ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ  
«РОССИЙСКИЙ УНИВЕРСИТЕТ ТРАНСПОРТА»  
(РУТ (МИИТ))

Институт транспортной техники и систем управления

Кафедра «Управление и защита информации»

ОТЧЁТ  
О ПРАКТИЧЕСКОЙ РАБОТЕ № 1

По дисциплине «Языки программирования»

Выполнил: ст. гр. ТКИ – 111

Сапожников Сергей Михайлович

Проверил: к.т.н., доц.

Васильева М. А.

Москва 2023

Содержание

[Задание 4–1 3](#_Toc125074062)

# Задание 4–1

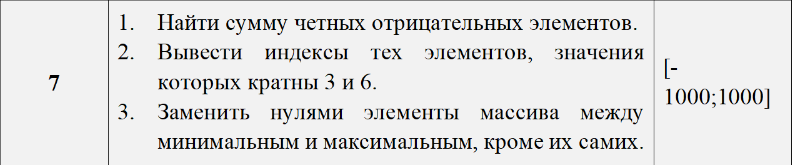
1. Формулировка задачи
2. Блок-схема алгоритма Блок-схемы алгоритмов функций представлены на рисунках (Рисунок 1, Рисунок 2, Рисунок 3,

Рисунок 4, Рисунок 5, Рисунок 6, Рисунок 7 , Рисунок 8 , Рисунок 9 , Рисунок 10 , Рисунок 11 , Рисунок 12).

