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 «Введение в языки программирования»

Выполнил: ст. гр. ТКИ – 142

Плешанов И.С.

Проверил: к.т.н., доц.

Васильева М.А.

Москва 2024

Оглавление

[**Задание 4–1** 2](#_Toc91434532)

[**Условие задания**](#_Toc91434533) 2

[**Блок-схема алгоритма** 4](#_Toc91434534)

[**Программа на языке С** 11](#_Toc91434535)

[**Решение тестовых примеров на С** 15](#_Toc91434536)

[**Решение тестовых примеров в Python** 16](#_Toc91434537)

# **Задание 4–1**

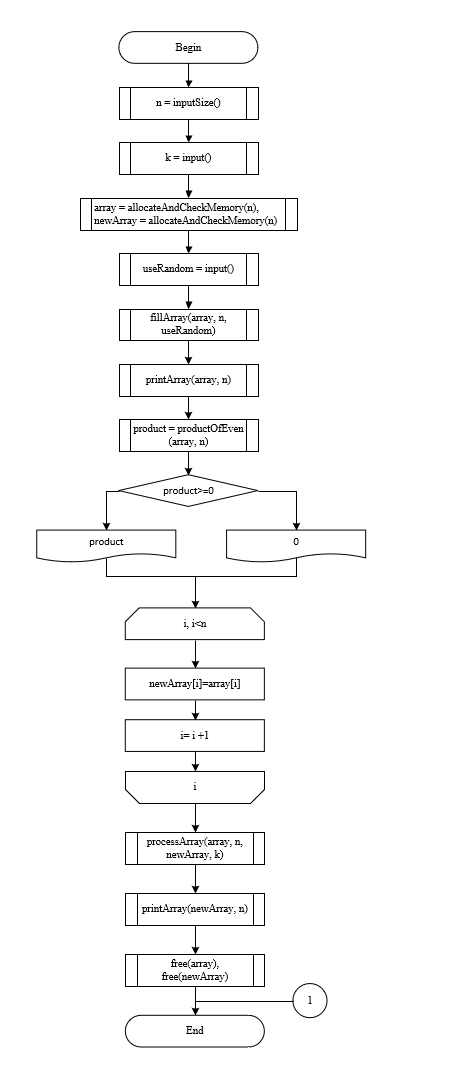
## **Условие задания**

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.

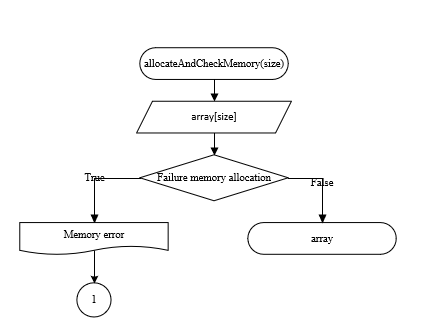
Таблица 1

|  |  |  |
| --- | --- | --- |
| 2 | 1. Найти произведение элементов, имеющих четное значение. 2. Заменить элементы массива с нечетными номерами на квадраты их номеров. 3. Определить, есть ли в данном массиве положительные элементы, делящиеся на заданное число *k* с остатком 2. | [-15;15] |

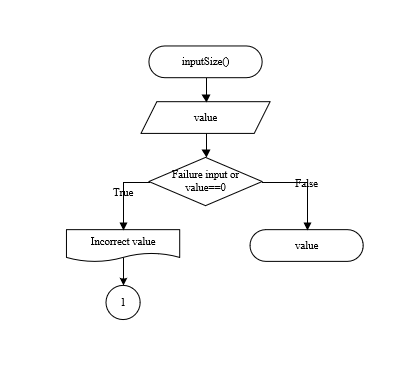
**Блок-схема алгоритма**

****

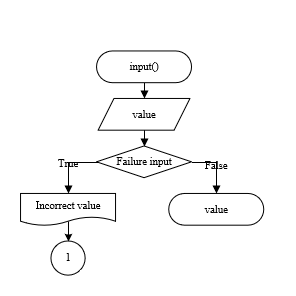
**Рисунок 1 – Блок-схема 4-1**



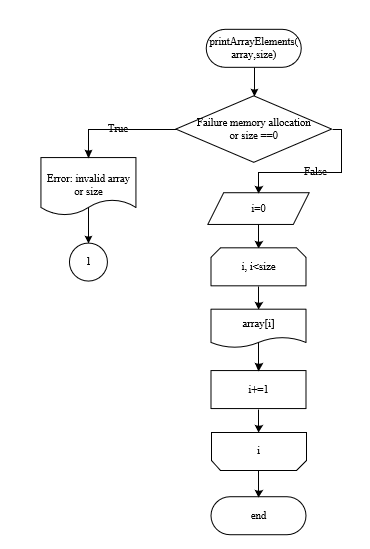
**Рисунок 2 – Блок схема allocateAndCheckMemory(size)**



