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

Институт транспортной техники и систем управления

Кафедра «Управление и защита информации»

ОТЧЁТ  
ПО ЛАБОРАТОРНОЙ РАБОТЕ №4

По дисциплине «Языки программирования»

ВАРИАНТ 12

Выполнил: ст. гр. ТКИ-141

Комаров Константин Сергеевич

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

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

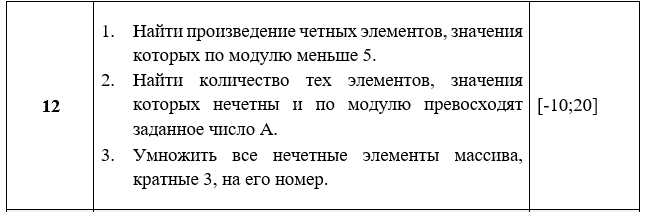
Москва 2024

**РЕШЕНИЕ ЗАДАЧИ 4-1**

* 1. **Формулировка задачи 4-1**

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



* 1. **Блок-схема алгоритма задания 4-1**

Блок-схема алгоритма представлена на рисунках (Рисунки 1-11).

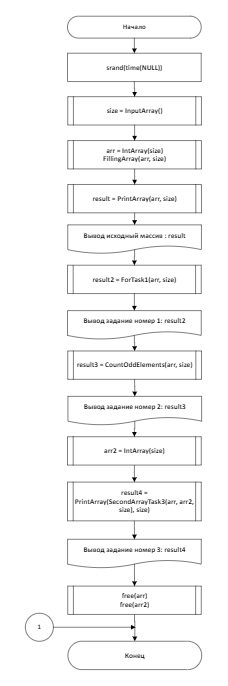


Рисунок 1 – Блок - схема основного алгоритма

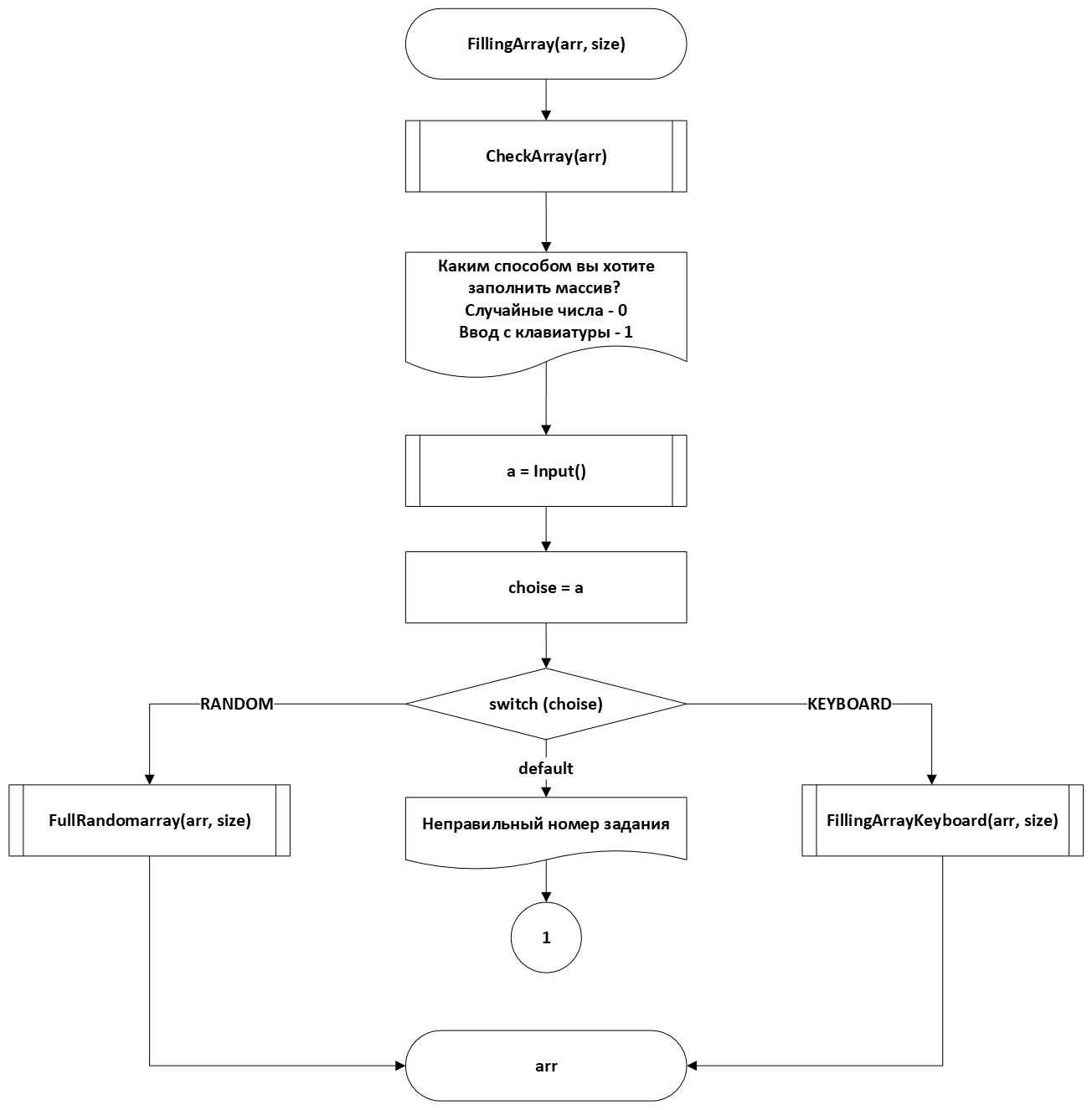


Рисунок 2 – Блок - схема функции ***FillingArray***

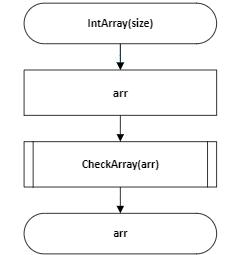
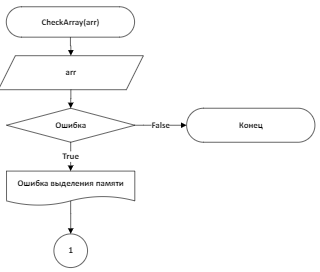
 

Рисунок 3 – Блок-схемы функций ***IntArray*** и ***CheckArray***

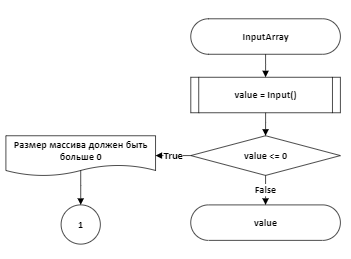
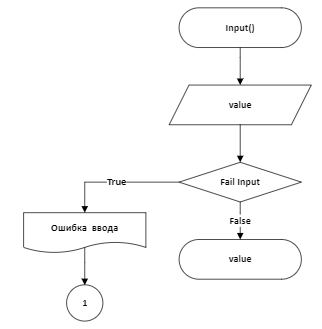