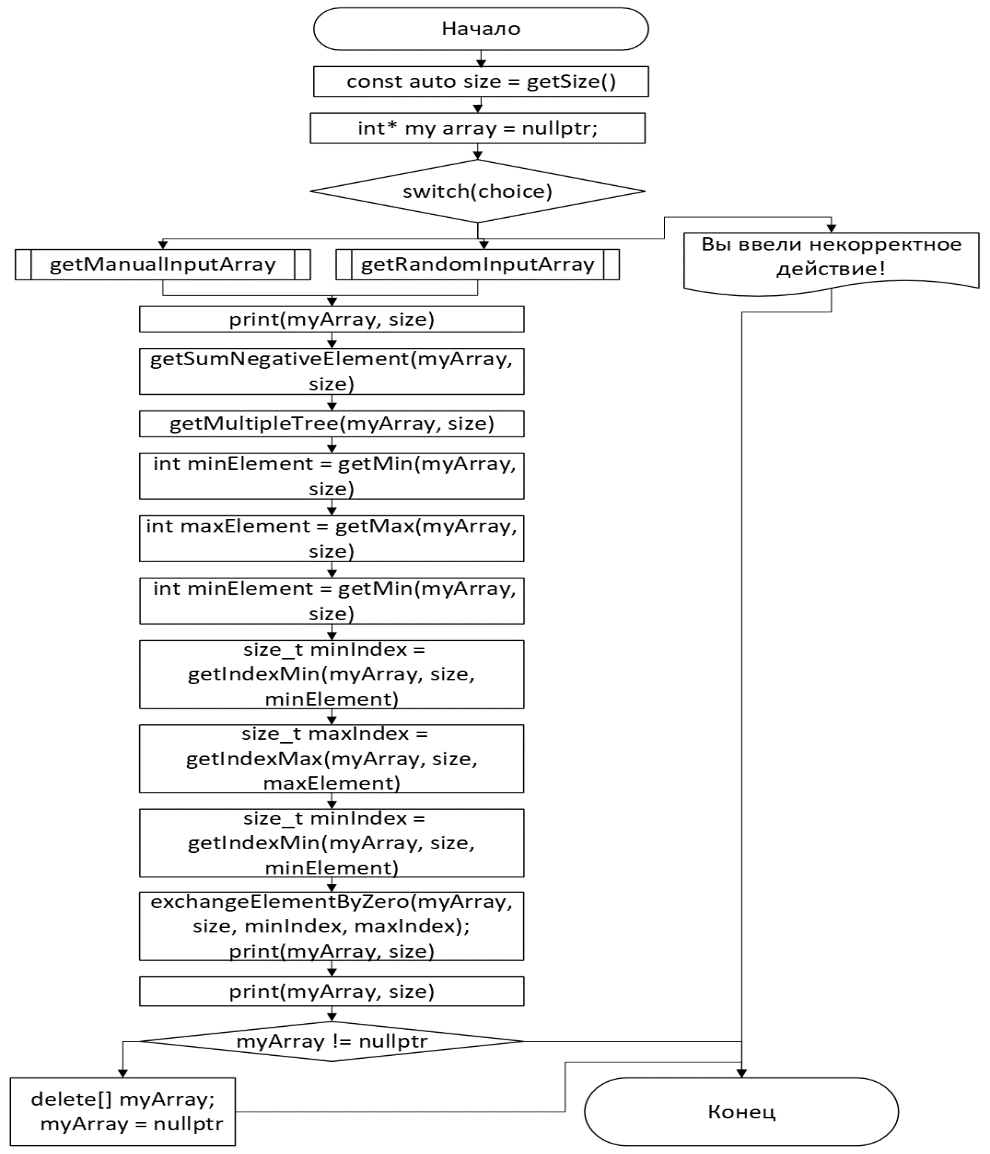


Рисунок 1 – Блок-схема функции main()

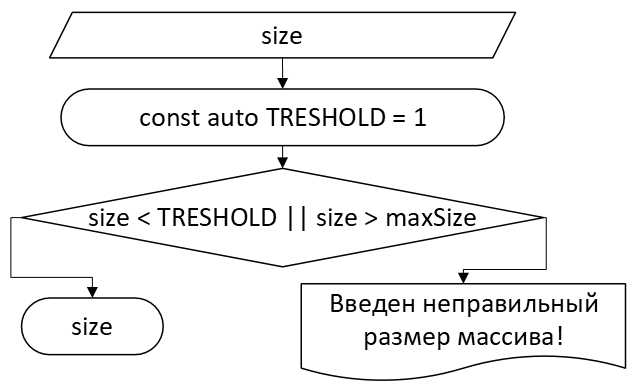


Рисунок 2 – Блок-схема функции getSize()

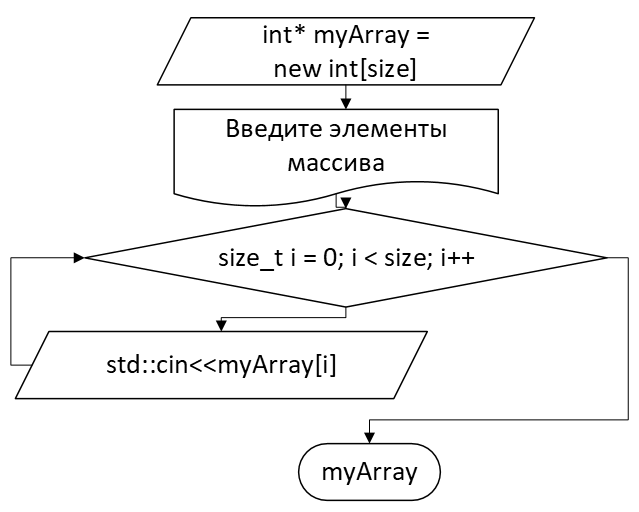


Рисунок 3 – Блок-схема функции int\* getManualInputArray()

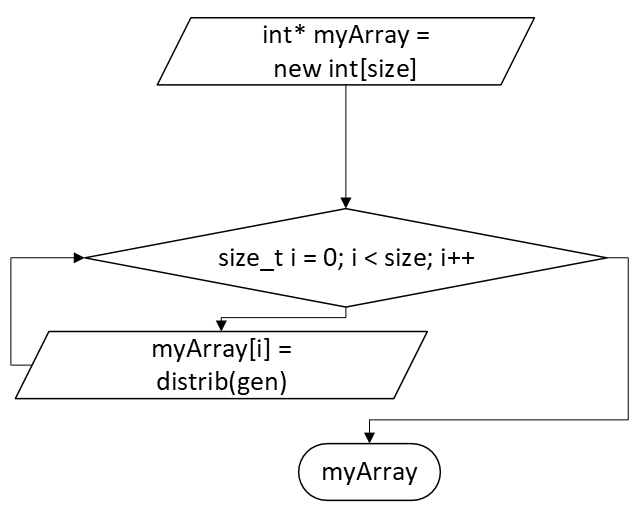


Рисунок 4 – Блок-схема функции int\* getRandomInputArray()

