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 ЛАБОРАТОРНОЙ РАБОТЕ №4

По дисциплине «Языки программирования»

ВАРИАНТ 8

Выполнил: ст. гр. ТКИ - 142

Лазарев Андрей Александрович

Проверил: к.т.н., доц. Васильева М. А.

(Проверил: к.т.н, доц. Балакина Е. П.)

Москва 2023

Оглавление

[1. Задание 4.1 3](#_Toc153797508)

[1.1. Формулировка задания 3](#_Toc153797509)

[1.2. Блок-схема алгоритма 4](#_Toc153797510)

[1.3. Код задания 4.1 9](#_Toc153797511)

[1.4. Решение тестового примера 15](#_Toc153797512)

[1.5. Зачёт задания в GitHub 18](#_Toc153797513)

[2. Задание 4.2 19](#_Toc153797514)

[2.1. Формулировка задания 19](#_Toc153797515)

[2.2. Блок-схема алгоритма 20](#_Toc153797516)

[2.3. Код задания 4.2 28](#_Toc153797517)

[2.4. Решение тестового примера 35](#_Toc153797518)

[2.5. Решение тестовых примеров в Excel 35](#_Toc153797519)

[2.6. Зачёт задания в GitHub 35](#_Toc153797520)

[3. Задание 4.3 35](#_Toc153797521)

[3.1. Формулировка задания 35](#_Toc153797522)

[3.2. Блок-схема алгоритма 35](#_Toc153797523)

[3.3. Код задания 4.3 35](#_Toc153797524)

[3.4. Решение тестового примера 35](#_Toc153797525)

[3.5. Решение тестовых примеров в Excel 35](#_Toc153797526)

[3.6. Зачёт задания в GitHub 35](#_Toc153797527)

# Задание 4.1

* 1. Формулировка задания

Создать одномерный массив из *n* целых чисел. Реализовать возможность заполнения массива, как случайными числами, так и с помощью клавиатуры по желанию пользователя. Пункты задания организовать в виде функций (методов). Составить блок-схему.

Таблица 1 –Формулировка задания 4.1

|  |  |  |
| --- | --- | --- |
| Вариант | Задачи | Интервал |
| 8 | 1. Заменить минимальный элемент массива на средний (количество элементов – нечетно). 2. Вывести индексы тех элементов, значения которых больше значения предыдущего элемента (начиная со второго). 3. Определить, есть ли две пары соседних элементов с одинаковыми знаками. | [20;40] |

* 1. Блок-схема алгоритма

Блок-схема алгоритма представлена на рисунках (Рисунок 1, Рисунок 2, Рисунок 3, Рисунок 4, Рисунок 5, Рисунок 6, Рисунок 7, Рисунок 8, Рисунок 9, Рисунок 10, Рисунок 11)

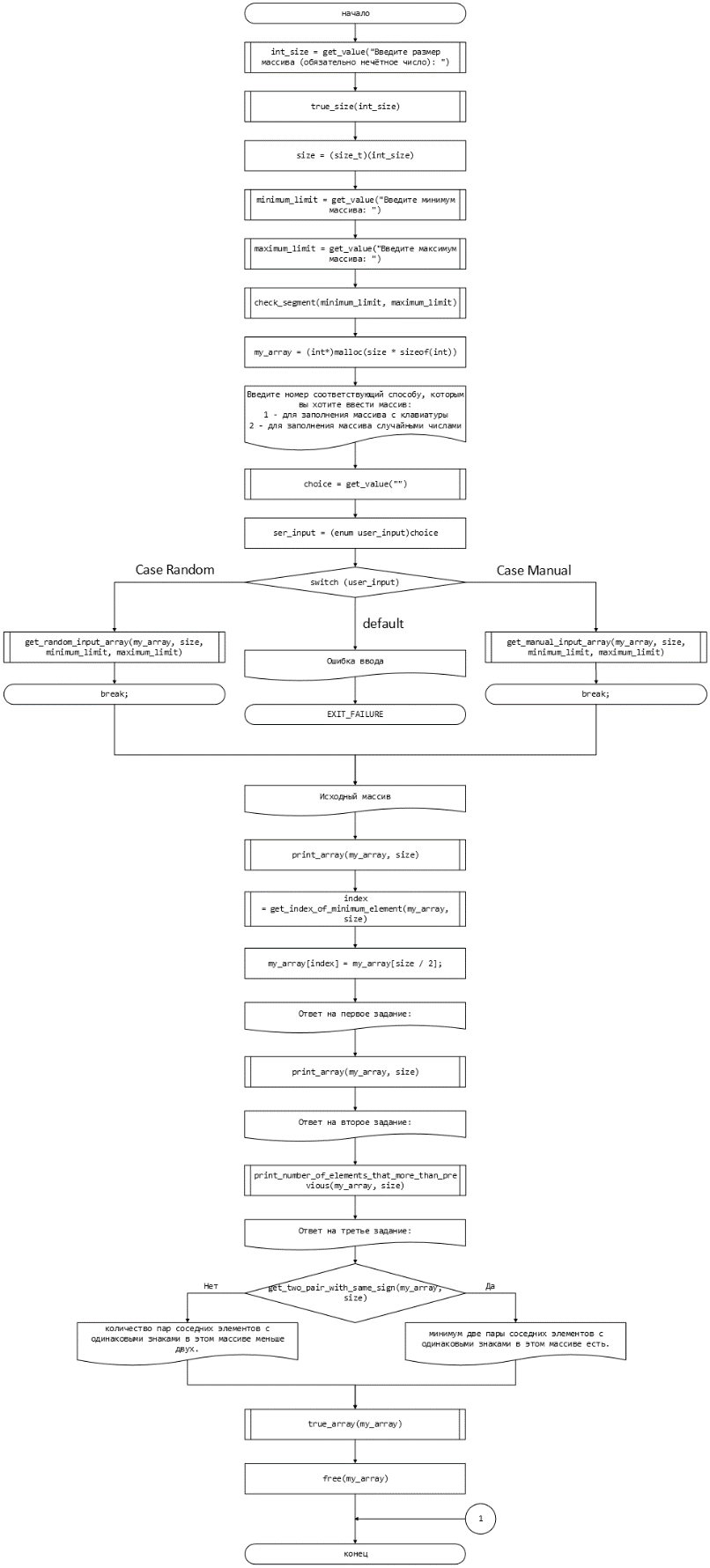


Рисунок 1 – Блок-схема функции main()

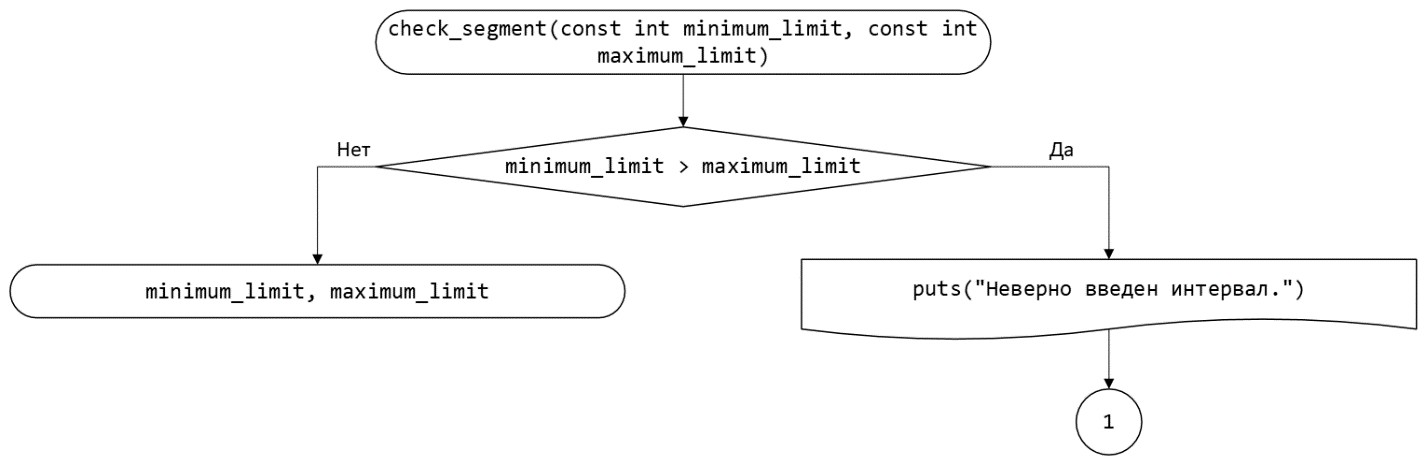


Рисунок 2 – Блок-схема функции check\_segment(minimum\_limit, maximum\_limit)

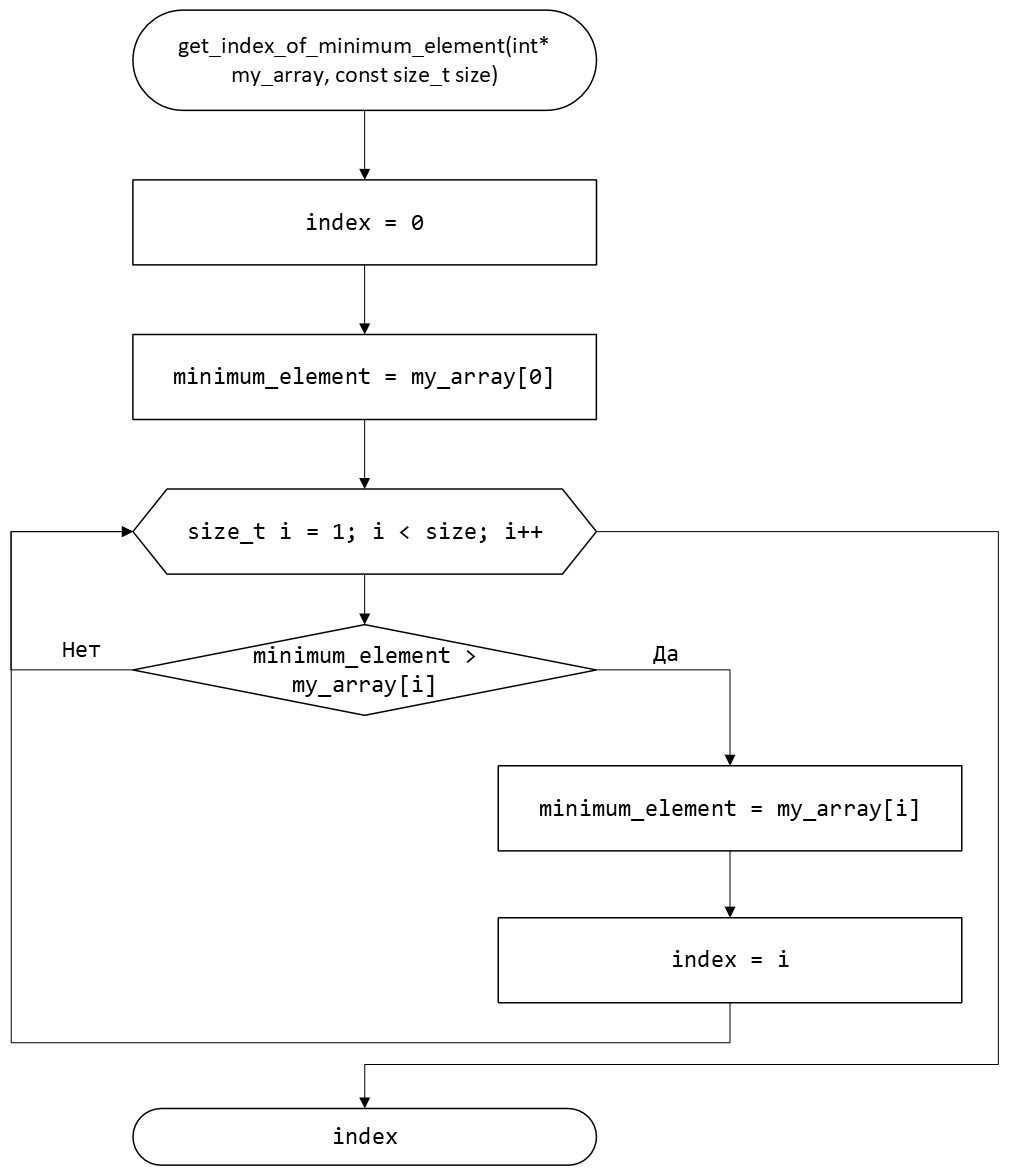


Рисунок 3 – Блок-схема функции get\_index\_of\_minimum\_element(my\_array, size)

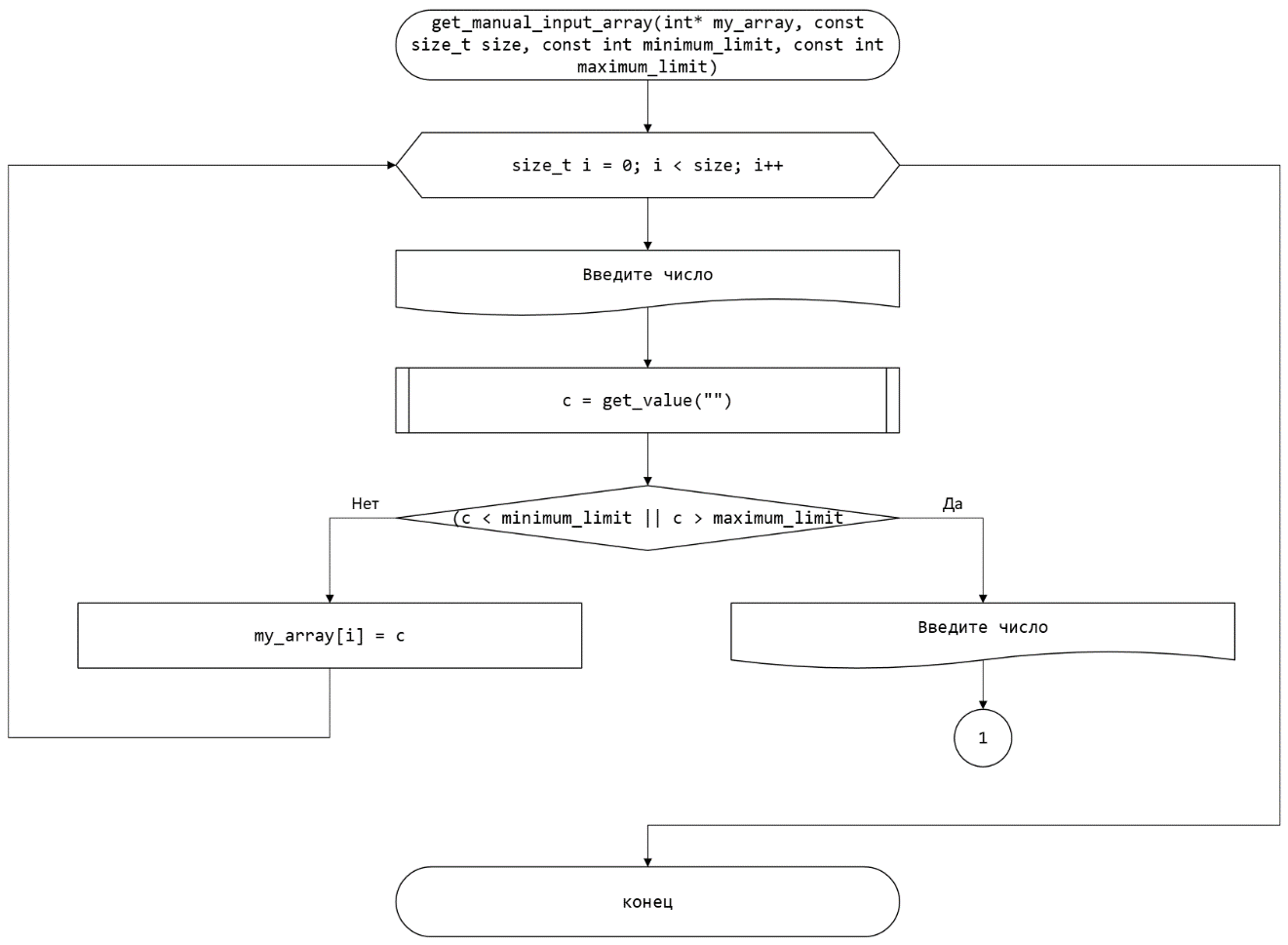


Рисунок 4 – Блок-схема функции get\_manual\_input\_array(my\_array, size, minimum\_limit, maximum\_limit)

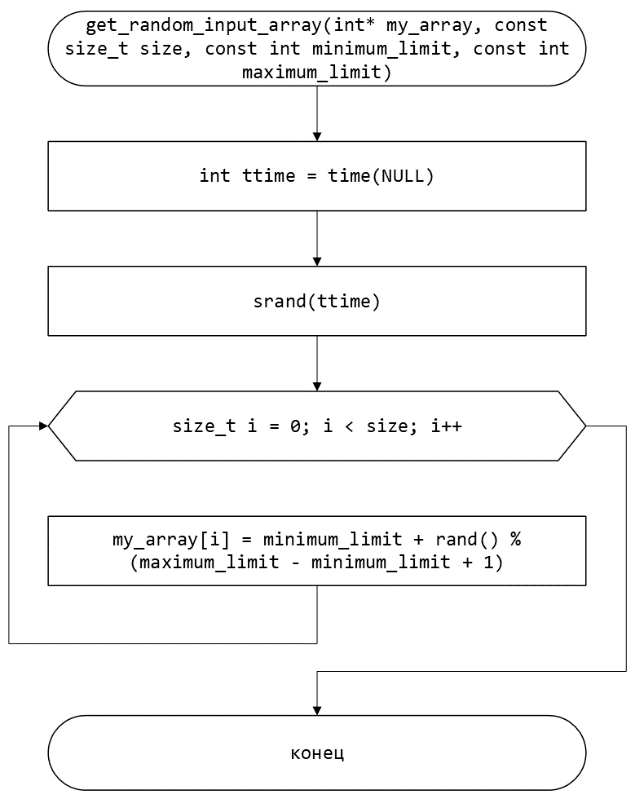


Рисунок 5 – Блок-схема функции get\_random\_input\_array(my\_array, size, minimum\_limit, maximum\_limit)

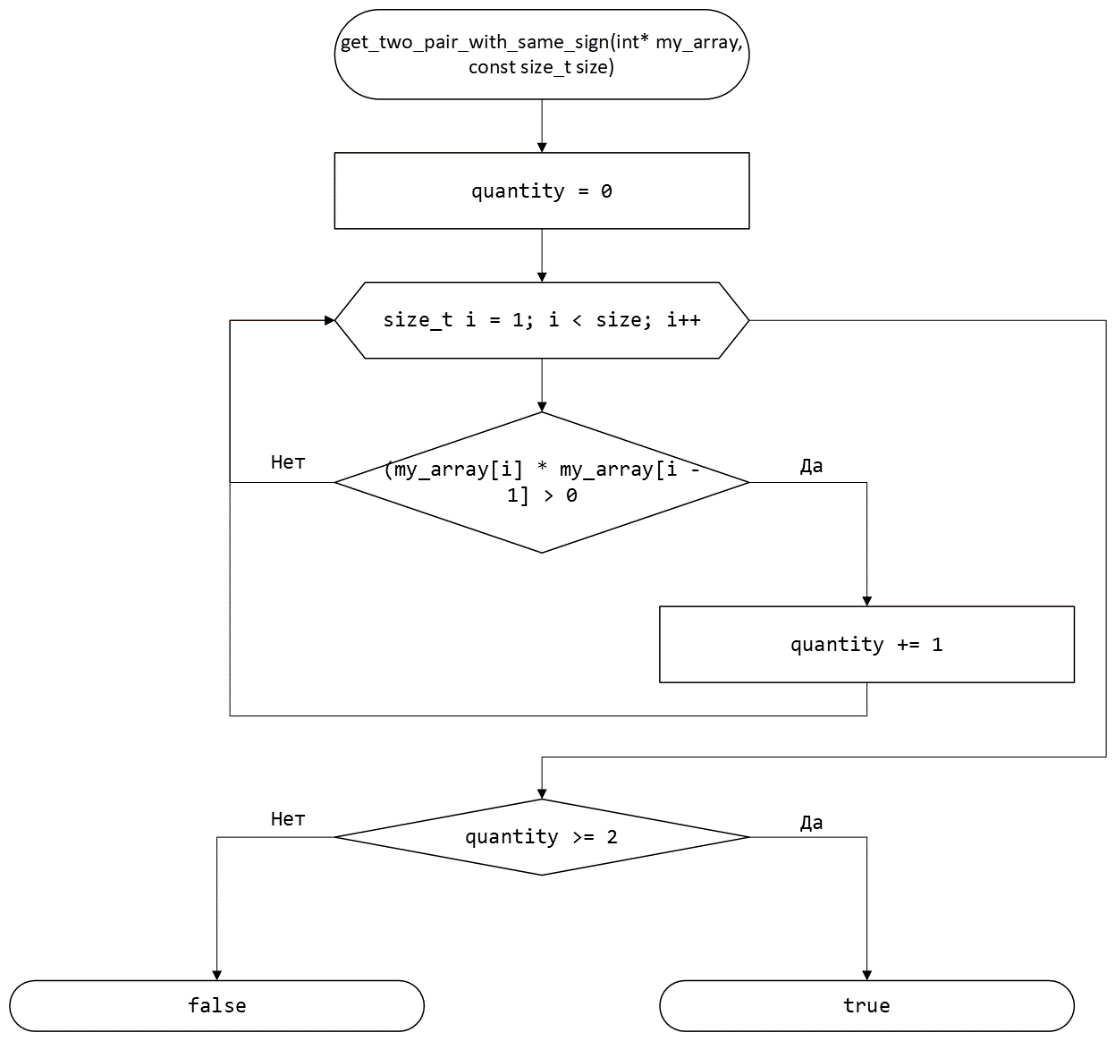


Рисунок 6 – Блок-схема функции get\_two\_pair\_with\_same\_sign(my\_array, size)

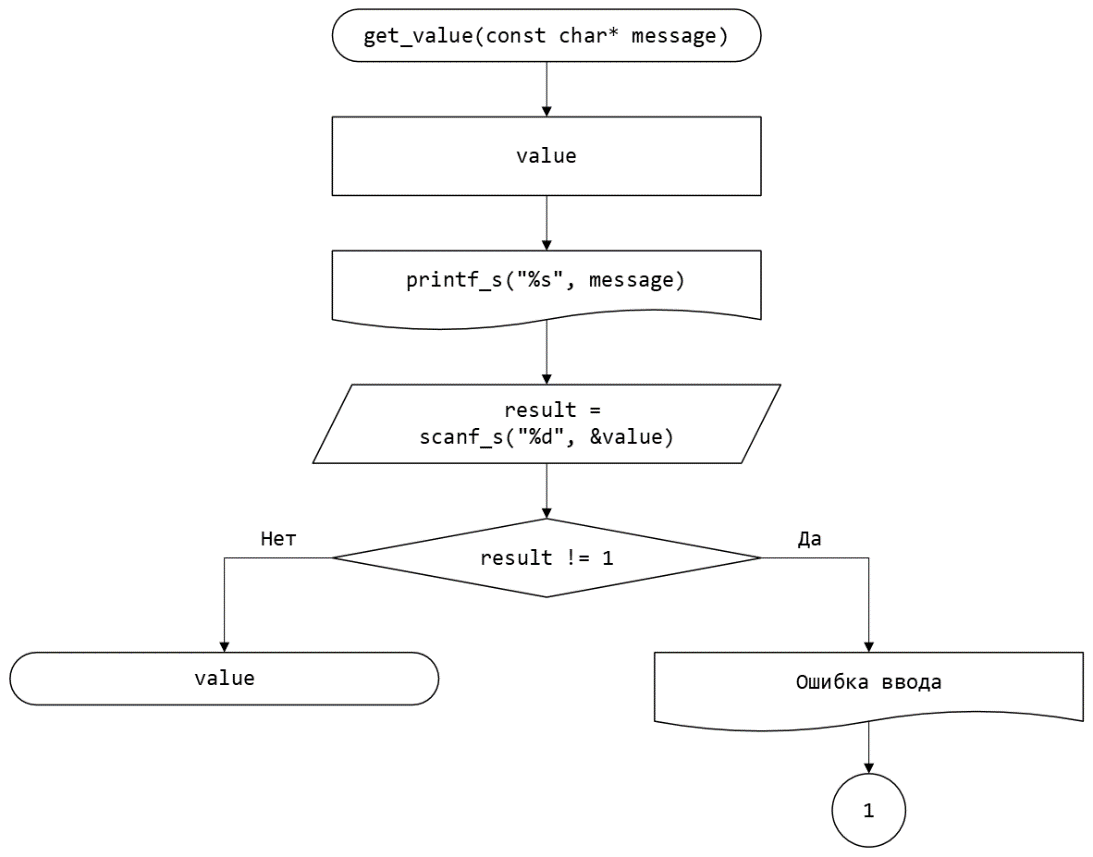


Рисунок 7 – Блок-схема функции get\_value(message)