Рисунок 4 – Блок-схемы функций проверки ввода и проверки ввода размера массива

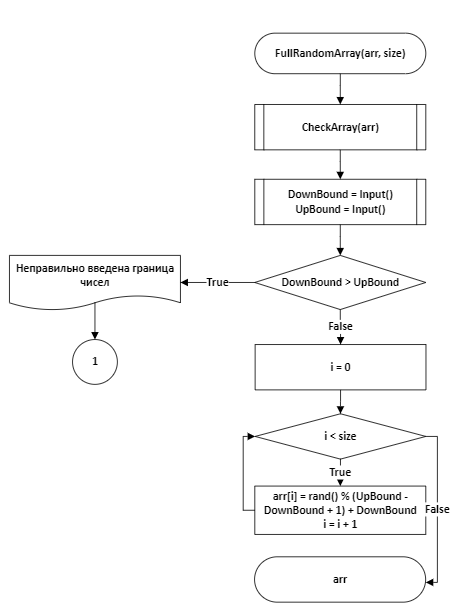


Рисунок 5 – Блок-схема функции ***FullRandomArray***

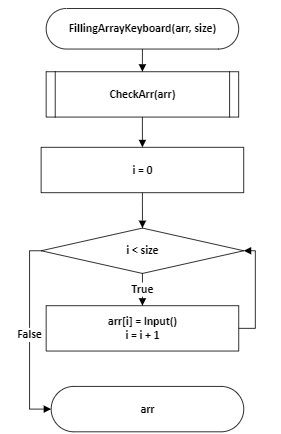


Рисунок 6 – Блок-схема функции ***FillingArrayKeyboard***

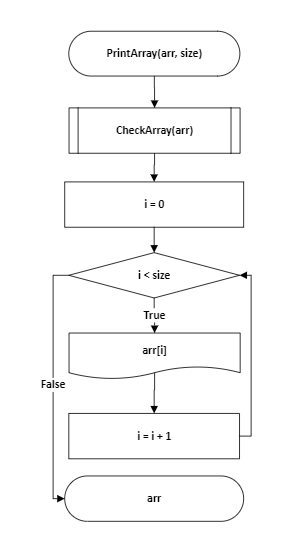


Рисунок 7 – Блок – схемы функций ***PrintArray***

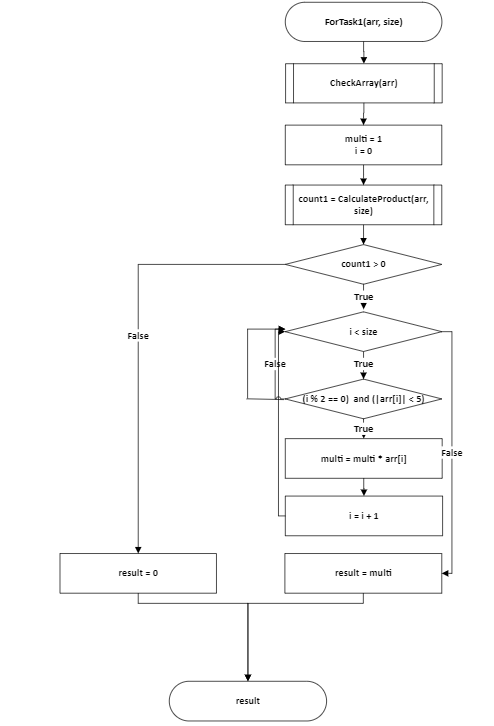


Рисунок 8 – Блок – схема функции ***ForTask1***

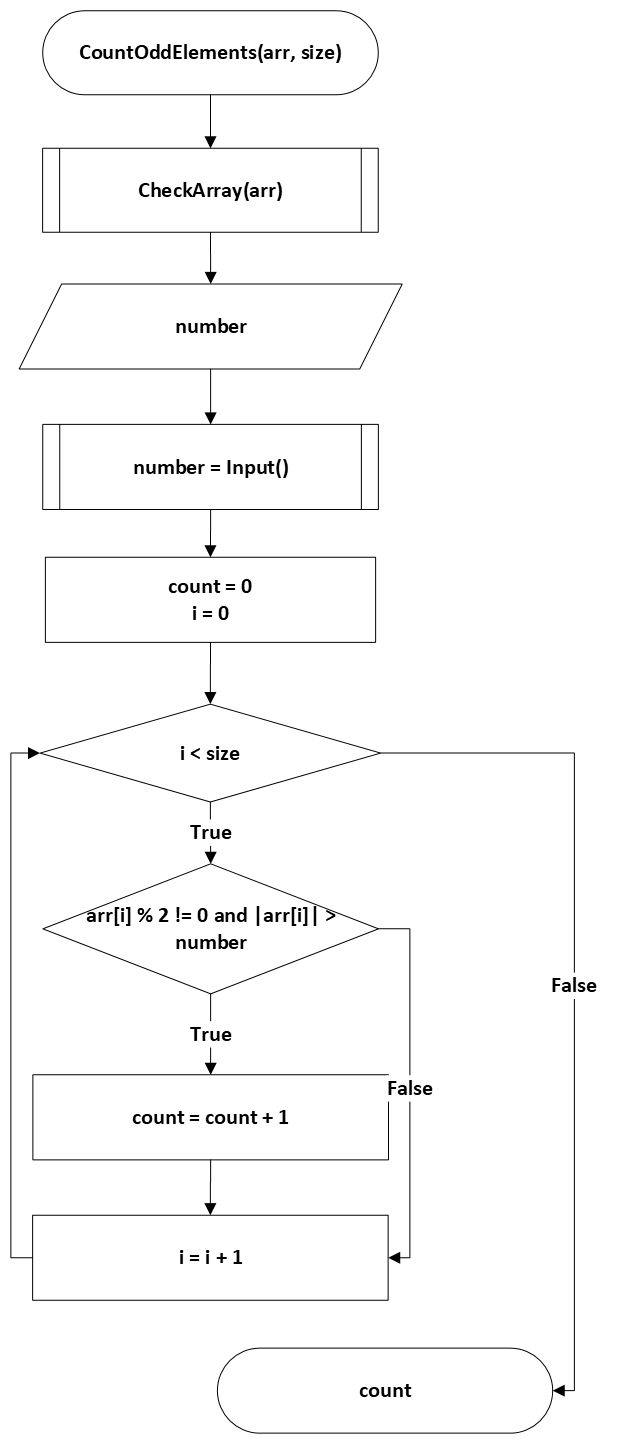
