

### 1. Цель работы

Целью работы является изучение структуры данных двумерный массив.

### 2. Задание

Согласно варианту №3:

Дана целочисленная прямоугольная матрица. Определить:

1. количество столбцов, содержащих хотя бы один нулевой элемент;
2. номер строки, в которой находится самая длинная серия одинаковых элементов.

### 3. Описание созданных функций

**Имя:** get\_num

**Назначение:** Запрос и проверка целочисленного числа на корректность

**Входные данные:**

- -

**Выходные данные:**

- x – Введённое число

**Побочный эффект:** Отсутствует

**Тестовые данные:**

Вход	Выход
5.5	Неверный ввод
5	5

**Прототип:** double get\_num()

**Псевдокод**

Запросить число

    Проверить на корректность

    В случае неудачи повторить процедуру

Вернуть число

**Блок-схема –**

**Имя:** request\_len

**Назначение:** Запрос и проверка неотрицательного целочисленного числа

**Входные данные:**

- -

**Выходные данные:**

- x – Введённое число

**Побочный эффект:** Отсутствует

**Тестовые данные:**

Вход	Выход
-5	Неверный ввод
5	5

**Прототип:** unsigned request\_len()**Псевдокод**

Запросить число

Проверить на корректность и неотрицаиельность

В случае неудачи повторить процедуру

Вернуть число

**Блок-схема –****Имя:** fill\_matrix**Назначение:** Заполнение матрицы целочисленными элементами**Входные данные:**

- matrix – Матрица для заполнения
- n – Размерность матрицы nxn

**Выходные данные:**

- -

**Побочный эффект:** Отсутствует**Тестовые данные:**

Вход	Выход
(1 2 0) (1 0 3f) 2	Неверный ввод
(1 2 0) (1 0 3) 2	
(1 2 0) (1 0 3) 2	(1 2 0) (1 0 3)

**Прототип:** fill\_matrix(int\*\* matrix, int n)**Псевдокод**

Перебор каждого элемента строки

Запрос числа

Проверка на корректность

В случае неудачи запрос заново

Присвоить число элементу строки

**Блок-схема –**

**Имя:** task1

**Назначение:** Подсчёт кол-ва столбцов с хотя бы одним нулевым элементом

**Входные данные:**

- matrix – Матрица для поиска столбцов
- n – Размерность матрицы nxn
- flag – Флаг наличия столбцов с нулевым элементом

