Лабораторная №8. Матрицы.

Темы, рассматриваемые в лабораторной работе:

Алгоритмы обработки матриц. Особенности передачи матриц-параметров.

Статические матрицы в языке C++ могут быть объявлены следующим образом:

```
int matr[5][5];
```

Так же, как и в случае со статическими одномерными массивами, размерами статической матрицы могут быть только константы или выражения, вычисляемые на этапе компиляции программы

```
const int MAX_ROWS = 5;
const int MAX_COLS = 4;

int matr[MAX_ROWS][MAX_COLS]; // создали матрицу из 5 строк и 4 столбцов
```

К элементам матриц можно обращаться через индексы в квадратных скобках [строка][столбец]

```
// например, заполним элемент matr[2][2] и выведем его
matr[2][2] = 2;
std::cout << matr[2][2];
```

Матрицу можно сразу заполнить при объявлении:

```
const int MAX_ROWS = 3;
const int MAX_COLS = 3;

int matr[MAX_ROWS][MAX_COLS] = {
    {1, 2, 3},
    {4, 5, 6},
    {7, 8, 9}
};
```

Рассмотрим передачу матрицы как аргумента функции. Пример функции, печатающей матрицу

```
const int MAX_ROWS = 3;
const int MAX_COLS = 3;

void printMatrix(int matr[MAX_ROWS][MAX_COLS], int rows, int cols) {
    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < cols; j++) {
            // Форматированный вывод для выравнивания столбцов
            std::cout << std::setw(std::cout.precision() * 2 + 1);
            std::cout << matr[i][j] << "\t";
        }
        std::cout << std::endl;
    }
}
```

В отличие от одномерных массивов для матриц нельзя использовать в качестве параметра функции указатель, можно только опустить размерность по первому измерению:

```
void printMatrix(int matr[][MAX_COLS], int rows, int cols)
```

Вместо громоздкого описания типа матрицы с размерами можно определить пользовательский тип:

```
// опишем тип для матрицы с элементами double размера максимум
// MAX_ROWS*MAX_COLS
typedef double matrix[MAX_ROWS][MAX_COLS];
void printMatrix(matrix matr, int rows, int cols) { ... }
```

Матрица хранится в памяти как одномерный массив, в котором строки размещаются последовательно друг за другом. Каждая строка - это статический одномерный массив. Например, используя функцию поиска максимума в одномерном массиве из прошлых лабораторных (arr_max), можно вычислить максимальный элемент первой строки матрицы:

```
int matr[MAX_ROWS][MAX_COLS];
// ... заполнение матрицы
std::cout << arr_max(matr[0], MAX_COLS);
```

Размеры статических матриц нельзя изменять в процессе выполнения программы, можно только менять их элементы. Если манипуляции с матрицей требуют удаления (или добавления) строк/столбцов, необходимо менять местами элементы исходной матрицы, а затем пересчитывать ее размеры (они должны передаваться по ссылке). Пример описания и использования функции, удаляющей первую строку матрицы:

```
// саму матрицу не требуется передавать по ссылке,
// даже если ее элементы будем менять, а не только считывать
void deleteFirstRow(matrix matr, int& rows, int cols) {
    // меняем элементы matr
    for (int i=0, j; i<rows-1; ++i)
     for (j=0; j<cols; ++j)
      matr[i][j] = matr[i+1][j];
//внутренний цикл можно заменить на вызов функции копирования массива
    rows--; // уменьшаем счетчик строк
}

int main() {
    // matrix - тип матрицы большого размера (с запасом, например 10*10)
    matrix matr = {
        {1, 2, 3},
        {4, 5, 6},
        {7, 8, 9}
    };
    int m = 3, n = 3; // размеры самой матрицы
    deleteFirstRow(matr, m, n);

    printMatrix(matr, m, n);
    // m должно быть 2, n должно быть 3, вывод - матрица без первой строки
}
```

При тестировании функций, меняющих значения элементов матриц, следует действовать так же, как и для одномерных массивов. Необходимо задать матрицы до и после выполнения теста, а затем в двойном цикле сравнивать элементы с помощью **Assert::AreEqual**). Счетчики новых размеров матриц также необходимо проверять с помощью **Assert::AreEqual**

Задания:

Решение каждой задачи должно сопровождаться представительным набором тестов

1. Реализовать функцию для вычисления количества строк матрицы, содержащих нули.
2. Реализовать функцию, транспонирующую квадратную матрицу.
3. Реализовать функцию проверки квадратной матрицы на симметричность относительно главной диагонали
4. Реализовать функцию удаления строки матрицы с заданным номером.
5. Реализовать функцию для добавления в матрицу строки (массив значений строки и позиция вставки K передаются как параметры функции вместе с самой матрицей и ее исходными размерами)

Задания повышенной сложности:

6. Реализовать функцию для вычеркивания из матрицы строки с индексом M и столбца с индексом N (M, N - параметры функции)
7. Реализовать функцию вычисления определителя квадратной матрицы