******

Рисунок 9 – Блок – схема функции ***CountOddElements***

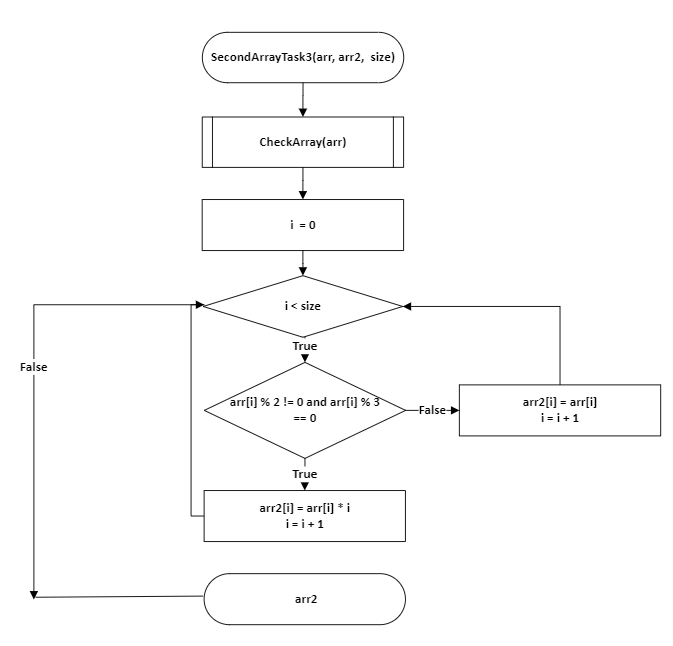


Рисунок 10 – Блок – схема функции ***SecondArrayTask3***

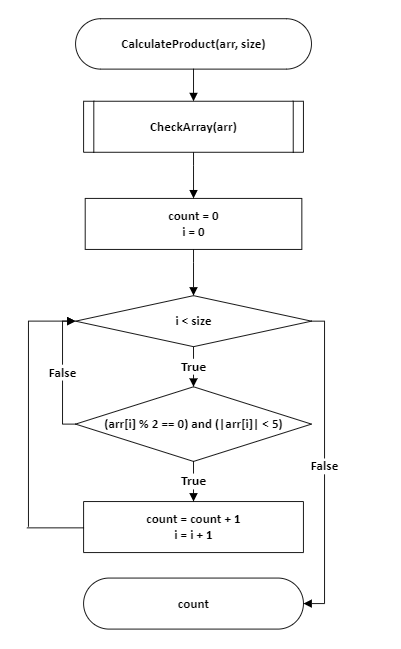


Рисунок 11 – Блок – схема функции ***CalculateProduct***

* 1. **Текст программы на языке программирования С задание 4-1**

#include <stdlib.h>

#include <time.h>

#include <stdio.h>

#include <locale.h>

#include <math.h>

/\*\*

\* @brief Выделяет память под массив целых чисел.