**Выходные данные:**

- s\_count – Кол-во столбцов с хотя бы одним нулевым элементом

**Побочный эффект:** Смена флага в зависимости от наличия столбцов с нулевым элементом:

true – Такие столбцы существуют

false – Таких столбцов не существует

**Тестовые данные:**

Вход	Выход
(1 2 3) (1 0 3)	0
(1 2 0) (1 0 3)	1

**Прототип:** unsigned task1(int\*\* matrix, int n, bool &flag)

**Псевдокод**

Перебор каждого элемента столбца

Если найден нулевой элементов

Обновить кол-во

Перейти к следующему столбцу

Поменять флаг в зависимости от наличия столбцов

Вернуть кол-во

**Блок-схема**

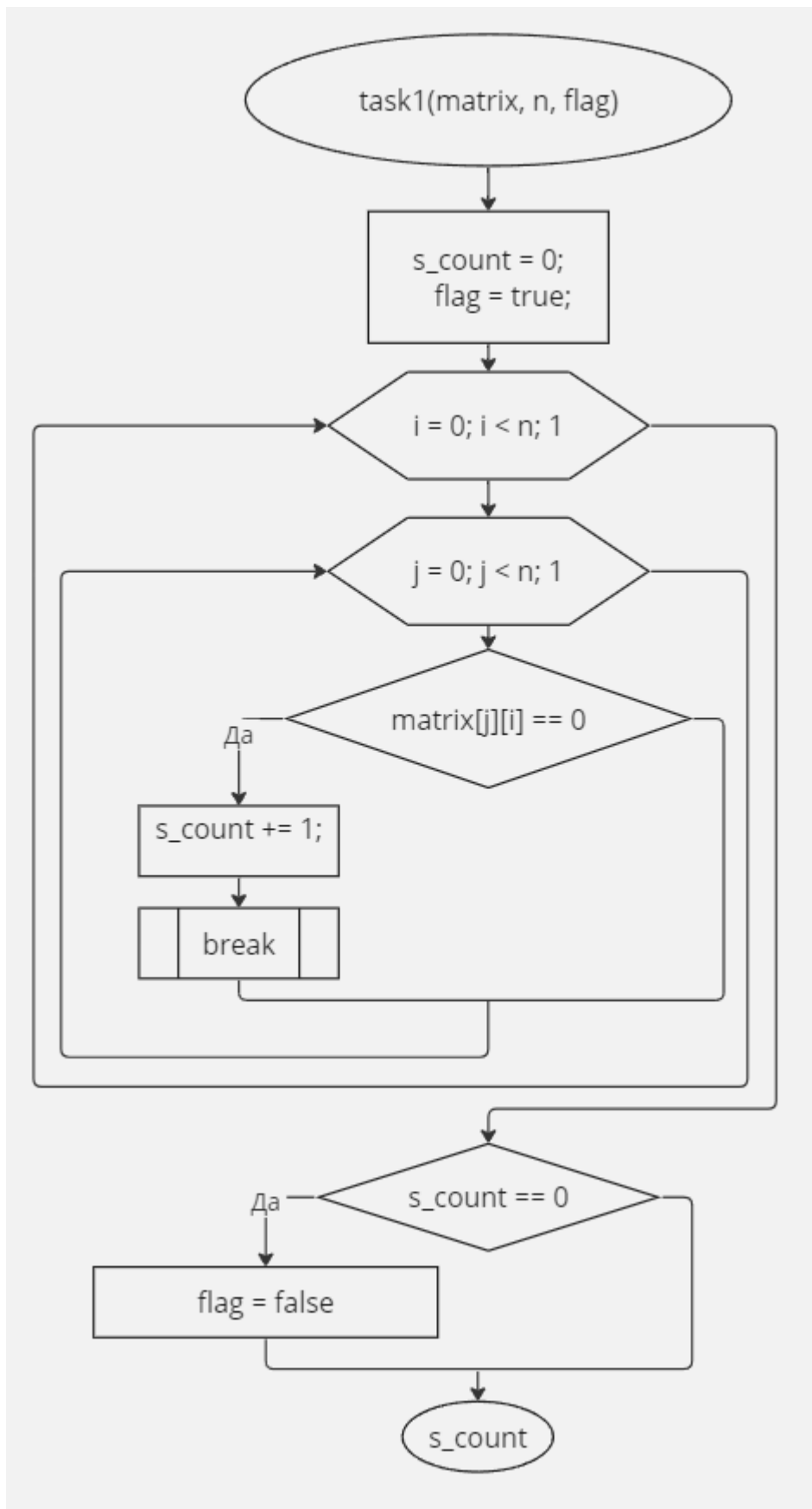


Рисунок 1 – Блок схема task 1

**Имя:** task2

**Назначение:** Нахождение строки с наибольшей серией одинаковых элементов

**Входные данные:**

- matrix – Матрица для поиска строки
- n – Размерность матрицы nxn
- flag – Флаг наличия строки с серией одинаковых элементов

