1. **Цель работы**

Целью работы является изучение структуры данных одномерный массив

1. **Задание**

Согласно варианту №3 необходимо:  
В одномерном массиве, состоящем из n целых элементов, вычислить:

1. произведение элементов массива с чѐтными 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).

1. **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:** requst\_arr

**Назначение:** Заполнение переданного массива числами

**Входные данные:**

* numbers – Массив для заполнения 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 requst\_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 34 4) 3 | 0 |
| (2 34 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 произвдение

Вернуть произведение

**Блок-схема**

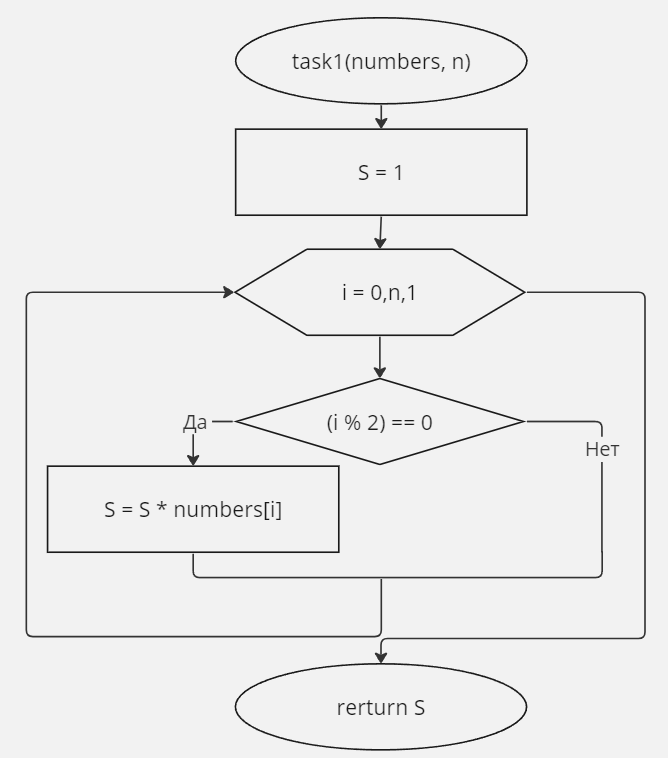


Рисунок 1 – Блок схема task1

**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 зависмости 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 цикил перебора индексов между нулями

