ГУАП

КАФЕДРА № 53

ОТЧЕТ   
ЗАЩИЩЕН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| к.т.н, доц.. |  |  |  | А.В. Туманова |
| должность, уч. степень, звание |  | подпись, дата |  | инициалы, фамилия |

|  |
| --- |
| ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ |
| Обработка числовых матриц |
| по дисциплине: ОСНОВЫ ПРОГРАММИРОВАНИЯ |
|  |
|  |

РАБОТУ ВЫПОЛНИЛ

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| СТУДЕНТ ГР. № | 4131 |  | 16.03.2022 |  | Д.А. Кузнецов |
|  |  |  | подпись, дата |  | инициалы, фамилия |

Санкт-Петербург 2022

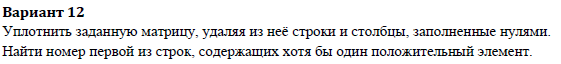
1. **Цель работы**

Целью работы является изучение структуры данных двумерный массив.

1. **Задание**

Задания на лабораторную работу приводятся в каждом варианте. При написании программ можно использовать как динамические, так и нединамические массивы. Размерность последних задаѐтся 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.



1. **Описание созданных функций**

**Имя:** main

**Назначение**: ввод/вывод матрицы и подсчёт промежуточных значений

**Входные данные:** int: k – счётчик

i – счётчик

m – счётчик для преобразованного массива

n – счётчик для преобразованного массива

double: l – элемент массива

sumstr – количество нулевых строк

sumcol – количество нулевых столбцов

double\*\*: array – двумерный массив

**Побочный эффект:** отсутствует

**Тестовые данные:** m = 5

n = 5

array = [ [1; 2; 0; 3; 4];

[5; 4; 0; 3; 2];

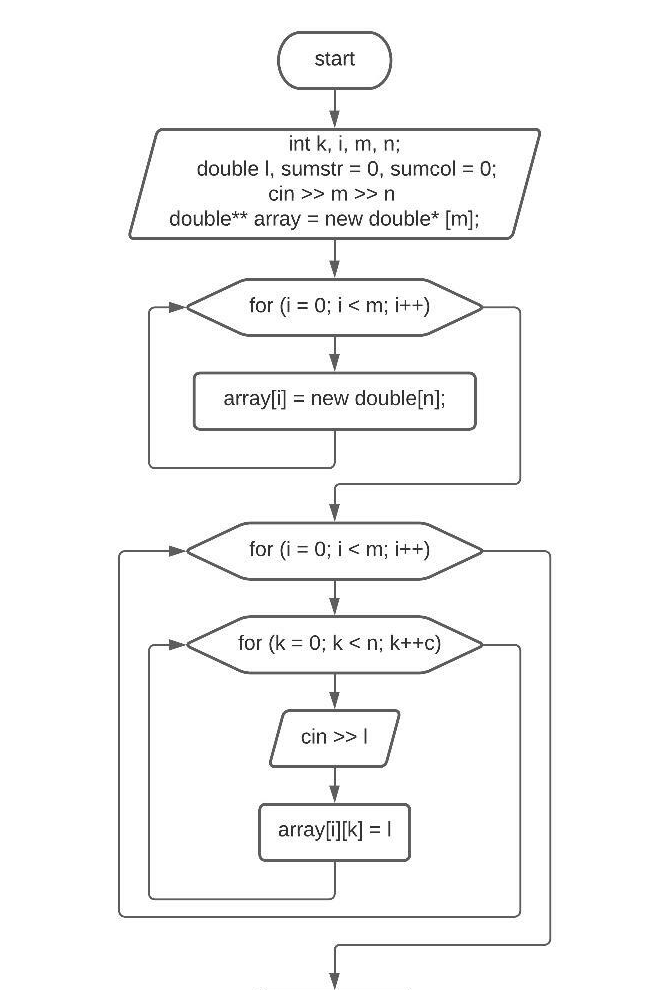
[0; 0; 0; 0; 0];

