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 №4.2

По дисциплине «Языки программирования»

ВАРИАНТ 3

Выполнил: ст. гр. ТКИ-142

Величко Иван Сергеевич

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

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

Москва 2024

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

Таблица 1 – Исходные данные

|  |  |  |
| --- | --- | --- |
| **Вариант** | **Задачи** | **Интервал** |
| 3 | 1. Заменить последний отрицательный элемент массива предпоследним элементом массива. 2. Удалить из него все элементы, первая цифра которых четная.  3. Из элементов массива D сформировать массив A той же размерности по правилу: если номер четный, то значение элемента находится по формуле Ai = Di2, а если нечетный, то по формуле Ai = 2Di. | [-10;20] |

1. Блок-схема алгоритма

Ниже представлены:

Блок-схема основного алгоритма (Рисунок 1).

Блок-схема основного алгоритма (Рисунок 1.2).

Блок-схема функции input() (Рисунок 2).

Блок-схема функции input\_positive() (Рисунок 3).

Блок-схема функции check\_array() (Рисунок 4).

Блок-схема функции random\_input() (Рисунок5).

Блок-схема функции input\_positive() (Рисунок 6).

Блок-схема функции print\_array() (Рисунок 7).

Блок-схема функции keyboard\_input() (Рисунок 8).

Блок-схема функции change() (Рисунок 9).

Блок-схема функции swap() (Рисунок 10).

Блок-схема функции delete\_elements() (Рисунок 11).

Блок-схема функции count\_remove\_elements() (Рисунок 12).

Блок-схема функции first\_digit() (Рисунок 13).

Блок-схема функции array\_filling() (Рисунок 14).

Блок-схема функции create\_array() (Рисунок 1).

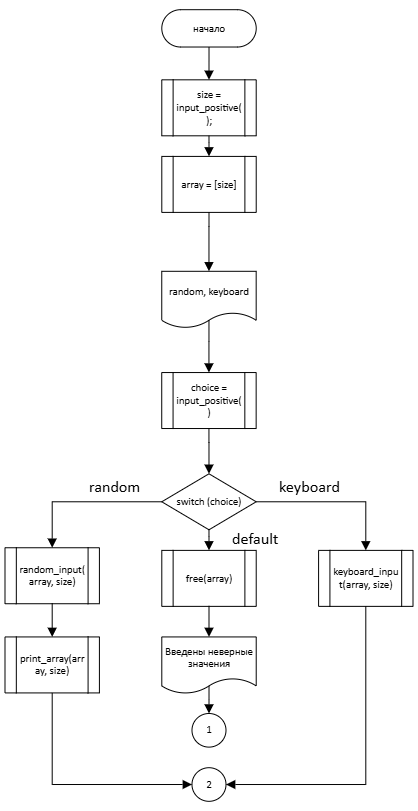


Рисунок 1 – Блок-схема основного алгоритма



Рисунок 1.1 – Блок-схема основного алгоритма



Рисунок 2 – Блок-схема функции input()

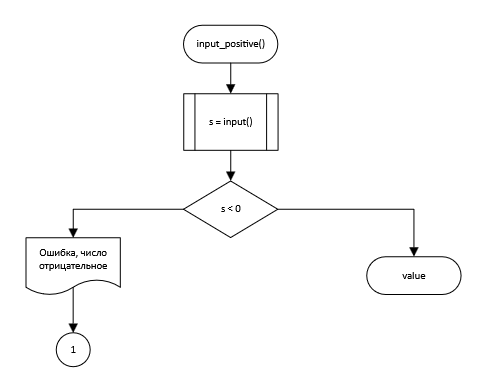


Рисунок 3 – Блок-схема функции input\_positive()

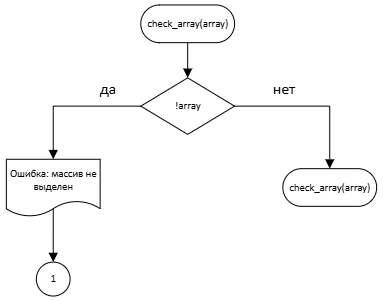


Рисунок 4 – Блок-схема функции check\_array()