**Выходные данные:**

- s\_count – Индекс строки+1 с наибольшей серией элементов

**Побочный эффект:** Смена флага в зависимости от наличия строки:

true – Такая строка существует

false – Такой строки не существует

**Тестовые данные:**

Вход	Выход
(7 7 7) (1 0 3)	1
(1 2 0) (1 0 3)	Флаг false

**Прототип:** int task2(int\*\* matrix, int n, bool& flag)

**Псевдокод**

Перебор каждого элемента строки

    Сравнить с предыдущим элементом

        Если они равны увеличить сумму

    Сравнить новую сумму и старую

        Если новая больше, то запомнить номер строки

Если серий не было найдено, изменить флаг

**Блок-схема**

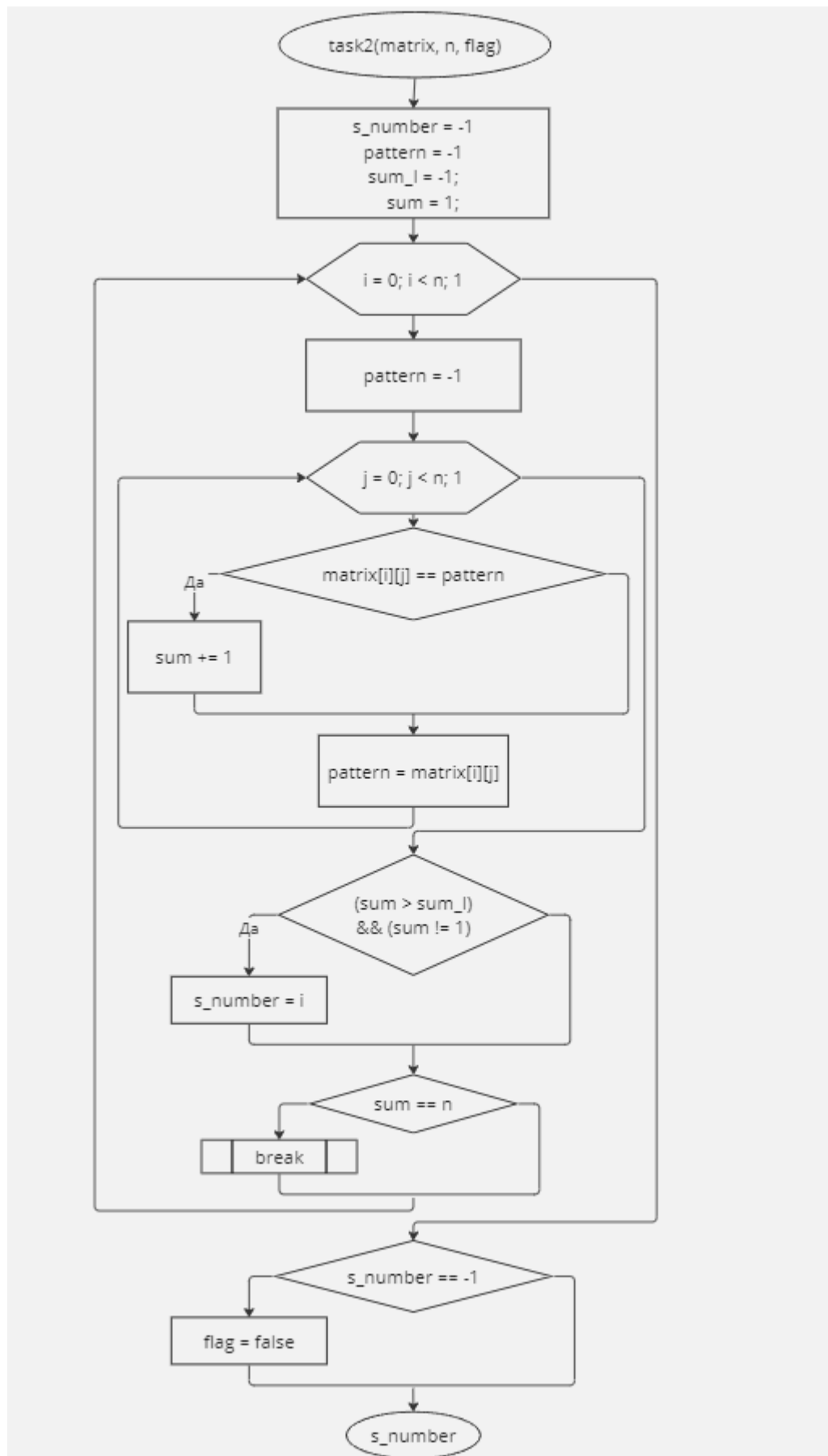


Рисунок 2 – Блок схема task2

#### 4. Текст программы

```

#define _CRTDBG_MAP_ALLOC
#include <stdlib.h>
#include <crtdbg.h>

#ifdef _DEBUG
#ifdef DBG_NEW
#define DBG_NEW new ( _NORMAL_BLOCK , __FILE__ , __LINE__ )
#define newDBG_NEW
#endif
#endif

#include<iostream>;

using namespace std;

int get_num() // Запрос и проверка числа int на корректность
{
    int x;

    cin >> x;
    while (cin.fail() || (cin.peek() != '\n')) // Проверка на корректность
    {
        cin.clear(); // Очищение флага ошибки
        cin.ignore(numeric_limits<streamsize>::max(), '\n'); // Очистка буфера
запроса
        cout << "Повторите ввод: ";
        cin >> x;
    }

    return x;
}

unsigned request_len() // Запрос размера массива
{
    int n;

    cout << "Введи размерность массива целым неотрицательным числом (!=0)!" << endl;
    cout << "n = ";

    n = get_num();

    while (n <= 0) // Проверка на корректный ввод
    {
        n = get_num();
    }

    return n;
}

void fill_matrix(int** matrix, int n) // Заполнение переданного массива с клавиатуры
