МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ  
федеральное государственное автономное образовательное учреждение высшего образования

«САНКТ-ПЕТЕРБУГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ  
АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ»

КАФЕДРА № 43

ОТЧЕТ   
ЗАЩИЩЕН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ассистент |  |  |  | М.А. Мурашова |
| должность, уч. степень, звание |  | подпись, дата |  | инициалы, фамилия |

|  |
| --- |
| ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ № 6 |
| «Обработка числовых матриц» |
| по курсу: Основы программирования |
|  |
|  |

РАБОТУ ВЫПОЛНИЛ

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| СТУДЕНТ ГР. № | 4134К |  |  |  | Опарин С.Н. |
|  |  |  | подпись, дата |  | инициалы, фамилия |

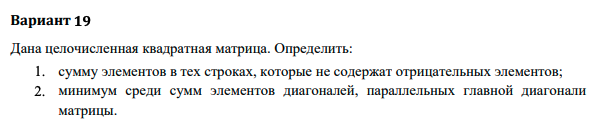
Санкт-Петербург 2022

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.



1. **Описание созданных функций**

Для реализации задания нам потребуется следующие функции:

**Имя:** fill

**Назначение:** проверка вводимых данных на корректность

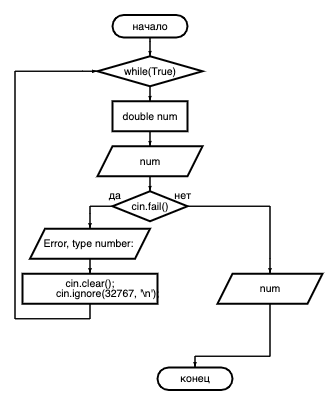
**Входные данные:** ввод с клавиатуры

**Выходные данные:** проверенный ввод с клавиатуры

**Побочный эффект:** отсутствует

**Тестовые данные:**

|  |  |
| --- | --- |
| **num** | **Результат** |
| **Числа** | Число |
| **Символы** | Error, type number: |

****

**Имя:** task\_1

**Назначение:** решение задачи, которая требуется в первом пункте

**Входные данные:** размер массива, массив

**Выходные данные:** сумма элементов строк, где отсутствуют отрицательные элементы

**Побочный эффект:** отсутствует

**Псевдокод:** Инициализируем переменные

От 0 до N

От 0 до N  
 Если значение неотрицательное и счетчик меньше размера, то   
 Переменную суммы увеличиваем на текущий элемент

Счетчик увеличиваем на единицу

Если счетчик равен размеру массива, то

Поднимаем флаг

Выходим

Иначе

Обнуляем счетчик и переменную суммы

Если сумма ненулевая и флаг поднят, то

Выведи сумму текущего ряда

Обнули переменные

Иначе

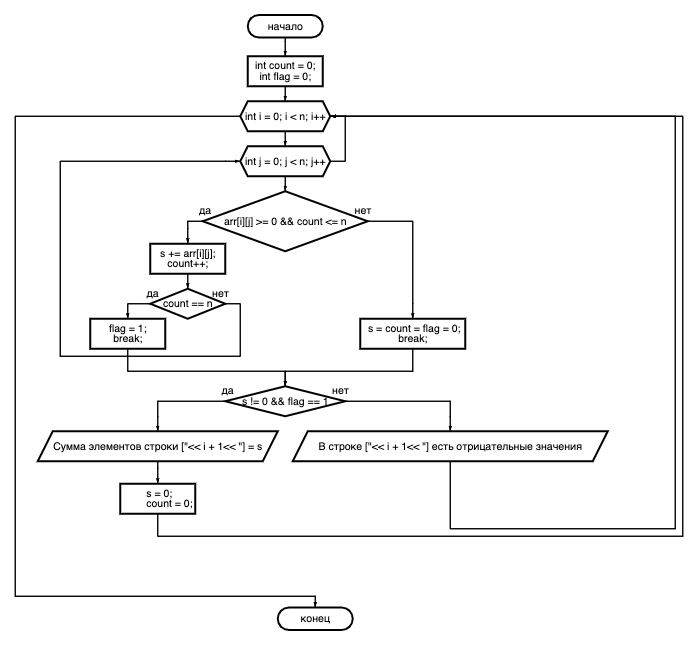
Выведи сообщение о присутствии отрицательных значений

**Прототип:** void task\_1(double\* arr[], int n)

**Тестовые данные:**

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| **1 2 3 4 5 6 7 8 9** | **Сумма элементов строки [1] = 6**  **Сумма элементов строки [2] = 15**  **Сумма элементов строки [3] = 24Сумма элементов строк** |
| **1 2 3 -4 5 6 7 8 9** | **Сумма элементов строки [1] = 6**  **В строке [2] есть отрицательные значения**  **Сумма элементов строки [3] = 24** |