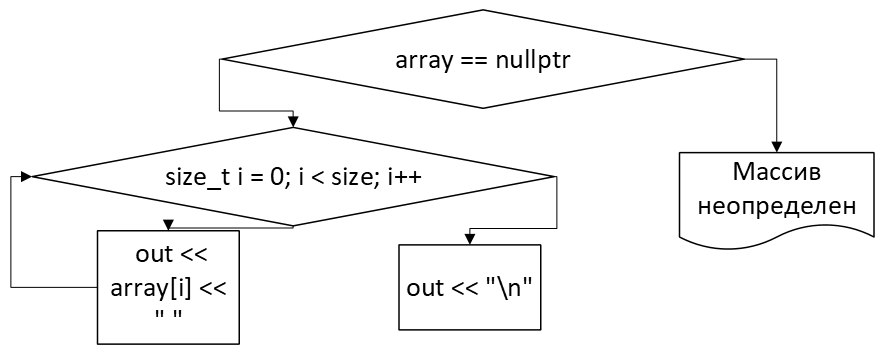


Рисунок 5 – Блок-схема функции print()

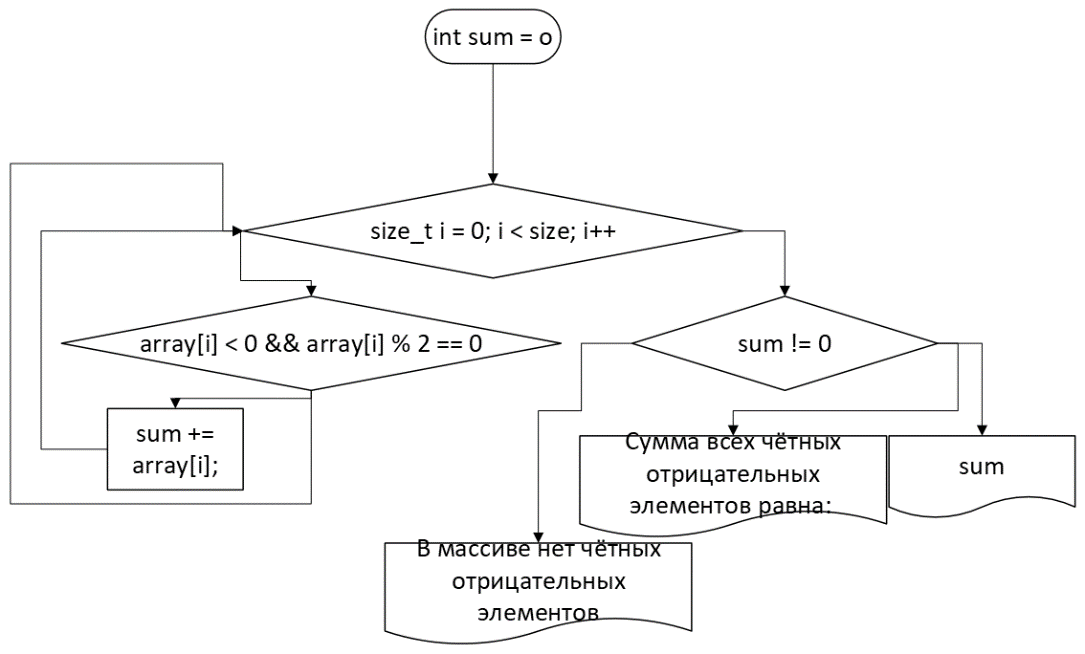


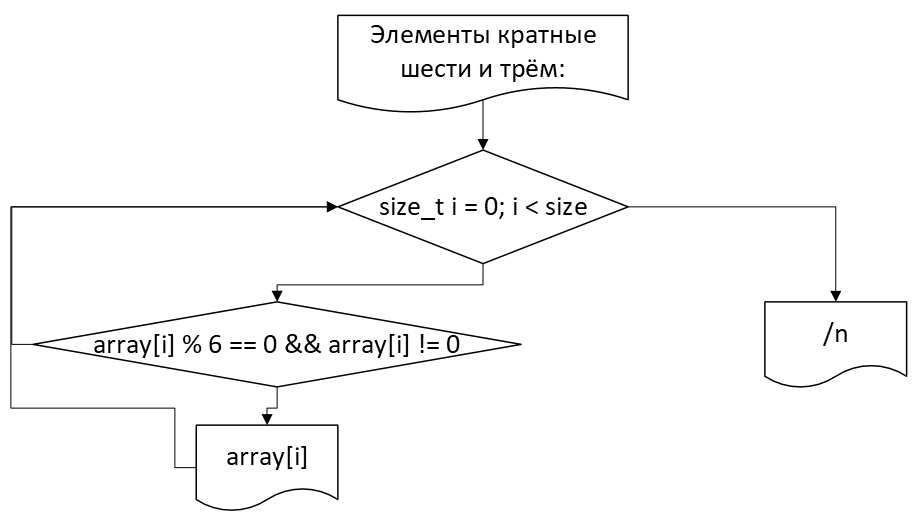
Рисунок 6 – Блок-схема функции getSumNegativeElement ()

