

1. Цель работы

Целью работы является изучение структуры данных одномерный массив

2. Задание

Согласно варианту №3 необходимо:

В одномерном массиве, состоящем из n целых элементов, вычислить:

1. произведение элементов массива с чётными номерами;
2. сумму элементов массива, расположенных между первым и последним нулевыми элементами.

Преобразовать массив таким образом, чтобы сначала располагались все положительные элементы, а потом – все отрицательные (элементы, равные 0, считать положительными).

3. Описание созданных функций

Имя: `get_num_int`

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

Входные данные:

- -

Выходные данные:

- x – Введённое число

Побочный эффект: Отсутствует

Тестовые данные:

Вход	Выход
-5.5	Неверный ввод
5	5

Прототип: `int get_num_int()`

Псевдокод

Запросить число

 Проверить на корректность

 В случае некорректности повторить процедуру

Вернуть число

Блок-схема –

Имя: `request_len`

Назначение: Запрос неотрицательного целого числа

Входные данные:

- -

Выходные данные:

- n – Введённое число

Побочный эффект: Отсутствует

Тестовые данные:

Вход	Выход
-5	Неверный ввод
5	5

Прототип: unsigned int request_len()

Псевдокод

Запросить число

 Проверить на корректность и неотрицательность

 В случае неудачи повторить процедуру

Вернуть число

Блок-схема –

Имя: request_arr

Назначение: Заполнение переданного массива числами

Входные данные:

- numbers – Массив для заполнения
- n – Размер переданного массива

Выходные данные:

- -

Побочный эффект: Переданный массив заполнится элементами

Тестовые данные:

Вход	Выход
(1 2 3) 3	Массив с элементами 1 2 3
g 3	Неверный ввод

Прототип: void request_arr(int* numbers, const unsigned int n)

Псевдокод

Запросить чисел n раз

Проверить на корректность

Добавить в массив

Блок-схема –

Имя: task1

Назначение: Подсчёт произведения элементов массива на чётных позициях

Входные данные:

- numbers – Массив для которого решается задание
- n – Размер массива

Выходные данные:

- S – Произведение элементов на чётных позициях

Побочный эффект: Отсутствует

Тестовые данные:

Вход	Выход
(0 3 4 4) 3	0
(2 3 4 4 5 5) 5	40

Прототип: int task1(int *numbers, const unsigned int n)