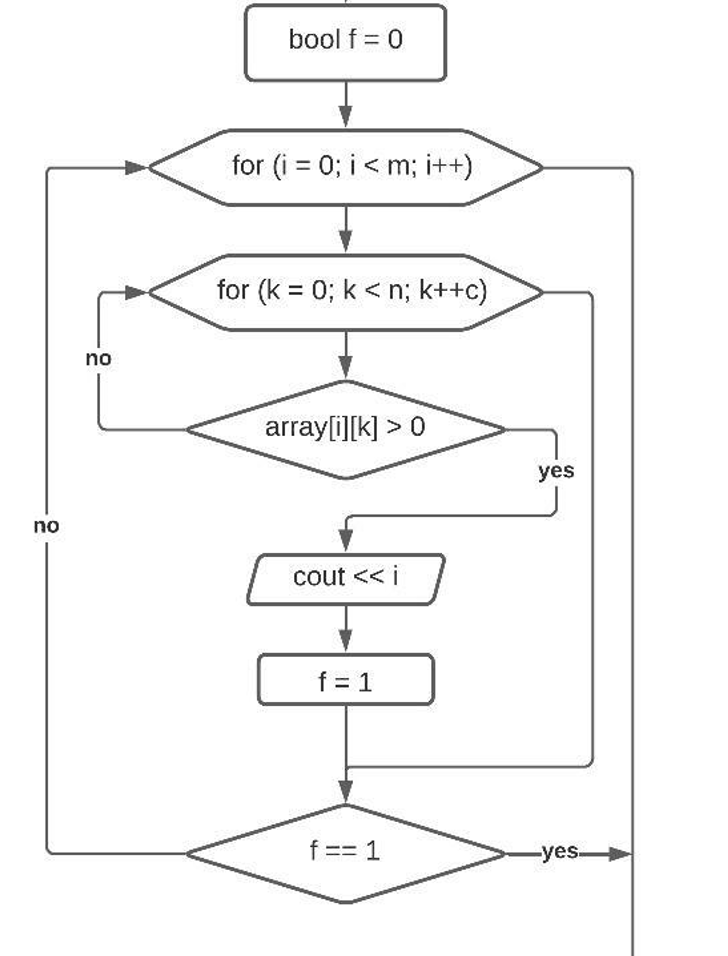
[1; 2; 0; 3; 4];

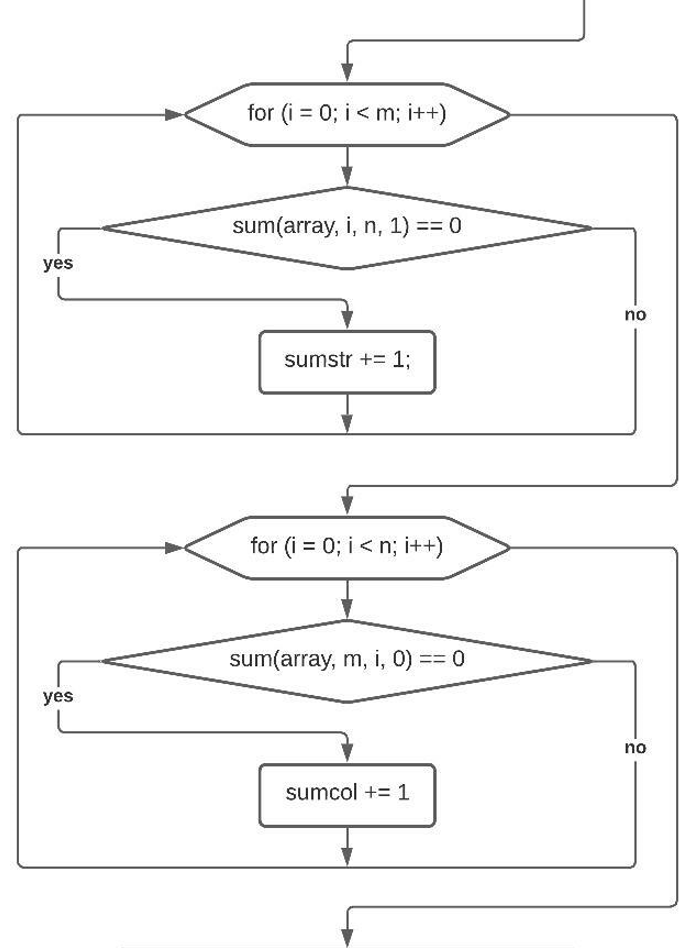
[5; 4; 0; 3; 2];]

