, 66, 14, 44,] [72, 69, 20, 71, 46,] [76, 85, 94, 84, 64,] [64, 48, 48, 82, 19,] [42, 43, 33, 45, 64]

Прототип: void shiftMatrix(int matrix[][MATRIX_SIZE], int nShifts)

Алгоритм:

- псевдокод
 - для каждого слоя матрицы от внешнего до внутреннего
 - записать все элементы слоя в вспомогательный одномерный массив
 - сдвинуть элементы во вспомогательном массиве, записав их во второй вспомогательный массив
 - записать второй вспомогательный массив обратно в матрицу
- блок-схема



4. Листинг программы

```
#include <iostream>
#include <stdlib.h>      /* srand, rand */
#include <time.h>        /* time */
using namespace std;

const int MATRIX_SIZE = 5; // Размерность матрицы

void fillMatrixWithRandomNumbers(int matrix[][MATRIX_SIZE], int min, int max) {
    // initialize random seed
    srand(time(NULL));

    // заполняет матрицу случайными числами от min до max
    for(int i = 0; i < MATRIX_SIZE; i++) {
        for(int j = 0; j < MATRIX_SIZE; j++) {
            matrix[i][j] = min + (rand() % (max - min + 1));
        }
    }
}

void printMatrix(int matrix[][MATRIX_SIZE]) {
    // печатает матрицу
    for(int i = 0; i < MATRIX_SIZE; i++) {
        for(int j = 0; j < MATRIX_SIZE; j++) {
            cout << matrix[i][j];
            if (i < MATRIX_SIZE && j < MATRIX_SIZE - 1) {
                cout << ", ";
            }
        }
        cout << endl;
    }
}

void shiftMatrix(int matrix[][MATRIX_SIZE], int nShifts) {
    int i, j, c, edge;
    int layerBeforeShift[4 * (MATRIX_SIZE - 1)];
    int layerAfterShift[4 * (MATRIX_SIZE - 1)];

    // Для каждого слоя матрицы,
    for(int k = 0; k < MATRIX_SIZE / 2; k++) {
        edge = MATRIX_SIZE - 1 - k;

        // записать слой в layerBeforeShift
        c = 0;
        for(j = k; j < edge; j++) {
            layerBeforeShift[c] = matrix[k][j];
            c++;
        }
        for(i = k; i < edge; i++) {
            layerBeforeShift[c] = matrix[i][edge];
            c++;
        }
        for(j = edge; j >= k; j--) {
            layerBeforeShift[c] = matrix[edge][j];
            c++;
        }
    }
}
```



```

        for(i = edge - 1; i > k; i--) {
            layerBeforeShift[c] = matrix[i][k];
            c++;
        }

        // переставить элементы и записать их в layerAfterShift
        for(i = 0; i < (4 * (MATRIX_SIZE - 2 * k - 1)); i++)
            layerAfterShift[(i + nShifts) % (4 * (MATRIX_SIZE - 2 * k - 1))] =
layerBeforeShift[i];

        // записать layerAfterShift обратно в матрицу
        c = 0;
        for(j = k; j < edge; j++) {
            matrix[k][j] = layerAfterShift[c];
            c++;
        }
        for(i = k; i < edge; i++) {
            matrix[i][edge] = layerAfterShift[c];
            c++;
        }
        for(j = edge; j >= k; j--) {
            matrix[edge][j] = layerAfterShift[c];
            c++;
        }
        for(i = edge - 1; i > k; i--) {
            matrix[i][k] = layerAfterShift[c];
            c++;
        }
    }
}

int main() {

    setlocale(LC_ALL, "russian");
    int matrix[MATRIX_SIZE][MATRIX_SIZE], nShifts;

    fillMatrixWithRandomNumbers(matrix, 10, 99);
    cout << "Исходная матрица:" << endl;
    printMatrix(matrix);

    cout << "Введите количество элементов, на которое " << endl;
    cout << "будет проведен циклический сдвиг матрицы: ";
    cin >> nShifts;

    shiftMatrix(matrix, nShifts);

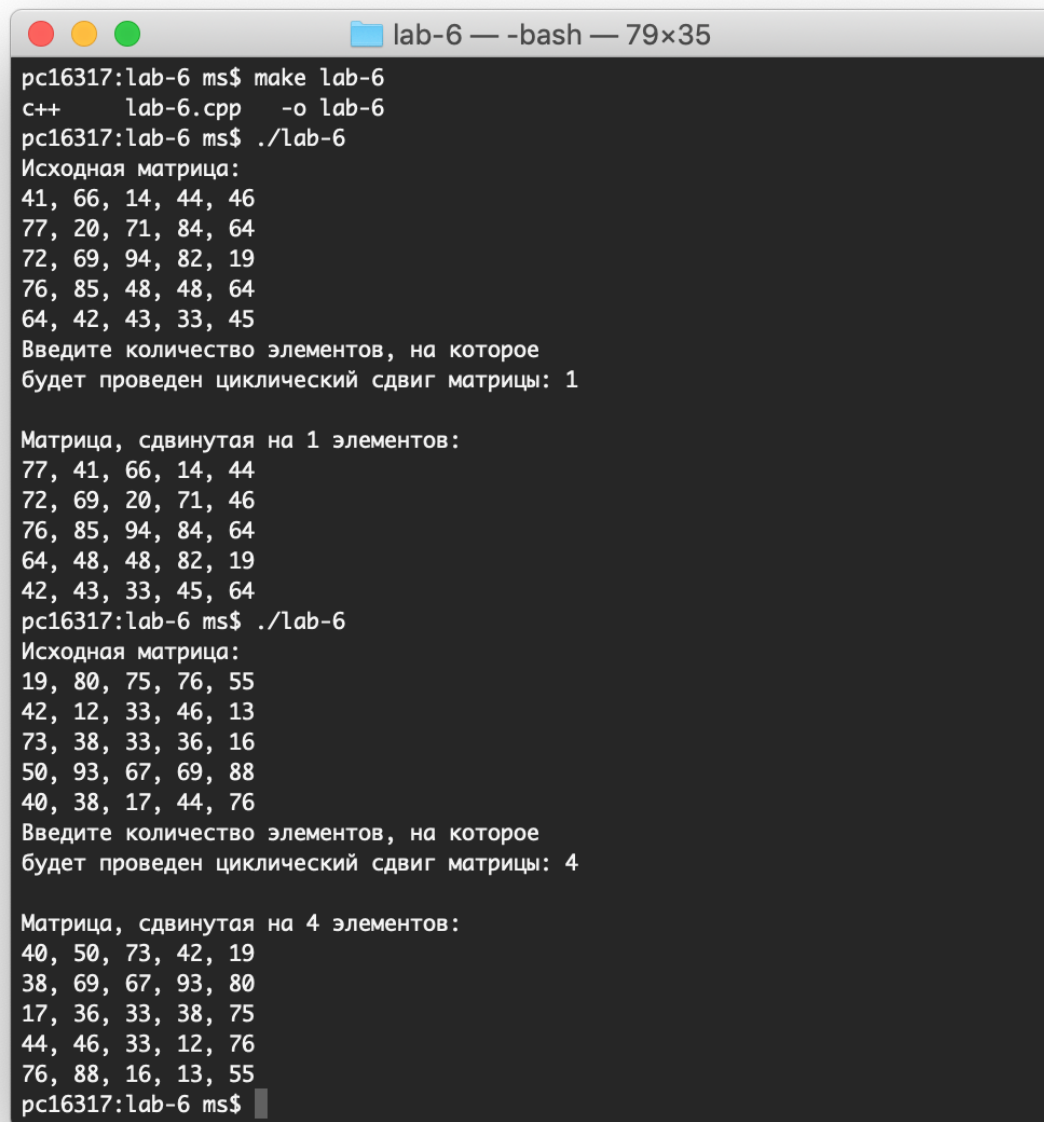
    cout << endl << "Матрица, сдвинутая на ";
    cout << nShifts << " элементов:" << endl;
    printMatrix(matrix);

    return 0;
}