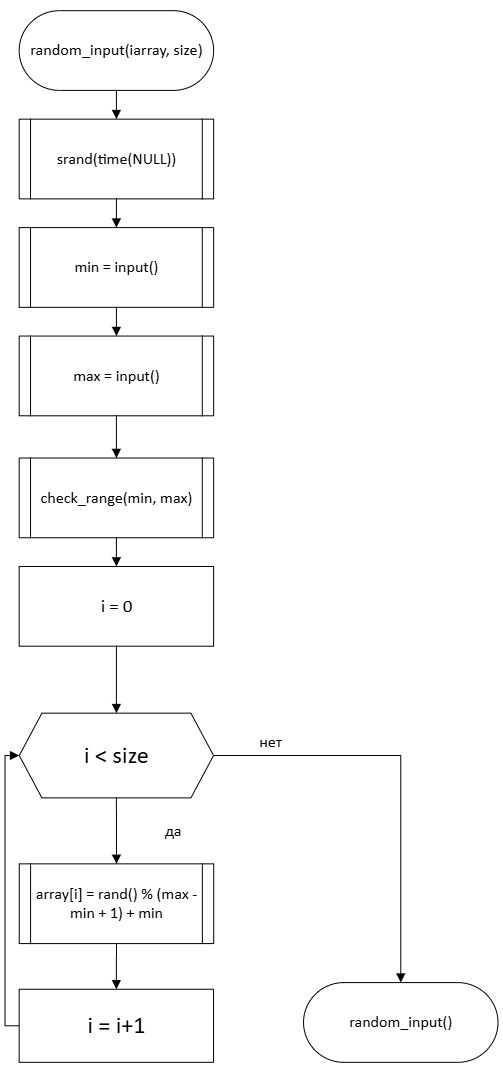


Рисунок 5 – Блок-схема функции random\_input()

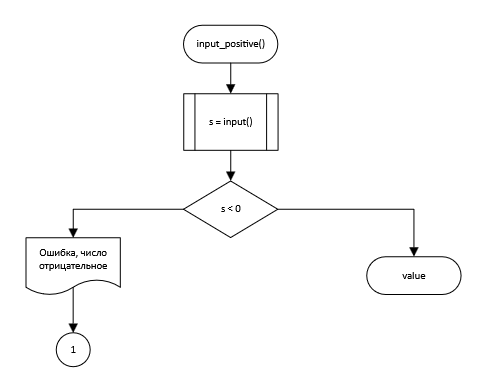


Рисунок 6 – Блок-схема функции input\_positive()

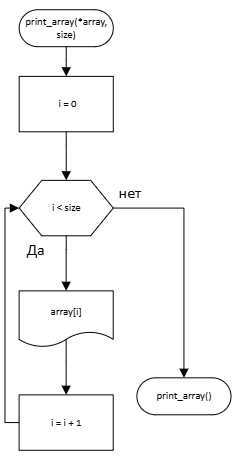


Рисунок 7 – Блок-схема функции print\_array()

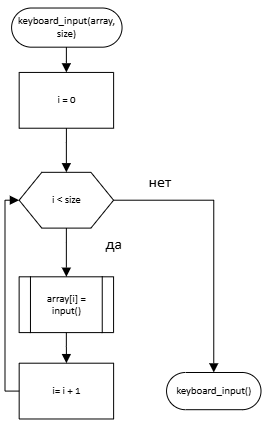


Рисунок 8 – Блок-схема функции keyboard\_input()