\* @param size Размер массива.

\* @return Указатель на выделенную память под массив.

\*/

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

/\*

\* @brief Функция проверки ввода.

\* @return Возвращает значение, если верное, иначе выдает ошибку.

\*/

int Input(void);

/\*

\* @brief Функция проверки ввода размера массива.

\* @return Возвращает значение размера массива, если верное, иначе выдает ошибку.

\*/

int InputArray(void);

/\*

\* @brief Функция, которая освоюождает память из под массива.

\*@param arr Указатель на массив.

\*/

void DeletedArray(int\* arr);

/\*\*

\* @brief Функция проверки массива.

\* @param arr - массив, который мы проверяем на NULL.

\* @return При непрохождении проверки выдает ошибку.

\*/

void CheckArray(int\* const arr);

/\*\*

\* @brief Заполняет массив случайными или введенными с клавиатуры числами.

\* @param arr Указатель на массив.

\* @param size Размер массива.

\*/

void FillingArray(int\* arr, const size\_t size);

/\*\*

\* @brief Заполняет массив, считывая значения с клавиатуры.

\* @param arr Указатель на массив.

\* @param size Размер массива.

\*/

void FillingArrayKeyboard(int\* arr, const size\_t size);

/\*\*

\* @brief Заполняет массив случайными числами в заданном диапазоне.

\* @param arr Указатель на массив.

\* @param size Размер массива.

\*/

void FullRandomArray(int\* arr, const size\_t size);

/\*\*

\* @brief Выводит массив на экран.

\* @param arr Указатель на массив.

\* @param size Размер массива.

\*/

void PrintArray(const int\* arr, const size\_t size);

/\*\*

\* @brief Считает колиичество четных элементов, значения модуля которых меньше пяти.

\* @param arr Указатель на массив.

\* @param size Размер массива.

\* @return Количество четных элементов, значения модуля которых меньше пяти.

\*/

int CalculateProduct(const int\* arr, const size\_t size);

/\*\*

\* @brief Выполняет задание 1: перемножает четные элементы, значение модуля которых меньше пяти.

\* @param arr Указатель на массив.

\* @param size Размер массива.

\* @return Произведение четных элементов, значение модуля которых меньше пяти, если таких элементов нет возвращает ноль.

\*/

int ForTask1(const int\* arr, const size\_t size);

/\*\*

\* @brief Выполняет задание 2: находит количество нечетных элементов, значение модуля которых больше заданного числа.

\* @param arr Указатель на массив.

\* @param size Размер массива.

\* @return Количество нечетных элементов, значение модуля которых больше заданного числа,если таких элементов нет возвращает ноль.

\*/

int CountOddElements(const int\* arr, const size\_t size);

/\*\*

\* @brief Выполняет задание 3: умножает нечетные и кратные трем элементы на их номер.

\* @param arr Указатель на массив.

\* @param arr2 Указатель на массив, в который будут записаны результаты умножения.

\* @param size Размер массива.

\* @return Новый массив после выполнения задания 3.

\*/

int\* SecondArrayTask3(const int\* arr, int\* arr2, const size\_t size);

/\*\*

\* @brief Перечисление для выбора способа заполнения массива.

\*/

enum TASK {

RANDOM,

KEYBOARD

};

/\*

\* @brief Точка входа в программу.

\* @return Возвращает значение функции с заданным в цикле корнем.