**Блок-схема:**

****

**Имя:** task\_2

**Назначение:** нахождение минимальной суммы побочной диагонали

**Входные данные:** размер массива, массив

**Выходные данные:** минимальная сумма побочной диагонали

**Побочный эффект:** отсутствует

**Псевдокод:** Инициализируем переменные

Присваиваем переменной суммы значение правого верхнего угла  
 От 1 до N

Обнуляем переменные для сравнения

От 0 до N-i   
 Прибавляем значение в одну переменную

Прибавляем значение в другую переменную

Если первая переменная меньше суммы, то

Пере присваиваем значение суммы  
 Если вторая переменная меньше суммы, то

Пере присваиваем значение суммы

Выведи минимальную сумму

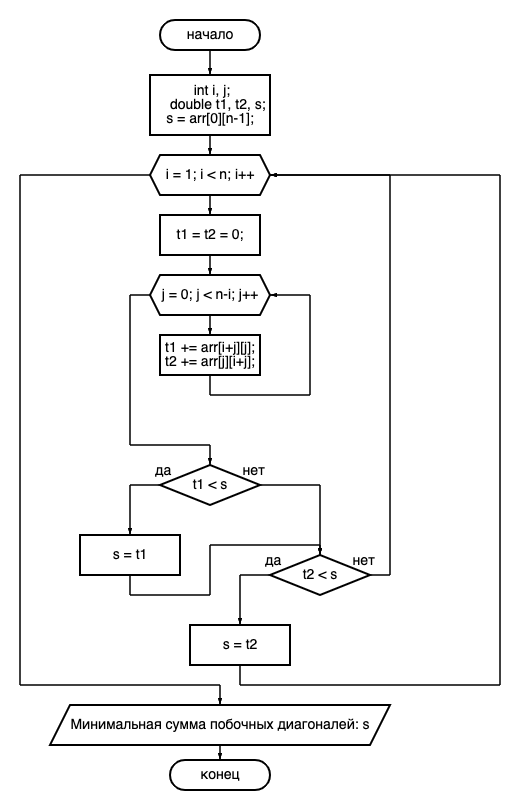
Верни 0

**Прототип:** double task\_2(double\* arr[], int n)

**Тестовые данные:**

|  |  |
| --- | --- |
| **Входные данные** | **Выходные данные** |
| **1 2 3 4 5 6 7 8 9** | **Минимальная сумма побочных диагоналей: 3** |
| **1 2 3 4 5 6 -7 8 9** | **Минимальная сумма побочных диагоналей: -7** |