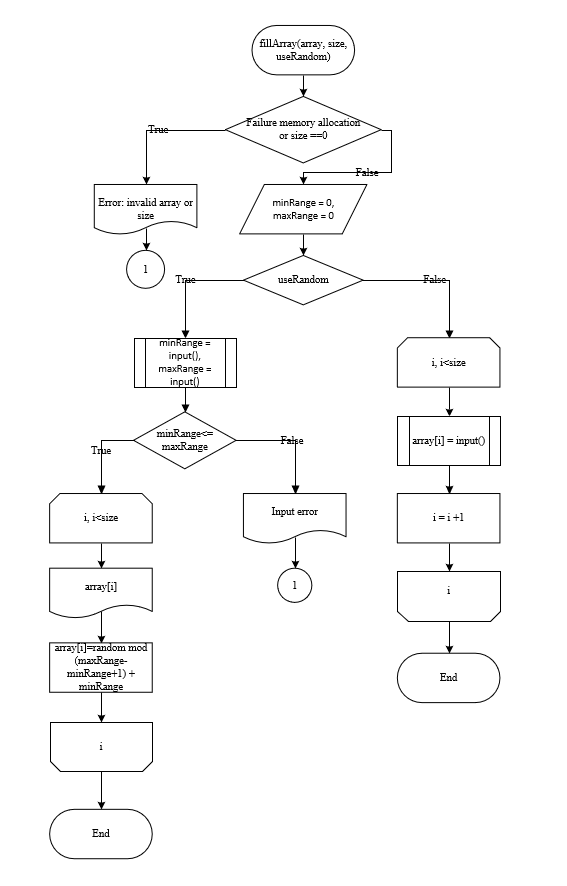
**Рисунок 3 – Блок схема inputSize()**

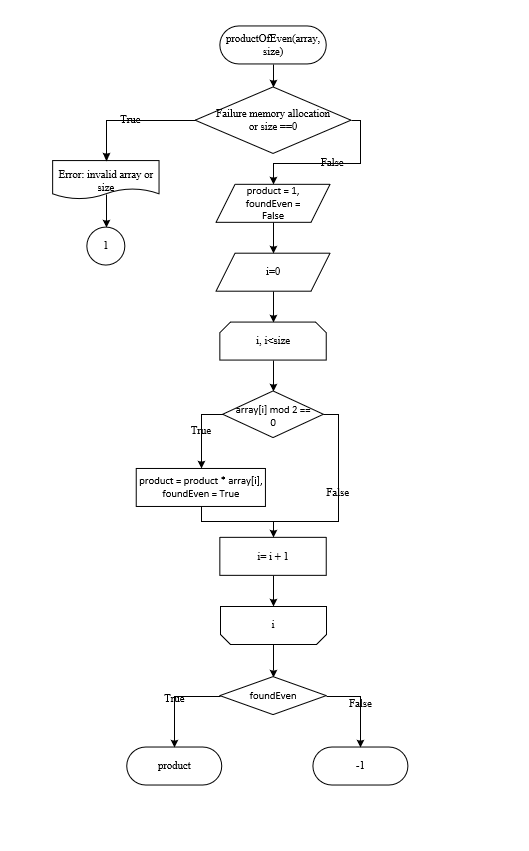


**Рисунок 4 – Блок схема input()**

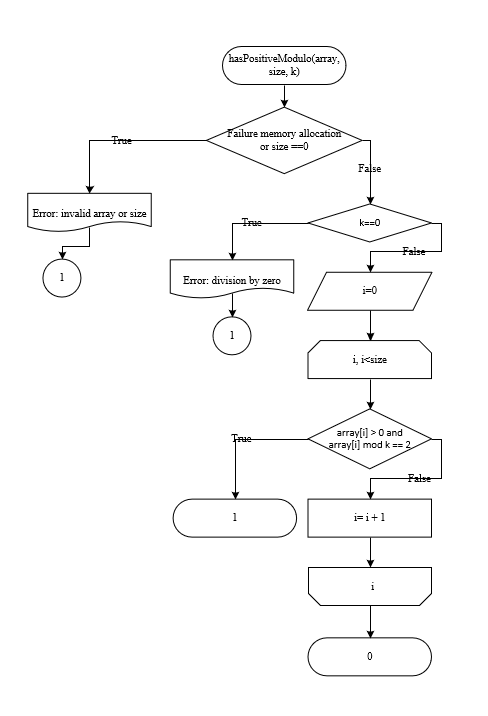