Псевдокод

Начать цикл перебора индексов

 Проверить индексы на чётность

 В случае успеха считать произведение

Вернуть произведение

Блок-схема

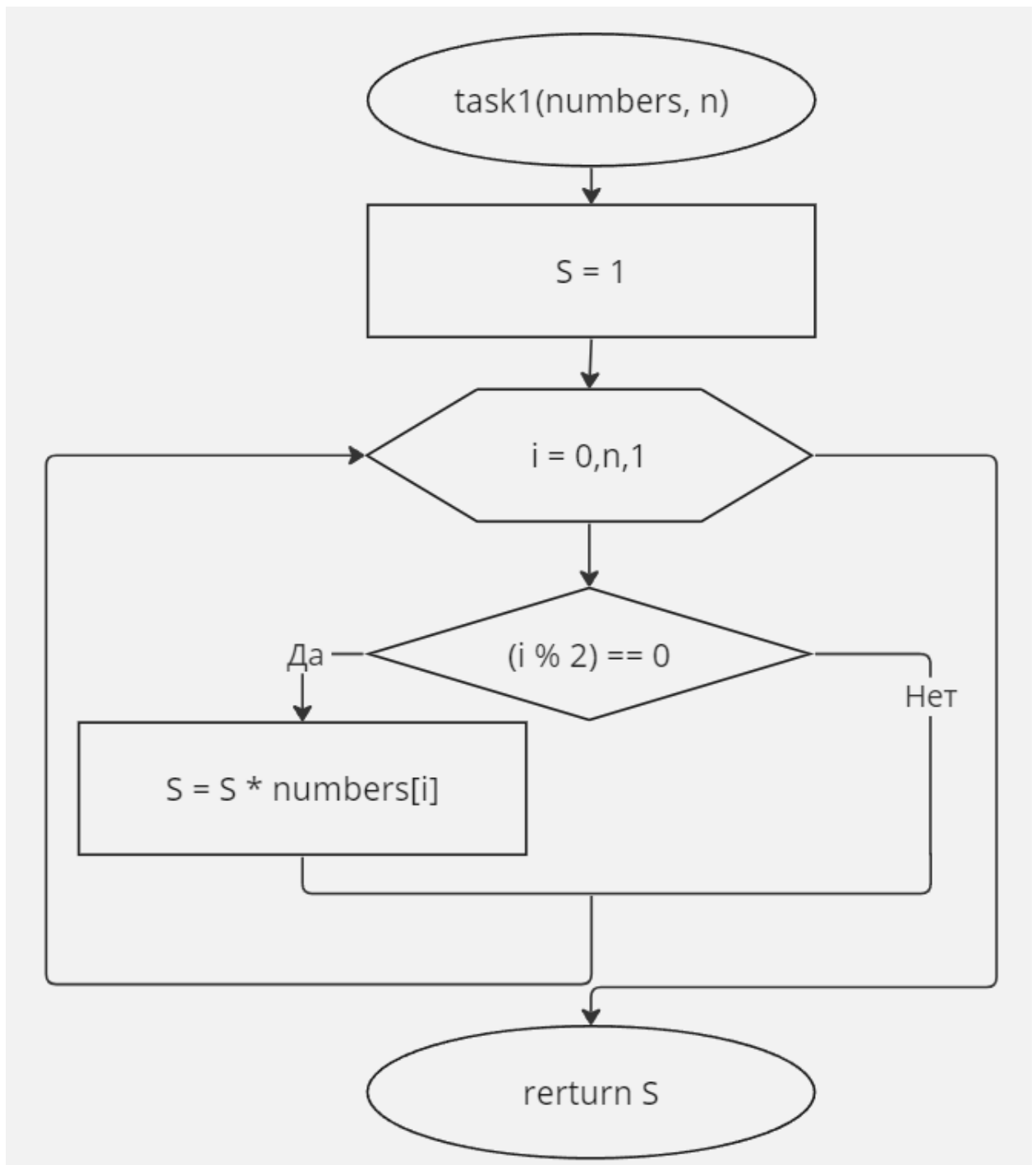


Рисунок 1 – Блок схема task1

Имя: task2

Назначение: Подсчёт суммы элементов между крайними нулями

Входные данные:

- `numbers` – Массив для которого решается задание
- `n` – Размер массива
- `flag` – флаг состояния наличия нулей в массиве

Выходные данные:

- `S` – Сумма элементов между крайними нулями

Побочный эффект: Установка flag в зависимости от нулей в массиве:

0 – В массиве существует два нуля

1 – В массиве не существует нулей

2 – В массиве существует один нуль

3 – В массиве нет элементов между нулями

Тестовые данные:

Вход	Выход
(0 34 4 0) 4	38
(0 34 4 5 5) 3	0 (флаг = 2)

Прототип: int task2(int *numbers, const unsigned int n, int &flag)