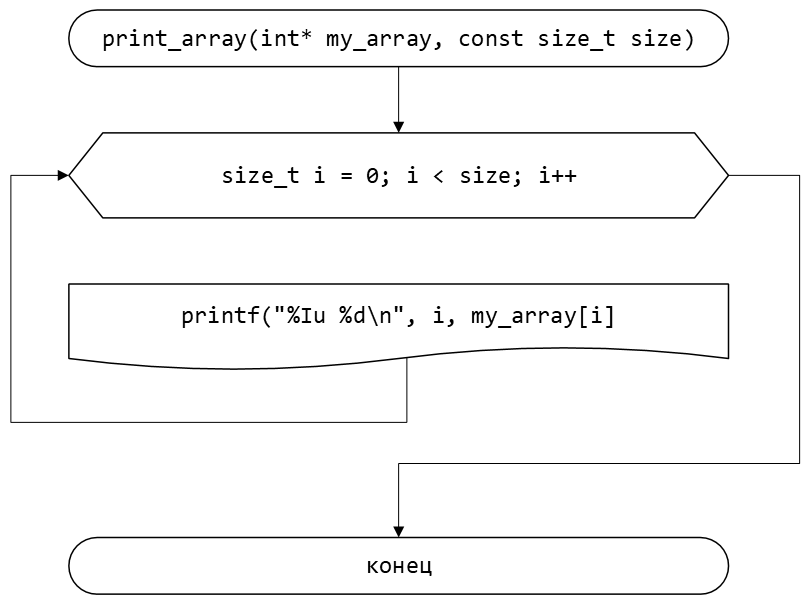


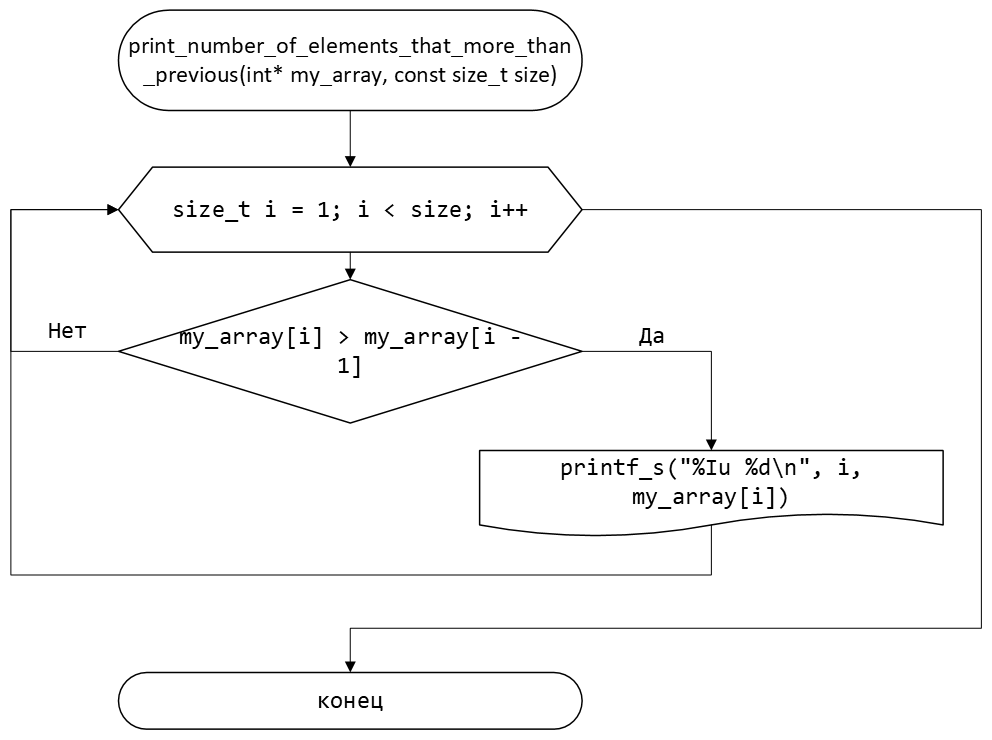
Рисунок 8 – Блок-схема функции print\_array(my\_array, size)

Рисунок 9 – Блок-схема функции print\_number\_of\_elements\_that\_more\_than\_previous(my\_array, size)

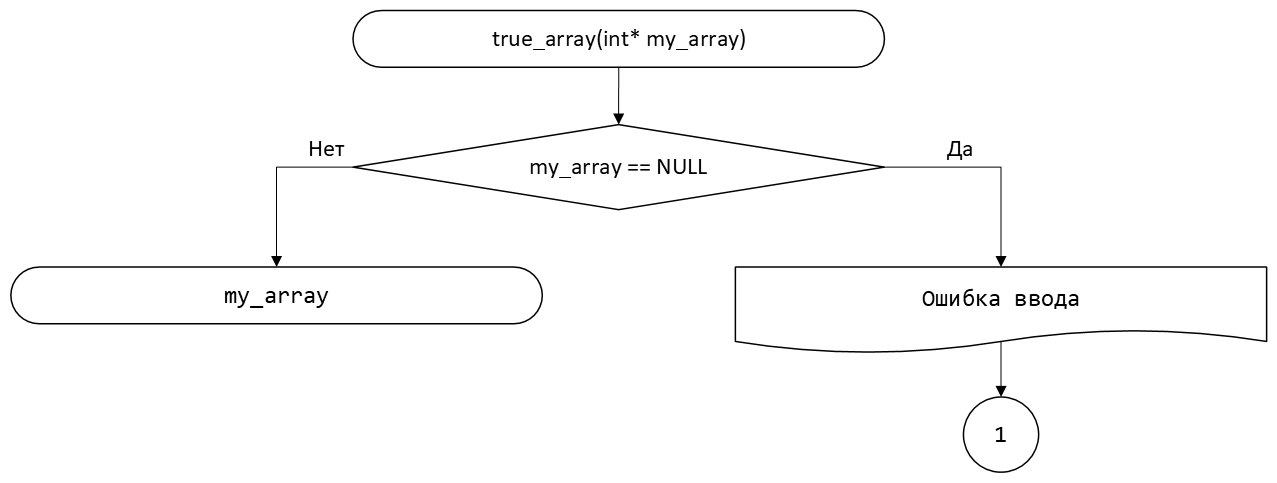


Рисунок 10 – Блок-схема функции true\_array(my\_array)

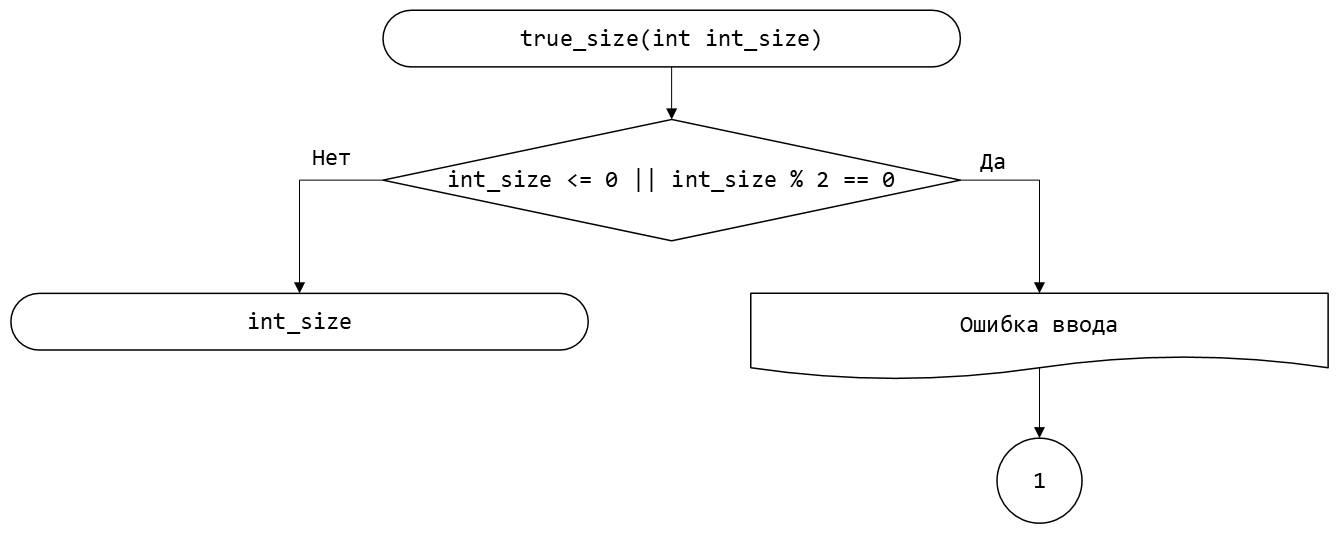


Рисунок 11 – Блок-схема функции true\_size(size)

* 1. Код задания 4.1

#include <stdio.h>

#include <locale.h>

#include <errno.h>

#include <math.h>

#include <float.h>

#include <stdlib.h>

#include <time.h>

#include <stdbool.h>

/\*\*

\* @brief Выбор заполнения массива.

\*/

enum array\_input\_choice

{

/\*\*

\* @brief Ручной способ.

\*/

MANUAL = 1,

/\*\*

\* @brief Заполнение массива случайными числами.

\*/

RANDOM = 2

};

/\*\*

\* @brief Функция для считывания значения с клавиатуры.

\* @param message - сообщение пользователю.

\* @remarks Экстренное завершение программы, в случае неправильного ввода.

\* @return Значение с клавиатуры.

\*/

int get\_value(const char\* message);

/\*\*

\* @brief Функция для проверки длинны массива.

\* @param int\_size - длинна массива.

\* @remarks Экстренное завершение программы, в случае неправильной длинны массива.

\*/

void true\_size(int int\_size);

/\*\*

\* @brief Функция для проверки массива.

\* @param my\_array - массив.

\* @remarks Экстренное завершение программы, в случае несуществования массива.

\*/

void true\_array(int\* my\_array);

/\*\*

\* @brief Функция для считывания элементов массива с клавиатуры.

\* @param size - длинна массива.

\* @param my\_array - массив.

\*/

void get\_manual\_input\_array(int\* my\_array, const size\_t size, const int minimum\_limit, const int maximum\_limit);

/\*\*

\* @brief Функция для заполнения массива случайными числами.

\* @param size - длинна массива.

\* @param my\_array - массив.

\*/

void get\_random\_input\_array(int\* my\_array, const size\_t size, const int minimum\_limit, const int maximum\_limit);

/\*\*

\* @brief Функция для печати массива.

\* @param my\_array - массив.

\* @param size - длинна массива.

\* @return Массив.

\*/

void print\_array(int\* my\_array, const size\_t size);

/\*\*

\* @brief Функция для нахождения минимального элемента массива и замены среднего элемента на него.

\* @param my\_array - массив.

\* @param size - длинна массива.

\*/

int get\_index\_of\_minimum\_element(int\* my\_array, const size\_t size);

/\*\*

\* @brief Функция вывода индексов элементов, значение которых больше предыдущего (начиная со второго).

\* @param my\_array - массив.

\* @param size - длинна массива.

\*/

void print\_number\_of\_elements\_that\_more\_than\_previous(int\* my\_array, const size\_t size);

/\*\*

\* @brief Функция для нахождения количества пар соседних элементов с разными знаками.

\* @param my\_array - массив.

\* @param size - длинна массива.

\* @return Возвращает истину в случае успеха.

\*/

bool get\_two\_pair\_with\_same\_sign(int\* my\_array, const size\_t size);

/\*\*

\* @brief Функция проверки интервала массива на правильность.

\* @param minimum\_limit - минимальный возможный элемент массива.

\* @param maximum\_limit - максимальный возможный элемент массива.

\*/

void check\_segment(const int minimum\_limit, const int maximum\_limit);

/\*\*

\* @brief Точка входа в программу.

\* @return \c 0 в случае успеха, иначе \c 1.

\*/

int main()

{

setlocale(LC\_ALL, "RU");

const int int\_size = get\_value("Введите размер массива (обязательно нечётное число): ");

true\_size(int\_size);

size\_t size = (size\_t)(int\_size);

const int minimum\_limit = get\_value("Введите минимум массива: ");

const int maximum\_limit = get\_value("Введите максимум массива: ");

check\_segment(minimum\_limit, maximum\_limit);

int\* my\_array = (int\*)malloc(size \* sizeof(int));

puts("Введите номер соответствующий способу, которым вы хотите ввести массив:\n");

printf\_s("%d - для заполнения массива с клавиатуры\n", MANUAL);

printf\_s("%d - для заполнения массива случайными числами\n", RANDOM);

int choice = get\_value("");

enum array\_input\_choice user\_input = (enum user\_input)choice;

switch (user\_input)

{

case MANUAL:

{

get\_manual\_input\_array(my\_array, size, minimum\_limit, maximum\_limit);

break;

}

case RANDOM:

{

get\_random\_input\_array(my\_array, size, minimum\_limit, maximum\_limit);

break;

}

default:

{

errno = EDOM;

perror("Ошибка ввода");

return EXIT\_FAILURE;

}

}

puts("Исходный массив");

print\_array(my\_array, size);

my\_array[get\_index\_of\_minimum\_element(my\_array, size)] = my\_array[size / 2];

puts("Ответ на первое задание:");

print\_array(my\_array, size);

printf\_s("Ответ на второе задание:\n");

print\_number\_of\_elements\_that\_more\_than\_previous(my\_array, size);

printf\_s("Ответ на третье задание: ");

if (get\_two\_pair\_with\_same\_sign(my\_array, size))

{

printf("%s", "минимум две пары соседних элементов с одинаковыми знаками в этом массиве есть.");

}

else

{

printf("%s", "количество пар соседних элементов с одинаковыми знаками в этом массиве меньше двух.");

}

true\_array(my\_array);

free(my\_array);

return 0;

}

int get\_value(const char\* message)

{

int value;

printf\_s("%s", message);

int result = scanf\_s("%d", &value);

if (result != 1)

{

errno = EIO;

perror("Ошибка ввода");

abort();

}

return value;

}

void true\_size(int int\_size)

{

if (int\_size <= 0 || int\_size % 2 == 0)

{

errno = EIO;

perror("Ошибка ввода");

abort();

}

}

void true\_array(int\* my\_array)

{

if (my\_array == NULL)

{

errno = EIO;

perror("Ошибка ввода");

abort();

}

}

void get\_manual\_input\_array(int\* my\_array, const size\_t size, const int minimum\_limit, const int maximum\_limit)

{

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

{

printf("Введите число от %d до %d: ", minimum\_limit, maximum\_limit);

int c = get\_value("");

if (c < minimum\_limit || c > maximum\_limit)

{

errno = EIO;

perror("Ошибка ввода");

abort();

}

my\_array[i] = c;

}

}

void get\_random\_input\_array(int\* my\_array, const size\_t size, const int minimum\_limit, const int maximum\_limit)

{

unsigned int ttime = time(NULL);

srand(ttime);

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

{

my\_array[i] = minimum\_limit + rand() % (maximum\_limit - minimum\_limit + 1);

}

}

int get\_index\_of\_minimum\_element(int\* my\_array, const size\_t size)

{

int index = 0;

int minimum\_element = my\_array[0];

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

{

if (minimum\_element > my\_array[i])

{

minimum\_element = my\_array[i];

index = i;

}

}

return index;

}

void print\_number\_of\_elements\_that\_more\_than\_previous(int\* my\_array, const size\_t size)

{

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

{

if (my\_array[i] > my\_array[i - 1])

{

printf\_s("%Iu %d\n", i, my\_array[i]);

}

}

}

bool get\_two\_pair\_with\_same\_sign(int\* my\_array, const size\_t size)

{

int quantity = 0;

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

{

if (my\_array[i] \* my\_array[i - 1] > 0)

{

quantity += 1;

}

}

if (quantity >= 2)

{

return true;

}

return false;

}

void print\_array(int\* my\_array, const size\_t size)

{

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

{

printf("%Iu %d\n", i, my\_array[i]);

}

}

void check\_segment(const int minimum\_limit, const int maximum\_limit)

{

if (minimum\_limit > maximum\_limit)

{

puts("Неверно введен интервал.");

abort();

}

}

* 1. Решение тестового примера

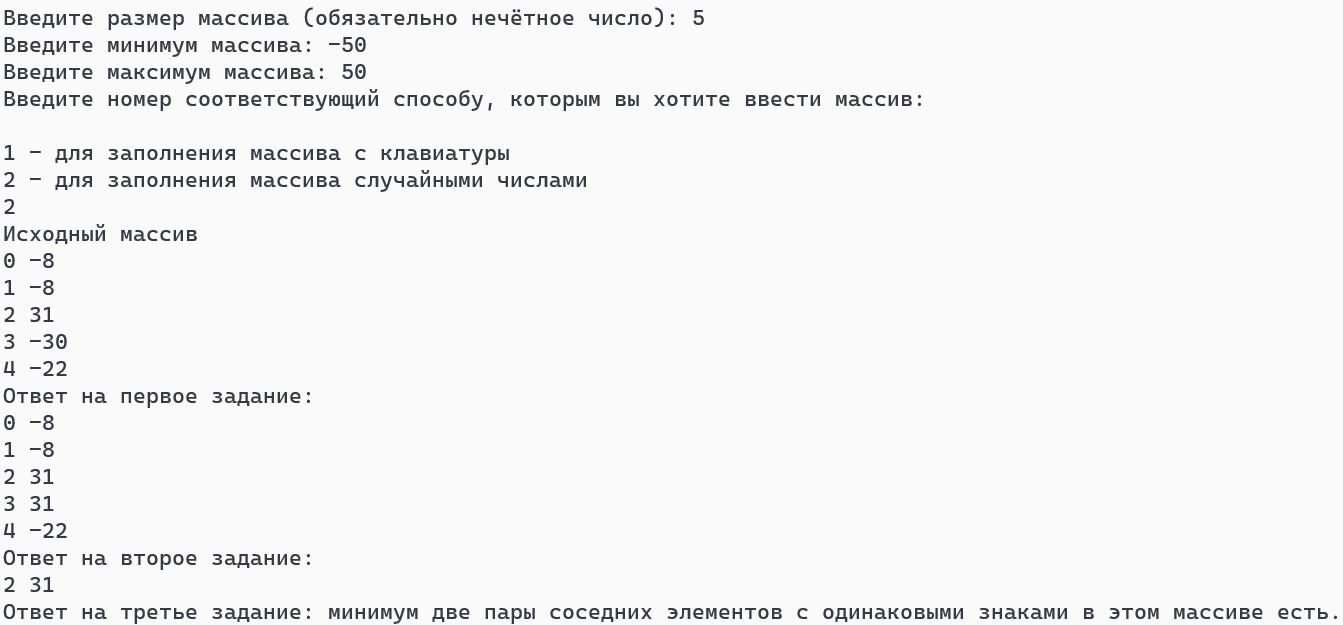


Рисунок 12 – Решение тестового примера рандомным заполнением массива

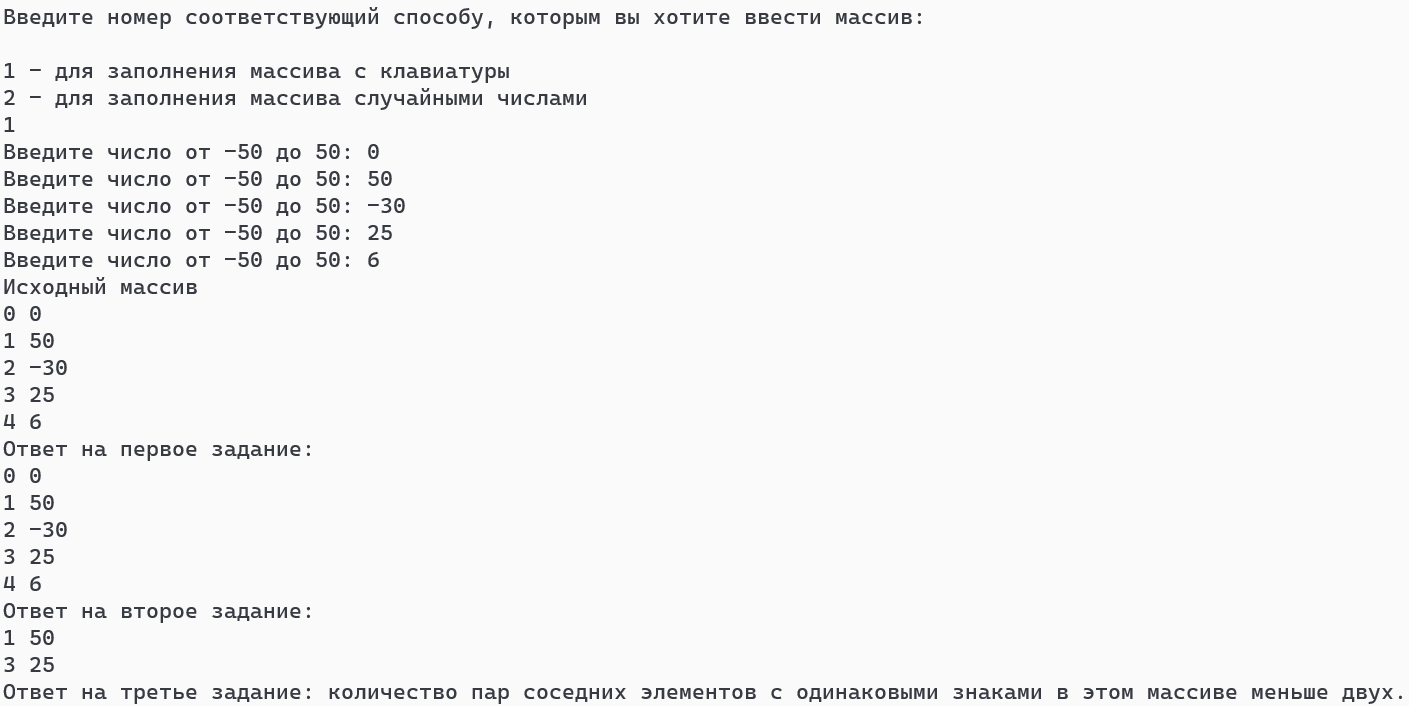


Рисунок 13 – Решение тестового примера пользовательским заполнением массива



Рисунок 14 – Вывод программы, когда размер массива – чётное число



Рисунок 15 – Вывод программы, когда размер массива – буква



Рисунок 16 – Вывод программы, когда размер массива – отрицательное число

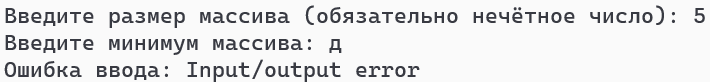


Рисунок 17 – Вывод программы, когда минимум массива – буква

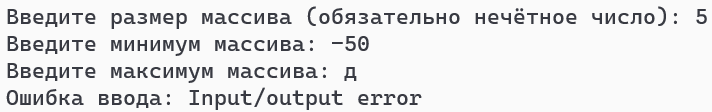


Рисунок 18 – Вывод программы, когда максимум массива – буква

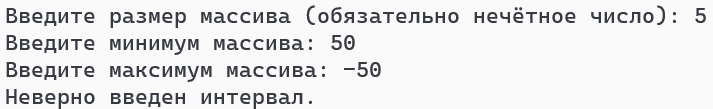


Рисунок 19 – Вывод программы, когда минимум массива болльше максимума

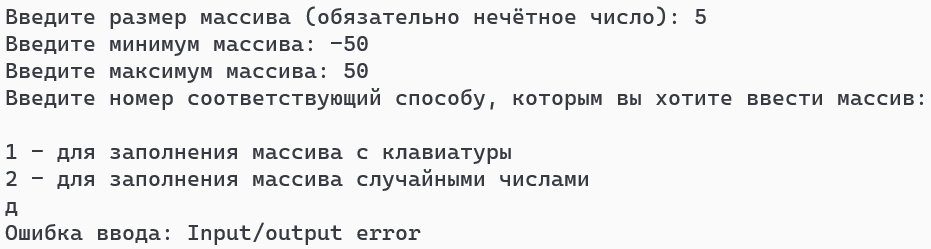


Рисунок 20 – Вывод программы, когда выбор способа заполнения массива – буква

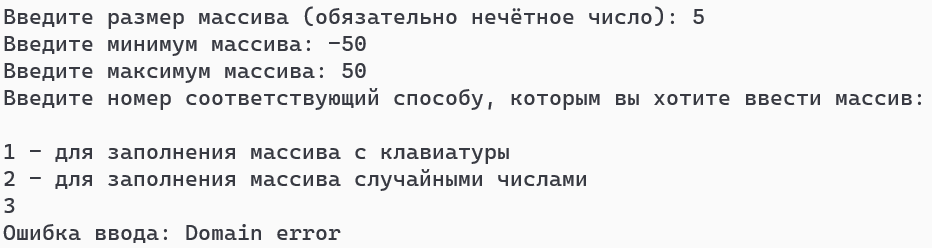


Рисунок 21 – Вывод программы, когда выбор способа заполнения массива не входит в указанные значения