****

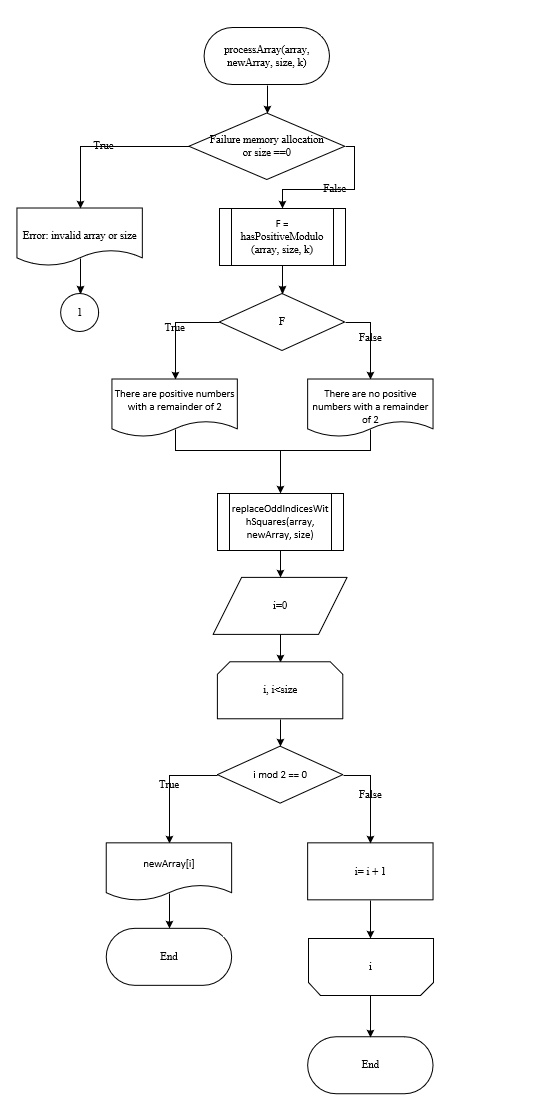
**Рисунок 5 – Блок схема printArrayElements(array,size)**

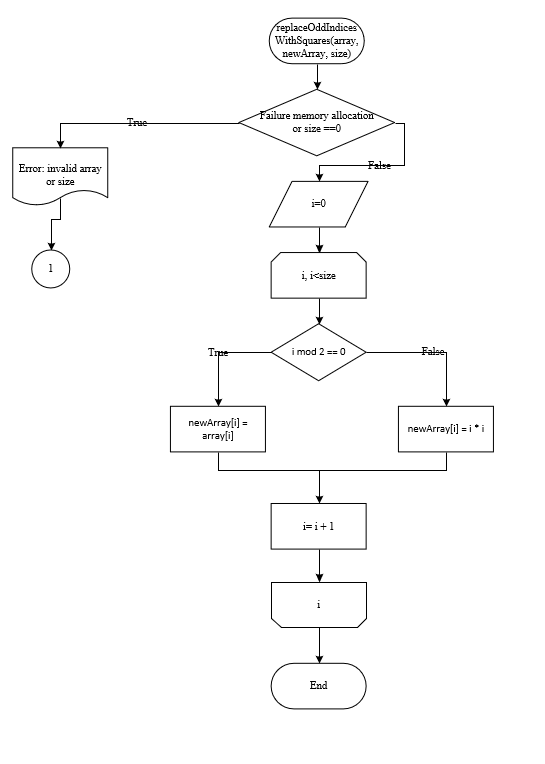
**Рисунок 6 – Блок схема fillArray(array, size, useRandom)**