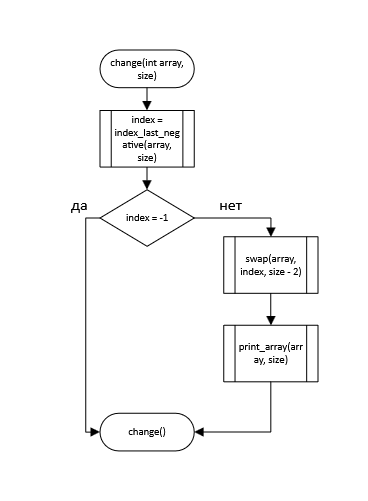


Рисунок 9 – Блок-схема функции change()

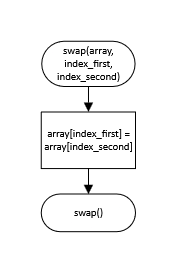


Рисунок 10 – Блок-схема функции swap()

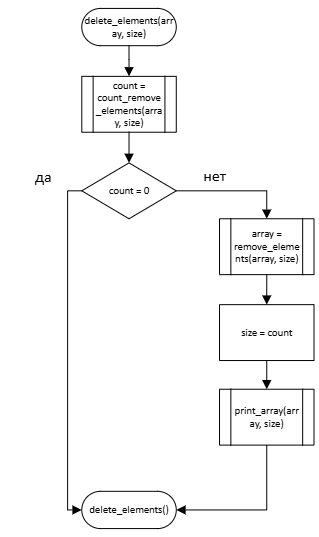


Рисунок 11 – Блок-схема функции delete\_elements()

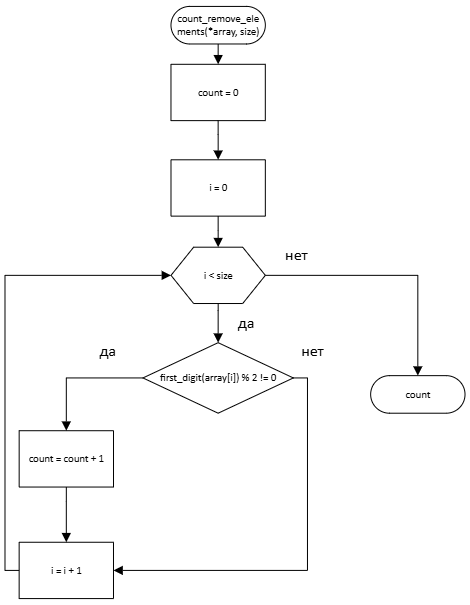


Рисунок 12 – Блок-схема функции count\_remove\_elements()

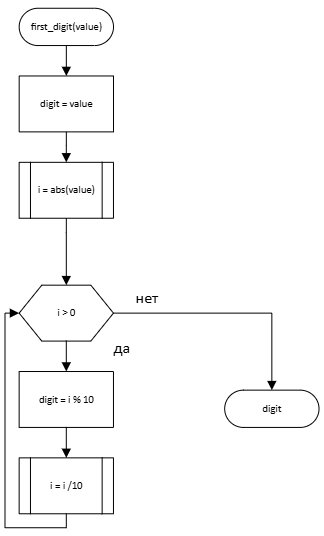


Рисунок 13 – Блок-схема функции first\_digit()

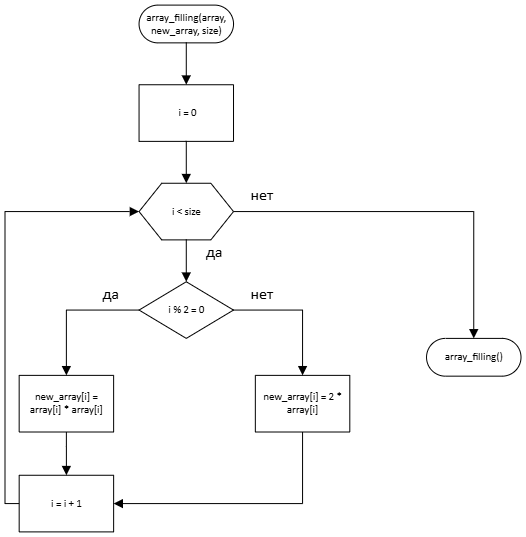


Рисунок 14 – Блок-схема функции create\_array()