Рисунок 7 – Блок-схема функции getMultipleTree ()

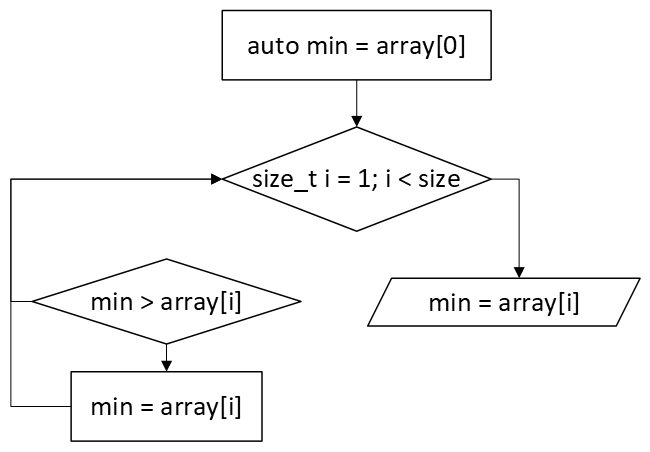


Рисунок 8 – Блок-схема функции getMin ()

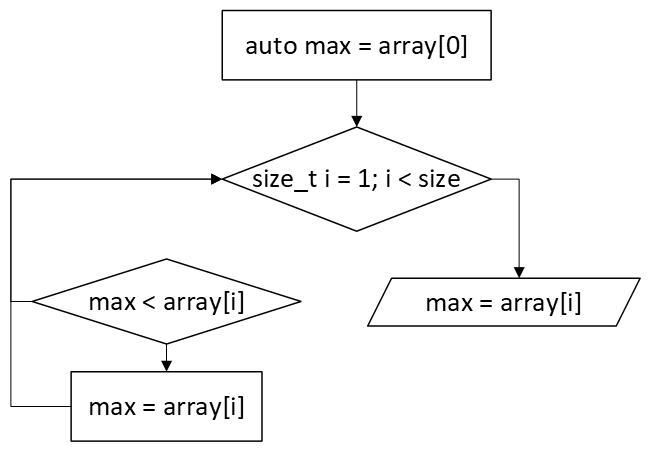


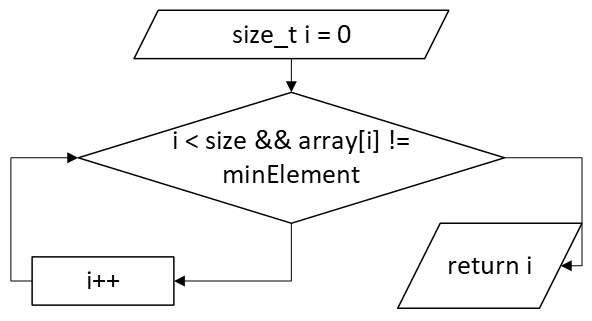
Рисунок 9 – Блок-схема функции getMax()