**Рисунок 7 – Блок схема productOfEven(array, size)**

**Рисунок 8 – Блок схема hasPositiveModulo(array, size, k)**

**Рисунок 9 – Блок схема processArray(array, newArray, size, k)**

**Рисунок 10 – Блок схема replaceOddIndicesWithSquares(array, newArray, size)**

## **Программа на языке С**

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

#include <locale.h>

#include <stdbool.h>

/\*

\* @brief Заполняет массив числами, случайными или пользовательскими.

\* @param array Указатель на первый элемент массива.

\* @param size Размер массива.

\* @param useRandom Если равно 1, заполняет случайными числами; если 0, запрашивает ввод от пользователя.

\*/

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

/\*

\* @brief Выводит массив на экран.

\* @param array Указатель на первый элемент массива.

\* @param size Размер массива.

\*/

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

/\*

\* @brief Вычисляет произведение четных элементов массива.

\* @param array Указатель на первый элемент массива.

\* @param size Размер массива.

\* @return Произведение четных элементов. Возвращает -1, если нет четных элементов.

\*/

long long productOfEven(const int\* array, const size\_t size);

/\*

\* @brief Заменяет элементы на нечетных индексах квадратами их индексов.

\* @param array Указатель на первый элемент массива.

\* @param size Размер массива.

\*/

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

/\*

\* @brief Проверяет, имеются ли положительные элементы с остатком 2 при делении на k.

\* @param array Указатель на первый элемент массива.

\* @param size Размер массива.

\* @param k Значение делителя.

\* @return 1, если найдены такие элементы, иначе 0.

\*/

int hasPositiveModulo(const int\* array, const size\_t size, const int k);

/\*

\* @brief Процедура обработки массива: проверяет положительные элементы и выполняет замену по индексам.

\* @param array Исходный массив.

\* @param size Размер массива.

\* @param newArray Модифицированный массив.

\* @param k Делитель для проверки с остатком 2.

\*/

void processArray(const int\* array, const size\_t size, int\* newArray, const int k);

/\*

\* @brief Запрашивает у пользователя ввод целого числа.

\* @param prompt Строка запроса для вывода пользователю.

\* @return Введенное пользователем целое число.

\*/

int inputInt(const char\* prompt);

/\*

\* @brief Запрашивает у пользователя ввод положительного целого числа (size\_t).

\* @param prompt Строка запроса для вывода пользователю.

\* @return Введенное пользователем положительное целое число типа size\_t.

\*/

size\_t inputSizeT(const char\* prompt);

/\*

\* @brief Выделяет память для массива целых чисел и проверяет успешность выделения.

\* @param size Количество элементов, для которых выделяется память.

\* @return Указатель на выделенную область памяти, если выделение успешно.

\* Программа завершает выполнение, если выделение памяти не удалось.

\*/

int\* allocateAndCheckMemory(size\_t size);

/\*

\* @brief Главная функция программы, выполняющая несколько операций над массивом целых чисел.

\* @return 0 при успешном завершении, 1 при ошибке.

\*/

int main(void) {

setlocale(LC\_ALL, "");

size\_t n = inputSizeT("Введите размер массива: ");

int k = inputInt("Введите число k: ");

int\* array = allocateAndCheckMemory(n);

int\* newArray = allocateAndCheckMemory(n);

int useRandom = inputInt("Заполнить массив случайными числами? (1 - да, 0 - нет): ");

fillArray(array, n, useRandom);

printf("Массив:\n");

printArray(array, n);

long long product = productOfEven(array, n);

printf("Произведение четных элементов: %lld\n", product >= 0 ? product : 0);

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

newArray[i] = array[i];

}

processArray(array, n, newArray, k);

printf("Массив после процесса:\n");

printArray(newArray, n);

free(array);

free(newArray);

return 0;

}

void fillArray(int\* array, const size\_t size, const int useRandom) {

if (array == NULL || size == 0) {

printf("Ошибка: некорректный массив или размер.\n");

return;

}

int minRange = 0;

int maxRange = 0;

if (useRandom) {

while (1) {

minRange = inputInt("Введите минимальное значение для случайных чисел: ");

maxRange = inputInt("Введите максимальное значение для случайных чисел: ");

if (minRange <= maxRange) {

break;