Псевдокод

Начать цикл перебора индексов

 Запомнить индексы крайних нулей

Поставить флаг в зависимости от индексов нулей

Начать цикл перебора индексов между нулями

Складывать значения

Вернуть сумму

Блок-схема

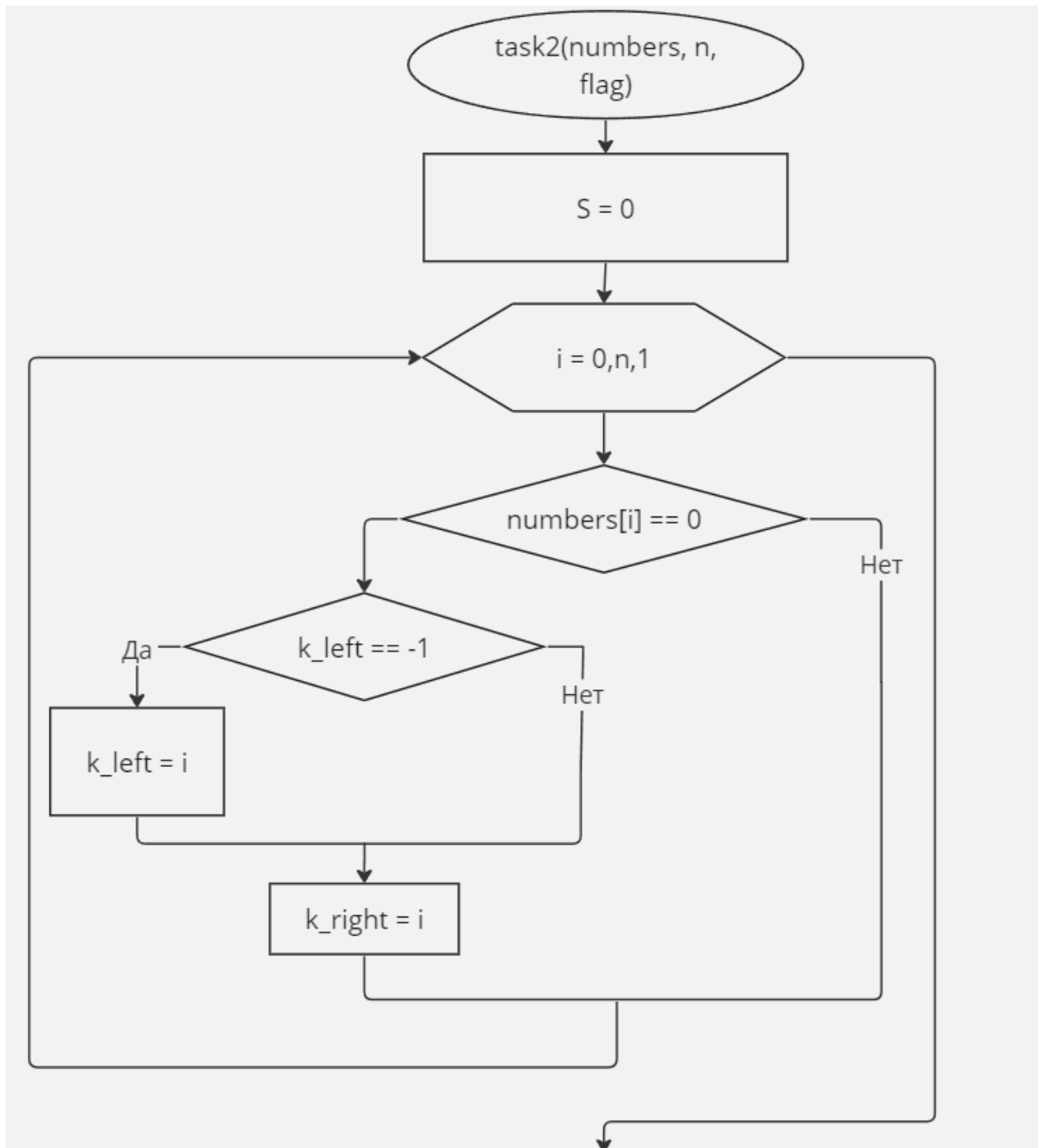


Рисунок 2 – блок схема task 2 I

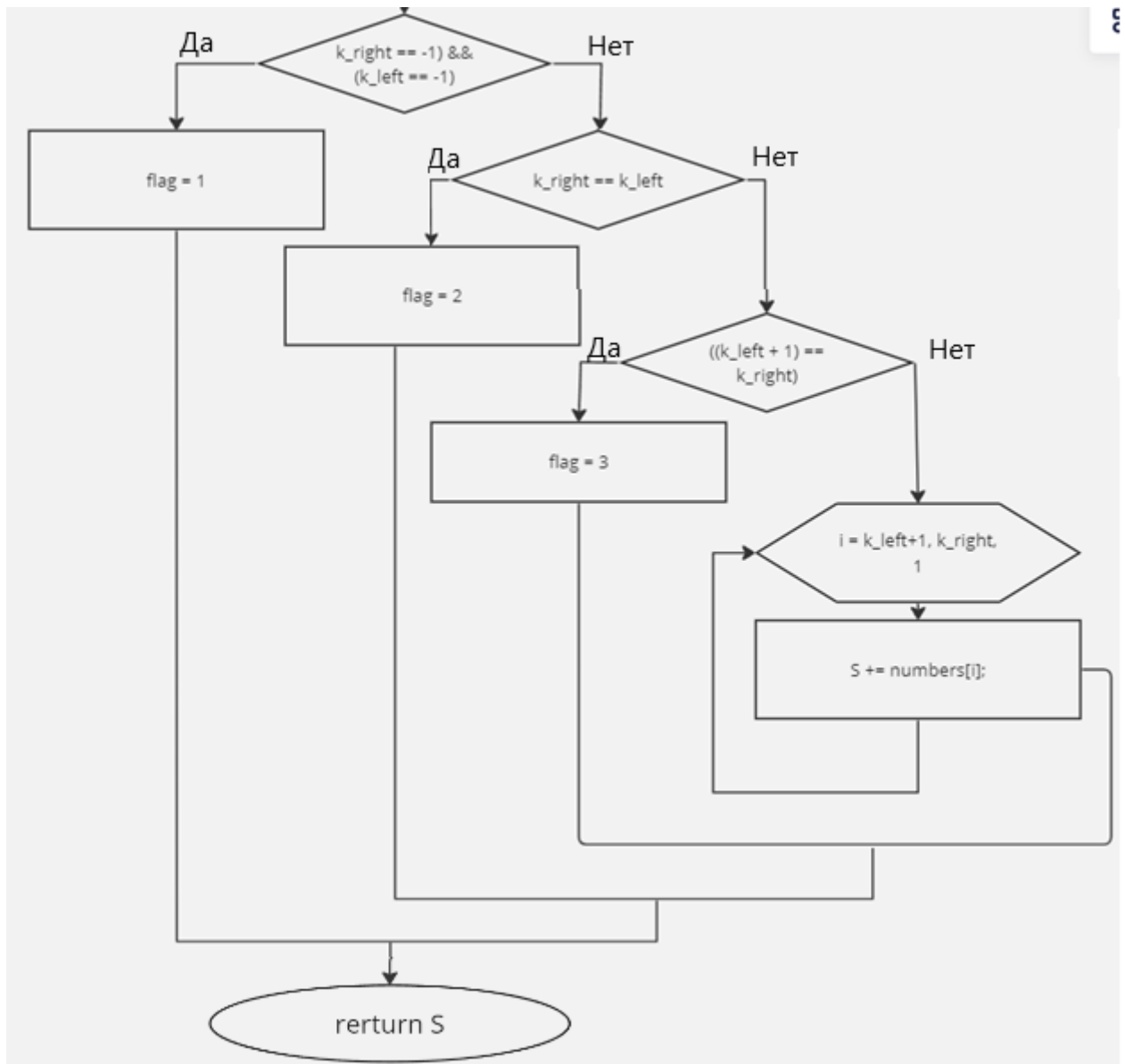


Рисунок 3 – блок схема task 2 II

Имя: task3

Назначение: Сортировка массива по убыванию

Входные данные:

- numbers_copу – Массив, в который скопируется исходных массив и произведётся сортировка
- numbers – Массив, который нужно сортировать
- double n – Размер массива

Выходные данные:

- -

Побочный эффект: numbers_copу заполнится отсортированным numbers

Тестовые данные:

Вход	Выход
(0 0 0) (30 0 90) 3	(90 30 0)
(0 0 0) (30 0 90 -4) 4	(90 30 0 -4)

Прототип: void task3(int *numbers_copy, int *numbers, const unsigned int n)