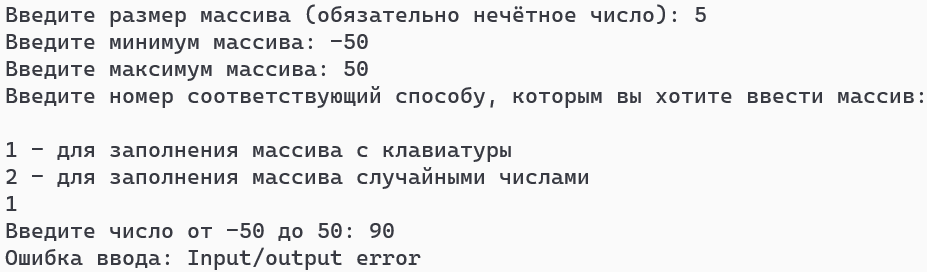


Рисунок 22 – Вывод программы, когда пользовательский ввод элемента массива не входит в указанный промежуток

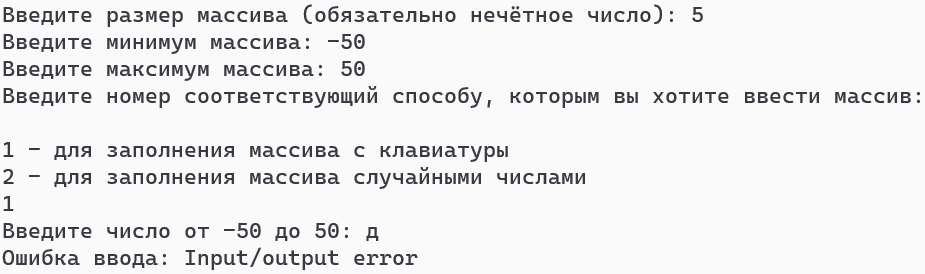


Рисунок 23 – Вывод программы, когда пользовательский ввод элемента массива – буква

* 1. Зачёт задания в GitHub

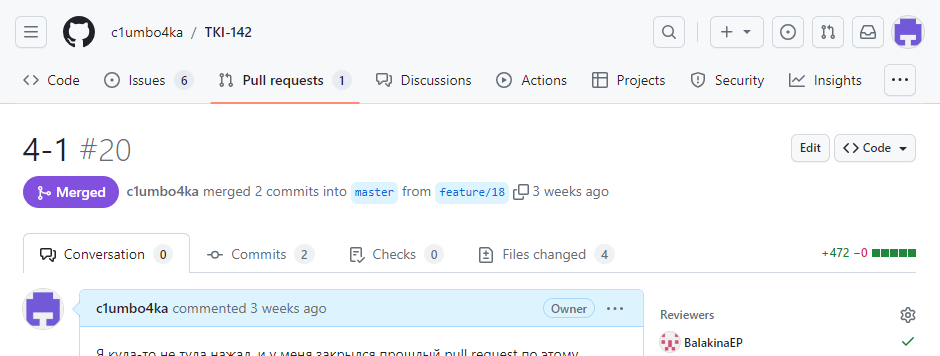


Рисунок 24 – Зачёт задания в GitHub

# Задание 4.2

* 1. Формулировка задания

Создать одномерный массив из *n* целых чисел. Реализовать возможность заполнения массива, как случайными числами, так и с помощью клавиатуры по желанию пользователя. Пункты задания организовать в виде функций (методов). Вывести массив на экран. Составить блок-схему.

|  |  |  |
| --- | --- | --- |
| Вариант | Задачи | Интервал |
| 8 | 1. Заменить минимальный по модулю положительный элемент массива последним элементом. 2. Удалить из него все элементы, в которых последняя цифра нечетная, а само число кратно 3. Из элементов массива C сформировать массив A той же размерности по правилу: все чётные элементы находятся по формуле: Ai=Ci+i, если нечетный, то Ai=Ci-i. | [-20;40] |

* 1. Блок-схема алгоритма

Блок-схема алгоритма представлена на рисунках (Рисунок 25, Рисунок 26, Рисунок 27, Рисунок 28, Рисунок 29, Рисунок 30, Рисунок 31, Рисунок 32, Рисунок 33, Рисунок 34, Рисунок 35, Рисунок 36, Рисунок 37, Рисунок 38, Рисунок 39, Рисунок 40)

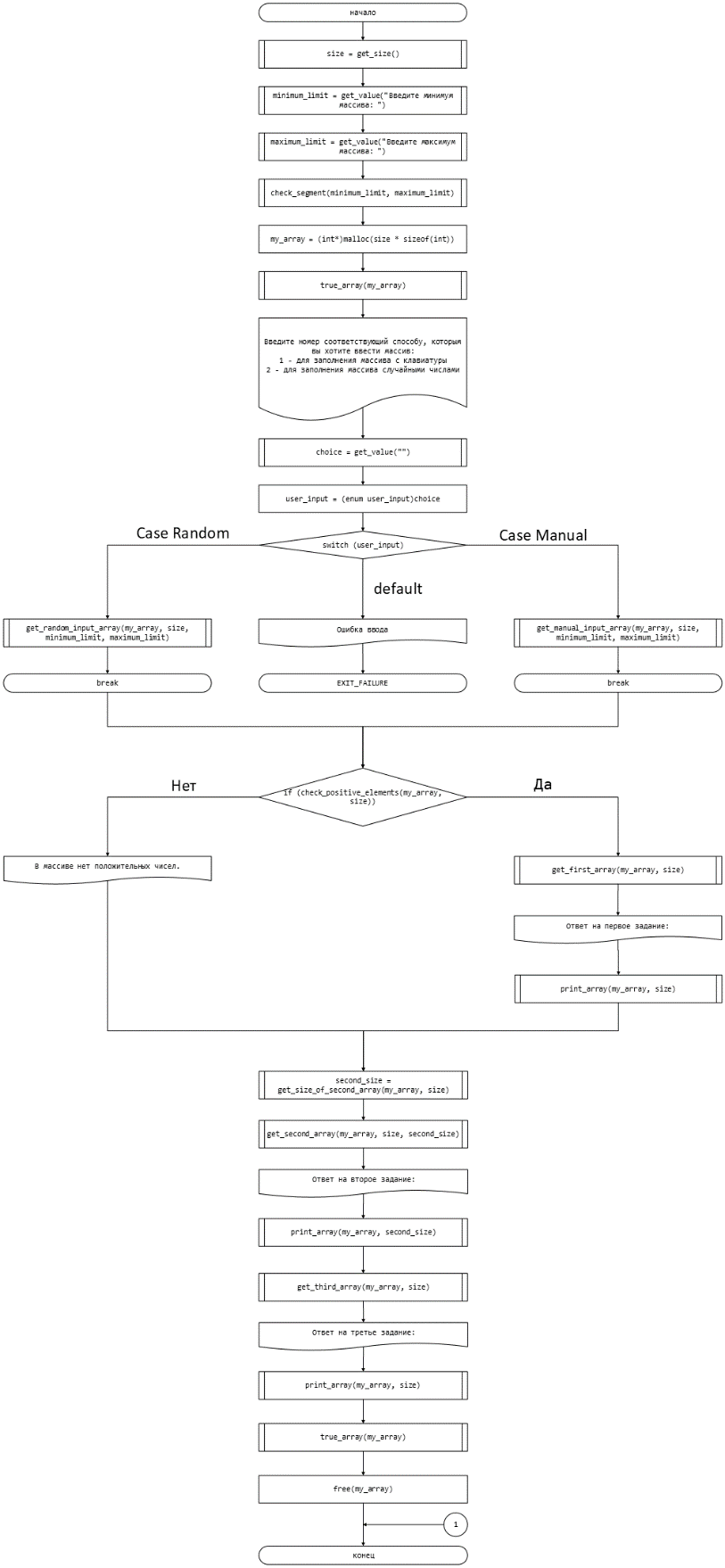


Рисунок 25 – Блок-схема функции main()

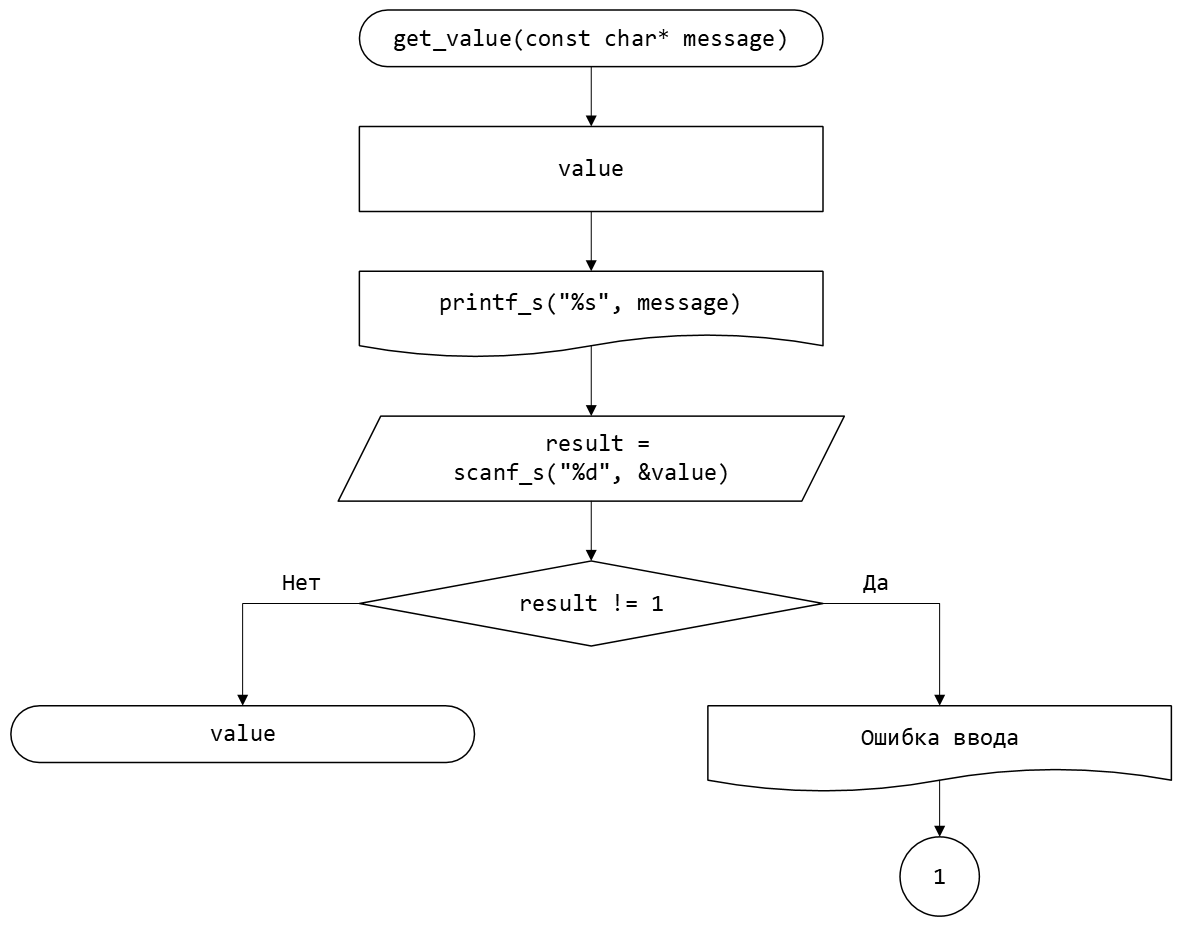


Рисунок 26 – Блок-схема функции get\_value(message)

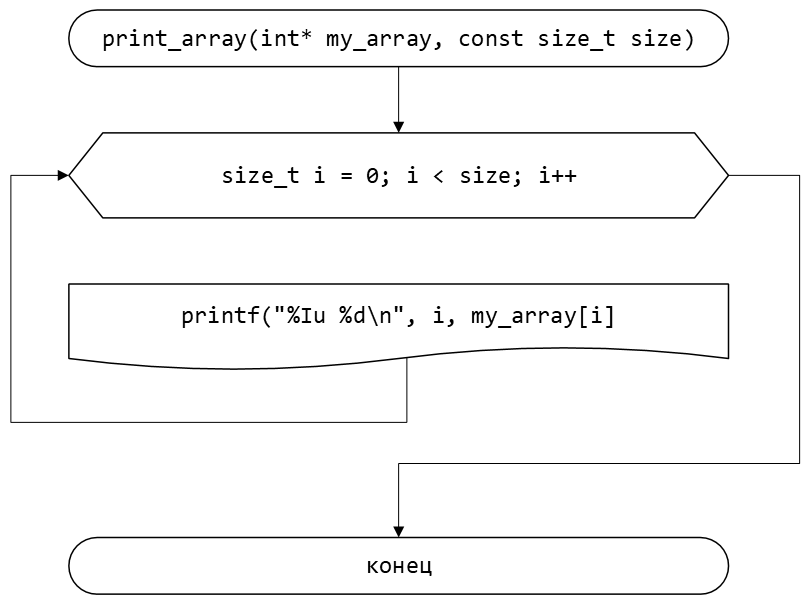


Рисунок 27 – Блок-схема функции print\_array(my\_array, size)

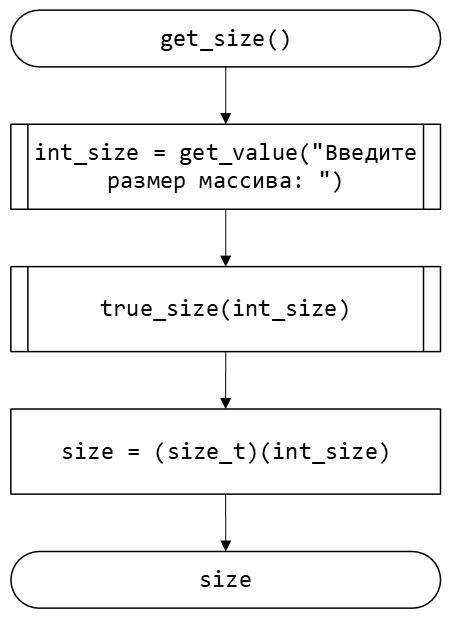


Рисунок 28 – Блок-схема функции get\_size()

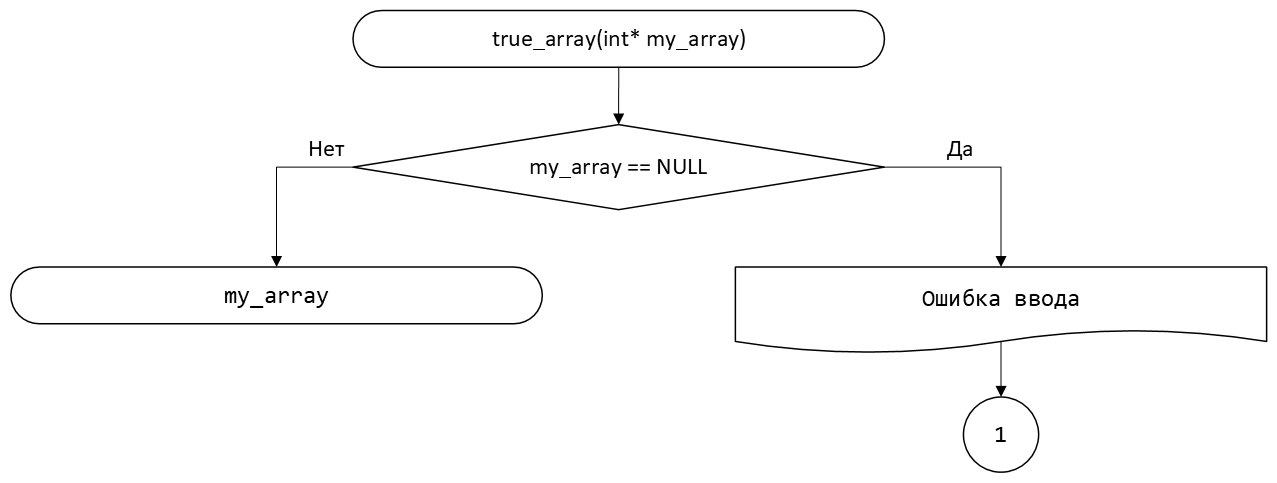


Рисунок 29 – Блок-схема функции true\_array(my\_array)

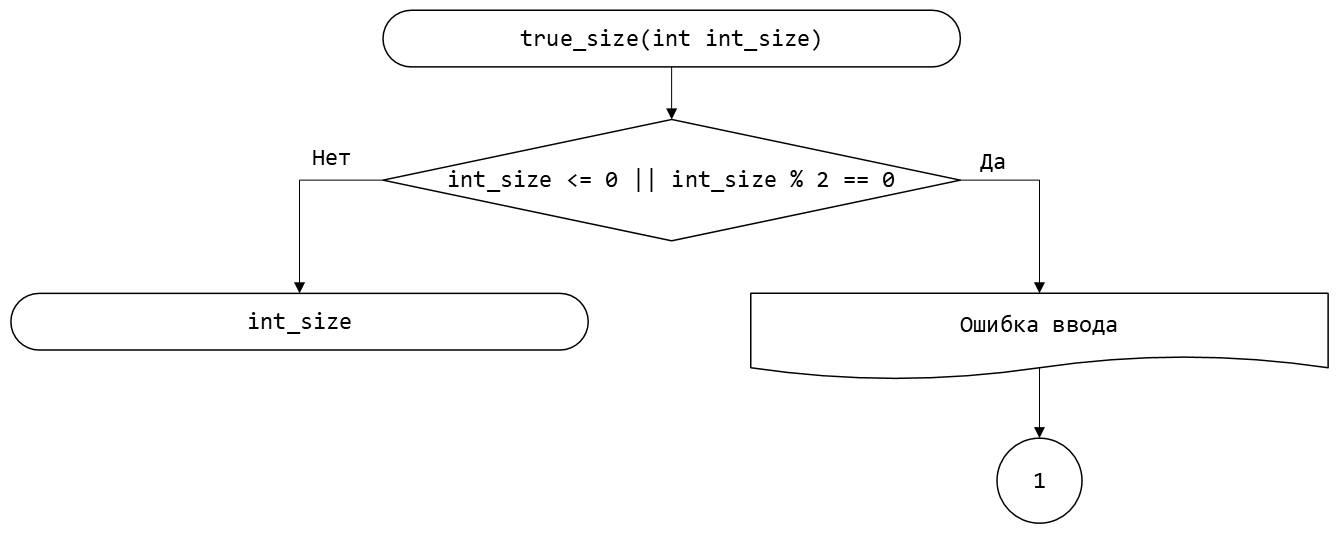


Рисунок 30 – Блок-схема функции true\_size(size)

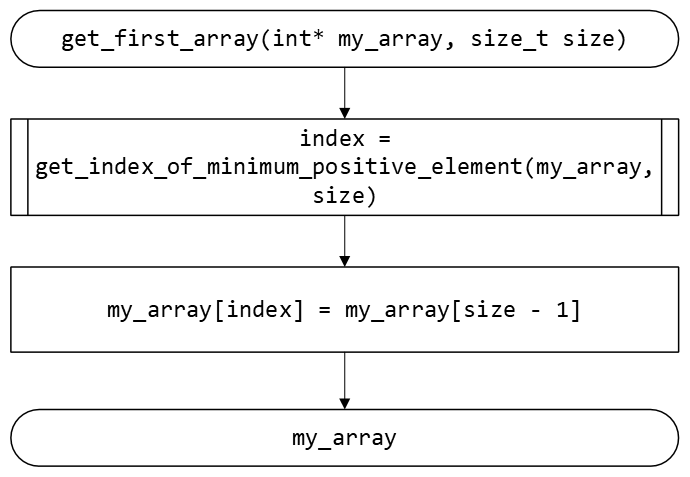


Рисунок 31 – Блок-схема функции get\_first\_array(my\_array, size)

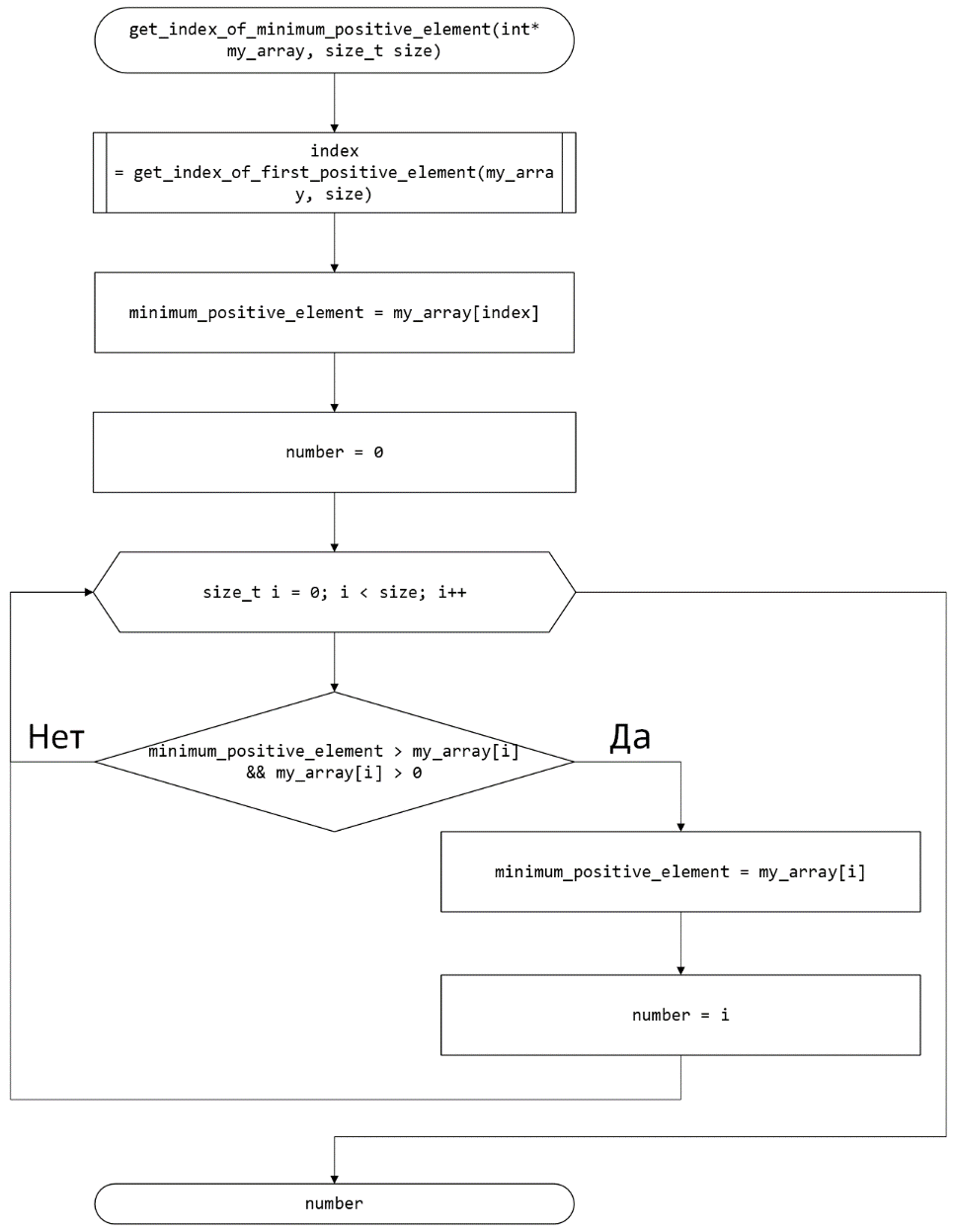


Рисунок 32 – Блок-схема функции get\_index\_of\_minimum\_positive\_element(my\_array, size)