}

else {

printf("Ошибка: минимальное значение не должно превышать максимальное. Попробуйте снова.\n");

}

}

srand((unsigned)time(NULL));

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

array[i] = minRange + rand() % (maxRange - minRange + 1);

}

}

else {

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

array[i] = inputInt("Введите элемент: ");

}

}

}

void printArray(const int\* array, const size\_t size) {

if (array == NULL || size == 0) {

printf("Ошибка: некорректный массив или размер.\n");

return;

}

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

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

}

printf("\n");

}

long long productOfEven(const int\* array, const size\_t size) {

if (array == NULL || size == 0) {

printf("Ошибка: некорректный массив или размер.\n");

return -1;

}

long long product = 1;

bool foundEven = false;

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

if (array[i] % 2 == 0) {

product \*= array[i];

foundEven = true;

}

}

return foundEven ? product : -1;

}

int hasPositiveModulo(const int\* array, const size\_t size, const int k) {

if (array == NULL || size == 0) {

printf("Ошибка: некорректный массив или размер.\n");

return 0;

}

if (k == 0) {

printf("Ошибка: деление на ноль.\n");

return 0;

}

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

if (array[i] > 0 && array[i] % k == 2) {

return 1;

}

}

return 0;

}

void replaceOddIndicesWithSquares(const int\* array, int\* newArray, const size\_t size) {

if (array == NULL || newArray == NULL || size == 0) {

printf("Ошибка: некорректные массивы или размер.\n");

return;

}

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

if (i % 2 != 0) {

newArray[i] = i \* i;

}

else {

newArray[i] = array[i];

}

}

}

void processArray(const int\* array, const size\_t size, int\* newArray, const int k) {

if (array == NULL || newArray == NULL || size == 0) {

printf("Ошибка: некорректные массивы или размер.\n");

return;

}

if (hasPositiveModulo(array, size, k)) {

printf("Есть положительные числа с остатком 2.\n");

}

else {

printf("Положительных чисел с остатком 2 нет.\n");

}

replaceOddIndicesWithSquares(array, newArray, size);

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

if (newArray[i] % 2 == 0) {

printf("Первый четный элемент найден: %d\n", newArray[i]);

abort();

}

}

}

int inputInt(const char\* prompt) {

int value = 0;

printf("%s", prompt);

if (scanf("%d", &value) != 1) {

perror("Ошибка ввода");

abort();

}

while (getchar() != '\n');

return value;

}

size\_t inputSizeT(const char\* prompt) {

size\_t value = 0;

printf("%s", prompt);

if (scanf("%zu", &value) != 1 || value == 0) {

perror("Ошибка ввода");

abort();

}

while (getchar() != '\n');

return value;

}

int\* allocateAndCheckMemory(size\_t size) {

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

if (array == NULL) {

perror("Ошибка выделения памяти");

exit(EXIT\_FAILURE);

}

return array;

}

## **Решение тестовых примеров на С**

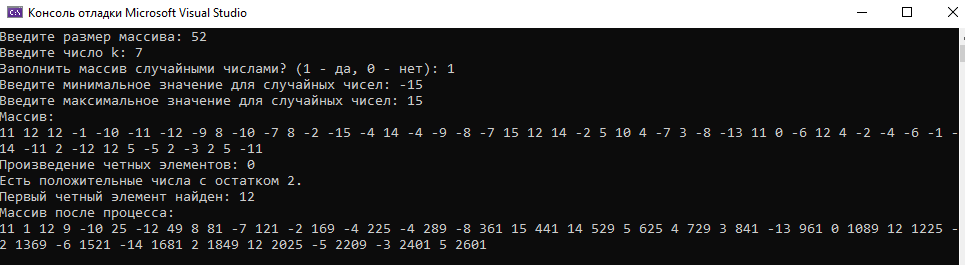
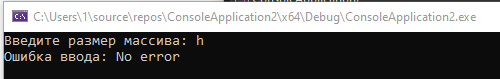


Рисунок 11 – Решение тестового примера для задания 4-1 на С



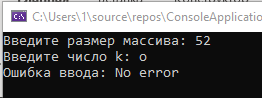
Рисунок 12 – ошибка решение тестового примера для задания 4-1 на С