Псевдокод

Скопировать массив в новый

Отсортировать по убыванию новый массив

Блок-схема

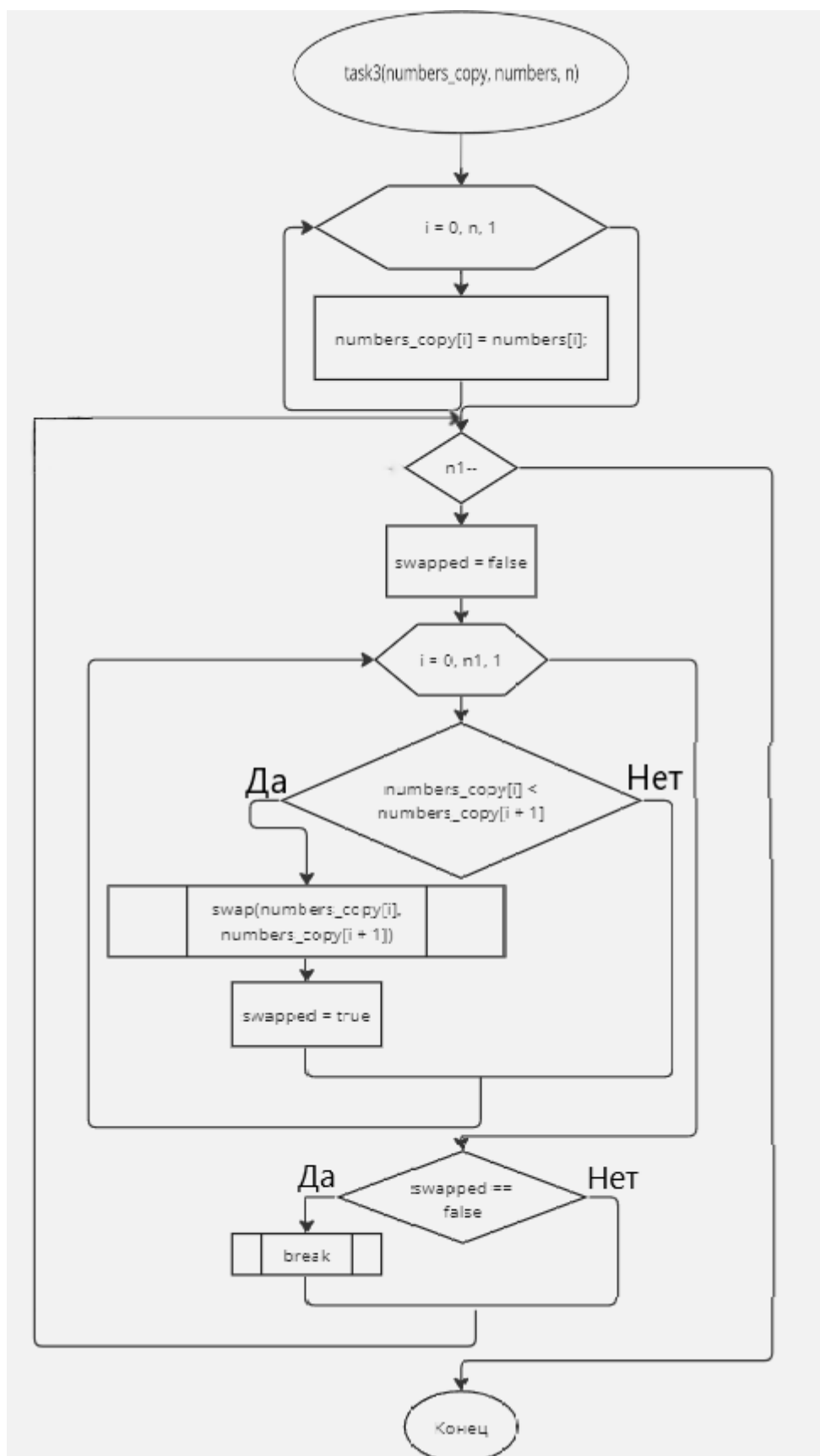


Рисунок 4 – Блок схема task3

4. Текст программы

```
// Для обнаружения утечек памяти
#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_int() // Запрос и проверка числа 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 int request_len()
{
    int n;

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

    n = get_num_int();

    while (n <= 0) // Проверка на корректный ввод
    {
        cout << "Повторите ввод: ";
        n = get_num_int();
    }

    return n;
}

void request_arr(int* numbers, const unsigned int n) // Заполнение массива
{
    unsigned int i = 0;

    cout << "Начинай вводить значения массива" << endl;

    while (i < n) // Заполнение массива
    {
        numbers[i] = get_num_int();
        i++;
    }
}
```

```

    cout << "Массив введён" << endl;
}

int task1(int *numbers, const unsigned int n) // Решение первого задания
(Произведение элементов на чётных позициях)
{
    cout << "Начинаем считать произведение элементов чётных позиций..." << endl;

    int S = 1;

    for (unsigned int i{ 0 }; i < n; i++) // Перебор каждого индекса
    {
        if ((i % 2) == 0) // Проверка индекса на чётность
        {
            S = S * numbers[i]; // Произведение
        }
    }

    return S;
}

int task2(int *numbers, const unsigned int n, int &flag) // Решение второго задания
(Сумма элементов между крайними нулями)
{
    cout << "Начинаем считать сумму элементов между первым и последним нулём..." <<
endl;

    int S = 0;
    unsigned int k_right = -1, k_left = -1;
    flag = 0;

    for (unsigned int i{ 0 }; i < n; i++)// Перебор элементов и нахождение двух
крайних нулей и присвоение их индексов переменным
    {
        if (numbers[i] == 0)
        {
            if (k_left == -1)
            {
                k_left = i;
            }
            k_right = i;
        }
    }

    if ((k_right == -1) && (k_left == -1)) // Вывод на каждый из случаев
    {
        flag = 1;
    }
    else if (k_right == k_left)
    {
        flag = 2;
    }
    else if ((k_left + 1) == k_right)
    {
        flag = 3;
    }
    else
    {
        for (unsigned int i{ k_left + 1 }; i < k_right; i++)// Перебор элементов
между нулями и сложение
        {
            S += numbers[i];
        }
    }
}