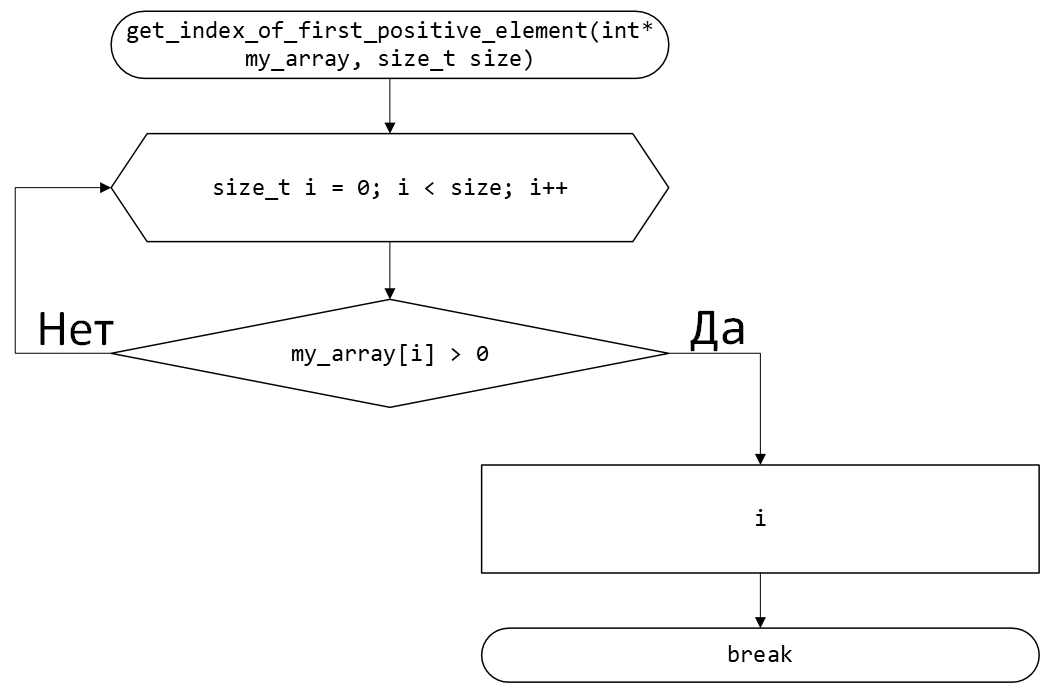


Рисунок 33 – Блок-схема функции get\_index\_of\_first\_positive\_element(my\_array, size)

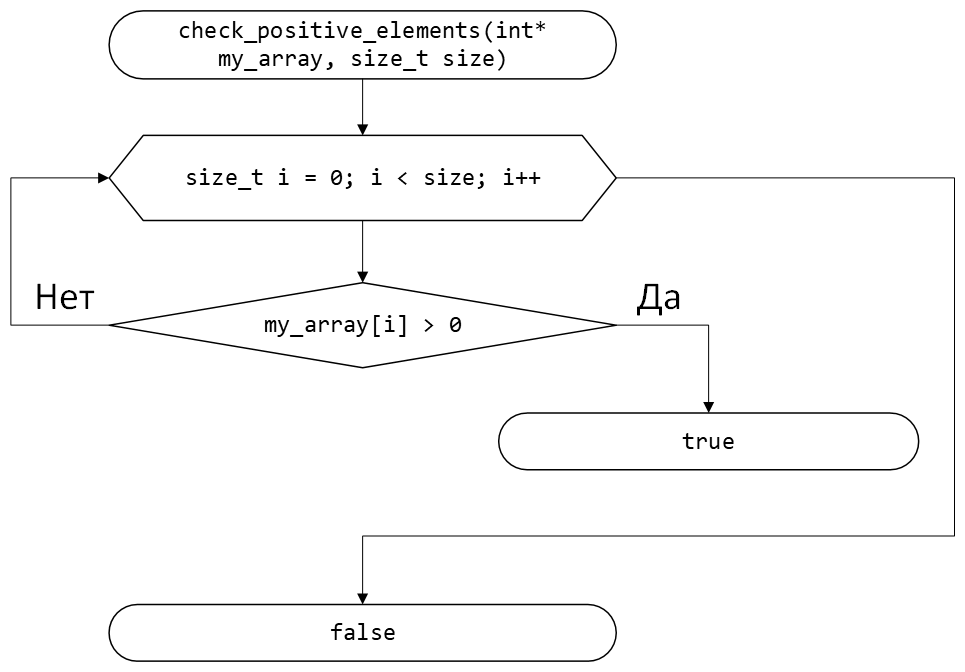


Рисунок 34 – Блок-схема функции check\_psitive\_elements(my\_array, size)

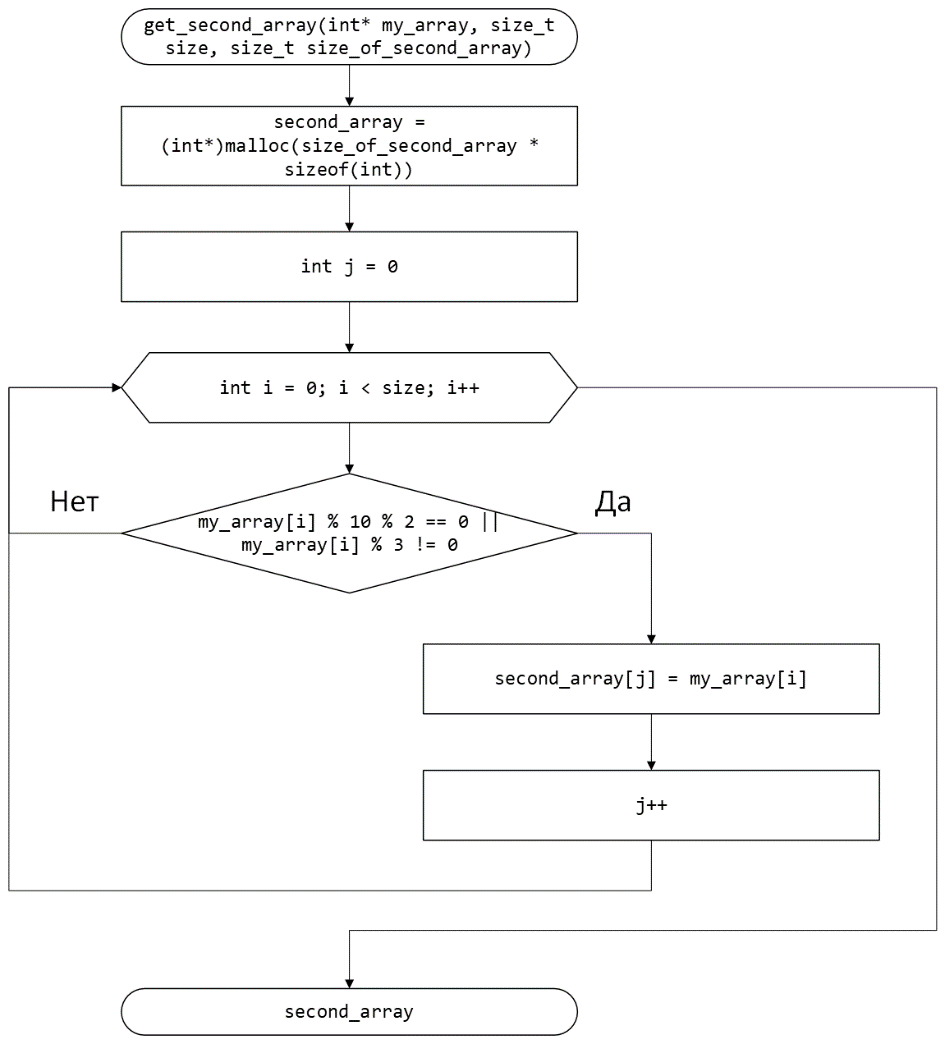


Рисунок 35 – Блок-схема функции get\_second\_array(my\_array, size, size\_of\_second\_array)

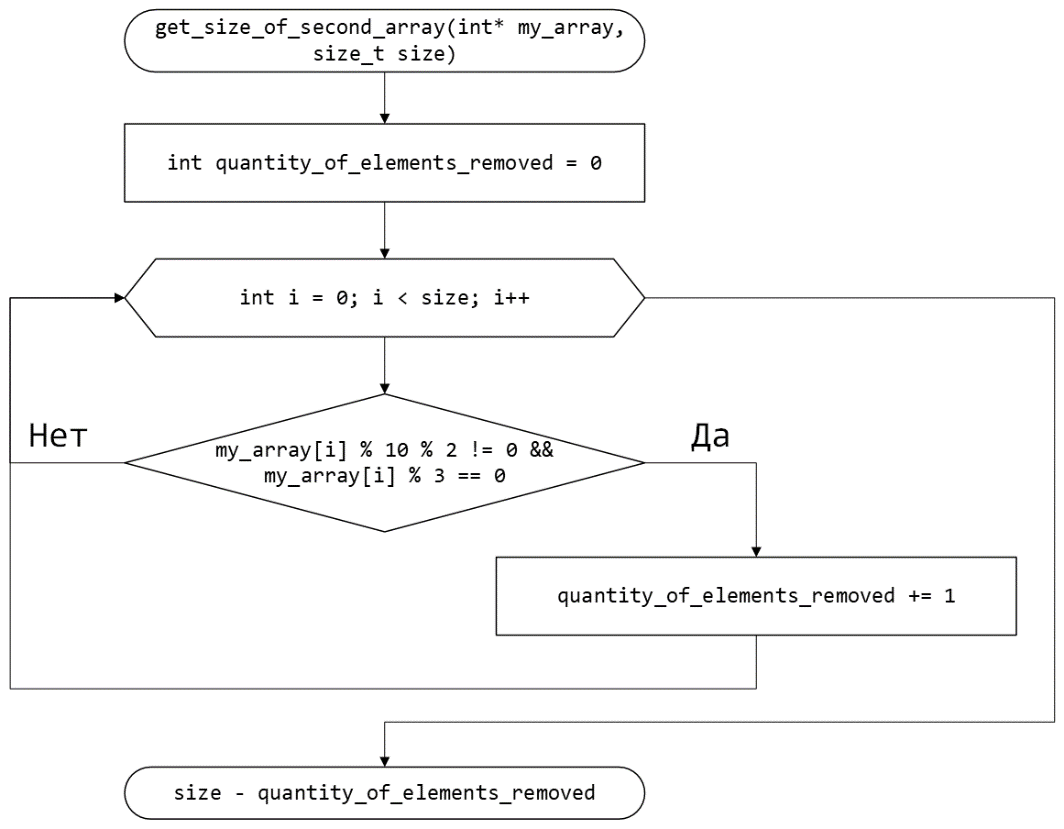


Рисунок 36 – Блок-схема функции get\_size\_of\_second\_array(my\_array, size)

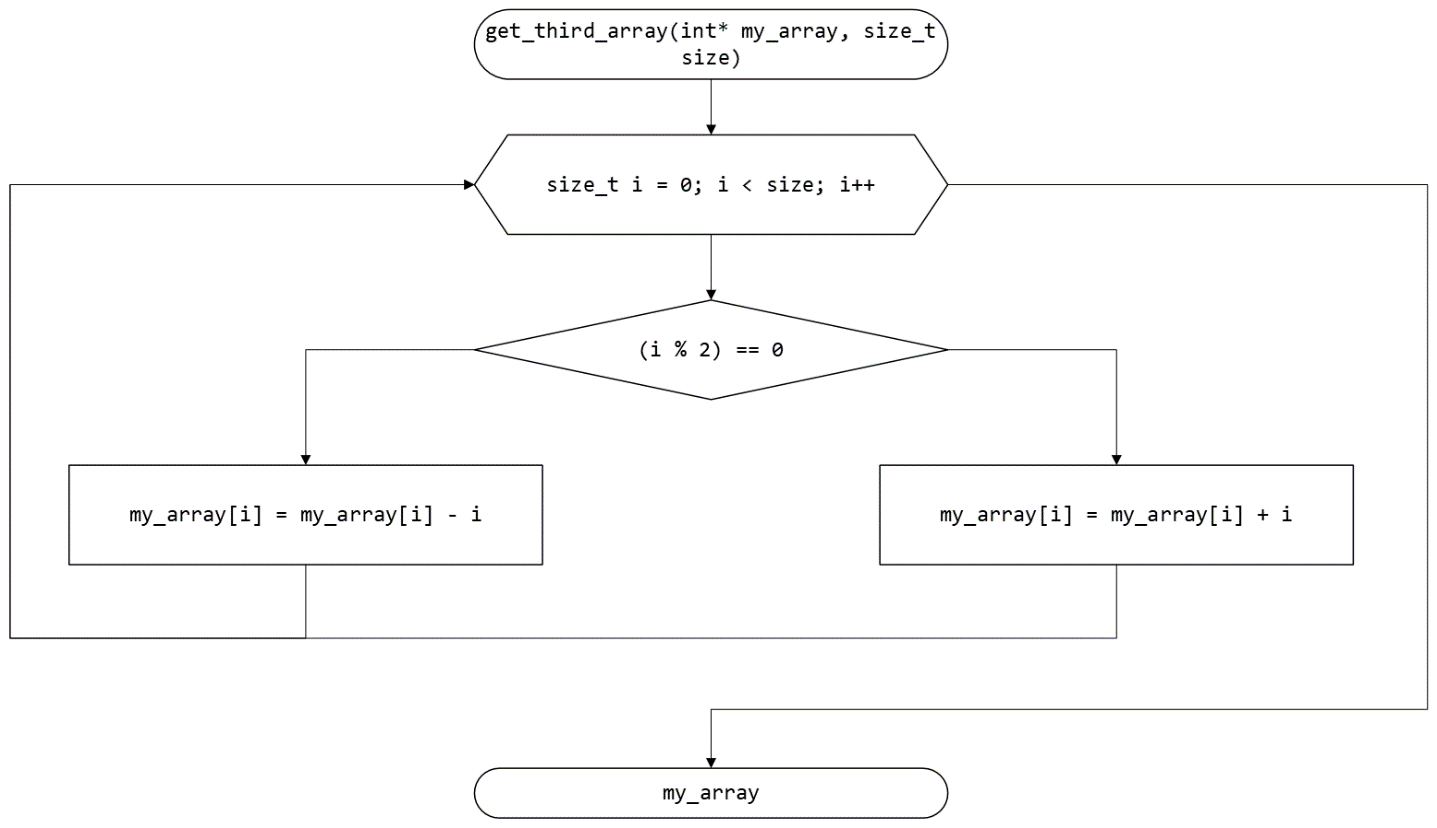


Рисунок 37 – Блок-схема функции get\_third\_array(my\_array, size)

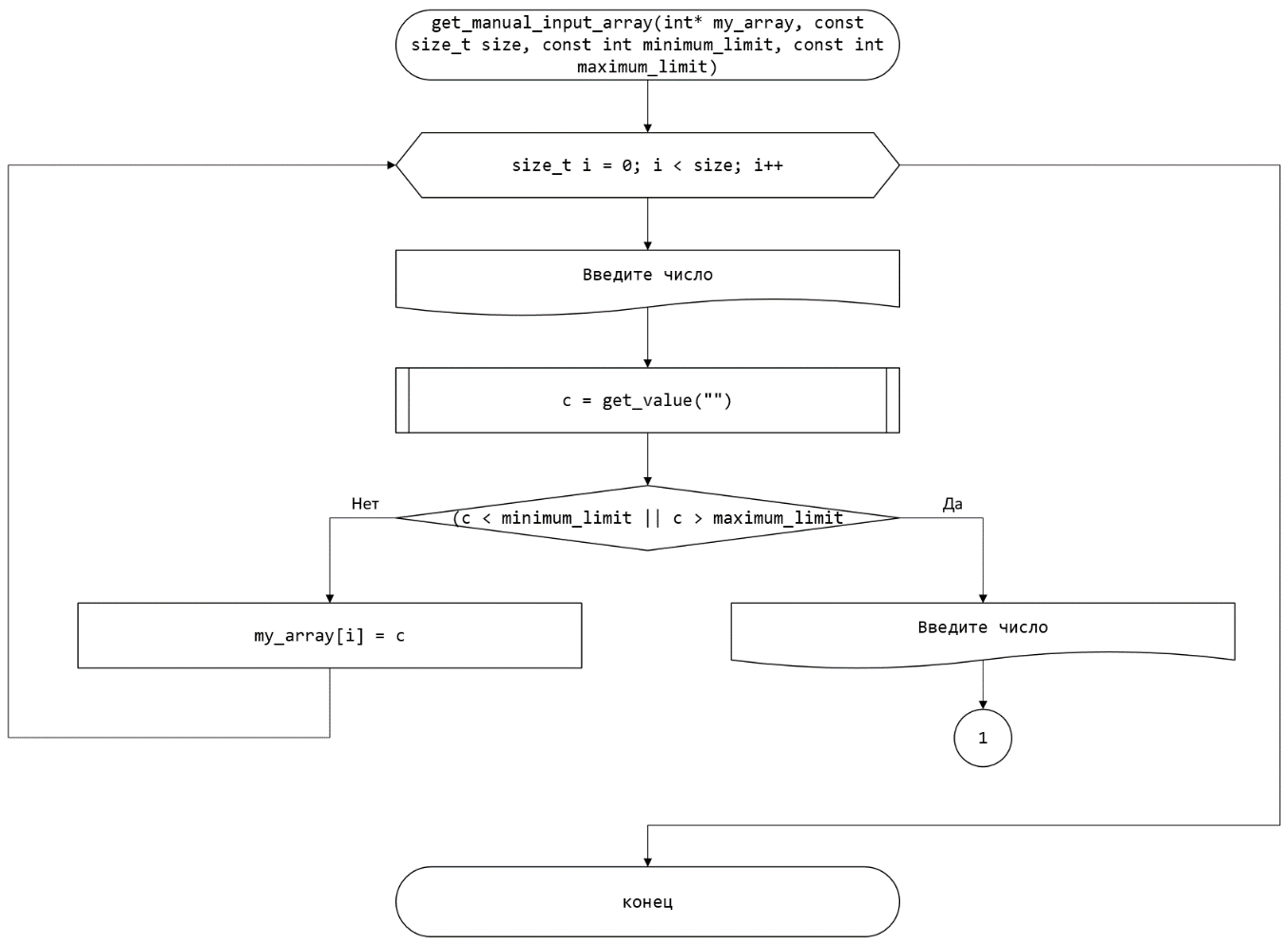


Рисунок 38 – Блок-схема функции get\_manual\_input\_array(my\_array, size, minimum\_limit, maximum\_limit)

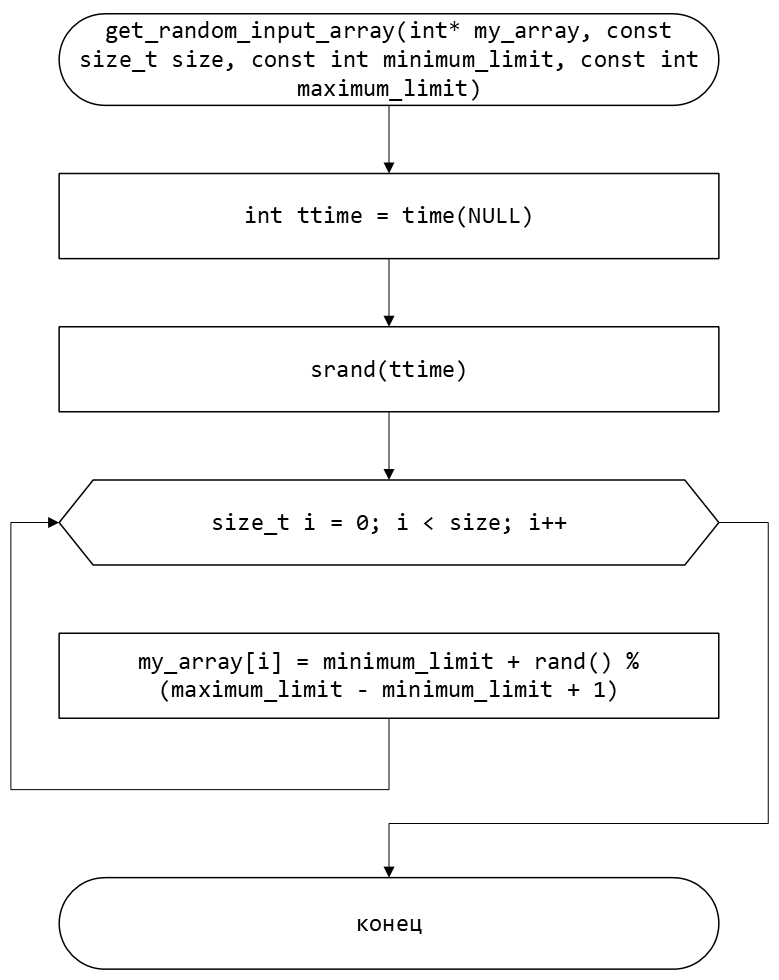


Рисунок 39 – Блок-схема функции get\_random\_input\_array(my\_array, size, minimum\_limit, maximum\_limit)

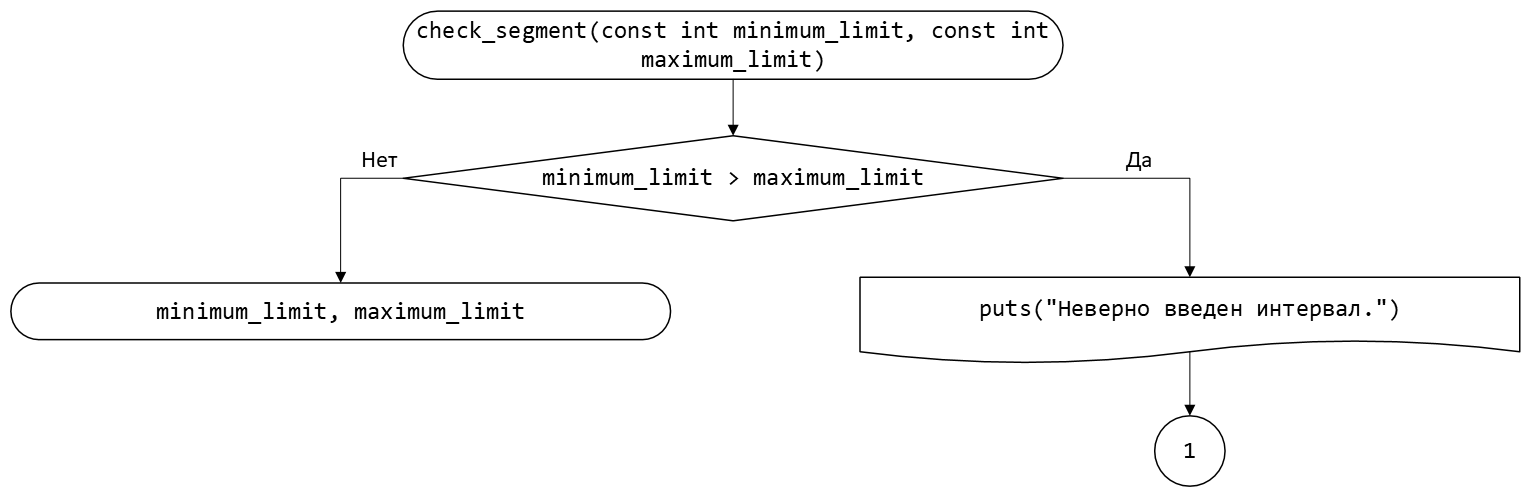


Рисунок 40 – Блок-схема функции check\_segment(minimum\_limit, maximum\_limit)

* 1. Код задания 4.2

#include <stdio.h>

#include <locale.h>

#include <errno.h>

#include <math.h>

#include <float.h>

#include <stdlib.h>

#include <time.h>

#include <stdbool.h>

/\*\*

\* @brief Выбор заполнения массива.

\*/

enum array\_input\_choice

{

/\*\*

\* @brief Ручной способ.

\*/

MANUAL = 1,

/\*\*

\* @brief Заполнение массива случайными числами.

\*/

RANDOM = 2

};

/\*\*

\* @brief Функция для считывания элементов массива с клавиатуры.

\* @param size - длина массива.

\* @param my\_array - массив.

\*/

void get\_manual\_input\_array(int\* my\_array, size\_t size, const int minimum\_limit, const int maximum\_limit);

/\*\*

\* @brief Функция для заполнения массива случайными числами.

\* @param size - длина массива.

\* @param my\_array - массив.

\*/

void get\_random\_input\_array(int\* my\_array, size\_t size, const int minimum\_limit, const int maximum\_limit);

/\*\*

\* @brief Функция для печати массива.