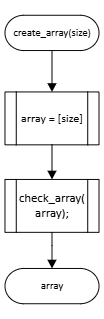


Рисунок 15 – Блок-схема функции create\_array(size)

1. Текст программы на языке C

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

#include <errno.h>

enum method {

random,

keyboard

};

void change(int\* array, size\_t size);

void delete\_elements(int\* array, size\_t size);

/\*\*

\* @brief принимает из stdin, а затем возвращает значение переменной value

\* @return числовое значение полученное из stdin

\*/

double input(void);

/\*\*

\* @brief принимает из stdin, а затем возвращает значение переменной value, проверяет является ли число положительным

\* @return положительное числовое значение

\*/

size\_t input\_positive(void);

/\*\*

\* @brief заполняет массив случайно сгенерироваными числами

\* @param array указатель на массив

\* @param size размер массива

\*/

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

/\*\*

\* @brief заполняет массив из стандартного потока ввода

\* @param array указатель на массив

\* @param size размер массива

\*/

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

/\*\*

\* @brief выводит содержимое массива

\* @param array указатель на массив

\* @param size размер массива

\*/

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

/\*\*

\* @brief проверяет правильность интервала

\* @param min минимально возможное значение элемента

\* @param max максимально возможное значение элемента

\*/

void check\_range(const int min, const int max);

/\*\*

\* @brief проверяет выделение памяти

\* @param array указатель на массив

\*/

void check\_array(int\* const array);

/\*\*

\* @brief возвращает индекс последнего отрицательного элемента

\* @param array указатель на массив

\* @param size размер массива

\*/

int index\_last\_negative(int\* const array, const size\_t size);

/\*\*

\* @brief меняет два элемента местами в массиве

\* @param array указатель на массив

\* @param index\_first номер первого элемента

\* \* @param index\_second номер второго элемента

\*/

void swap(int\* array, const int index\_first, const int index\_second);

/\*\*

\* @brief возвращает первую цифру числа

\* @param value число

\*/

int first\_digit(const int value);

/\*\*

\* @brief возвращает количество четных элементов(которые нужно удалить)

\* @param array указатель на массив

\* @param size размер массива

\*/

int count\_remove\_elements(int\* array, const size\_t size);

/\*\*

\* @brief копирует элементы массива array в массив new\_array

\* @param array указатель на массив

\* @param new\_array указатель на массив

\* @param size размер массива

\*/

void copy\_array(int\* new\_array, int const\* array, const size\_t size);

/\*\*

\* @brief возвращает новый массив с без удаленных элементов

\* @param array указатель на массив

\* @param size размер массива

\*/

int\* remove\_elements(int\* array, const size\_t size);

/\*\*

\* @brief заполняет массив new\_array значениями из array

\* @param array указатель на массив

\* @param new\_array указатель на массив

\* @param size размер массива

\*/

void array\_filling(int const\* array, int\* new\_array, const size\_t size);

/\*\*

\* @brief создает массив

\* @param size размер создаваемого массива

\* @return указатель на созданный массив

\*/

int\* create\_array(const size\_t size);

int main() {

puts("Введите размер массива:");

size\_t size = input\_positive();

int\* array = create\_array(size);

printf("Выберите способ заполнения массива:\n%d - случайными числами\n%d - вводом с клавиатуры\n", (int)random, (int)keyboard);

enum method choice = (enum method)input\_positive();

switch (choice) {

case random:

random\_input(array, size);

puts("Массив: ");

print\_array(array, size);

break;

case keyboard:

keyboard\_input(array, size);

break;

default:

free(array);

errno = EIO;

perror("Введены неверные значения\n");

exit(1);

}

change(array, size);

delete\_elements(array, size);

int\* new\_array = create\_array(size);

array\_filling(array, new\_array, size);

puts("\nПолученный массив A: \n");

print\_array(new\_array, size);

free(array);

free(new\_array);

return 0;