Рисунок 10 – Блок-схема функции getIndexMin ()

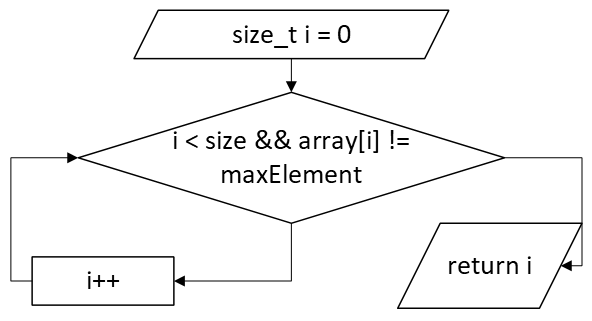


Рисунок 11 – Блок-схема функции getIndexMax ()

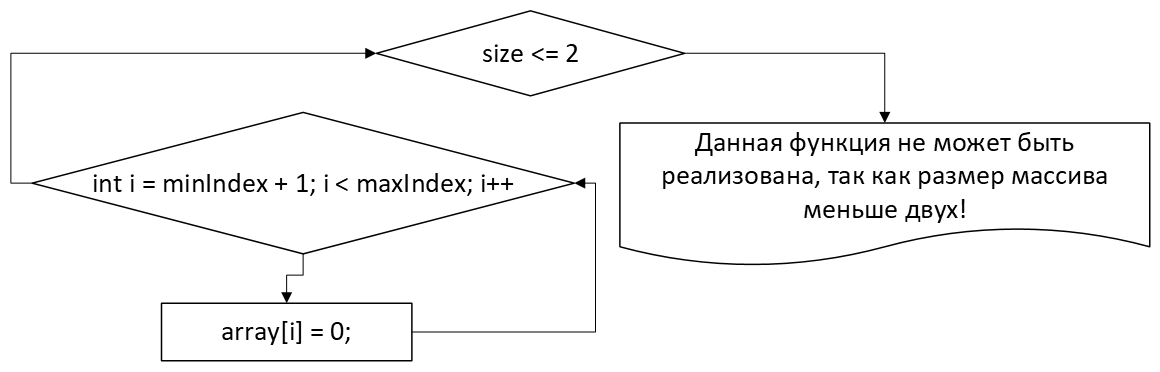


Рисунок 12 – Блок-схема функции exchangeElementByZero ()

1. Решение задачи на языке программирования C++

#include <string>

#include <iostream>

#include <random>

/\*

\* \brief Считывает размер массива.

\* \param message Сообщение пользователю.

\* \return Размер массива.

\*/

size\_t getSize(const std::string& message = "", size\_t maxSize = std::numeric\_limits<std::size\_t>::max());

/\*

\* \brief Считывает значения элементов массива с клавиатуры.

\* \param size длина массива.

\* \return Массив.

\*/

int\* getManualInputArray(const size\_t size);

/\*

\* \brief Заполнение массива случайными числами.

\* \param size Длина массива.

\* \param minValue Минимальное значение массива.

\* \param maxValue Максимальное значение массива.

\* \return Массив.

\*/

int\* getRandomInputArray(const size\_t size, const int minValue = -10, const int maxValue = 10);

/\*

\* \brief Печатает массив.

\* \param array Массив.

\* \param size длина массива.

\* \param out Поток вывода.

\*/

void print(int\* array, const size\_t size, std::ostream& out = std::cout);

/\*

\* \brief Считает сумму чётных отрицательных элементов.

\* \param array Массив.

\* \param size длина массива.

\* \return сумма чётных отрицательных элементов.

\*/

void getSumNegativeElement(int\* array, const size\_t size);

/\*

\* \brief Находит элементы кратные трём и шести и выводит их.

\* \param array Массив.

\* \param size длина массива.

\* \return максимальный по значению элемент.

\*/

void getMultipleTree(int\* array, const size\_t size);

/\*

\* \brief Находит минимальный по значению элемент.

\* \param array Массив.

\* \param size длина массива.

\* \return Минимальный по значению элемент.