\* @param array - массив.

\* @param size - длина массива.

\*/

void print\_array(int\* array, size\_t size);

/\*\*

\* @brief Функция для считывания значения с клавиатуры.

\* @param message - сообщение пользователю.

\* @remarks Экстренное завершение программы, в случае неправильного ввода.

\* @return Значение с клавиатуры.

\*/

int get\_value(const char\* message);

/\*\*

\* @brief Функция для проверки длинны массива.

\* @param int\_size - длина массива.

\* @remarks Экстренное завершение программы, в случае неправильной длинны массива.

\*/

void true\_size(int int\_size);

/\*\*

\* @brief Функция для проверки массива.

\* @param my\_array - первый массив.

\* @param second\_array - второй массив.

\* @param third\_array - третий массив.

\* @remarks Экстренное завершение программы, в случае несуществования массива.

\*/

void true\_array(int\* my\_array);

/\*\*

\* @brief Функция проверки интервала массива на правильность.

\* @param minimum\_limit - минимальный возможный элемент массива.

\* @param maximum\_limit - максимальный возможный элемент массива.

\*/

void check\_segment(const int minimum\_limit, const int maximum\_limit);

/\*\*

\* @brief Функция для нахождения индекса минимального положительного элемента массива.

\* @param my\_array - первый массив.

\* @param size - длина массива.

\* @return Индекс минимального положительного элемента массива.

\*/

size\_t get\_index\_of\_minimum\_positive\_element(int\* my\_array, size\_t size);

/\*\*

\* @brief Функция для нахождения первого положительного элемента массива.

\* @param my\_array - первый массив.

\* @param size - длина массива.

\* @return Максимальный элемент массива.

\*/

size\_t get\_index\_of\_first\_positive\_element(int\* my\_array, size\_t size);

/\*\*

\* @brief Функция для нахождения длины второго массива.

\* @param my\_array - первый массив.

\* @param size - длина первого массива.

\* @return Длина второго массива.

\*/

int get\_size\_of\_second\_array(int\* my\_array, size\_t size);

/\*\*

\* @brief Функция для заполнения третьего массива.

\* @param my\_array - первый массив.

\* @param size - старая длина массива.

\* @return Третий массив.

\*/

int\* get\_third\_array(int\* my\_array, size\_t size);

/\*\*

\* @brief Функция для получения второго массива.

\* @param my\_array - первый массив.

\* @param size - старая длина массива.

\* @param size\_of\_second\_array - длина второго массива.

\* @return Второй массив.

\*/

int\* get\_second\_array(int\* my\_array, size\_t size, size\_t size\_of\_second\_array);

/\*\*

\* @brief Функция считывания размера массива.

\* return Размер массива

\*/

size\_t get\_size();

/\*\*

\* @brief Функция проверки наличия положительных элементов в массиве.

\* @param my\_array - массив.

\* @param size - размер массива.

\* @return Возвращает истину в случае успеха.

\*/

bool check\_positive\_elements(int\* my\_array, size\_t size);

/\*\*

\* @brief Функция для заполнения первого массива.

\* @param my\_array - массив.

\* @param size - размер массива.

\* @return Первый массив.

\*/

int\* get\_first\_array(int\* my\_array, size\_t size);

/\*\*

\* @brief Точка входа в программу.

\* @return Возвращает 0 в случае успеха.

\*/

int main()

{

setlocale(LC\_ALL, "RU");

size\_t size = get\_size();

const int minimum\_limit = get\_value("Введите минимум массива: ");

const int maximum\_limit = get\_value("Введите максимум массива: ");

check\_segment(minimum\_limit, maximum\_limit);

int\* my\_array = (int\*)malloc(size \* sizeof(int));

true\_array(my\_array);

puts("Введите номер соответствующий способу, которым вы хотите ввести массив:\n");

printf\_s("%d - для заполнения массива с клавиатуры\n", MANUAL);

printf\_s("%d - для заполнения массива случайными числами\n", RANDOM);

int choice = get\_value("");

enum array\_input\_choice user\_input = (enum user\_input)choice;

switch (user\_input)

{

case MANUAL:

{

get\_manual\_input\_array(my\_array, size, minimum\_limit, maximum\_limit);

break;

}

case RANDOM:

{

get\_random\_input\_array(my\_array, size, minimum\_limit, maximum\_limit);

break;

}

default:

{

errno = EDOM;

perror("Ошибка ввода");

return EXIT\_FAILURE;

}

}

if (check\_positive\_elements(my\_array, size))

{

puts("Ответ на первое задание:");

print\_array(get\_first\_array(my\_array, size), size);

}

else

{

puts("В массиве нет положительных чисел.");

}

puts("Ответ на второе задание:");

print\_array(get\_second\_array(my\_array, size, get\_size\_of\_second\_array(my\_array, size)), get\_size\_of\_second\_array(my\_array, size));

puts("Ответ на третье задание:");

print\_array(get\_third\_array(my\_array, size), size);

true\_array(my\_array);

free(my\_array);

return 0;

}

int get\_value(const char\* message)

{

int value;

printf\_s("%s", message);

int result = scanf\_s("%d", &value);

if (result != 1)

{

errno = EIO;

perror("Ошибка ввода");

abort();

}

return value;

}

void true\_size(int int\_size)

{

if (int\_size <= 0)

{

errno = EIO;

perror("Ошибка ввода");

abort();

}

}

void true\_array(int\* my\_array)

{

if (my\_array == NULL)

{

errno = EIO;

perror("Ошибка ввода");

abort();

}

}

void get\_manual\_input\_array(int\* my\_array, size\_t size, const int minimum\_limit, const int maximum\_limit)

{

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

{

printf("Введите число от %d до %d: ", minimum\_limit, maximum\_limit);

int c = get\_value("");

if (c < minimum\_limit || c > maximum\_limit)

{

errno = EIO;

perror("Ошибка ввода");

abort();

}

my\_array[i] = c;

}

}

void get\_random\_input\_array(int\* my\_array, size\_t size, const int minimum\_limit, const int maximum\_limit)

{

unsigned int ttime = time(NULL);

srand(ttime);

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

{

my\_array[i] = minimum\_limit + rand() % (maximum\_limit - minimum\_limit + 1);

}

}

void check\_segment(const int minimum\_limit, const int maximum\_limit)

{

if (minimum\_limit > maximum\_limit)

{

puts("Неверно введен интервал.");

abort();

}

}

void print\_array(int\* array, size\_t size)

{

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

{

printf("%Iu %d\n", i, array[i]);

}

}

size\_t get\_index\_of\_minimum\_positive\_element(int\* my\_array, size\_t size)

{

{

int minimum\_positive\_element = my\_array[get\_index\_of\_first\_positive\_element(my\_array, size)];

int number = 0;

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

{

if (minimum\_positive\_element > my\_array[i] && my\_array[i] > 0)

{

minimum\_positive\_element = my\_array[i];

number = i;

}

}

return number;

}

}

size\_t get\_index\_of\_first\_positive\_element(int\* my\_array, size\_t size)

{

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

{

if (my\_array[i] > 0)

{

return i;

break;

}

}

}

int\* get\_third\_array(int\* my\_array, size\_t size)

{

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

{

if ((i % 2) == 0)

{

my\_array[i] = my\_array[i] + i;

}

else

{

my\_array[i] = my\_array[i] - i;

}

}

return my\_array;

}

int get\_size\_of\_second\_array(int\* my\_array, size\_t size)

{

int quantity\_of\_elements\_removed = 0;

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

{

if (my\_array[i] % 10 % 2 != 0 && my\_array[i] % 3 == 0)

{

quantity\_of\_elements\_removed += 1;

}

}

return (size - quantity\_of\_elements\_removed);

}

int\* get\_second\_array(int\* my\_array, size\_t size, size\_t size\_of\_second\_array)

{

int\* second\_array = (int\*)malloc(size\_of\_second\_array \* sizeof(int));

int j = 0;

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

{

if (my\_array[i] % 10 % 2 == 0 || my\_array[i] % 3 != 0)

{

second\_array[j] = my\_array[i];

j++;

}

}

return second\_array;

}

size\_t get\_size()

{

int int\_size = get\_value("Введите размер массива: ");

true\_size(int\_size);

size\_t size = (size\_t)(int\_size);

return size;

}

bool check\_positive\_elements(int\* my\_array, size\_t size)

{

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

{

if (my\_array[i] > 0)

{

return true;

break;

}

}

return false;

}

int\* get\_first\_array(int\* my\_array, size\_t size)

{

my\_array[get\_index\_of\_minimum\_positive\_element(my\_array, size)] = my\_array[size - 1];

return my\_array;

}

* 1. Решение тестового примера

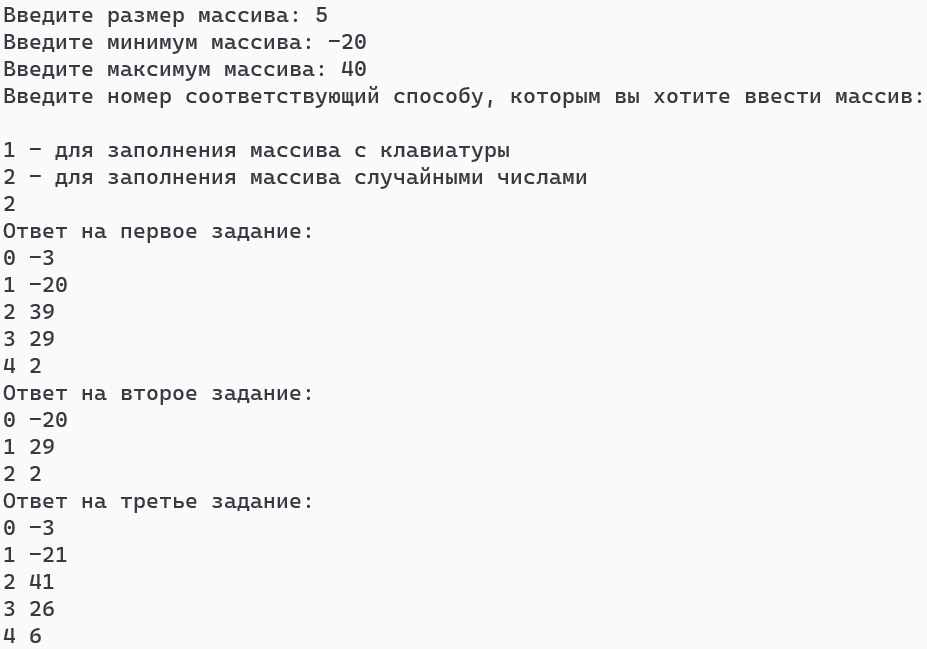


Рисунок 41 – Решение тестового примера рандомным заполнением массива

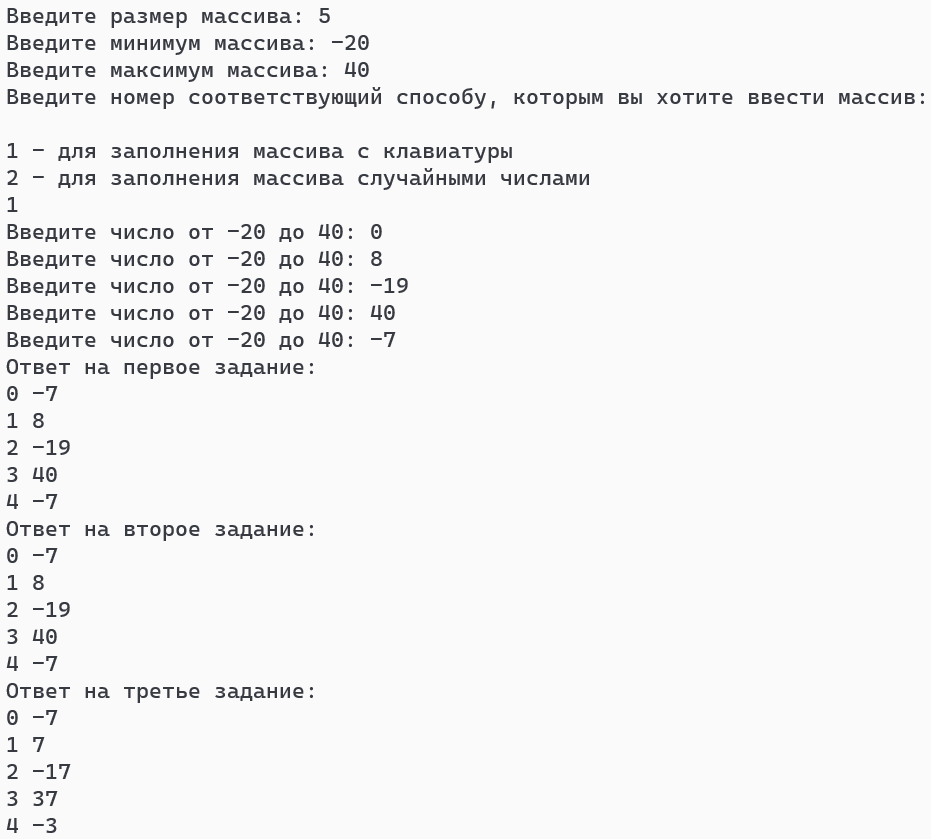


Рисунок 42 – Решение тестового примера пользовательским заполнением массива



Рисунок 43 – Вывод программы, когда размер массива отрицательный



Рисунок 44 – Вывод программы, когда размер массива – буква

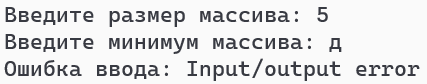


Рисунок 45 – Вывод программы, когда минимум массива – буква

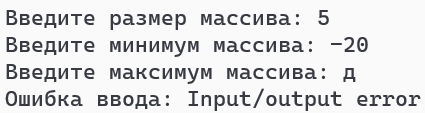


Рисунок 46– Вывод программы, когда максимум массива – буква

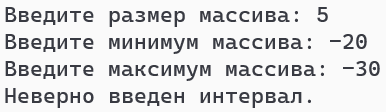


Рисунок 47– Вывод программы, когда максимум массива меньше минимума

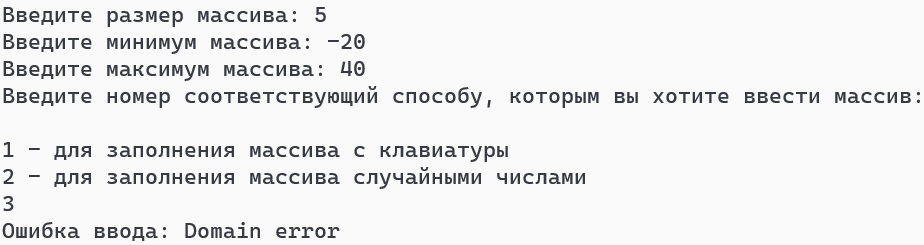


Рисунок  48 – Вывод программы, когда выбор способа заполнения массива не соответствует указанным значениям

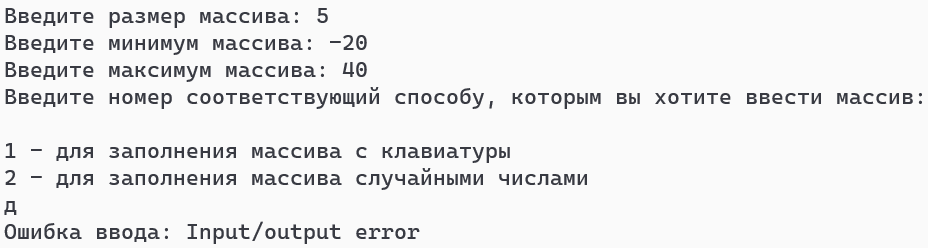


Рисунок  49 – Вывод программы, когда выбор способа заполнения массива – буква

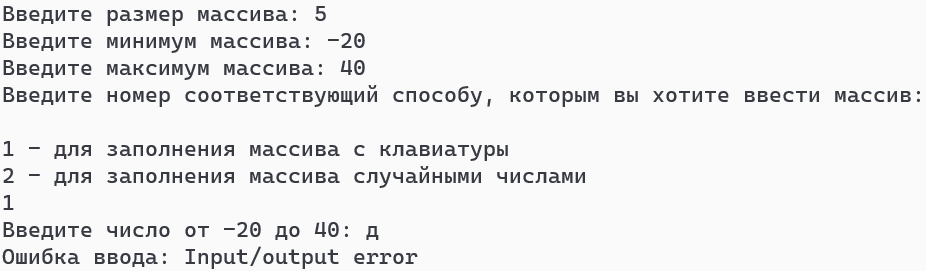


Рисунок  50 – Вывод программы, когда элемент массива при пользовательском вводе – буква

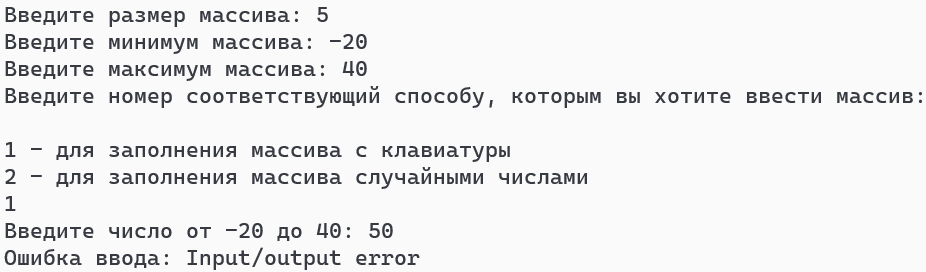


Рисунок  51 – Вывод программы, когда элемент массива при пользовательском вводе не входит промежуток

* 1. Зачёт задания в GitHub

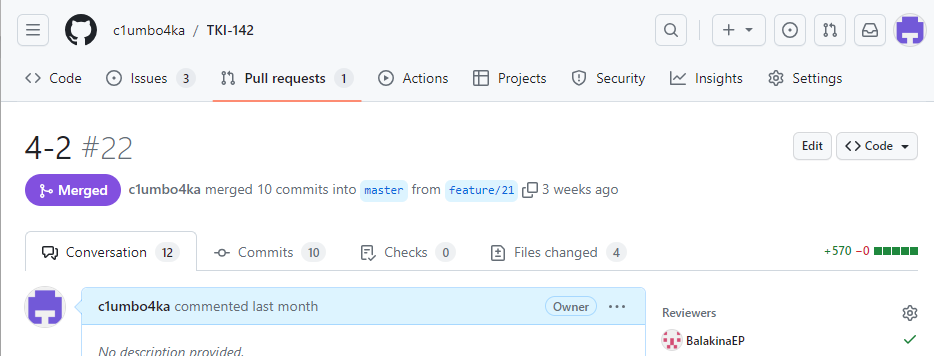


Рисунок 52 – Зачёт задания в GitHub

# Задание 4.3

* 1. Формулировка задания

Создать многомерный массив nˣm из *n* целых чисел. Реализовать возможность заполнения массива, как случайными числами, так и с помощью клавиатуры по желанию пользователя. Пункты задания организовать в виде функций (методов). Вывести массив на экран.

Таблица 2 – Формулировка задания 4.3

|  |  |
| --- | --- |
| Вариант | Задачи |
| 8 | 1. Заменить максимальный элемент каждого столбца нулем. 2. Вставить после всех строк, содержащих максимальный по модулю элемент, первую строку. |

* 1. Блок-схема алгоритма

Блок-схема алгоритма представлена на рисунках (Рисунок 53, Рисунок 54, Рисунок 55, Рисунок  56, Рисунок  57, Рисунок 58, Рисунок 59, Рисунок 60, Рисунок 61, Рисунок 62, Рисунок 63, Рисунок 64, Рисунок 65, Рисунок 66, Рисунок 68, Рисунок 69, Рисунок 70)

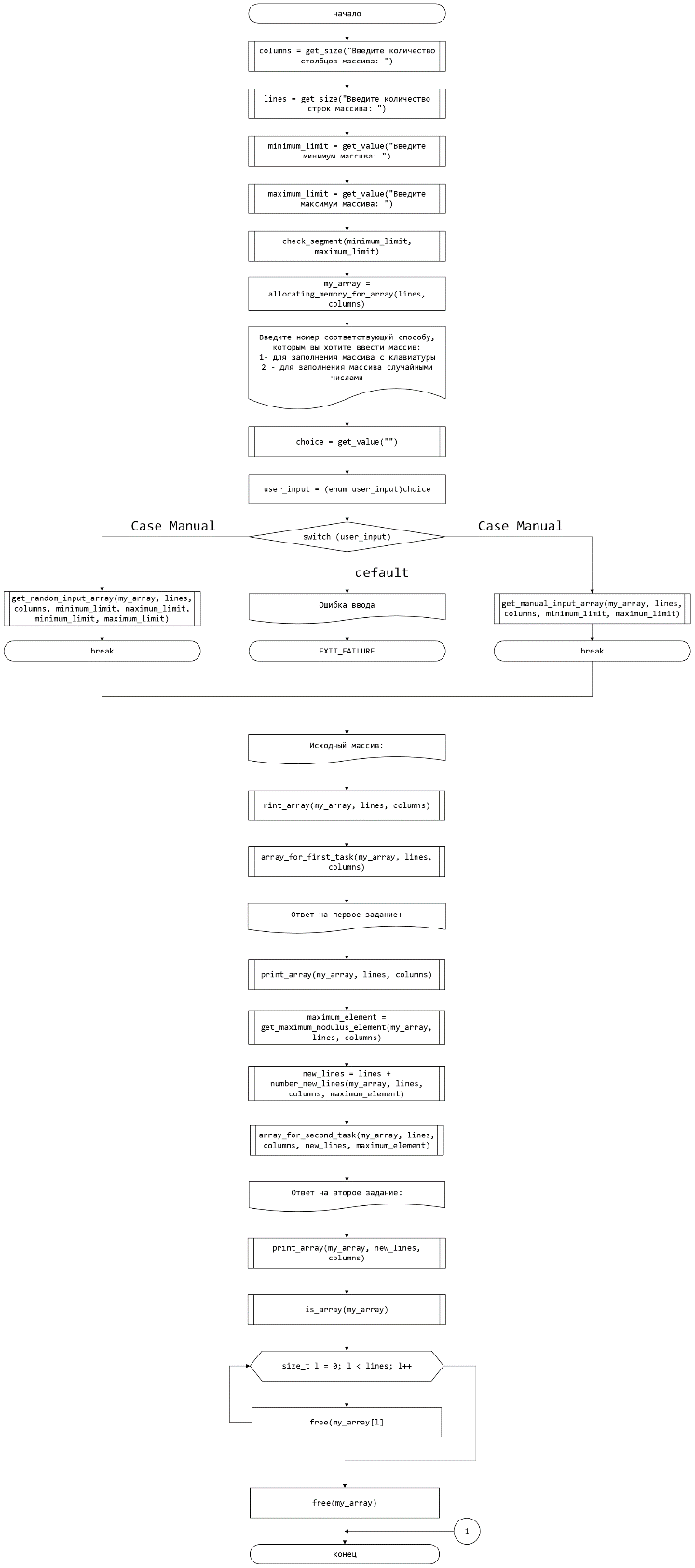


Рисунок 53 – Блок-схема функции main()