\*/

int main(void)

{

srand(time(NULL));

setlocale(LC\_ALL, "Russian");

puts("Введите количество элементов в массиве: ");

size\_t size = (size\_t)InputArray();

int\* arr = IntArray(size);

FillingArray(arr, size);

printf("\nИсходный массив: ");

PrintArray(arr, size);

printf("\nПроизведение четных элементов, значение модуля которых меньше пяти: %d\n", ForTask1(arr, size));

printf("\nКоличество нечетных элементов, значения модуля которых больше заданного числа: %d\n", CountOddElements(arr, size));

printf("\nМассив после умножения нечетных элементов кратных трем на их номер: ");

int\* arr2 = IntArray(size);

PrintArray(SecondArrayTask3(arr, arr2, size), size);

free(arr);

free(arr2);

return 0;

}

int Input(void)

{

int input = 0;

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

{

puts("Ошибка ввода\a");

exit(EXIT\_FAILURE);

}

return input;

}

int InputArray(void)

{

int value = Input();

if (value <= 0)

{

puts("Размер массива должен быть больше нуля.\a");

exit(EXIT\_FAILURE);

}

return value;

}

void CheckArray(int\* const arr)

{

if (arr == NULL)

{

printf("Ошибка выделения памяти");

exit(EXIT\_FAILURE);

}

}

int\* IntArray(const size\_t size)

{

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

CheckArray(arr);

return arr;

}

void FullRandomArray(int\* arr, const size\_t size)

{

CheckArray(arr);

puts("Введите минимальную границу случайных чисел: ");

const int DownBound = Input();

puts("Введите максимальную границу случайных чисел: ");

const int UpBound = Input();

if (DownBound > UpBound)

{

puts("Неправильно введена граница чисел!\a");

exit(EXIT\_FAILURE);

}

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

{

arr[i] = rand() % (UpBound - DownBound + 1) + DownBound;

}

}

void FillingArrayKeyboard(int\* arr, const size\_t size)

{

CheckArray(arr);

printf("\nВведите %zu целых чисел для заполнения массива: ", size);

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

{

arr[i] = Input();

}

}

void FillingArray(int\* arr, const size\_t size)

{

CheckArray(arr);

printf("\nКаким способом вы хотите заполнить массив?\n"

"Случайные числа - %d\n"

"Ввод с клавиатуры - %d\n", RANDOM, KEYBOARD);

int a = Input();

enum TASK choice = (enum TASK)a;

switch (choice) {

case RANDOM:

FullRandomArray(arr, size);

break;

case KEYBOARD:

FillingArrayKeyboard(arr, size);

break;

default:

printf("Неправильный номер задания!\a");

exit(EXIT\_FAILURE);

}

}

void PrintArray(const int\* arr, const size\_t size)

{

CheckArray(arr);

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

{

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

}

printf("\n");

}

int CalculateProduct(const int\* arr, const size\_t size)

{

int count = 0;

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

{

if ((arr[i] % 2 == 0) && (abs(arr[i]) < 5))

{

count++;

}

}

return count;

}

int ForTask1(const int\* arr, const size\_t size)

{

int mylti = 1;

int count1 = CalculateProduct(arr, size);

if (count1 > 0)

{

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

{

if ((arr[i] % 2 == 0) && (abs(arr[i]) < 5))

{

mylti \*= arr[i];

}

}

return mylti;

}

else

{

return 0;

}

}

int CountOddElements(const int\* arr, const size\_t size)

{

CheckArray(arr);

puts("Ввидите любое целое число: ");

int number = Input();

int count = 0;

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

{

if (arr[i] % 2 != 0 && abs(arr[i]) > number)

{

count++;

}

}

return count;

}

int\* SecondArrayTask3(const int\* arr, int\* arr2, const size\_t size)

{

CheckArray(arr);

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

{

if (arr[i] % 2 != 0 && arr[i] % 3 == 0)

{

arr2[i] = arr[i] \* i;

}

else

{

arr2[i] = arr[i];

}

}

return arr2;

}

* 1. **Результаты выполнения программы**

Результаты выполнения программы представленны ниже (Рисунки 12-17)