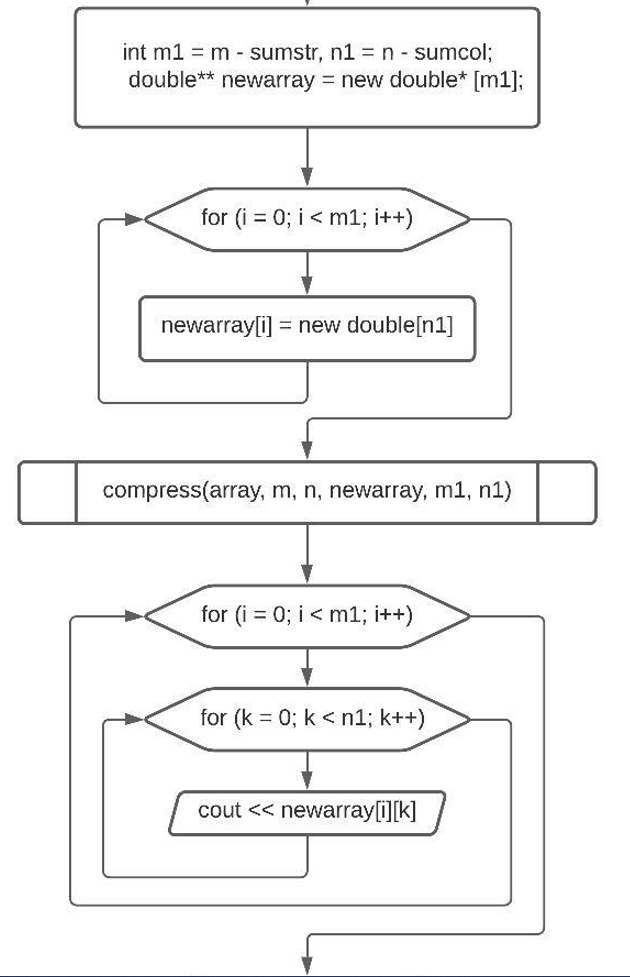
**Прототип:** int main()