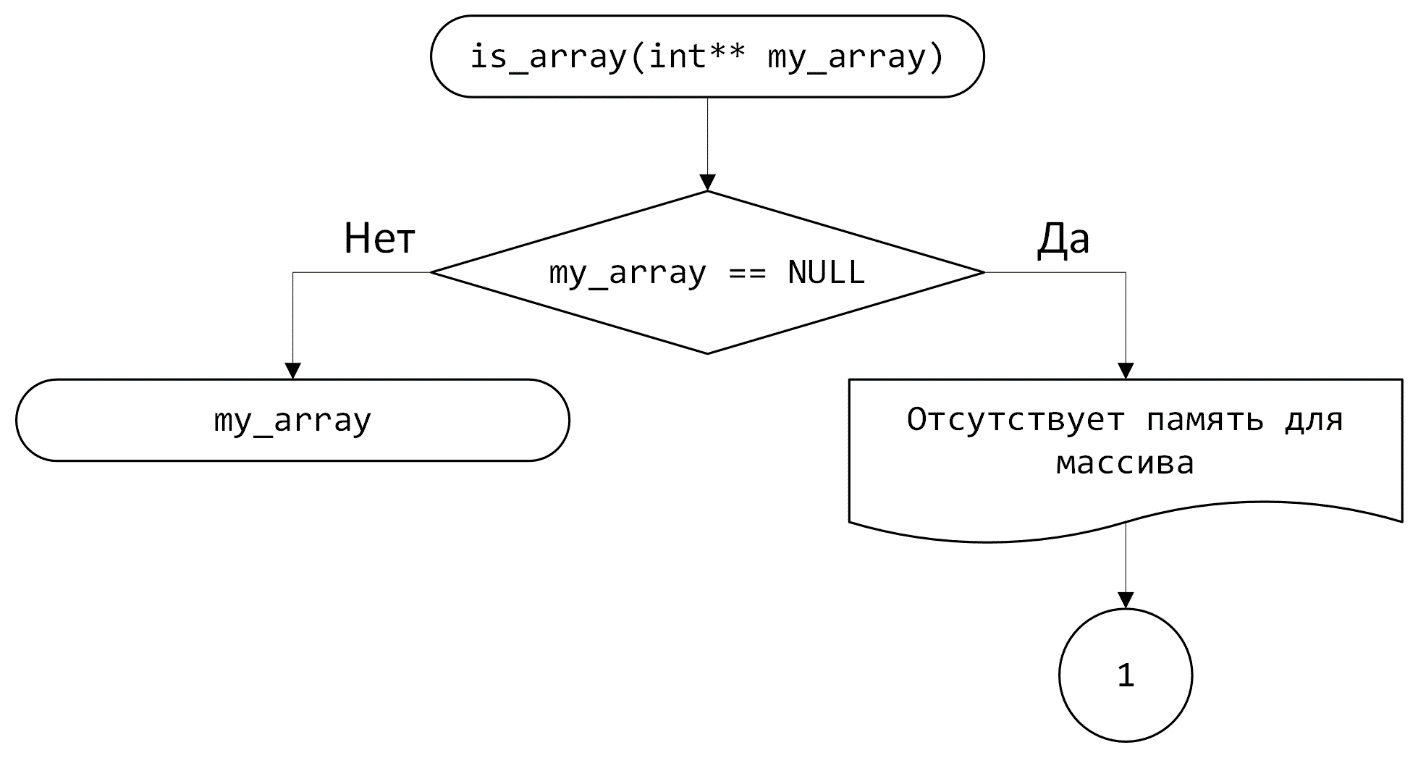


Рисунок 54 – Блок-схема функции is\_array(my\_array)

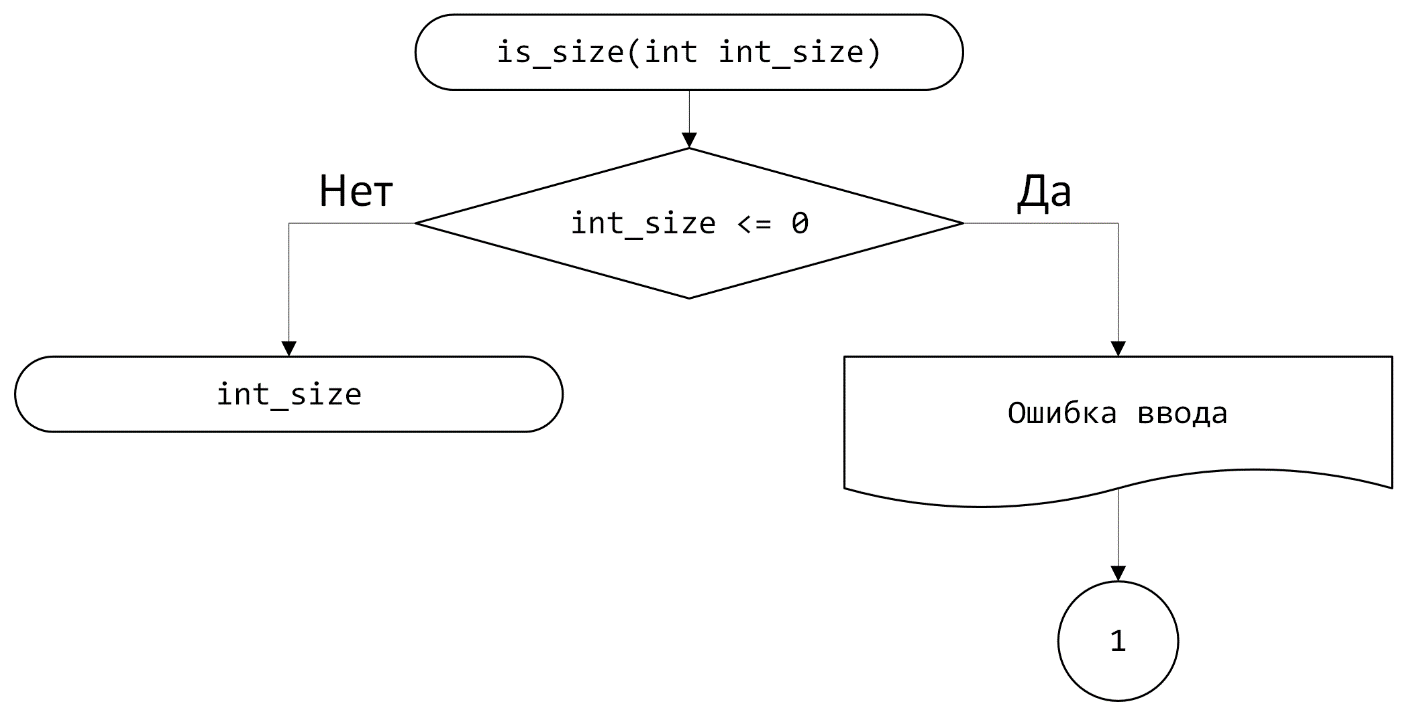


Рисунок 55 – Блок-схема функции is\_size(int\_size)

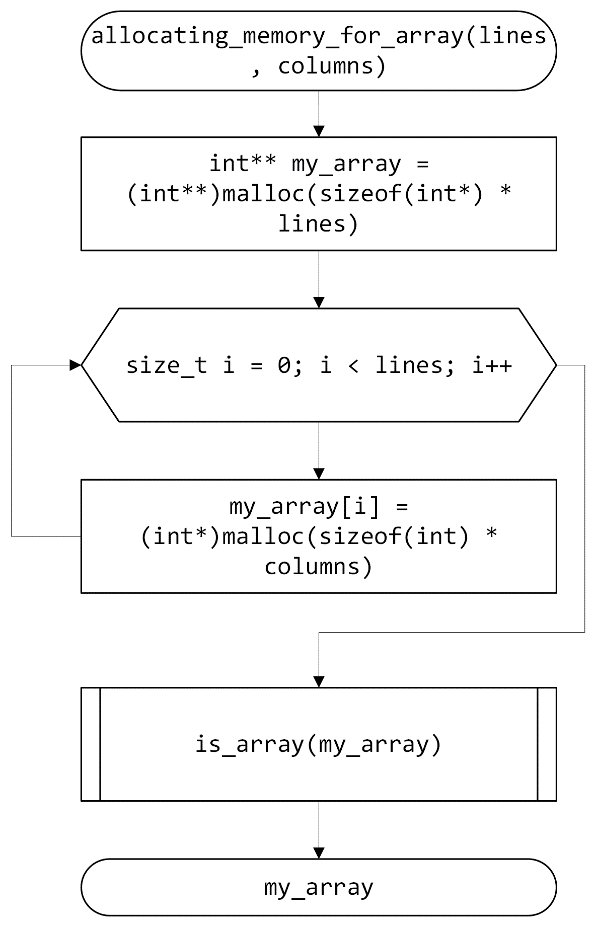


Рисунок  56 – Блок-схема функции allocating\_memory\_for\_array(lines, columns)

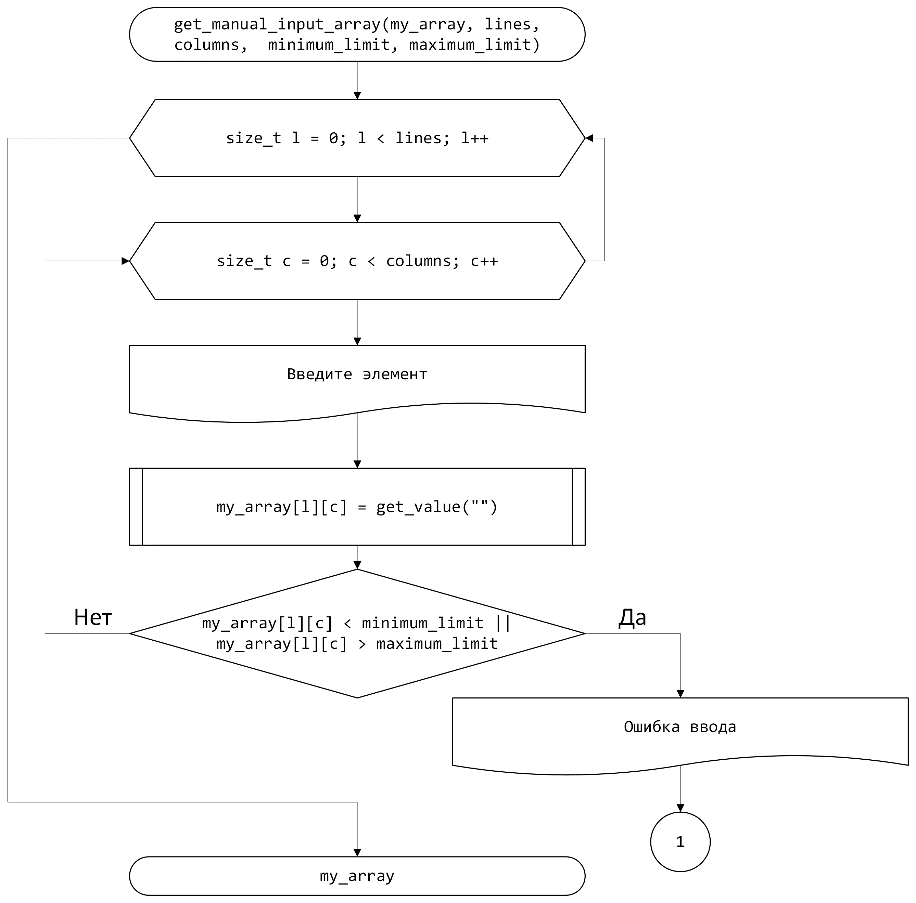


Рисунок  57 – Блок-схема функции get\_manual\_input\_array(my\_array, lines, columns, minimum\_limit, maximum\_limit)

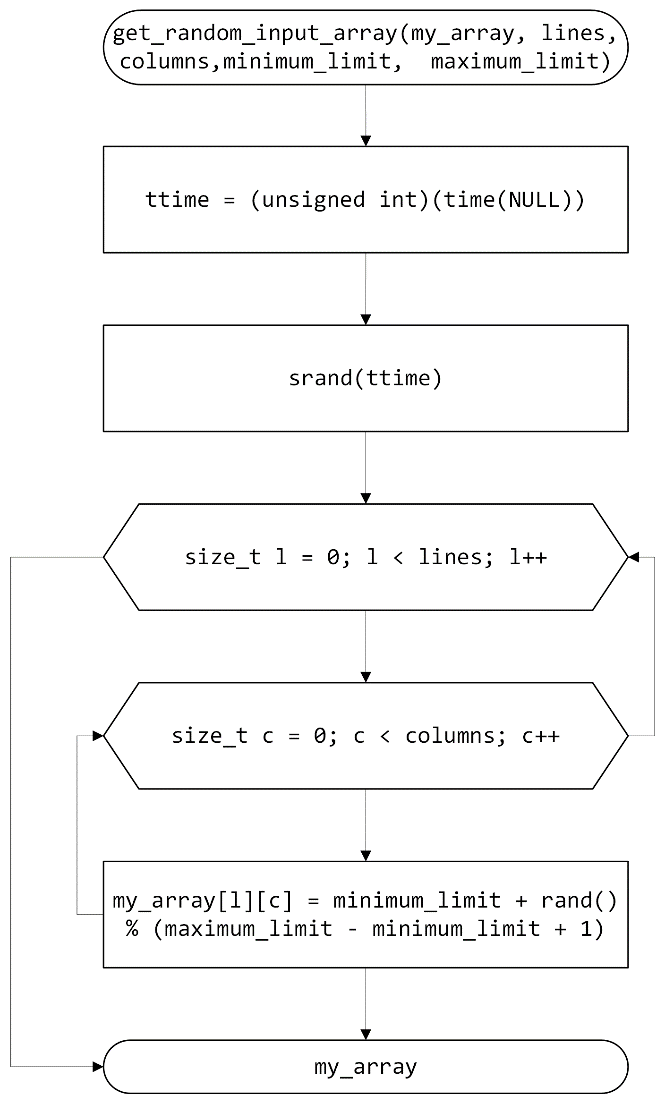


Рисунок 58 – Блок-схема функции get\_random\_input\_array(my\_array, lines, columns, minimum\_limit, maximum\_limit)

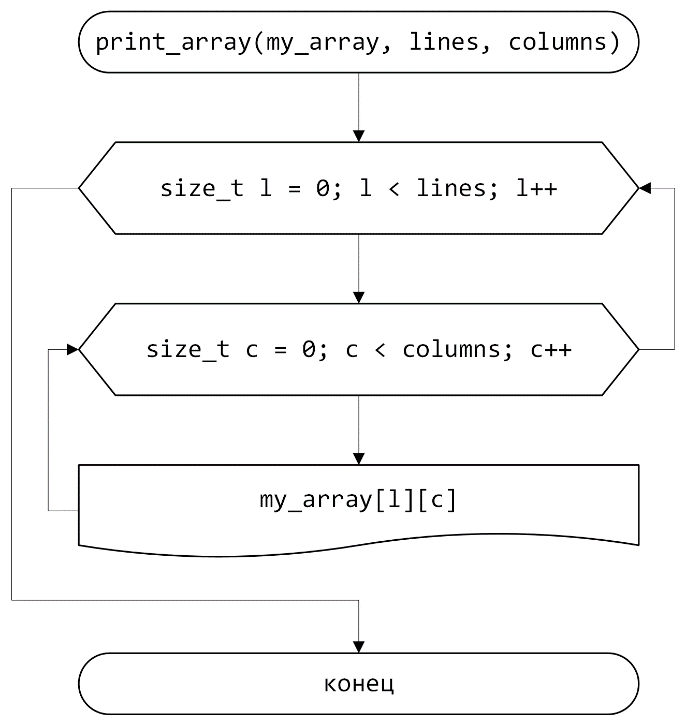


Рисунок 59 – Блок-схема функции print\_array(my\_array, lines, columns)

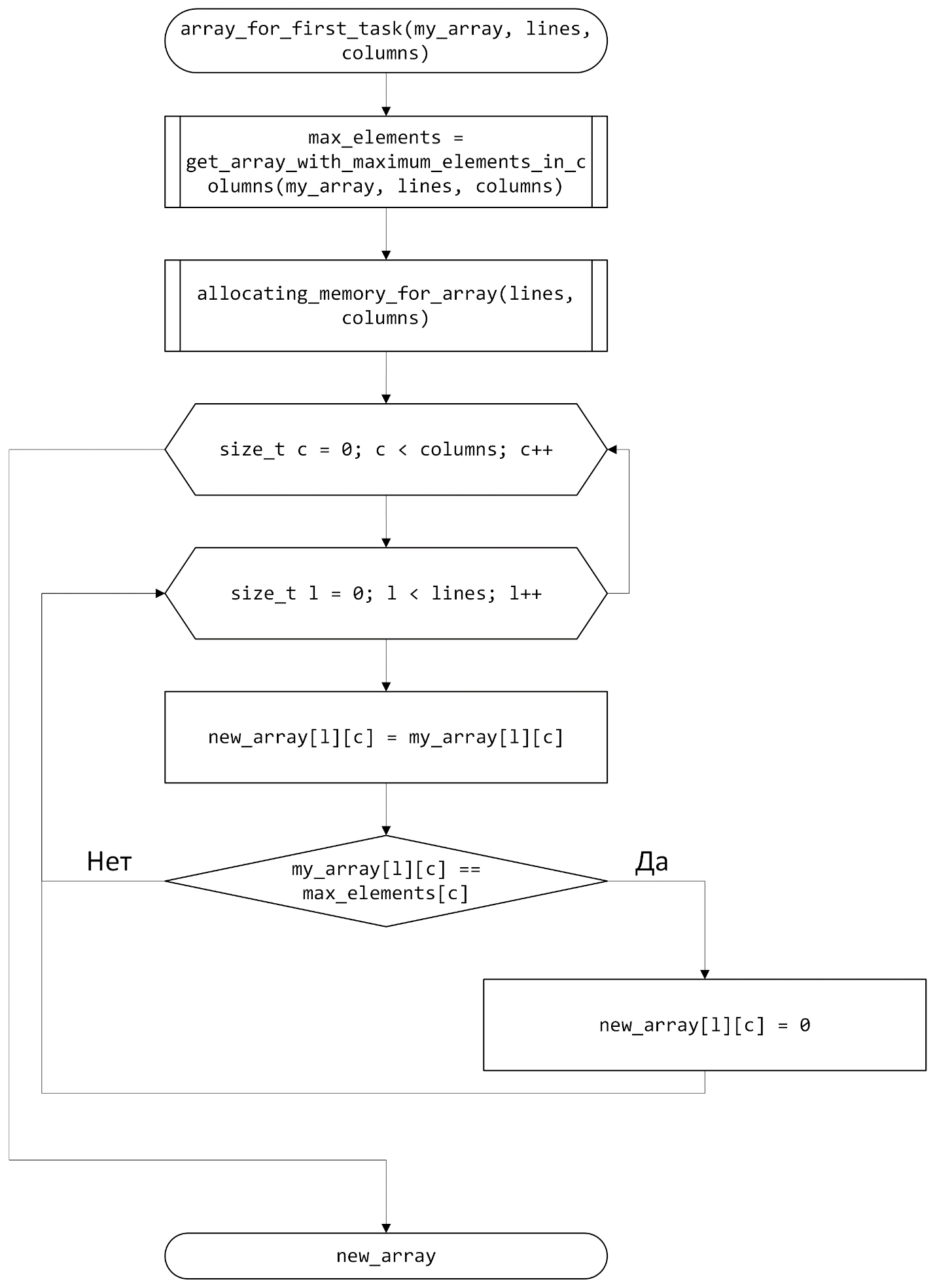


Рисунок 60 – Блок-схема функции array\_for\_first\_task(my\_array, lines, columns)

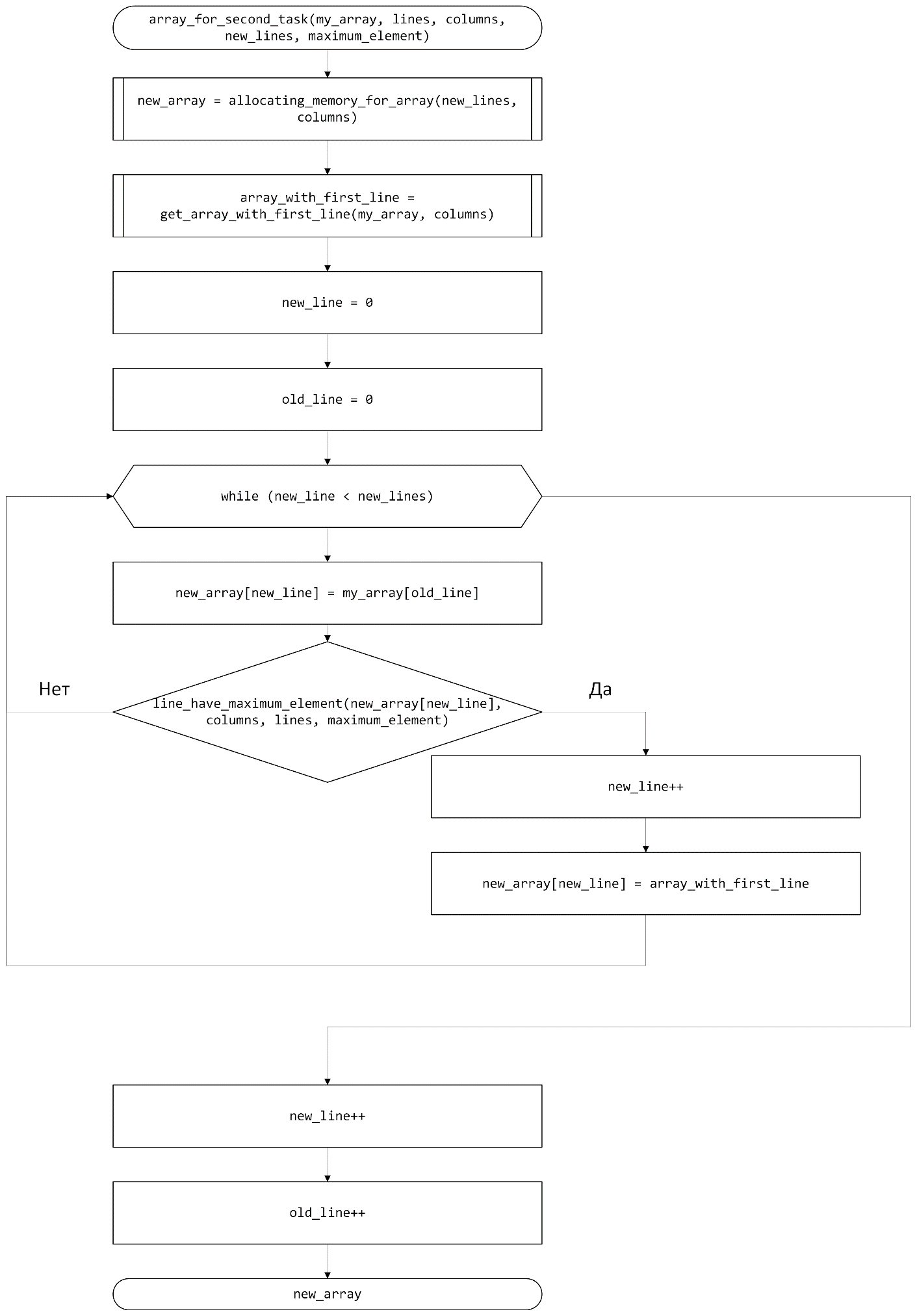


Рисунок 61 – Блок-схема функции array\_for\_second\_task(my\_array, lines, columns, new\_lines, maximum\_element)

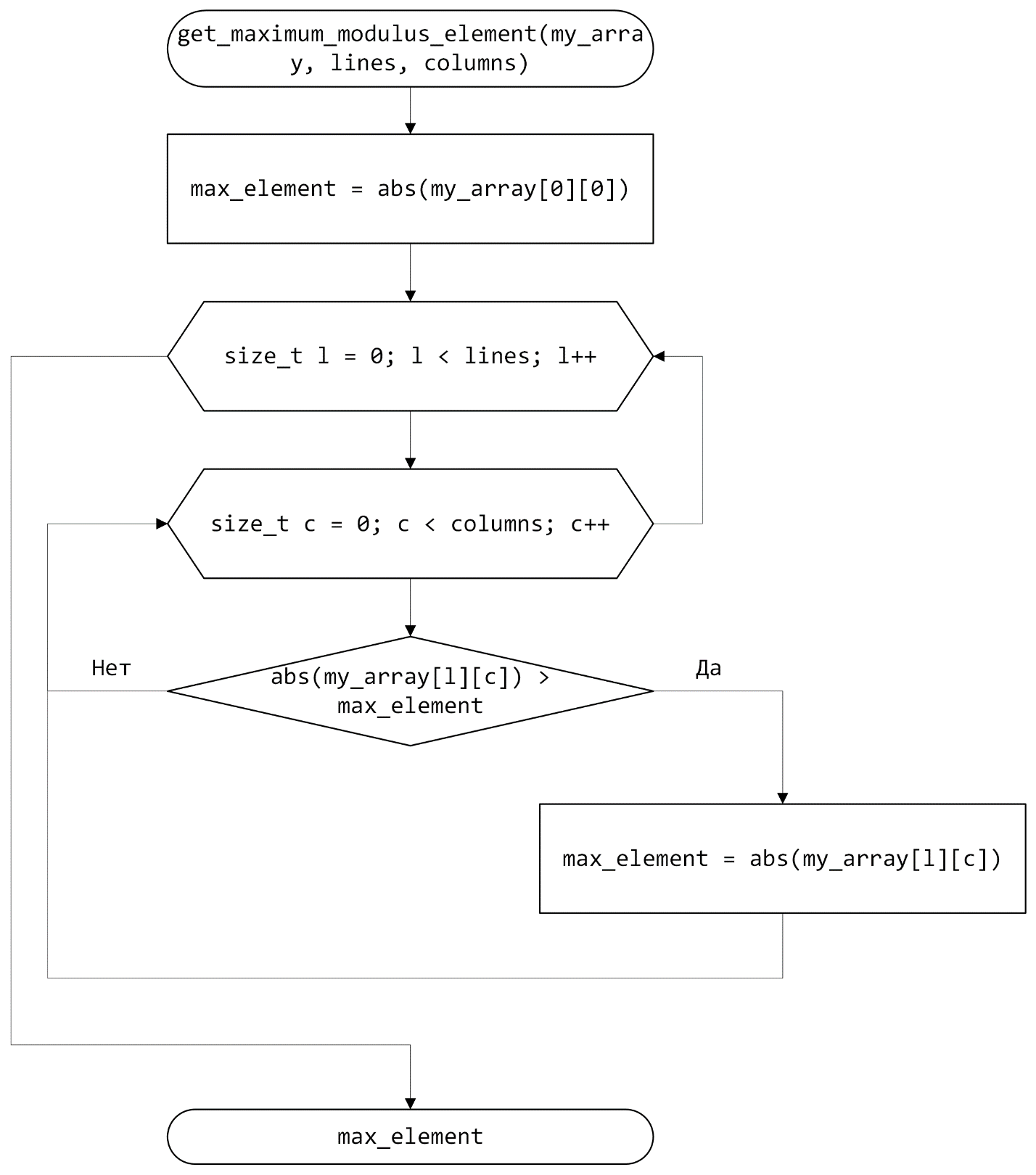


Рисунок 62 – Блок-схема функции get\_maximum\_modulus\_element(my\_array, lines, columns)