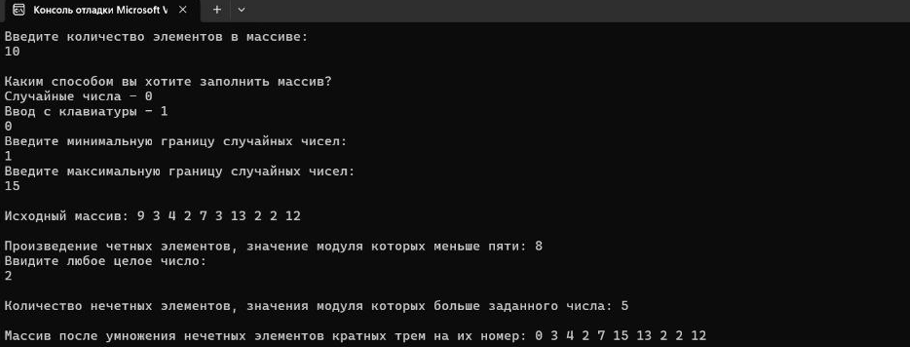
****

Рисунок 12 – Результаты выполнения программы случайным вводом

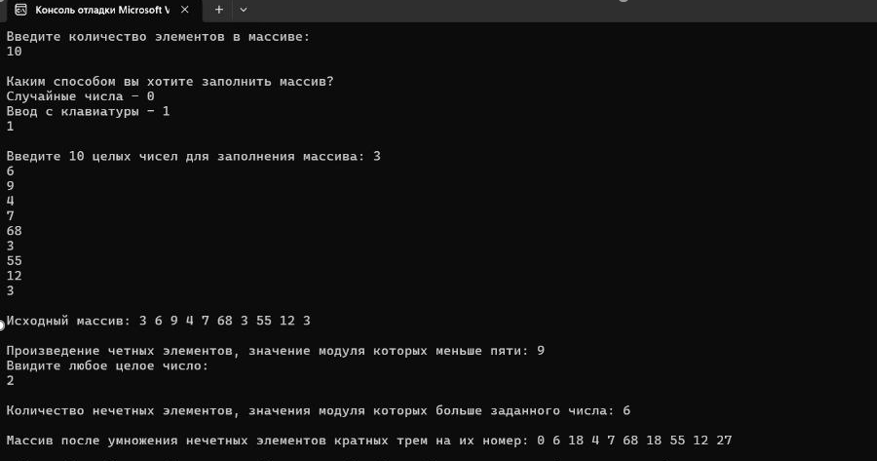


Рисунок 13 – Результаты выполнения программы ручным вводом

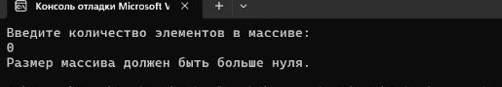


Рисунок 14 – Вывод ошибки ввода размера массива

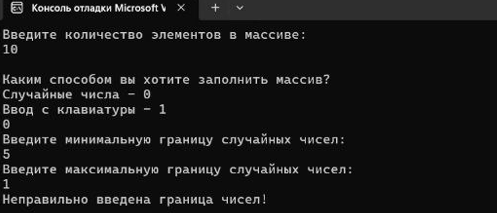


Рисунок 15 – Вывод ошибки ввода границы случайных чисел

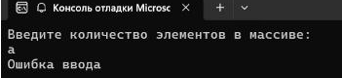


Рисунок 16 – Вывод ошибки ввода

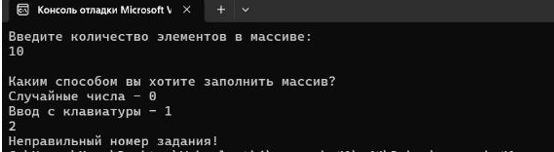


Рисунок 17 – Вывод ошибки выбора способа заполнения

* 1. **Выполнение тестовых примеров задание 4-1**

В программе Microsoft Visual Studio выполнены тестовые примеры. Результаты их выполнения представленны ниже (Рисунки 18-19)