**Блок-схема:**

****

**Имя:** main

**Назначение:** вызов всех функций, формирование матрицы и вывод ответов на задания по варианту

**Входные данные:** размер массива, значения в каждый элемент

**Выходные данные:** решения всех заданий по варианту и оформленный вывод

**Побочный эффект:** отсутствует

**Псевдокод:** Инициализируем переменные

Пока переменная меньше или равно 0, то

Выведи сообщение с просьбой о вводе

Вызови функцию проверки и введи значение

Формируем массив

От 1 до N

Формируем массивы в массиве

Выведи сообщение о просьбе заполнить матрицу

От 1 до N

От 1 до N

Выведи сообщение с просьбой о вводе значения матрицы

Введи значение матрицы с помощью функции проверки

От 1 до N

От 1 до N

Выведи текущее значение матрицы

Вызов функции решения первого задания

Вызов функции решения второго задания

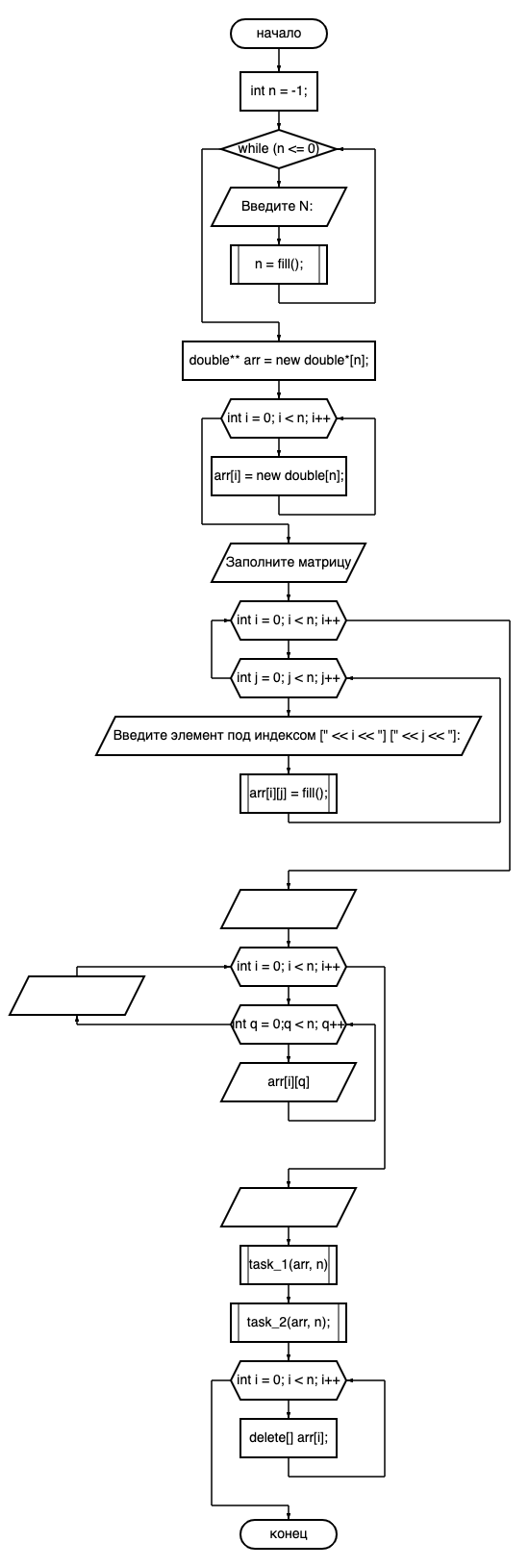
Удали матрицу

**Прототип:** int main()

**Тестовые данные:**

|  |  |
| --- | --- |
| **Входные данные** | **Выходные данные** |
| **3 1 2 3 4 5 6 -7 8 9** | **1 2 3**  **4 5 6**  **-7 8 9**  **Сумма элементов строки [1] = 6**  **Сумма элементов строки [2] = 15**  **В строке [3] есть отрицательные значения**  **Минимальная сумма побочных диагоналей: -7** |
| **3 10 11 12 13 14 15 16 17 18** | **10 11 12**  **13 14 15**  **16 17 18**  **Сумма элементов строки [1] = 33**  **Сумма элементов строки [2] = 42**  **Сумма элементов строки [3] = 51**  **Минимальная сумма побочных диагоналей: 12** |

**Блок-схема:**

****

1. **Текст программы**

#define \_CRTDBG\_MAP\_ALLOC

#include <stdlib.h>

//#include <crtdbg.h>

#ifdef \_DEBUG

#ifndef DBG\_NEW

#define DBG\_NEW new ( \_NORMAL\_BLOCK , \_\_FILE\_\_ , \_\_LINE\_\_ )

#define newDBG\_NEW

#endif

#endif

#include <iostream>

#include <string>

double fill() {

while (true) {

double num;

std::cin >> num;

if (std::cin.fail()) {

std::cin.clear();

std::cin.ignore(32767, '\n');

}

else {

return num;

}

}

}

double s = 0;

void task\_1(double\* arr[], int n) {

int count = 0;

int flag = 0;

for (int i = 0; i < n; i++) {

for (int j = 0; j < n; j++) {

if (arr[i][j] >= 0 && count <= n){

s += arr[i][j];

count++;

if(count == n){

flag = 1;

break;

}

}

else {

s = count = flag = 0;

break;

}

}

if (s != 0 && flag == 1){

std::cout << "Сумма элементов строки ["<< i + 1<< "] = " << s << std::endl;

s = 0;

count = 0;

}

else {

std::cout << "В строке ["<< i + 1<< "] есть отрицательные значения"<< std::endl;

}

}

}

double task\_2(double\* arr[], int n){

int i, j;

double t1, t2, s;

s = arr[0][n-1];

for(i = 1; i < n; i++){

t1 = t2 = 0;

for(j = 0; j < n-i; j++){

t1 += arr[i+j][j];

t2 += arr[j][i+j];

}

if(t1 < s) s = t1;

if(t2 < s) s = t2;

}

std::cout << "Минимальная сумма побочных диагоналей: " << s << std::endl;

return 0;

}

int main()

{

int n = -1;

while (n <= 0) {

std::cout << "Введите N: ";

n = fill();

}

double\*\* arr = new double\*[n];

for (int i = 0; i < n; i++) {

arr[i] = new double[n];

}

std::cout << "Заполните матрицу\n";

for (int i = 0; i < n; i++)

{

for (int j = 0; j < n; j++)

{

std::cout << "Введите элемент под индексом [" << i << "] [" << j << "]: ";

arr[i][j] = fill();

}

}

std::cout << std::endl;

for (int i = 0; i < n; i++) {

for (int q = 0;q < n; q++) {

std::cout << arr[i][q] << " ";

}

std::cout << std::endl;

}

std::cout << std:: endl;

task\_1(arr, n);

task\_2(arr, n);

for (int i = 0; i < n; i++) {

delete[] arr[i];

}

delete[] arr;

/\*

// 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();

\*/

}

1. **Пример выполнения программы**

Ниже показан пример выполнения программы

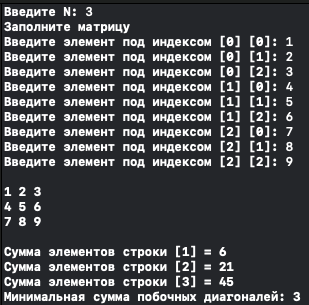


Рис. 1 - Пример выполнения программы

Видно, что результаты расчетов совпадают с тестовыми данными.

1. **Анализ результатов и выводы**

В результате выполнения лабораторной работы были изучены структуры данных двумерного массива

К достоинствам программы можно отнести:

* Решение каждого выражения реализовано в виде отдельной функции
* Есть проверка на корректность данных

Из недостатков можно отметить:

* Недостаточно оптимизирован код