\*/

int getMin(int\* array, const size\_t size);

/\*

\* \brief Находит максимальный по значению элемент.

\* \param array Массив.

\* \param size длина массива.

\* \return максимальный по значению элемент.

\*/

int getMax(int\* array, const size\_t size);

/\*

\* \brief Находит индекс минимального по значению элемента.

\* \param array Массив.

\* \param size длина массива.

\* \param minElement минимальный элемент

\* \return индекс минимального по значению элемента.

\*/

size\_t getIndexMin(int\* array, const size\_t size, int minElement);

/\*

\* \brief Находит индекс максимального по значению элемента.

\* \param array Массив.

\* \param size длина массива.

\* \param maxElement максимальный элемент

\* \return индекс максимального по значению элемента.

\*/

size\_t getIndexMax(int\* array, const size\_t size, int maxElement);

/\*

\* \brief Меняет все элементы массива между Минимальным и максимальным элементом на нули.

\* \param array Массив.

\* \param size длина массива.

\* \param minIndex индекс минимального элемента

\* \param maxIndex индекс максимального элемента

\* \return Массив измененный.

\*/

void exchangeElementByZero(int\* array, const size\_t size, size\_t minIndex, size\_t maxIndex);

/\*

\* \brief Класс для выбора заполнения массива.

\* \param MANUAL Ручной способ.

\* \param RANDOM Заполнение массива случайными числами.

\*/

enum class arrayInputChoice

{

MANUAL,

RANDOM

};

/\*

\* \brief Точка входа в программу.

\* \return В случае успеха возвращает 0.

\*/

int main()

{

try {

const auto size = getSize("Введите размер массива = ");

int\* myArray = nullptr;

std::cout << "Введите требуемое действие \n" << static\_cast<int>(arrayInputChoice::MANUAL) << " для ручного ввода массива\n"

<< static\_cast<int>(arrayInputChoice::RANDOM) << " для заполнения массива случайными числами\n";

int userInput = 0;

std::cin >> userInput;

const auto choice = static\_cast<arrayInputChoice>(userInput);

switch (choice)

{

case arrayInputChoice::MANUAL:

{

myArray = getManualInputArray(size);

break;

}

case arrayInputChoice::RANDOM:

{

myArray = getRandomInputArray(size);

break;

}

default:

std::cerr << "Вы ввели некорректное действие!";

return 1;

}

print(myArray, size);

getSumNegativeElement(myArray, size);

getMultipleTree(myArray, size);

int minElement = getMin(myArray, size);

int maxElement = getMax(myArray, size);

size\_t minIndex = getIndexMin(myArray, size, minElement);

size\_t maxIndex = getIndexMax(myArray, size, maxElement);

exchangeElementByZero(myArray, size, minIndex, maxIndex);

print(myArray, size);

if (myArray != nullptr)

{

delete[] myArray;

myArray = nullptr;

}

return 0;

}

catch (std::exception& e)

{

std::cerr << e.what();

return 1;

}

}

size\_t getSize(const std::string& message, size\_t maxSize)

{

const auto TRESHOLD = 1;

int size = TRESHOLD;

std::cout << message;

std::cin >> size;

if (size < TRESHOLD || size > maxSize)

{

throw std::out\_of\_range("Введен неправильный размер массива!");

}

return size;

}

int\* getManualInputArray(const size\_t size)

{

int\* myArray = new int[size];

std::cout << "Введите элементы массива\n";

for (size\_t i = 0; i < size; i++)

{

std::cout << "A[" << i + 1 << "] = ";

std::cin >> myArray[i];

}

return myArray;

}

int\* getRandomInputArray(const size\_t size, const int minValue, const int maxValue)

{

std::random\_device random;

std::mt19937 gen(random());

std::uniform\_int\_distribution<> distrib(minValue, maxValue);

int\* myArray = new int[size];

for (size\_t i = 0; i < size; i++)

{

myArray[i] = distrib(gen);

}

return myArray;

}

void print(int\* array, const size\_t size, std::ostream& out)

{

if (array == nullptr)

{

throw std::out\_of\_range("Массив не определен!");

}

out << "Массив:\n";

for (size\_t i = 0; i < size; i++)