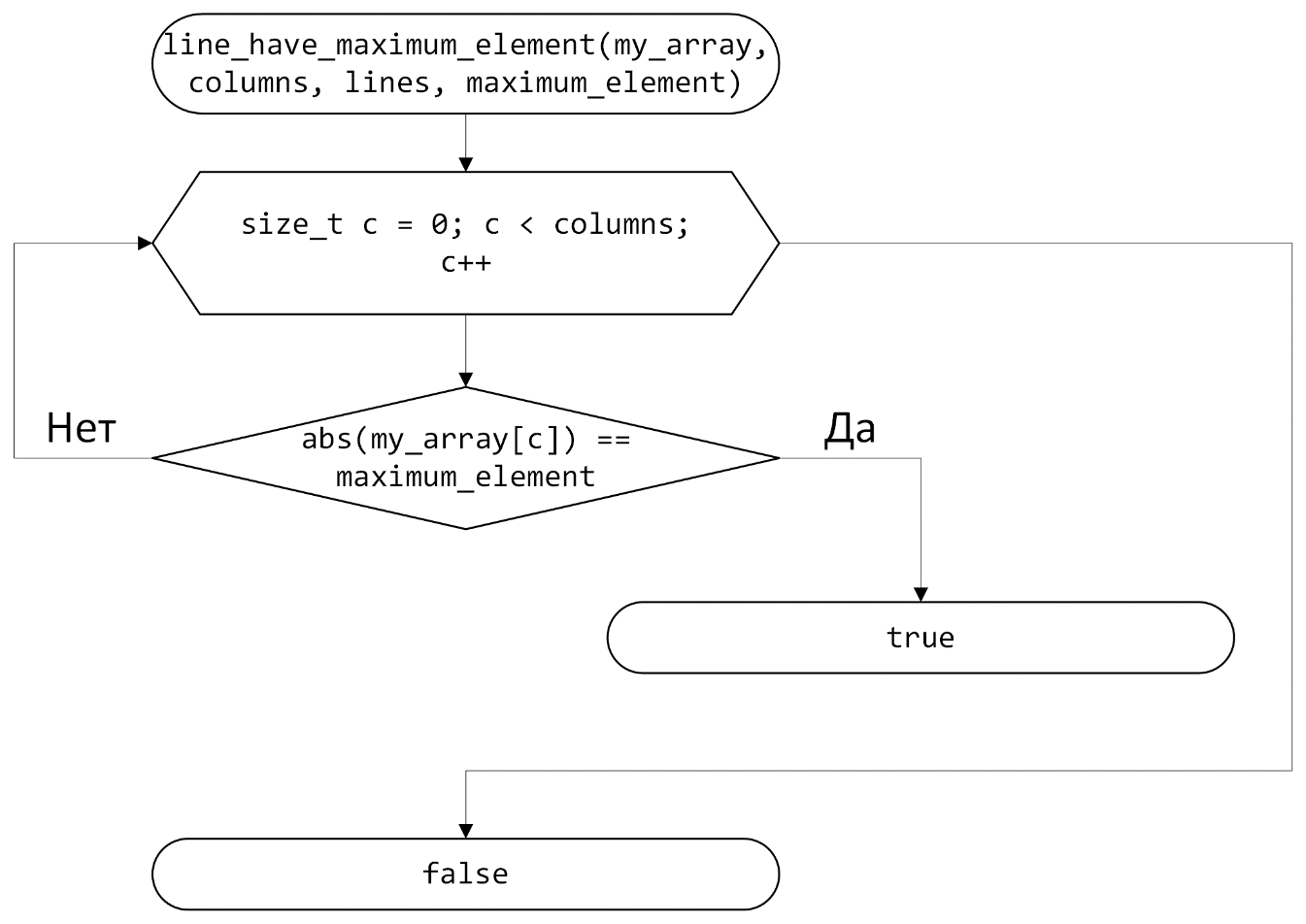


Рисунок 63 – Блок-схема функции line\_have\_maximum\_element(my\_array, columns, lines, maximum\_element)

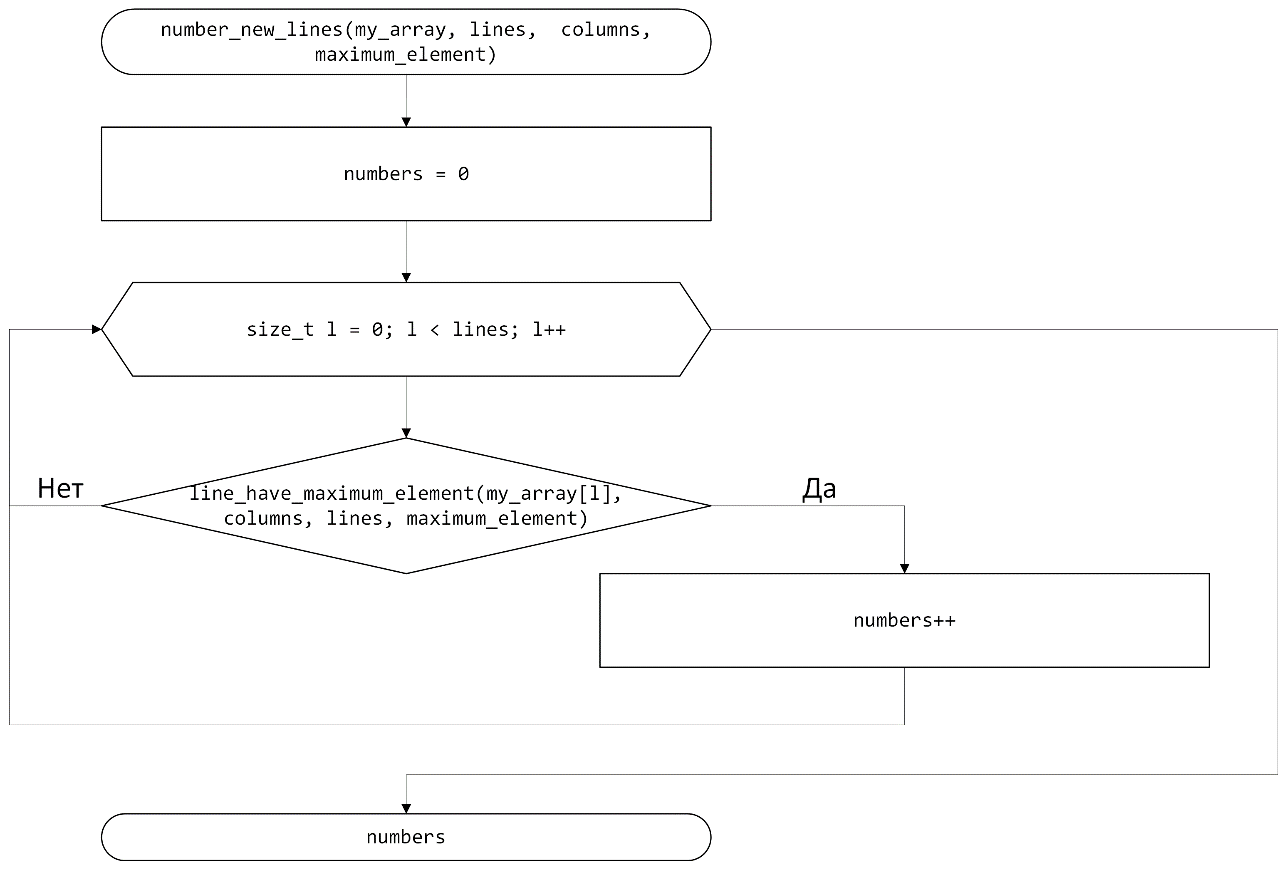


Рисунок 64 – Блок-схема функции number\_new\_lines(my\_array, lines, columns, maximum\_element)

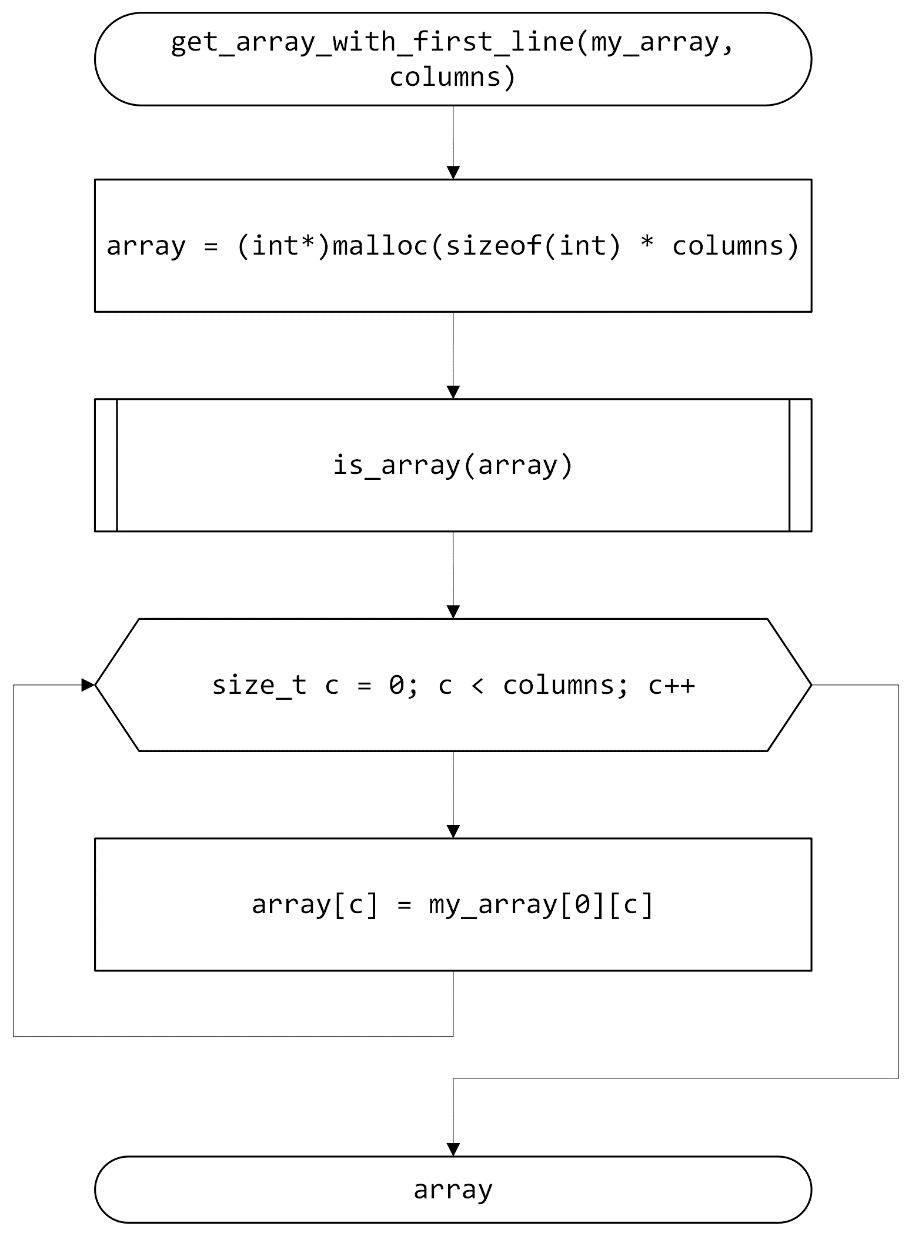


Рисунок 65 – Блок-схема функции get\_array\_with\_first\_line(my\_array, columns)

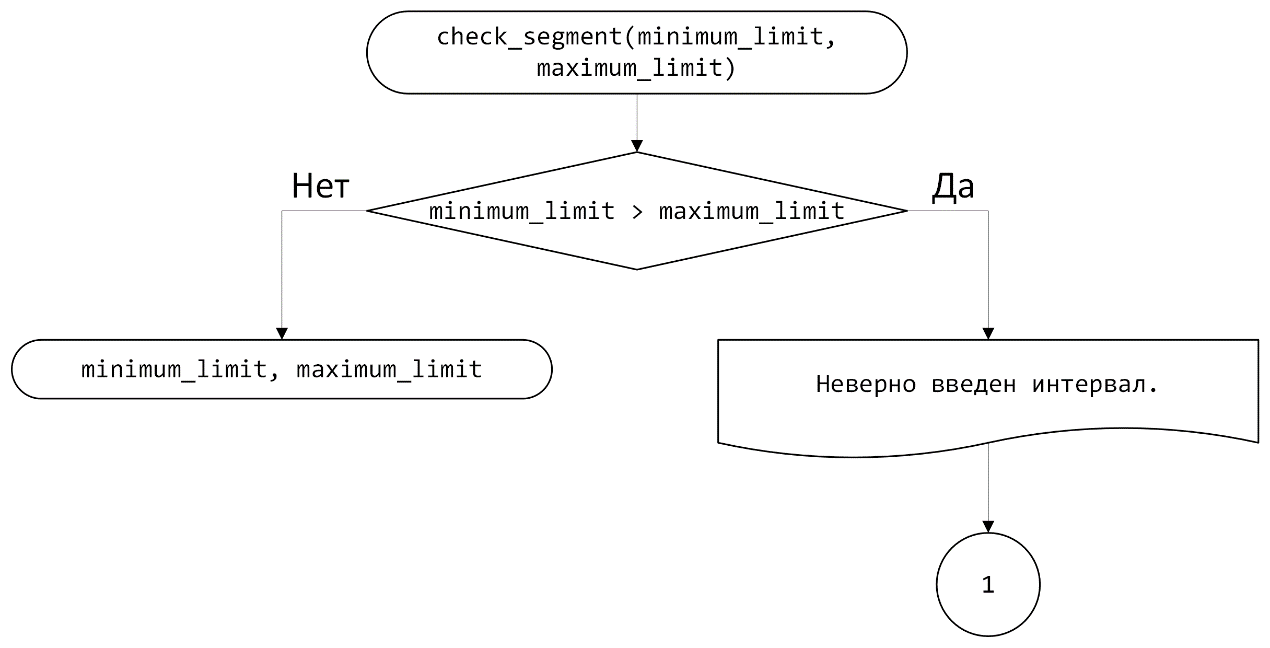


Рисунок 66 – Блок-схема функции check\_segment(minimum\_limit, maximum\_limit)

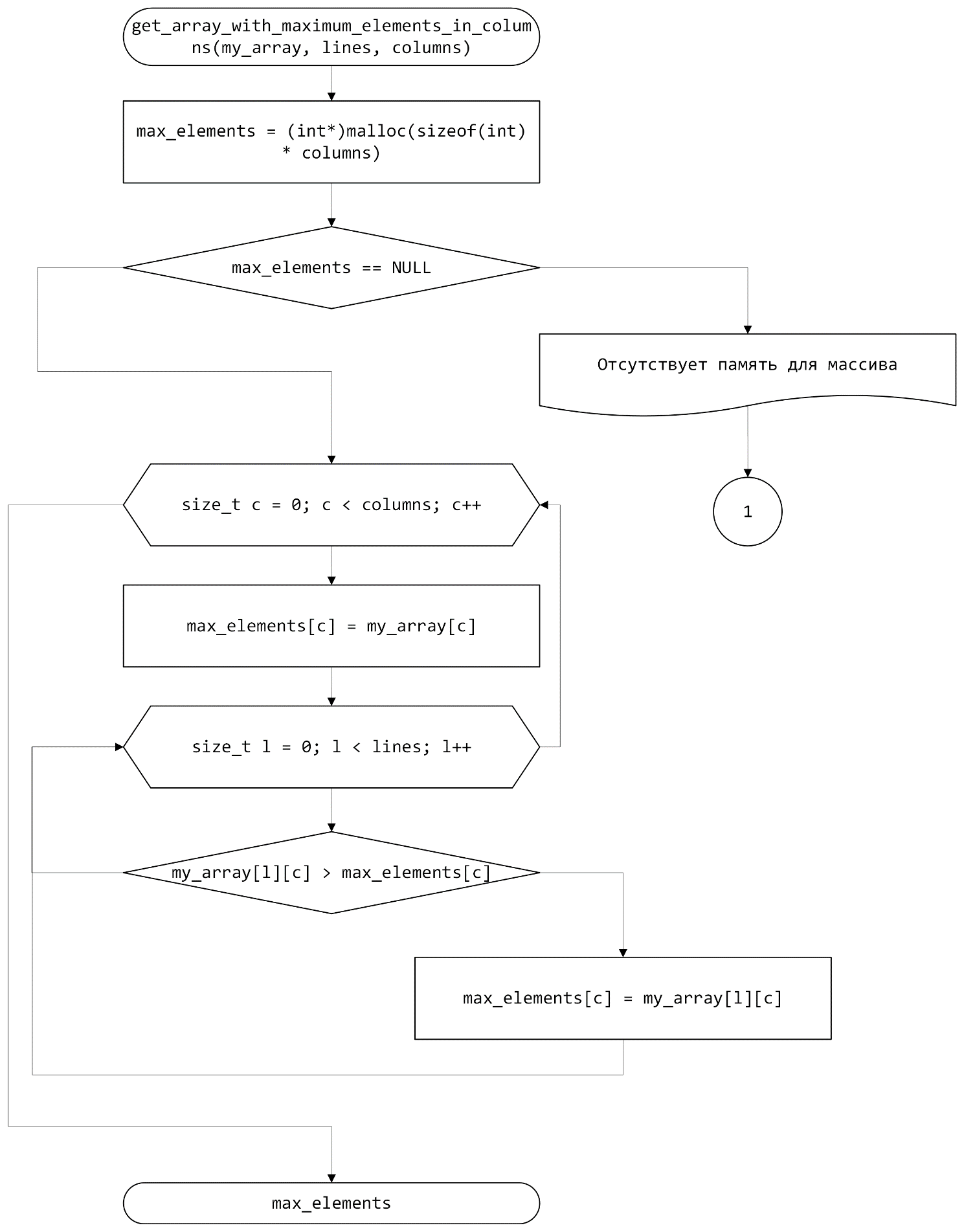


Рисунок 67 – Блок-схема функции get\_array\_with\_maximum\_elements\_in\_columns(my\_array, lines, columns)

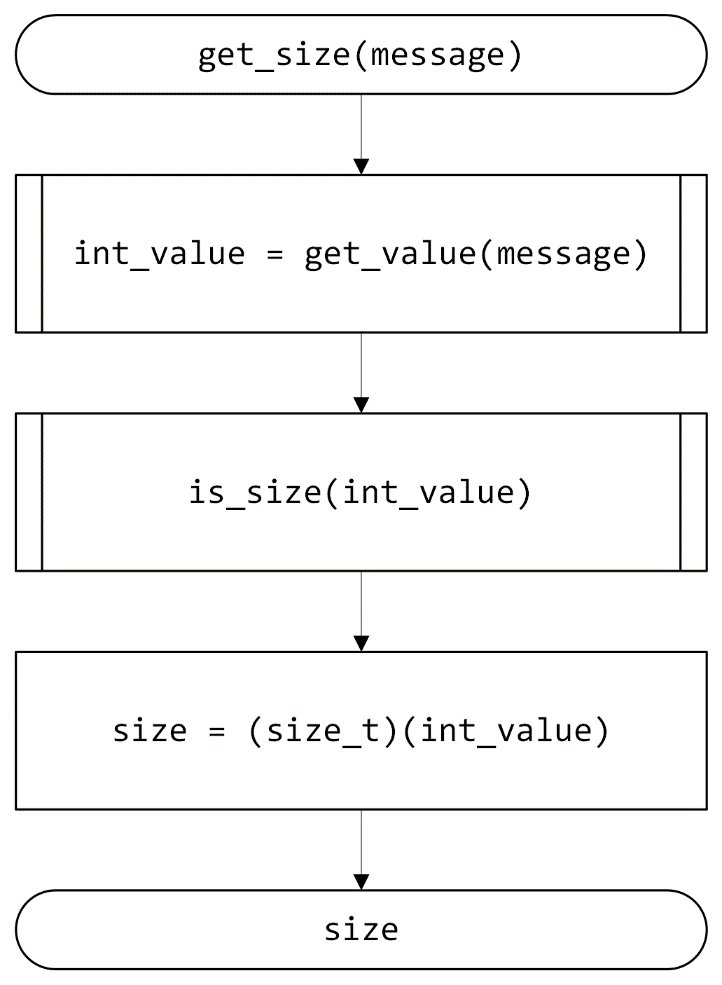


Рисунок 68 – Блок-схема функции get\_size(message)

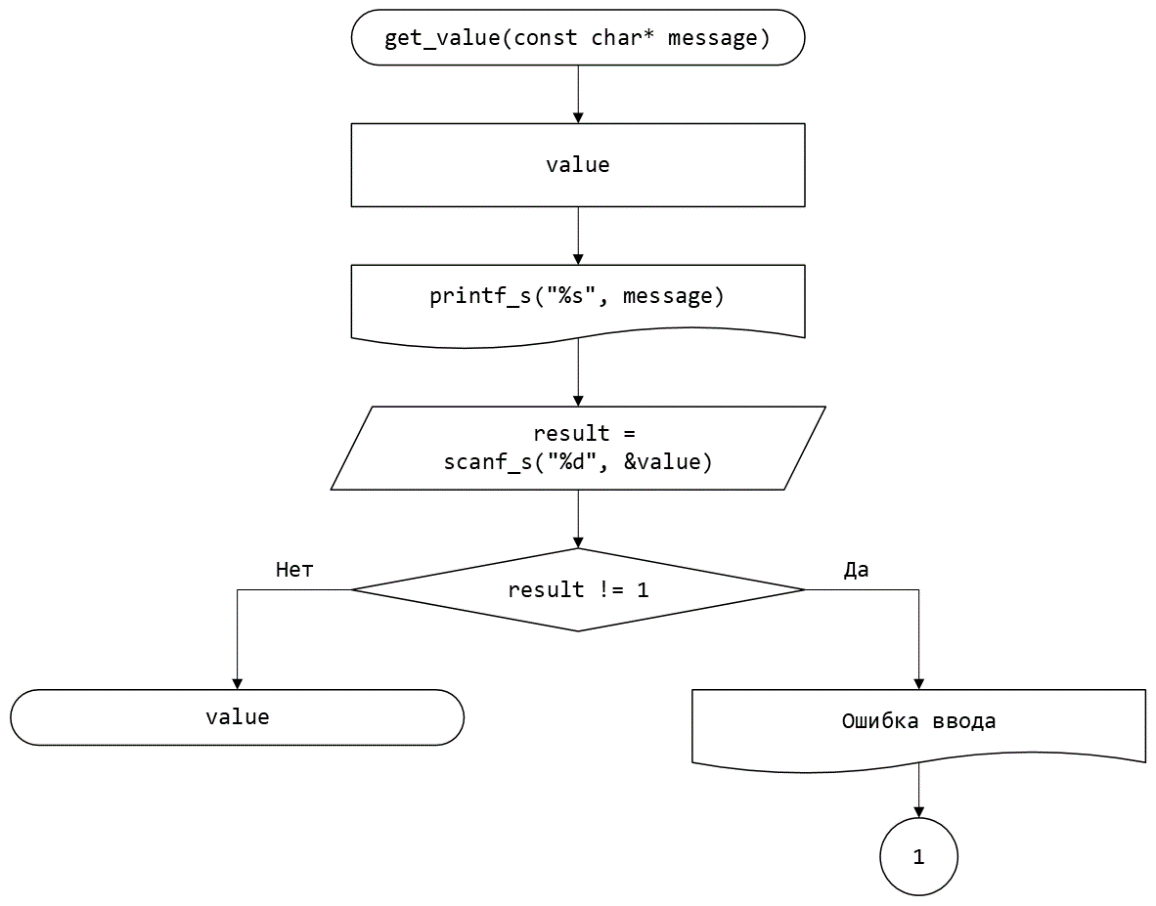


Рисунок 69 – Блок-схема функции get\_value(message)

* 1. Код задания 4.3

#include <stdio.h>

#include <locale.h>

#include <errno.h>

#include <math.h>

#include <stdlib.h>

#include <float.h>

#include <time.h>

#include <stdbool.h>

/\*\*

\* @brief Выбор заполнения массива.