{
    cout << "Матрица принимает только целые значения" << endl;
    for (int i = 0; i < n; i++) // строки
    {
        cout << "Ввод " << i + 1 << " строки.." << endl;
        for (int j = 0; j < n; j++) //столбцы массива
        {
            cout << "Ввод " << j + 1 << " элемента: ";
            matrix[i][j] = get_num(); //запоминаем введенное значение
        }
    }
}

```

```

unsigned task1(int** matrix, int n, bool &flag) // 1 Задание - Найти кол-во столбцов
с хотя бы одним нулём
{
    unsigned s_count = 0;
    flag = true;

    cout << "Начинаем считать количество столбцов с нулями..." << endl;
    for (int i = 0; i < n; i++) // столбцы
    {
        for (int j = 0; j < n; j++) // строки
        {
            if (matrix[j][i] == 0)
            {
                s_count += 1;
                break;
            }
        }
    }

    if (s_count == 0)
    {
        flag = false;
    }

    return s_count;
}

int task2(int** matrix, int n, bool& flag) // 2 Задание - Найти номер строки в
которой наибольшая серия одинаковых элементов
{
    int s_number = -1, pattern = -1, sum_l = -1;
    int sum = 1;

    flag = true;

    cout << "Начинаем искать строку с наибольшей серией одинаковых элементов..." <<
endl;
    for (int i = 0; i < n; i++) // строки
    {
        pattern = -1;

        for (int j = 0; j < n; j++) // столбцы
        {
            if (matrix[i][j] == pattern)
            {
                sum += 1;
            }
            pattern = matrix[i][j];
        }

        if ((sum > sum_l) && (sum != 1))
        {
            s_number = i;
        }

        if (sum == n)
        {
            break;
        }

        sum_l = sum;
        sum = 1;
    }
}