}

void change(int\* array, size\_t size) {

int index = index\_last\_negative(array, size);

if (index == -1) {

puts("Невозможно заменить последний отрицательный элемент массива предпоследним элементом массива. В массиве нет отрицательных элентов\n");

}

else {

swap(array, index, size - 2);

puts("\nМассив после замены: \n");

print\_array(array, size);

}

}

void delete\_elements(int\* array, size\_t size) {

int count = count\_remove\_elements(array, size);

if (count == 0) {

puts("\nНет улементов для удаления: \n");

}

else {

array = remove\_elements(array, size);

size = count;

puts("\nМассив после удаления элементов,первая цифра которых четная: \n");

print\_array(array, size);

}

}

double input(void) {

double value = 0.0;

int s = scanf\_s("%lf", &value);

if (s != 1) {

errno = EIO;

perror("Ошибка, не числовое значение\n");

exit(1);

}

return value;

}

size\_t input\_positive() {

int s = input();

if (s < 0) {

errno = EIO;

perror("Ошибка, число отрицательное\n");

exit(EXIT\_FAILURE);

}

return (size\_t)s;

}

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

srand(time(NULL));

puts("Введите минимальное значение элемента массива:");

int min = input();

puts("Введите максимальное значение элемента массива:");

int max = input();

check\_range(min, max);

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

array[i] = rand() % (max - min + 1) + min;

}

}

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

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

array[i] = input();

}

}

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

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

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

}

}

void check\_range(const int min, const int max) {

if (min > max) {

errno = EIO;

perror("Не верно указан интервал\n");

exit(1);

}

}

void check\_array(int\* const array) {

if (!array) {

perror("Ошибка: массив не выделен");

exit(EXIT\_FAILURE);

}

}

int index\_last\_negative(int\* const array, const size\_t size) {

int index = -1;

for (int i = (int)size - 1; i >= 0; i--) {

if (array[i] < 0) {

index = i;

break;

}

}

return index;

}

void swap(int\* array, const int index\_first, const int index\_second) {

array[index\_first] = array[index\_second];

}

int first\_digit(const int value) {

int digit = value;

for (size\_t i = abs(value); i > 0; i /= 10) {

digit = i % 10;

}

return digit;

}

int count\_remove\_elements(int\* array, const size\_t size) {

int count = 0;

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

if (first\_digit(array[i]) % 2 != 0) {

count++;

}

}

return count;

}

void copy\_array(int\* new\_array, int const\* array, const size\_t size) {

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

new\_array[i] = array[i];

}

}

int\* remove\_elements(int\* array, const size\_t size) {

int current\_index = 0;

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

if (first\_digit(array[i]) % 2 != 0) {

array[current\_index] = array[i];

current\_index++;

}

}

int\* array = create\_array(size);

copy\_array(new\_array, array, current\_index);

return new\_array;

}

void array\_filling(int const\* array, int\* new\_array, const size\_t size) {

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

if (i % 2 == 0) {

new\_array[i] = array[i] \* array[i];

}

else {

new\_array[i] = 2 \* array[i];

}

}

}

int\* create\_array(const size\_t size) {

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

check\_array(array);

return array;

}

1. Результаты выполнения программы

Результаты выполнения программы представлены ниже (Рисунок 6.1, Рисунок 6.2).