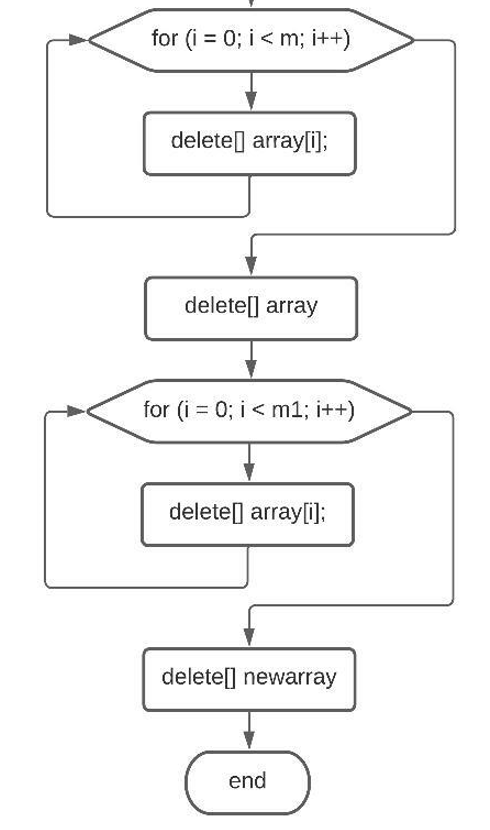
**Блок-схема:**

****

****

****

****

****

1. **Текст программы:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99  100  101  102  103  104  105  106  107  108  109  110  111  112  113  114  115  116 | // Для обнаружения утечек памяти  #define CRTDBG\_MAP\_ALLOC  #include <cstdlib>  #include <crtdbg.h>  #ifdef \_DEBUG  #ifndef DBG\_NEW  #define DBG\_NEW new ( \_NORMAL\_BLOCK , \_\_FILE\_\_ , \_\_LINE\_\_ )  #define NEW\_DBG\_NEW  #endif  #endif  // Основной код  #include <iostream>  #include "functions.h"  **int** **main**() {    setlocale(LC\_ALL, "rus");  **int** k, i, m, n;  **double** l, sumstr = **0**, sumcol = **0**;  std::cout << "Введите размерность матрицы (кол-во строк, столбцов): ";  std::cin >> m >> n;  // Создание двумерной матрицы  **double**\*\* array = **new** **double**\* [m];  **for** (i = **0**; i < m; i++) {  array[i] = **new** **double**[n];  }  // Заполнение двумерной матрицы  std::cout << "Введите свою матрицу:" << std::endl;  **for** (i = **0**; i < m; i++) {  **for** (k = **0**; k < n; k++) {  std::cin >> l;  array[i][k] = l;  }  std::cout << std::endl;  }  // Нахождение первой строки, содержащей хотя бы один положительный  элемент  **bool** f = **0**;  **for** (i = **0**; i < m; i++) {  **for** (k = **0**; k < n; k++) {  **if** (array[i][k] > **0**) {  std::cout << "Номер первой строки, содержащей  положительный элемент: " << i << std::endl;  f = **1**;  **break**;  }  }  **if** (f == **1**) { **break**; }  }  //Считаем количество нулевых строк  **for** (i = **0**; i < m; i++) {  **if** (sum(array, i, n, **1**) == **0**) {  sumstr += **1**;  }  }  //Считаем количство нулевых столбцов  **for** (i = **0**; i < n; i++) {  **if** (sum(array, m, i, **0**) == **0**) {  sumcol += **1**;  }  }  //Выделение памяти для сжатой матрицы (c учетом удаленных элементов)  **int** m1 = m - sumstr, n1 = n - sumcol;  **double**\*\* newarray = **new** **double**\* [m1];  **for** (i = **0**; i < m1; i++) {  newarray[i] = **new** **double**[n1];  }  // Сжатие матрицы  compress(array, m, n, newarray, m1, n1);  //Вывод сжатой матрицы  **for** (i = **0**; i < m1; i++) {  **for** (k = **0**; k < n1; k++) {  std::cout << newarray[i][k] << " ";  }  std::cout << std::endl;  }  //Очистка памяти    //Очистка старого массива  **for** (i = **0**; i < m; i++) {  **delete**[] array[i];  }  **delete**[] array;  //Очистка нового массива  **for** (i = **0**; i < m1; i++) {  **delete**[] newarray[i];  }  **delete**[] newarray;  // Для обнаружения утечек памяти  \_CrtSetReportMode(\_CRT\_WARN, \_CRTDBG\_MODE\_FILE);  \_CrtSetReportFile(\_CRT\_WARN, \_CRTDBG\_FILE\_STDOUT);  \_CrtSetReportMode(\_CRT\_ERROR, \_CRTDBG\_MODE\_FILE);  \_CrtSetReportFile(\_CRT\_ERROR, \_CRTDBG\_FILE\_STDOUT);  \_CrtSetReportMode(\_CRT\_ASSERT, \_CRTDBG\_MODE\_FILE);  \_CrtSetReportFile(\_CRT\_ASSERT, \_CRTDBG\_FILE\_STDOUT);  \_CrtDumpMemoryLeaks();  } | |  |  |  |
|  |  |

**Имя:** sum

**Назначение**: подсчёт суммы строки/столбца

**Входные данные**: double\*\*: array – входной массив

int: m – количество строк массива

n – количество столбцов массива

i – счётчик

bool: b – флаг

double: sum – сумма элементов строки/столбца

**Побочный эффект:** отсутствует

**Тестовые данные:** array = [ [1; 2; 0; 3; 4];

[5; 4; 0; 3; 2];

[0; 0; 0; 0; 0];

[1; 2; 0; 3; 4];

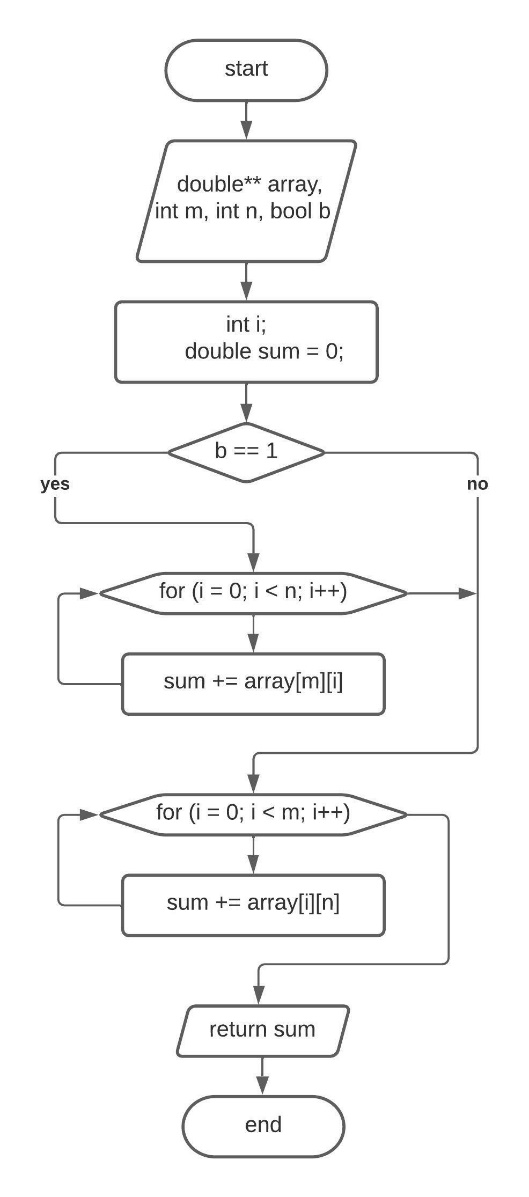
[5; 4; 0; 3; 2];]