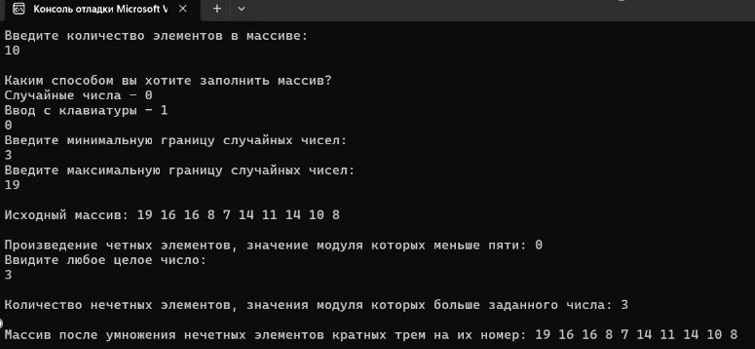


Рисунок 18 – Результат выполнения тестового примера программы случайным вводом

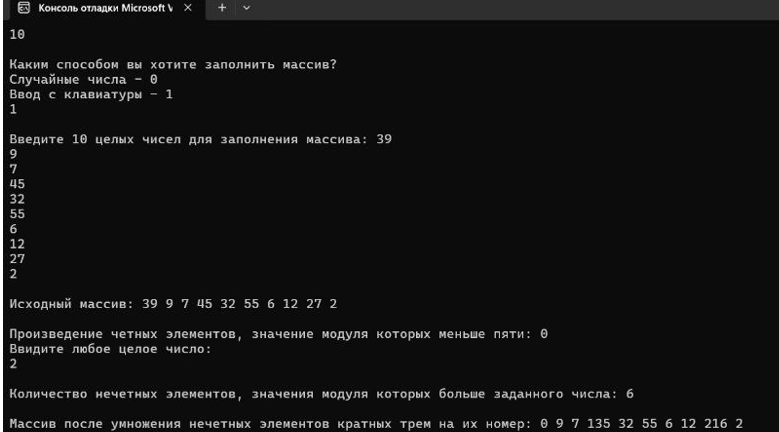


Рисунок 19 – Результат выполнения тестовго примера программы ручным вводом

* 1. **Отметка о выполнении задания в веб-хостинге системы контроля версий задание 4-1**

Ниже представлено доказательство того, что задание 4.1 было принято (Рисунок 20)

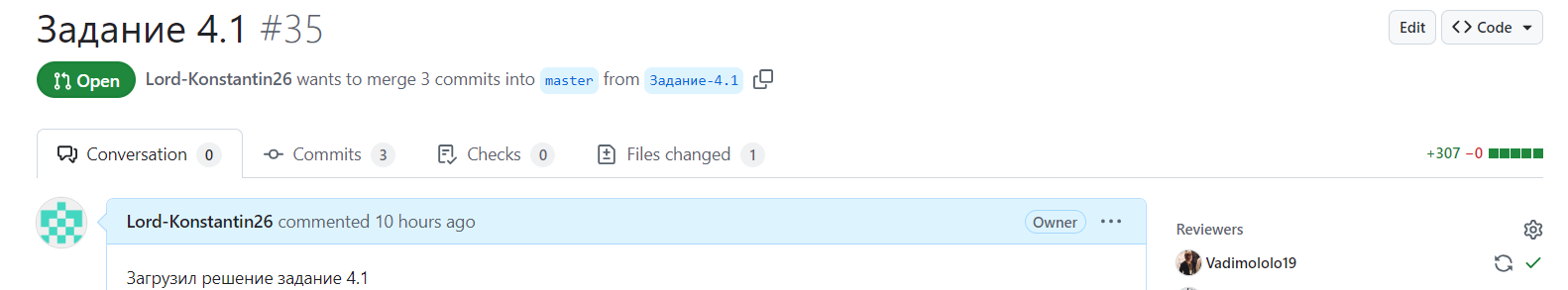


Рисунок 20 – Approve задачи 4.1