```

```

    return S;
}

void task3(int *numbers_copy, int *numbers, const unsigned int n) // Сортировка
массива по убыванию методом bubblesort
{
    cout << "Начинаем пересобирать массив..." << endl;

    int n1 = n;

    for (unsigned int i = 0; i < n; i++)
    {
        numbers_copy[i] = numbers[i];
    }

    while (n1--)
    {
        bool swapped = false;

        for (int i = 0; i < n1; i++)
        {
            if (numbers_copy[i] < numbers_copy[i + 1])
            {
                swap(numbers_copy[i], numbers_copy[i + 1]);
                swapped = true;
            }
        }

        if (swapped == false)
            break;
    }
}

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

    unsigned int n = request_len();
    int S;
    int *numbers = new int[n];
    int *numbers_copy = new int[n];
    int f;

    request_arr(numbers, n);

    cout << "Произведение элементов на чётных позициях: " << task1(numbers, n) <<
endl;

    S = task2(numbers, n, f);

    if (f == 0)
    {
        cout << "Сумма элементов перед первым и последним нулём: " << S << endl;
    }
    else if (f == 1)
    {
        cout << "В массиве нет нулей" << endl;
    }
    else if (f == 2)
    {
        cout << "В массиве один нуль" << endl;
    }
    else
    {
        cout << "Нет элементов между нулями" << endl;
    }
}

```

```

}

task3(numbers_copy, numbers, n);

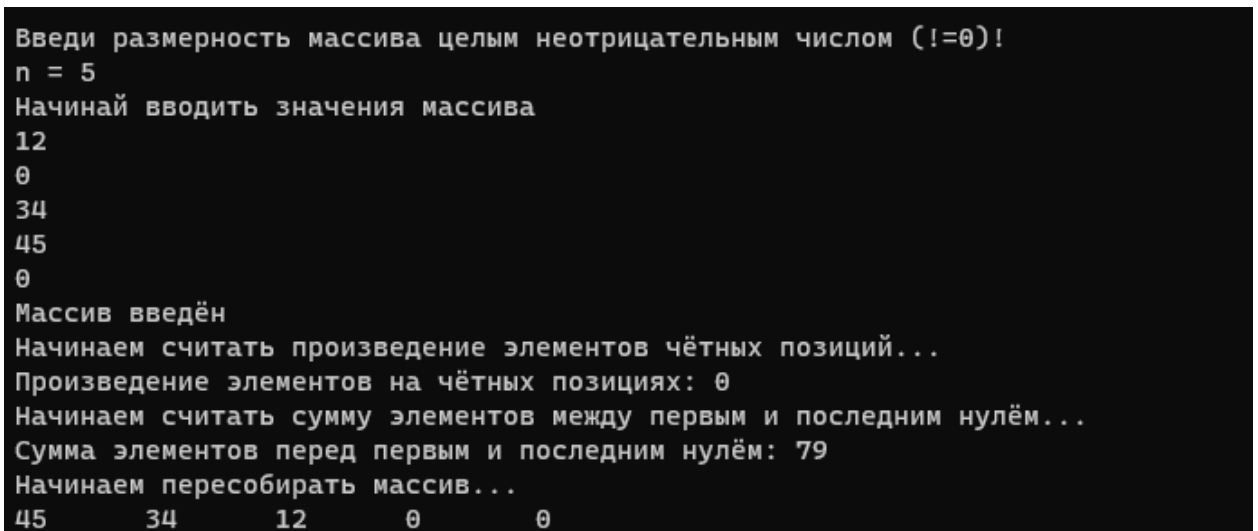
for (unsigned int i{ 0 }; i < n; i++)
{
    cout << numbers_copy[i] << '\t';
}

delete[] numbers;
delete[] numbers_copy;

// 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 = 5
Начинай вводить значения массива
12
0
34
45
0
Массив введен
Начинаем считать произведение элементов чётных позиций...
Произведение элементов на чётных позициях: 0
Начинаем считать сумму элементов между первым и последним нулём...
Сумма элементов перед первым и последним нулём: 79
Начинаем пересобирать массив...
45      34      12      0      0

```

Рисунок 5 – 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, есть ли нули в массиве(Задание 2)