Рисунок 13 – ошибка решение тестового примера для задания 4-1 на С

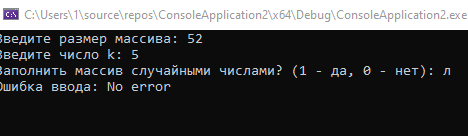


Рисунок 14 – ошибка решение тестового примера для задания 4-1 на С

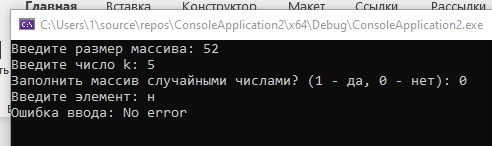


Рисунок 15 – ошибка решение тестового примера для задания 4-1 на С

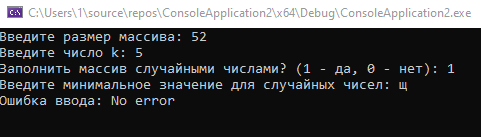


Рисунок 16 – ошибка решение тестового примера для задания 4-1 на С

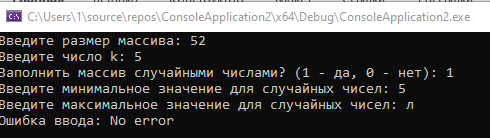


Рисунок 17 – ошибка решение тестового примера для задания 4-1 на С

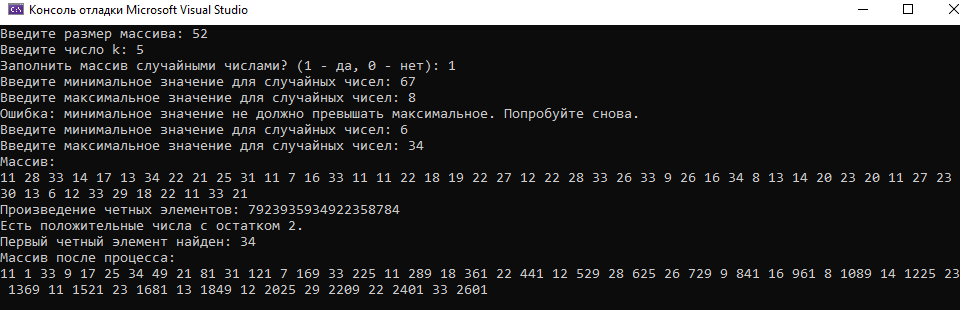


Рисунок 18 – ошибка решение тестового примера для задания 4-1 на С

## **Решение тестовых примеров в Python**

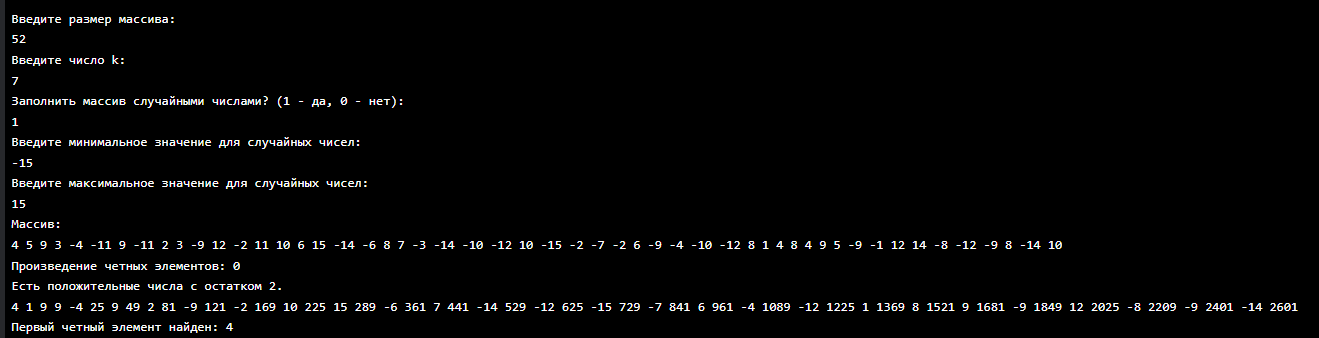


Рисунок 19 - Решение тестового примера задания 4-1 в Pyth