m = 5;

n = 5;

b = 1;

**Прототип:** double sum(double\*\* array, int m, int n, bool b)

**Блок-схема:**

1. **Текст программы:**

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16 | **double** **sum**(**double**\*\* array, **int** m, **int** n, **bool** b) {  **int** i;  **double** sum = **0**;  **if** (b) {  **for** (i = **0**; i < n; i++) {  sum += array[m][i];  }  }  **else** {  **for** (i = **0**; i < m; i++) {  sum += array[i][n];  }  }  **return** sum;  } |

**Имя:** compress

**Назначение**: сжатие матрицы посредством удаление нулевых столбцов и строк

**Входные данные**: double\*\*: array – входной массив

newarray – пустой массив для сохранения преобразованного

int: m – количество строк массива

n – количество столбцов массива

m1 – количество строк нового массива

n1 – количество столбцов нового массива

i – счётчик

k – счётчик

i1 – счётчик для нового массива

k1 – счётчик для нового массива

**Побочный эффект:** отсутствует

**Тестовые данные:** array = [ [1; 2; 0; 3; 4];

[5; 4; 0; 3; 2];

[0; 0; 0; 0; 0];

[1; 2; 0; 3; 4];

[5; 4; 0; 3; 2]]

newarray = [ [0; 0; 0; 0];

[0; 0; 0; 0];

[0; 0; 0; 0];

[0; 0; 0; 0]]

m = 5;

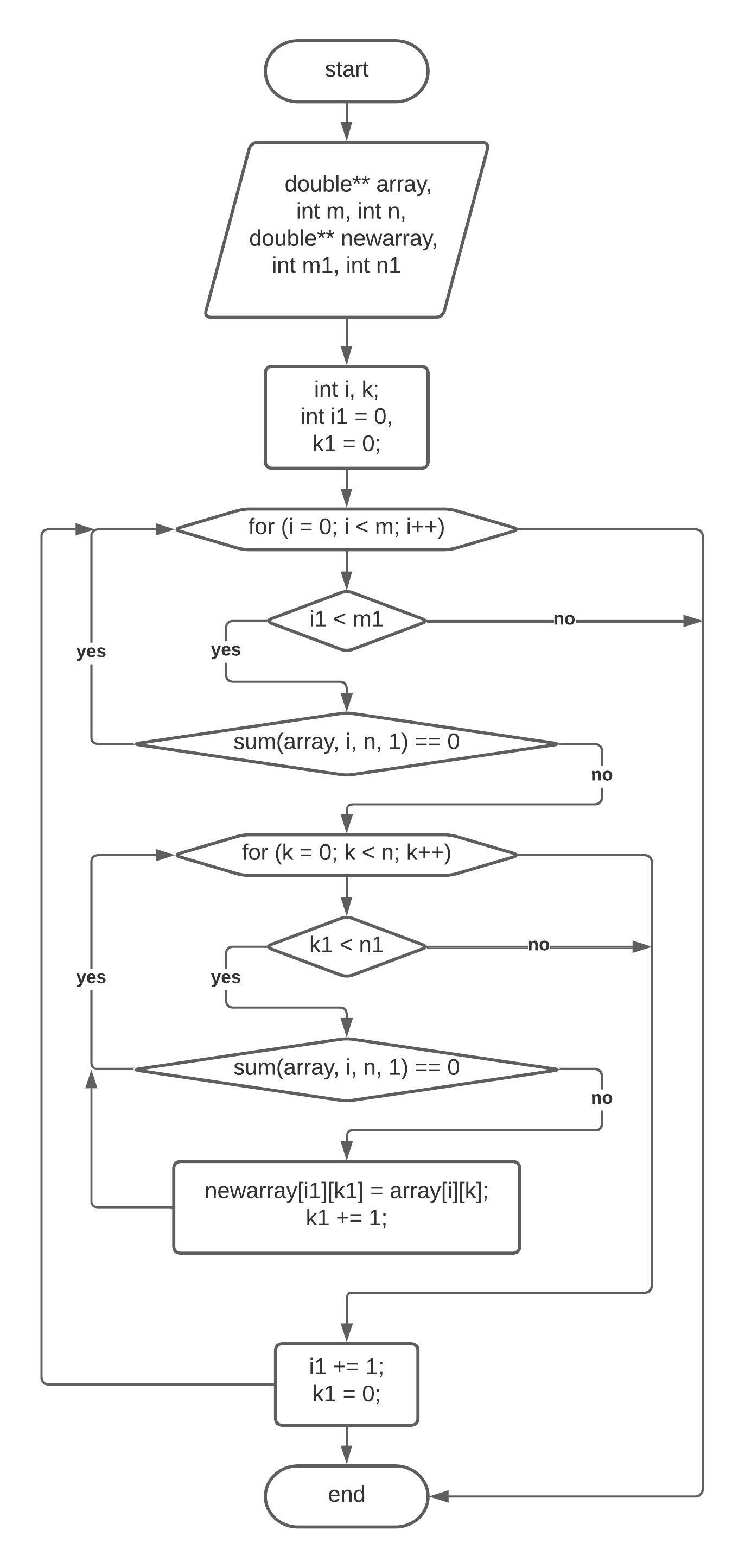
n = 5;

