Выделить память под многомерный массив, заполнить его произвольным образом, передать обрабатывающей функции и получить её результат (входная матрица остаётся неизменной, на выходе должен быть новый массив или число).

Вариант	Массив, функция
1	Вычислить след матрицы
2	В наличии прямоугольная плоская решётка с шагом 1. В узлах решётки находятся точечные массы . Считая матрицу *M* наперёд заданной, вычислить координаты центра масс решётки
3	Дана матрица. Получить вектор, в котором содержатся максимальные элементы из каждой строки матрицы
4	Дана матрица. Считая каждый столбец матрицы вектором, получить новую матрицу, в которой столбцы отсортированы по норме
5	Построить из двух одинаковых треугольников Паскаля произвольного размера «ромб», симметричный относительно основания треугольника
6	Реализовать операцию умножения матрицы на вектор
7	Удалить из матрицы все нулевые строки и столбцы
8	Написать функцию, возвращающую число столбцов матрицы, в которых нули чередуются с ненулевыми элементами.
9	Дана матрица. Получить вектор, каждая компонента которого содержит сумму элементов из соответствующей строки
10	Проверить, является ли матрица нильпотентной (можно ограничиться некоторой наперёд заданной степенью)
11	Дана матрица. Получить вектор, каждая компонента которого содержит среднее геометрическое элементов из соответствующего столбца
12	Проверить, является ли матрица идемпотентной
13	Вернуть сумму всех чётных элементов матрицы
14	Дана матрица. Получить вектор, каждая компонента которого содержит среднее элементов из соответствующей строки
15	Привести матрицу к верхнетреугольному виду по методу Гаусса
16	Вычислить определитель матрицы
17	Получить транспонированную матрицу
18	В наличии решётка в виде параллелепипеда. В узлах решётки находятся точечные массы (с шагом 1 по *x, y, z*). Считая массив *M* наперёд известным, вычислить координаты центра масс решётки
19	Даны матрица Грамма и два вектора. Найти скалярное произведение векторов
20	Удалить все строки матрицы, в которых есть нулевые элементы
21	Применить ко всем элементам матрицы некоторую функцию, переданную по указателю
22	Отсортировать элементы во всех строках матрицы
23	Коэффициенты входной матрицы считать высотами некоторого участка поверхности. Получить на выходе матрицу, в которой отмечены локальные минимумы карты.
24	Найти все вхождения некоторого элемента element в матрице. Вернуть массив индексов.
25	Дана матрица. Получить вектор, в котором компонента нулевая, если соответствующая строка матрицы не отсортирована, и ненулевая, если все элементы строки расположены в порядке возрастания или убывания
26	Удалить из матрицы все строки и столбцы, содержащие элементы, большие некоторого параметра border
27	Удалить все строки матрицы, в которых есть некоторый наперёд заданный

	элемент
28	Сгладить все элементы матрицы - заменить их средним всех соседних.
29	Написать функцию, возвращающую число столбцов матрицы, не содержащих ни одного нуля.
30	Написать функцию, переворачивающую матрицу слева направо (вокруг воображаемой вертикальной оси).
31	В строках матрицы, в которых есть отрицательные элементы произвести циклический сдвиг вправо на число этих элементов (для каждой строки оно своё).
32	Найти все седловые элементы матрицы. Вернуть двумерный массив их координат.
33	Для каждого столбца матрицы посчитать: среднее, медиану и среднеквадратическое отклонение.
34	Дана матрица. Считая каждый столбец матрицы вектором, получить все вектора, у которых норма меньше средней.
35	Дана квадратная матрица. Свинуть циклически все элементы на диагоналях, параллельных главной, на k (входной параметр).
36	Подсчитать число элементов, которые отличаются от среднего значения более чем на два среднеквадратических отклонения.

Пример кода:

```
#include <stdlib.h>

/**
 * Выделение памяти под треугольник Паскаля
 * @param matrix  Массив строк треугольника Паскаля
 *
 * @return Незаполненный массив под треугольник
 **/
double** alloc_triangle(int rows)
{
    int i;
        //выделение памяти под указатель - сначала как массив указателей
        double** matrix = (double**)calloc(rows, sizeof(double*));

        //каждая строка - тоже указатель - но уже типа int
        for(i=0;i<rows;i++)
        {
            // заметьте, что для каждой следующей строки выделяется больше памяти,
            // таким образом, получается не прямоугольная матрица, а "треугольник"
                matrix[i] = (double*)calloc(i+1, sizeof(double));
        }

        return matrix;
}

/**
 * Освобождение памяти для треугольника Паскаля
 * @param matrix  Массив строк треугольника Паскаля
```

```
 * @param rows    Число строк треугольника Паскаля
 **/
void free_triangle(double** matrix, int rows)
{
    int i;
    // Освобождение памяти для двумерного массива
        for(i=0;i<rows;i++)
        {
                free(matrix[i]);
        }
        free(matrix);
}


/**
 * Заполнение треугольника Паскаля
 * @param matrix  Массив строк треугольника Паскаля
 * @param rows    Число строк треугольника Паскаля
 **/
void input_triangle(double** matrix, int rows)
{
        int i,j;
        for(i=0;i<rows;i++)
        {
                for(j=0;j < i + 1;j++)
                {
                        if(j==0 || j==i)
                                matrix[i][j] = 1;
                        else
                                matrix[i][j] = matrix[i-1][j-1]+matrix[i-1][j];
                }
        }
}

/**
 * Возводит элементы треугольника Паскаля в квадрат
 * @param matrix  Массив строк треугольника Паскаля
 * @param rows    Число строк треугольника Паскаля
 *
 * @return Новый треугольник с элементами в квадрате
 **/
double** square_triangle(double** matrix, int rows)
{
        // выделяем память под новый массив
        double** new_matrix = alloc_triangle(rows);

        int i,j;
        for(i=0;i<rows;i++)
        {
                for(j=0;j < i + 1;j++)
                {
                        new_matrix[i][j] = matrix[i][j]*matrix[i][j];
                }
        }
```

```
        return new_matrix;
}

/**
 * Печать треугольника Паскаля в консоль
 * @param matrix  Массив строк треугольника Паскаля
 * @param rows    Число строк треугольника Паскаля
 **/
void output_triangle(double** matrix, int rows)
{
        int i,j;
        for(i=0;i<rows;i++)
        {
                printf("\n");
                int spaces = (rows - i - 1);
                for(j=0;j<spaces;j++)
                        printf("\t");

                for(j=0;j < i + 1;j++)
                        printf("%lf\t", matrix[i][j]);

                for(j=0;j<spaces;j++)
                        printf("\t");

                printf("\n");
        }
}

int main()
{
        //объявление указателя для двумерного массива
        double** matr;
        int rows;
        printf("\nEnter the height of the Pascal triangle\n");
        scanf("%d", &rows);
        int i,j;

        // выделяем память
        matr = alloc_triangle(rows);
    // заполняем
        input_triangle(matr, rows);
        // выводим
        output_triangle(matr, rows);
        // изменяем
        double** square_matr = square_triangle(matr, rows);
        // выводим
        output_triangle(square_matr, rows);
        // освобождаем память
        free_triangle(matr, rows);
        free_triangle(square_matr, rows);

        system("pause");

        return 0;
}
```