```



```

    if (s_number == -1)
    {
        flag = false;
    }

    return s_number;
}

int main()
{
    setlocale(LC_ALL, "rus");

    unsigned n = request_len();
    unsigned s_count;
    int s_number;
    bool f;

    int** matrix; // Создание матрицы nхn с целочисленными данными
    matrix = new int* [n];
    for (unsigned i = 0; i < n; i++)
        matrix[i] = new int[n];

    fill_matrix(matrix, n);

    cout << "Ваша матрица:" << endl;
    for (unsigned i = 0; i < n; i++)
    {
        for (unsigned j = 0; j < n; j++)
        {
            cout << matrix[i][j] << '\t';
        }
        cout << endl;
    }

    s_count = task1(matrix, n, f);
    if (!f)
    {
        cout << "В матрице нет столбцов с нулевыми элементами" << endl;
    }
    else
    {
        cout << "Количество столбцов: " << s_count << endl;
    }

    s_number = task2(matrix, n, f);
    if (!f)
    {
        cout << "В матрице нет строк с серией одинаковых элементов" << endl;
    }
    else
    {
        cout << "Номер строки: " << s_number + 1 << endl;
    }

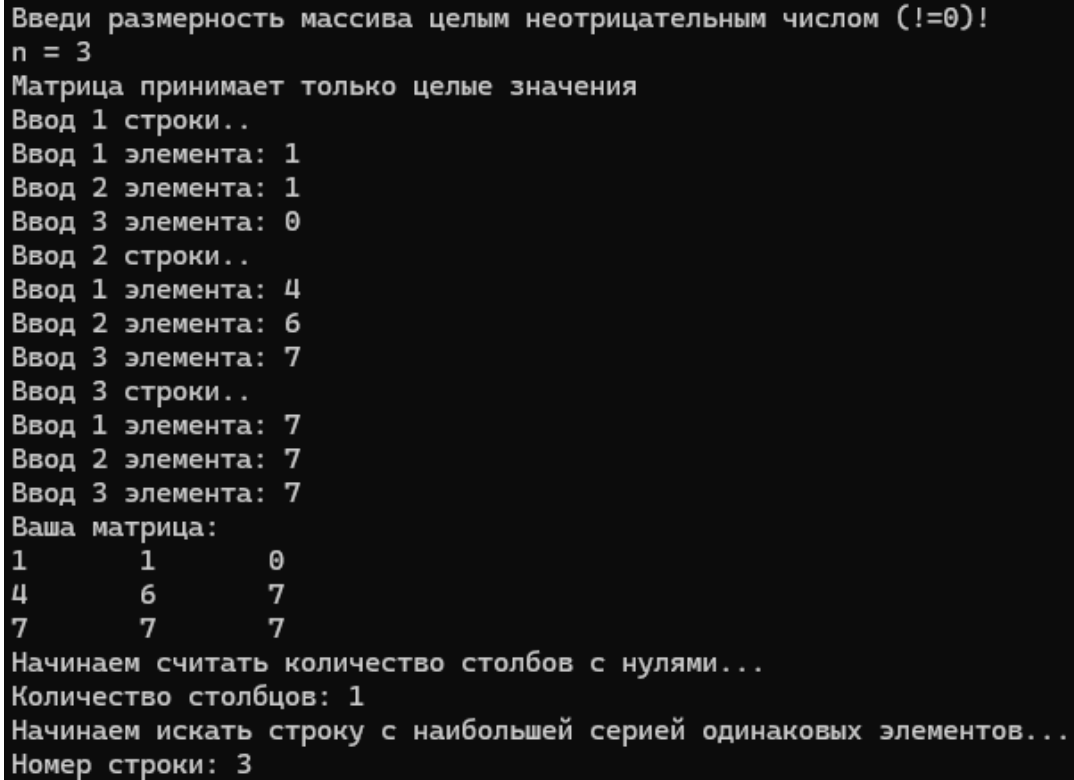
    for (unsigned i = 0; i < n; i++) // Освобождение памяти
        delete matrix[i];
    delete[] matrix;

    // 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();  
}
```

## 5. Пример работы программы



```
Введи размерность массива целым неотрицательным числом (!=0)!  
n = 3  
Матрица принимает только целые значения  
Ввод 1 строки..  
Ввод 1 элемента: 1  
Ввод 2 элемента: 1  
Ввод 3 элемента: 0  
Ввод 2 строки..  
Ввод 1 элемента: 4  
Ввод 2 элемента: 6  
Ввод 3 элемента: 7  
Ввод 3 строки..  
Ввод 1 элемента: 7  
Ввод 2 элемента: 7  
Ввод 3 элемента: 7  
Ваша матрица:  
1      1      0  
4      6      7  
7      7      7  
Начинаем считать количество столбцов с нулями...  
Количество столбцов: 1  
Начинаем искать строку с наибольшей серией одинаковых элементов...  
Номер строки: 3
```

Рисунок – Результат работы программы

Полученные данные совпадают с действительными

## 6. Анализ результатов и выводы

В процессе лабораторной работы была изучена структура данных одномерный массив

Достоинства программы:

- Проверка данных на корректность
- Возможность использования подпрограмм в других разработках
- Используются флаги для пометки состояния выполнения задания