\*/

enum array\_input\_choice

{

/\*\*

\* @brief Ручной способ.

\*/

MANUAL = 1,

/\*\*

\* @brief Заполнение массива случайными числами.

\*/

RANDOM = 2

};

/\*\*

\* @brief Функция для считывания значения с клавиатуры.

\* @param message - сообщение пользователю.

\* @remarks Экстренное завершение программы, в случае неправильного ввода.

\* @return Значение с клавиатуры.

\*/

int get\_value(const char\* message);

/\*\*

\* @brief Функция для проверки длинны или ширины массива.

\* @param int\_size - количество строк или столбцов массива.

\* @remarks Экстренное завершение программы, в случае неправильной длинны или ширины массива.

\*/

void is\_size(int int\_size);

/\*\*

\* @brief Функция для выделения памяти для массива.

\* @param lines - количество строк массива.

\* @param columns - количество столбцов массива.

\* @return Указатель на выделенную для массива память.

\*/

int\*\* allocating\_memory\_for\_array(const size\_t lines, const size\_t columns);

/\*\*

\* @brief Функция для проверки массива.

\* @param my\_array - массив.

\* @remarks Экстренное завершение программы, если место для массива не выделено.

\*/

void is\_array(int\*\* my\_array);

/\*\*

\* @brief Функция для считывания элементов массива с клавиатуры.

\* @param my\_array - массив.

\* @param lines - количество строк массива.

\* @param columns - количество столбцов массива.

\* @param minimum\_limit - минимальный элемент массива.

\* @param maximum\_limit - максимальный элемент массива.

\*/

void get\_manual\_input\_array(int\*\* my\_array, const size\_t lines, const size\_t columns, const int minimum\_limit, const int maximum\_limit);

/\*\*

\* @brief Функция для заполнения массива случайными числами.

\* @param my\_array - массив.

\* @param lines - количество строк массива.

\* @param columns - количество столбцов массива.

\* @param minimum\_limit - минимальный элемент массива.

\* @param maximum\_limit - максимальный элемент массива.

\*/

void get\_random\_input\_array(int\*\* my\_array, const size\_t lines, const size\_t columns, const int minimum\_limit, const int maximum\_limit);

/\*\*

\* @brief Функция для печати массива.

\* @param my\_array - массив.

\* @param lines - количество строк массива.

\* @param columns - количество столбцов массива.

\*/

void print\_array(int\*\* my\_array, size\_t lines, size\_t columns);

/\*\*

\* @brief Функция для замены максимальных элементов каждого столбца нулем.

\* @param my\_array - массив.

\* @param lines - количество строк массива.

\* @param columns - количество столбцов массива.

\* @return Массив с заменёнными числами.

\*/

int\*\* array\_for\_first\_task(int\*\* my\_array, size\_t lines, size\_t columns);

/\*\*

\* @brief Функция для создания и заполнения массива, в котором после строк, содержащих минимальный элемент, идут строки с порядковыми числами.

\* @param my\_array - исходный массив.

\* @param lines - количество строк массива.

\* @param columns - количество столбцов массива.

\* @param maximum\_element - максимальный по модулю элемент массива.

\* @param new\_lines - количество строк массива, в котором будут содержаться строки с порядковыми числами.

\* @return Массив, содержащий строки с элементами первой строки исходного массива.

\*/

int\*\* array\_for\_second\_task(int\*\* my\_array, size\_t lines, size\_t columns, size\_t new\_lines, int maximum\_element);

/\*\*

\* @brief Функция для нахождения максимального по модулю элемента массива.

\* @param my\_array - массив.

\* @param lines - количество строк массива.

\* @param columns - количество столбцов массива.

\* @return Максимальный по модулю элемент массива.

\*/

int get\_maximum\_modulus\_element(int\*\* my\_array, size\_t lines, size\_t columns);

/\*\*

\* @brief Функция для определения содержит строка максимальный по модулю элемент или нет.

\* @param my\_array - строка массива.

\* @param columns - количество столбцов массива.

\* @param maximum\_element - максимальный по модулю элемент массива.

\* @return true если строка содержит минимальный элемент массива, иначе false.

\*/

bool line\_have\_maximum\_element(int\* my\_array, size\_t columns, size\_t lines, int maximum\_element);

/\*\*

\* @brief Функция для подсчёта количества строк массива с минимальным элементом.

\* @param my\_array - массив.

\* @param lines - количество строк массива.

\* @param columns - количество столбцов массива.

\* @param maximum\_element - максимальный по модулю элемент массива.

\* @return Количество строк массива с максимальным по модулю элементо массива.

\*/

size\_t number\_new\_lines(int\*\* my\_array, size\_t lines, size\_t columns, int maximum\_element);

/\*\*

\* @brief Функция для создания массива с первой строкой исходного массива.

\* @param my\_array - исходный массив.

\* @param columns - количество столбцов массива.

\*/

int\* get\_array\_with\_first\_line(int\*\* my\_array, size\_t columns);

/\*\*

\* @brief Функция для проверки отрезка массива.

\* @param minimum\_limit - минимальный элемент массива.

\* @param maximum\_limit - максимальный элемент массива.

\*/

void check\_segment(const int minimum\_limit, const int maximum\_limit);

/\*\*

\* @brief Функция для создания и заполнения массива с максимальными жлементами каждого столбца исходного массива.

\* @param my\_array - исходный массив.

\* @param lines - количество строк массива.

\* @param columns - количество столбцов массива.

\*/

int\* get\_array\_with\_maximum\_elements\_in\_columns(int\*\* my\_array, size\_t lines, size\_t columns);

/\*\*

\* @brief Функция для считывания размера массива с клавиатуры.

\* @param message - сообщение пользователю.

\* @return Размер массива.

\*/

size\_t get\_size(const char\* message);

/\*\*

\* @brief Точка входа в программу.

\* @return Возвращает 0 в случае успеха.

\*/

int main()

{

setlocale(LC\_ALL, "RU");

size\_t columns = get\_size("Введите количество столбцов массива: ");

size\_t lines = get\_size("Введите количество строк массива: ");

const int minimum\_limit = get\_value("Введите минимум массива: ");

const int maximum\_limit = get\_value("Введите максимум массива: ");

check\_segment(minimum\_limit, maximum\_limit);

int\*\* my\_array = allocating\_memory\_for\_array(lines, columns);

puts("Введите номер соответствующий способу, которым вы хотите ввести массив:");

printf\_s("%d - для заполнения массива с клавиатуры\n", MANUAL);

printf\_s("%d - для заполнения массива случайными числами\n", RANDOM);

int choice = get\_value("");

enum array\_input\_choice user\_input = (enum user\_input)choice;

switch (user\_input)

{

case MANUAL:

{

get\_manual\_input\_array(my\_array, lines, columns, minimum\_limit, maximum\_limit);

break;

}

case RANDOM:

{

get\_random\_input\_array(my\_array, lines, columns, minimum\_limit, maximum\_limit, minimum\_limit, maximum\_limit);

break;

}

default:

{

errno = EDOM;

perror("Ошибка ввода");

return EXIT\_FAILURE;

}

}

puts("Исходный массив: ");

print\_array(my\_array, lines, columns);

puts("Ответ на первое задание: ");

print\_array(array\_for\_first\_task(my\_array, lines, columns), lines, columns);

int maximum\_element = get\_maximum\_modulus\_element(my\_array, lines, columns);

size\_t new\_lines = lines + number\_new\_lines(my\_array, lines, columns, maximum\_element);

puts("Ответ на второе задание: ");

print\_array(array\_for\_second\_task(my\_array, lines, columns, new\_lines, maximum\_element), new\_lines, columns);

is\_array(my\_array);

for (size\_t l = 0; l < lines; l++)

{

free(my\_array[l]);

}

free(my\_array);

return EXIT\_SUCCESS;

}

int get\_value(const char\* message)

{

int value;

printf("%s", message);

int res = scanf\_s("%d", &value);

if (res != 1)

{

errno = EIO;

perror("Ошибка ввода");

abort();

}

return value;

}

void is\_size(int int\_size)

{

if (int\_size <= 0)

{

errno = EIO;

perror("Ошибка ввода");

abort();

}

}

int\*\* allocating\_memory\_for\_array(const size\_t lines, const size\_t columns)

{

int\*\* my\_array = (int\*\*)malloc(sizeof(int\*) \* lines);

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

{

my\_array[i] = (int\*)malloc(sizeof(int) \* columns);

}

is\_array(my\_array);

return my\_array;

}

void is\_array(int\*\* my\_array)

{

if (my\_array == NULL)

{

errno = ENOMEM;

perror("Отсутствует память для массива");

abort();

}

}

void get\_manual\_input\_array(int\*\* my\_array, const size\_t lines, const size\_t columns, const int minimum\_limit, const int maximum\_limit)

{

for (size\_t l = 0; l < lines; l++)

{

for (size\_t c = 0; c < columns; c++)

{

printf("Введите элемент a[%d][%d] от %d до %d: ", l, c, minimum\_limit, maximum\_limit);

my\_array[l][c] = get\_value("");

if (my\_array[l][c] < minimum\_limit || my\_array[l][c] > maximum\_limit)

{

errno = EIO;

perror("Ошибка ввода");

abort();

}

}

}

}

void get\_random\_input\_array(int\*\* my\_array, const size\_t lines, const size\_t columns, const int minimum\_limit, const int maximum\_limit)

{

unsigned int ttime = (unsigned int)(time(NULL));

srand(ttime);

for (size\_t l = 0; l < lines; l++)

{

for (size\_t c = 0; c < columns; c++)

{

my\_array[l][c] = minimum\_limit + rand() % (maximum\_limit - minimum\_limit + 1);

}

}

}

void print\_array(int\*\* my\_array, size\_t lines, size\_t columns)

{

for (size\_t l = 0; l < lines; l++)

{

for (size\_t c = 0; c < columns; c++)

{

printf\_s("%d\t", my\_array[l][c]);

}

printf\_s("\n");

}

}

int\*\* array\_for\_first\_task(int\*\* my\_array, size\_t lines, size\_t columns)

{

int\* max\_elements = get\_array\_with\_maximum\_elements\_in\_columns(my\_array, lines, columns);

int\*\* new\_array = allocating\_memory\_for\_array(lines, columns);

for (size\_t c = 0; c < columns; c++)

{

for (size\_t l = 0; l < lines; l++)

{

new\_array[l][c] = my\_array[l][c];

if (my\_array[l][c] == max\_elements[c])

{

new\_array[l][c] = 0;

}

}

}

return new\_array;

}

int\*\* array\_for\_second\_task(int\*\* my\_array, size\_t lines, size\_t columns, size\_t new\_lines, int maximum\_element)

{

int\*\* new\_array = allocating\_memory\_for\_array(new\_lines, columns);

int\* array\_with\_first\_line = get\_array\_with\_first\_line(my\_array, columns);

size\_t new\_line = 0;

size\_t old\_line = 0;

while (new\_line < new\_lines)

{

new\_array[new\_line] = my\_array[old\_line];

if (line\_have\_maximum\_element(new\_array[new\_line], columns, lines, maximum\_element))

{

new\_line++;

new\_array[new\_line] = array\_with\_first\_line;

}

new\_line++;

old\_line++;

}

return new\_array;

}

int get\_maximum\_modulus\_element(int\*\* my\_array, size\_t lines, size\_t columns)

{

int max\_element = abs(my\_array[0][0]);

for (size\_t l = 0; l < lines; l++)

{

for (size\_t c = 0; c < columns; c++)

{

if (abs(my\_array[l][c]) > max\_element)

{

max\_element = abs(my\_array[l][c]);

}

}

}

return max\_element;

}

bool line\_have\_maximum\_element(int\* my\_array, size\_t columns, size\_t lines, int maximum\_element)

{

for (size\_t c = 0; c < columns; c++)

{

if (abs(my\_array[c]) == maximum\_element)

{

return true;

}

}

return false;

}

size\_t number\_new\_lines(int\*\* my\_array, size\_t lines, size\_t columns, int maximum\_element)

{

size\_t numbers = 0;

for (size\_t l = 0; l < lines; l++)

{

if (line\_have\_maximum\_element(my\_array[l], columns, lines, maximum\_element))

{

numbers++;

}

}

return numbers;

}

int\* get\_array\_with\_first\_line(int\*\* my\_array, size\_t columns)

{

int\* array = (int\*)malloc(sizeof(int) \* columns);

is\_array(array);

for (size\_t c = 0; c < columns; c++)

{

array[c] = my\_array[0][c];

}

return array;

}

void check\_segment(const int minimum\_limit, const int maximum\_limit)

{

if (minimum\_limit > maximum\_limit)

{

puts("Неверно введен интервал.");

abort();

}

}

int\* get\_array\_with\_maximum\_elements\_in\_columns(int\*\* my\_array, size\_t lines, size\_t columns)

{

int\* max\_elements = (int\*)malloc(sizeof(int) \* columns);

if (max\_elements == NULL)

{

errno = EIO;

perror("Отсутствует память для массива");

abort();

}

for (size\_t c = 0; c < columns; c++)

{

max\_elements[c] = my\_array[c];

for (size\_t l = 0; l < lines; l++)

{

if (my\_array[l][c] > max\_elements[c])

{

max\_elements[c] = my\_array[l][c];

}

}

}

return max\_elements;

}

size\_t get\_size(const char\* message)

{

int int\_value = get\_value(message);

is\_size(int\_value);

size\_t size = (size\_t)(int\_value);

return size;

}

* 1. Решение тестового примера

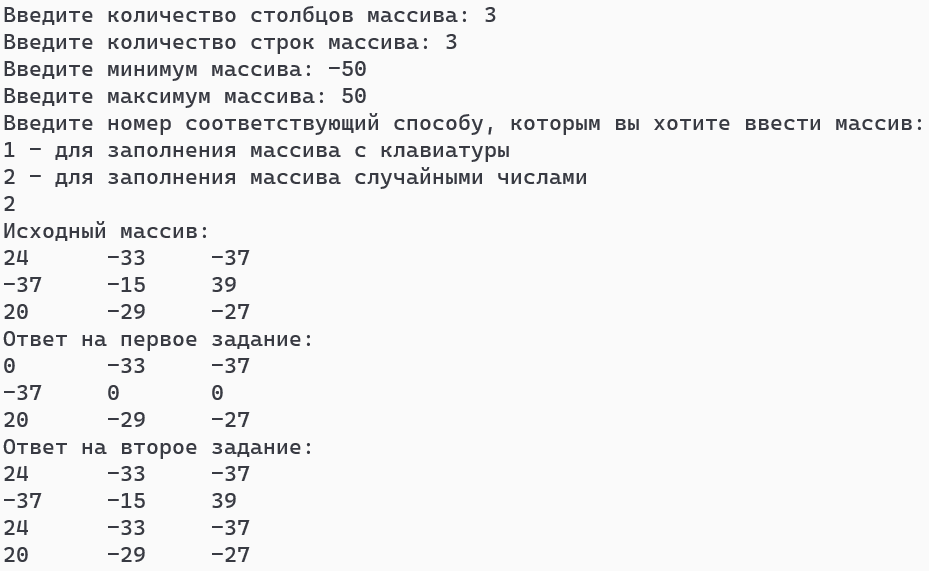


Рисунок 70 – Решение тестового примера при рандомном заполнении массива

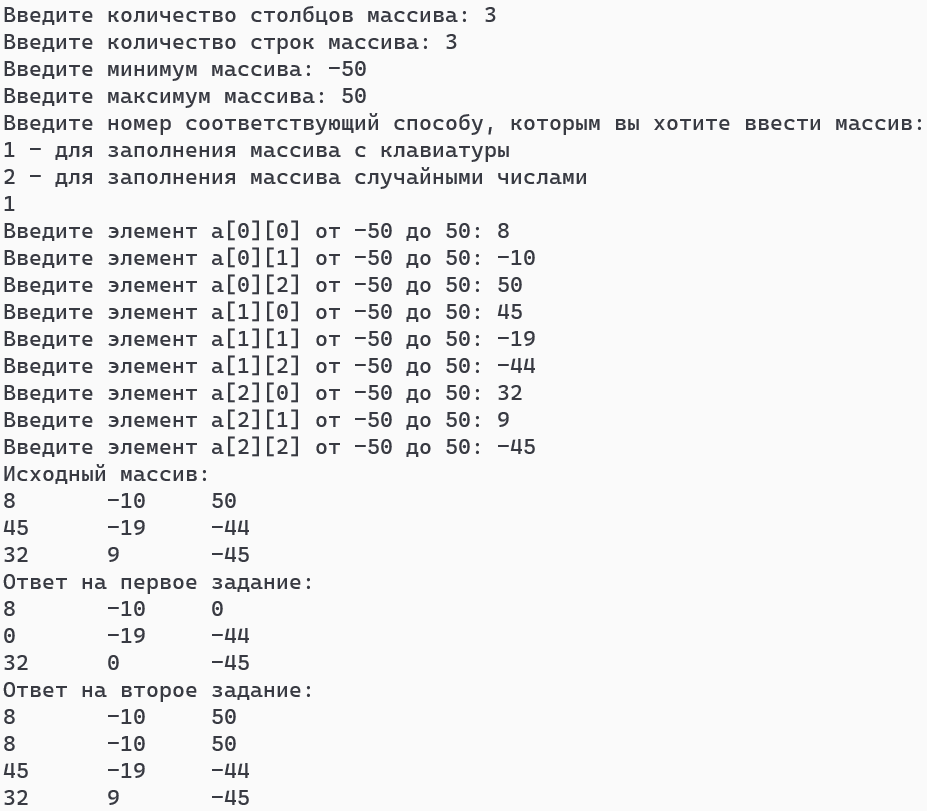


Рисунок 71 – Решение тестового примера при пользовательском заполнении массива



Рисунок 72 – Вывод программы, когда количество столбцов массива отрицательно



Рисунок 73 – Вывод программы, когда количество столбцов массива – буква

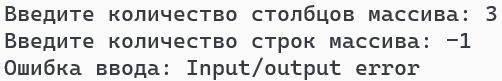


Рисунок 74 – Вывод программы, когда количество строк массива отрицательно

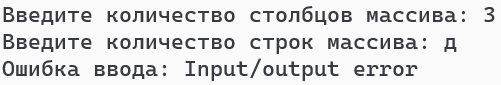


Рисунок 75 – Вывод программы, когда количество строк массива – буква

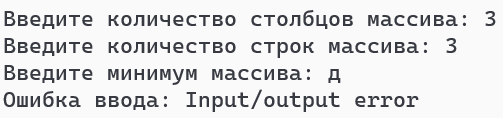


Рисунок 76 – Вывод программы, когда минимум массива – буква

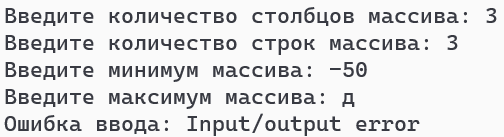


Рисунок 77 – Вывод программы, когда максимум массива – буква

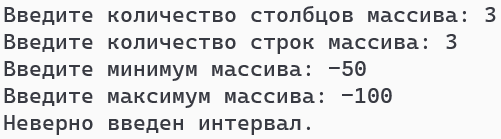


Рисунок 78 – Вывод программы, когда минимум массива больше максимума

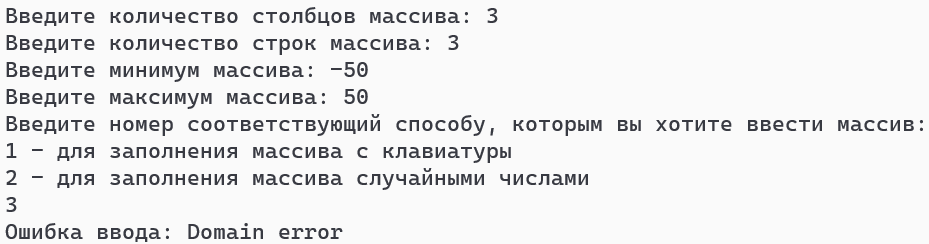


Рисунок 79 – Вывод программы, когда выбор способа заполнения массива не соответствует указанным значениям

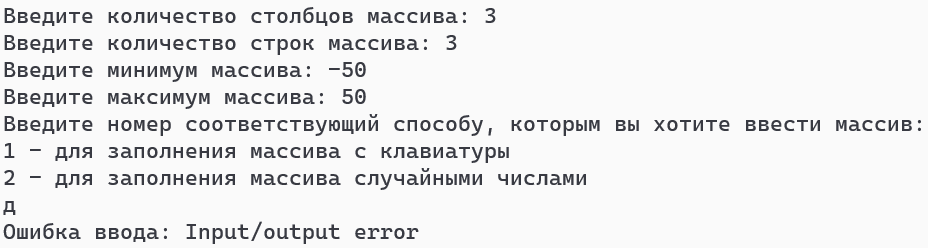


Рисунок 80 – Вывод программы, когда выбор способа заполнения массива – буква

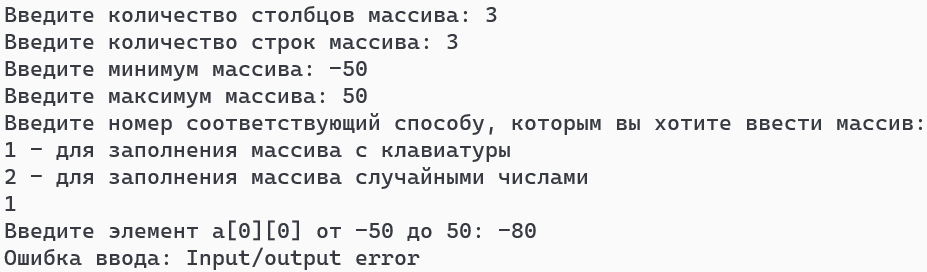


Рисунок 81 – Вывод программы, когда элемент массива при пользовательском заполнении массива не входит в промежуток

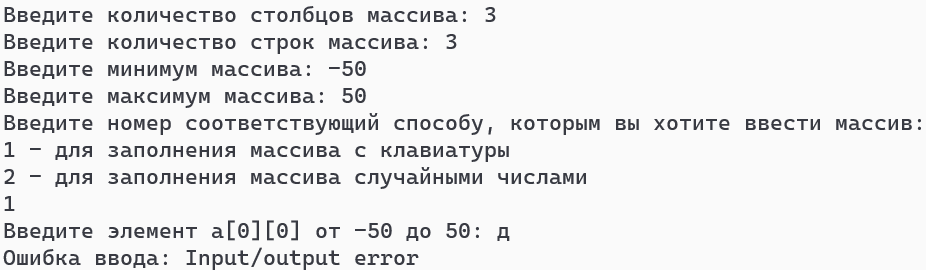


Рисунок 82 – Вывод программы, когда элемент массива при пользовательском заполнении – буква

* 1. Зачёт задания в GitHub

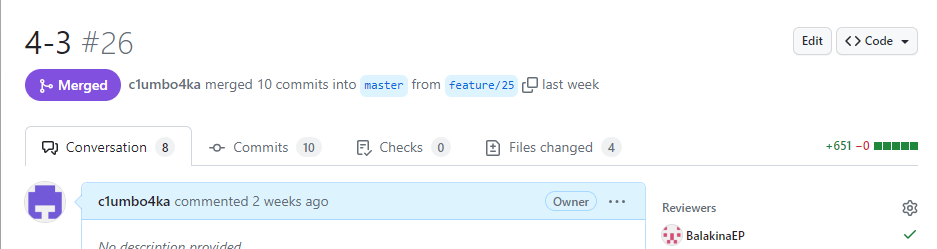


Рисунок 83 – Зачёт задания в GitHub