Складывать значения

Вернуть сумму

**Блок-схема**

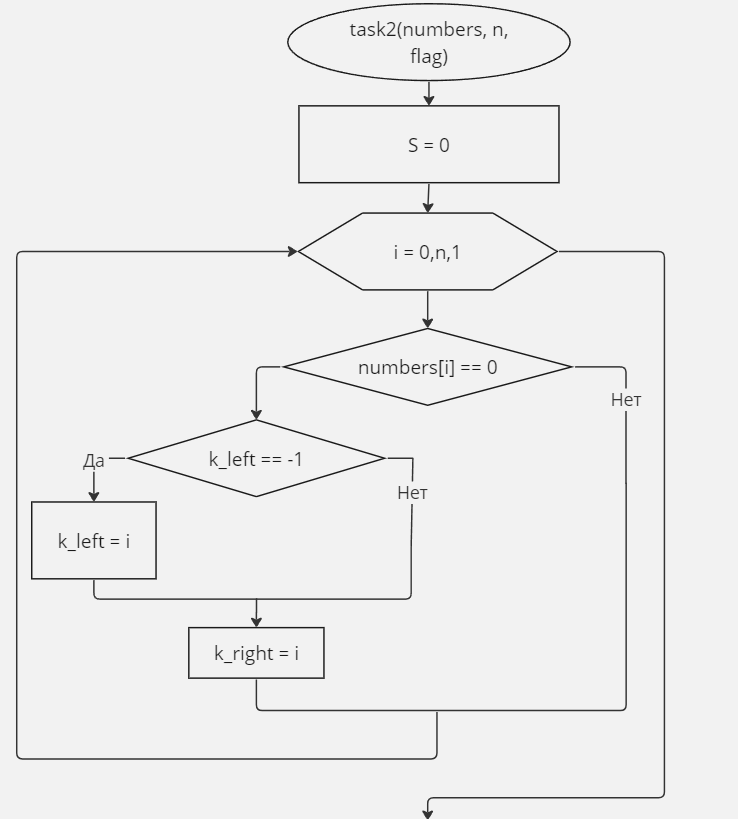


Рисунок 2 – блок схема task 2 I

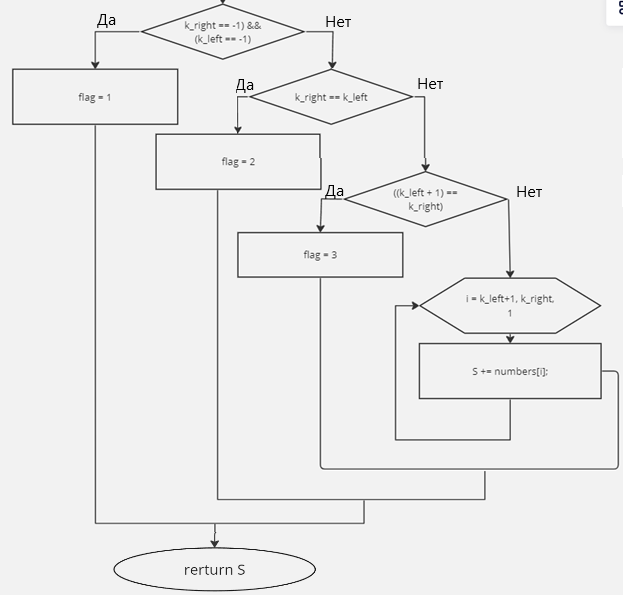
****

Рисунок 3 – блок схема task 2 II

**Имя:** task3

**Назначение:** Сортировка массива по убыванию

**Входные данные:**

* numbers\_copy – Массив, в который скопируется исходных массив и произведётся сортировка
* numbers – Массив, который нужно сортировать
* double n – Размер массива

**Выходные данные:**

* -

**Побочный эффект:** numbers\_copy 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**

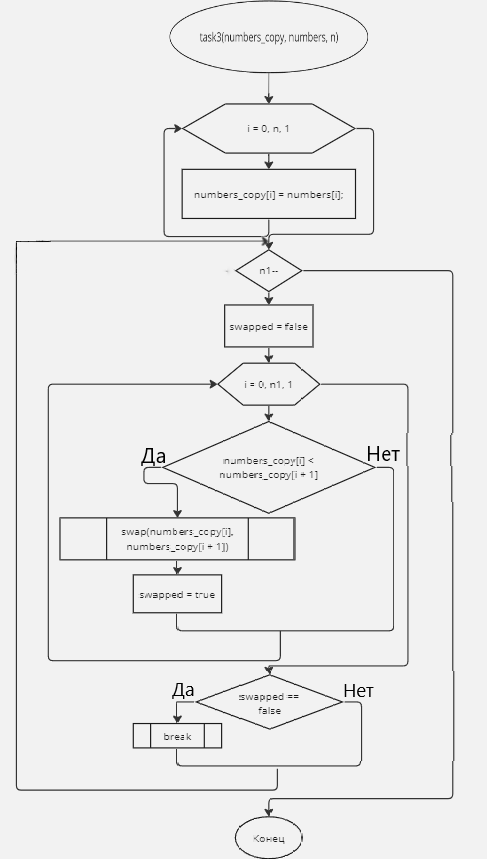
****

Рисунок 4 – Блок схема task3

1. **Текст программы**

// Для обнаружения утечек памяти

#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>;

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 requst\_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) // Cортировка 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;

requst\_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();

}

1. **Пример работы программы**

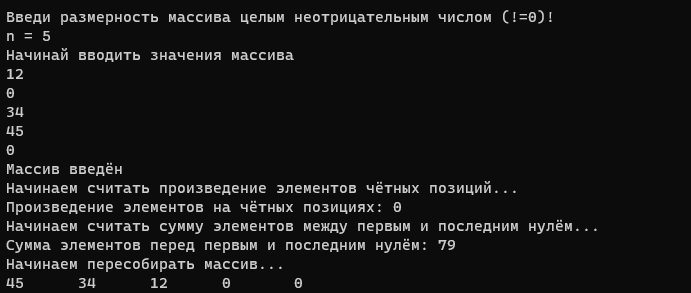


Рисунок 5 – Результат работы программы

Полученные данные совпадают с действительными

1. **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)