m1 = 4;

n1 = 4;

**Прототип:** void compress(double\*\* array, int m, int n, double\*\* newarray, int m1, int n1)

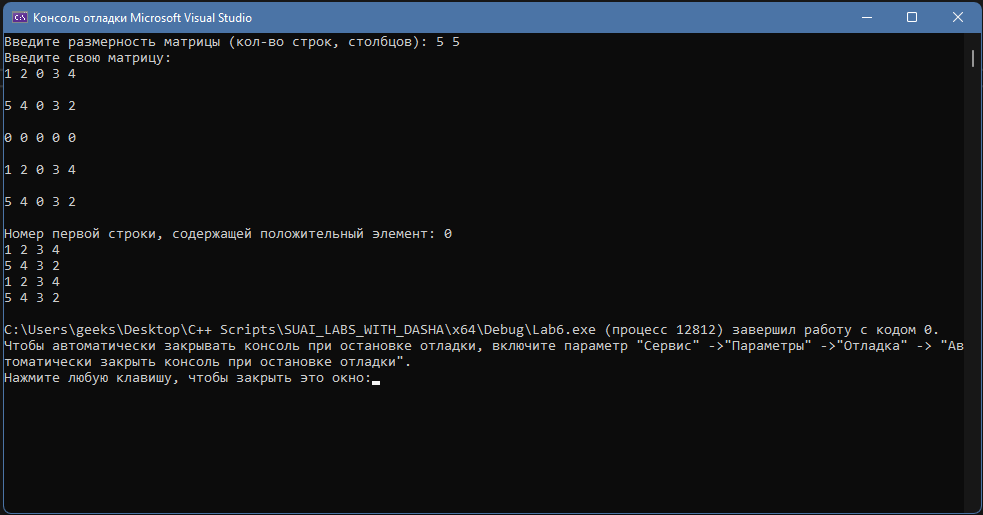
**Блок-схема:**



1. **Текст программы:**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29 | **void** **compress**(**double**\*\* array, **int** m, **int** n, **double**\*\* newarray, **int** m1, **int** n1) {  **int** i, k; // счётчики для старой матрицы  **int** i1 = **0**, k1 = **0**; // счётчки для новой матрицы  **for** (i = **0**; i < m; i++) {  **if** (i1 < m1) {  **if** (sum(array, i, n, **1**) == **0**) {  **continue**;  }  **for** (k = **0**; k < n; k++) {  **if** (k1 < n1) {  **if** (sum(array, m, k, **0**) == **0**) {  **continue**;  }  newarray[i1][k1] = array[i][k];  k1 += **1**;  }  **else** {  **break**;  }  }  i1 += **1**;  k1 = **0**;  }  **else** {  **break**;  }  }  } |

1. **Пример выполнения программы**

****

1. **Анализ результатов и выводы**

В ходе лабораторной работы я изучил структуру данных двумерный массив.