{

out << array[i] << " ";

}

out << "\n";

}

void getSumNegativeElement(int\* array, const size\_t size)

{

int sum = 0;

for (size\_t i = 0; i < size; i++)

{

if (array[i] < 0 && array[i] % 2 == 0)

{

sum += array[i];

}

}

if (sum != 0)

{

std::cout << "Сумма всех чётных отрицательных элементов равна: " << sum << "\n";

}

else

std::cout << "В массиве нет чётных отрицательных элементов\n";

}

void getMultipleTree(int\* array, const size\_t size)

{

std::cout << "Элементы кратные шести и трём:";

for (size\_t i = 0; i < size; i++)

{

if (array[i] % 6 == 0 && array[i] != 0)

{

std::cout << " " << array[i];

}

}

std::cout << "\n";

}

int getMin(int\* array, const size\_t size)

{

auto min = array[0];

for (size\_t i = 1; i < size; i++)

{

if (min > array[i])

{

min = array[i];

}

}

return min;

}

int getMax(int\* array, const size\_t size)

{

auto max = array[0];

for (size\_t i = 1; i < size; i++)

{

if (max < array[i])

{

max = array[i];

}

}

return max;

}

void exchangeElementByZero(int\* array, const size\_t size, size\_t minIndex, size\_t maxIndex)

{

if (size <= 2)

{

std::cout << "Данная функция не может быть реализована, так как размер массива меньше двух!\n";

return;

}

else

{

for (int i = minIndex + 1; i < maxIndex; i++)

{

array[i] = 0;

}

}

}

size\_t getIndexMin(int\* array, const size\_t size, int minElement)

{

size\_t i = 0;

while (i < size && array[i] != minElement )

{

i++;

}

return i;

}

size\_t getIndexMax(int\* array, const size\_t size, int maxElement)

{

size\_t i = 0;

while (i < size && array[i] != maxElement )

{

i++;

}

return i;

}

1. Решение тестовых примеров

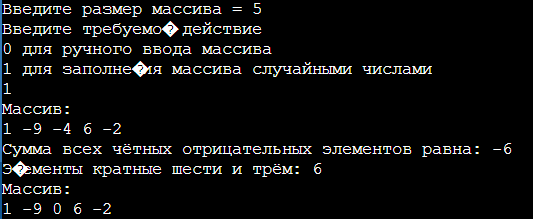


Рисунок 13 – Решение тестового примера

1. Решение тестовых примеров

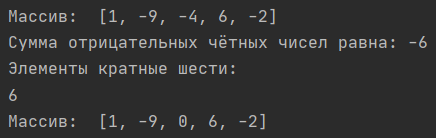
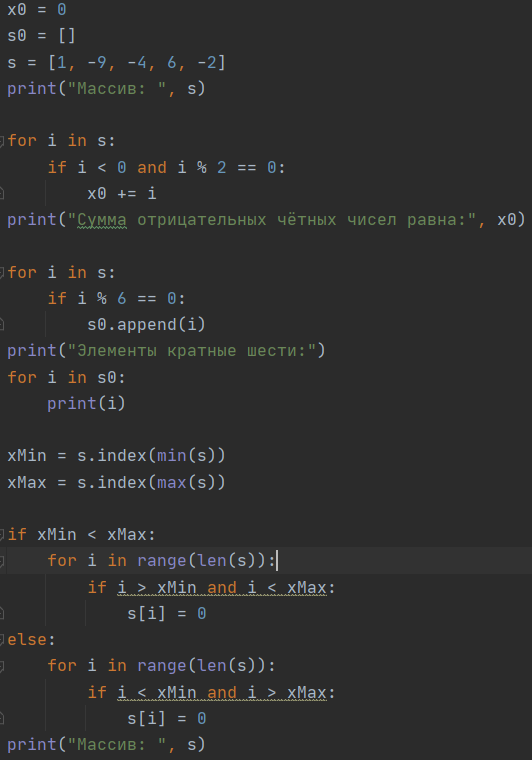


Рисунок 14 –Расчет значения

1. Зачет задания в GitHub

Жду апрув 4.1:)

Рисунок 15– Зачет задания