```

5. Пример выполнения программы

Ниже показан пример выполнения программы.



```
pc16317:lab-6 ms$ make lab-6
c++      lab-6.cpp      -o lab-6
pc16317:lab-6 ms$ ./lab-6
Исходная матрица:
41, 66, 14, 44, 46
77, 20, 71, 84, 64
72, 69, 94, 82, 19
76, 85, 48, 48, 64
64, 42, 43, 33, 45
Введите количество элементов, на которое
будет проведен циклический сдвиг матрицы: 1

Матрица, сдвинутая на 1 элементов:
77, 41, 66, 14, 44
72, 69, 20, 71, 46
76, 85, 94, 84, 64
64, 48, 48, 82, 19
42, 43, 33, 45, 64
pc16317:lab-6 ms$ ./lab-6
Исходная матрица:
19, 80, 75, 76, 55
42, 12, 33, 46, 13
73, 38, 33, 36, 16
50, 93, 67, 69, 88
40, 38, 17, 44, 76
Введите количество элементов, на которое
будет проведен циклический сдвиг матрицы: 4

Матрица, сдвинутая на 4 элементов:
40, 50, 73, 42, 19
38, 69, 67, 93, 80
17, 36, 33, 38, 75
44, 46, 33, 12, 76
76, 88, 16, 13, 55
pc16317:lab-6 ms$
```

Видно, что результаты расчётов совпадают с тестовыми данными.

6. Анализ результатов и выводы

К достоинствам программы можно отнести:

- Программа выполняет поставленную задачу и работает без ошибок (для корректных тестовых данных).
- Задание реализовано в виде отдельных функций.

Из недостатков можно отметить:

- Не производится проверка входных данных.