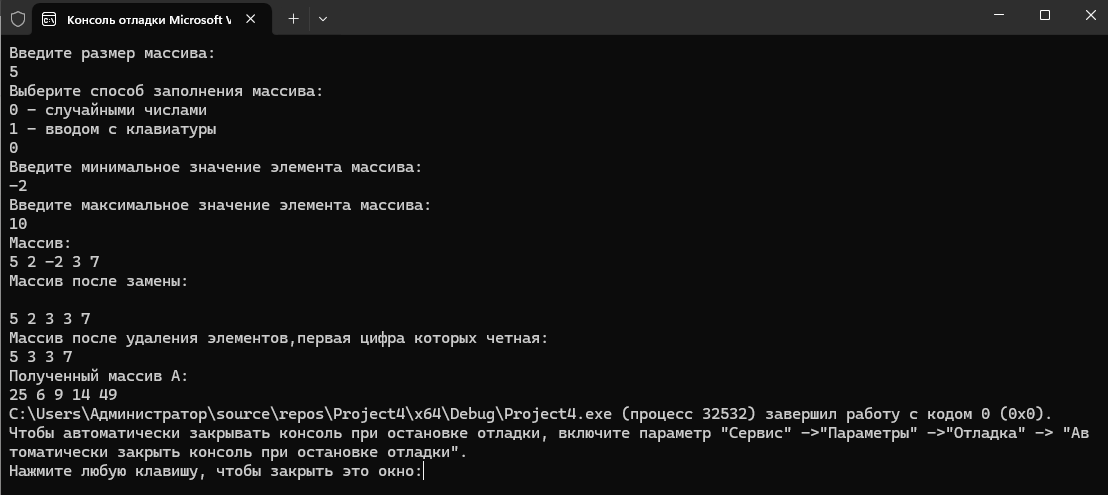


Рисунок 16 – Результат выполнения программы при выборе заполнения массива случайными числами

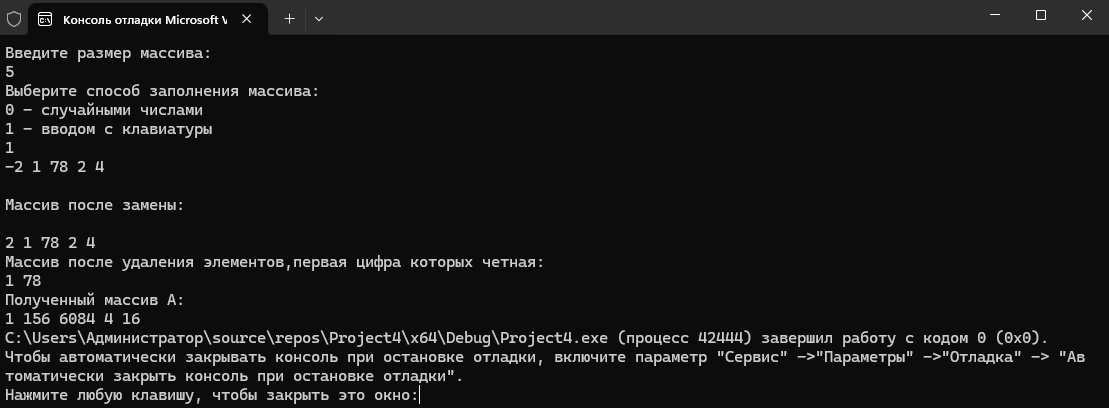


Рисунок 17 – Результат выполнения программы при выборе заполнения массива из stdin

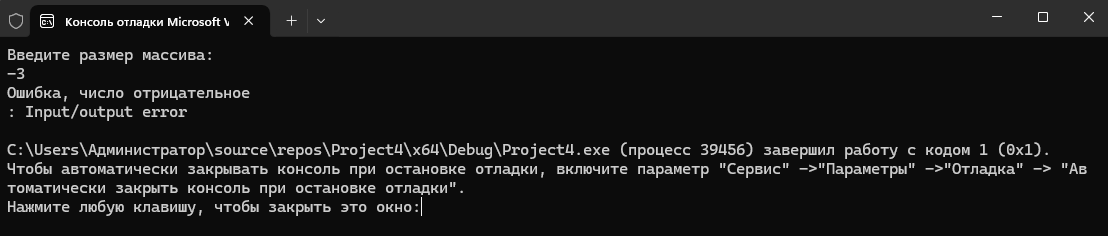


Рисунок 18 – Ввод отрицательного значения при выборе размера массива

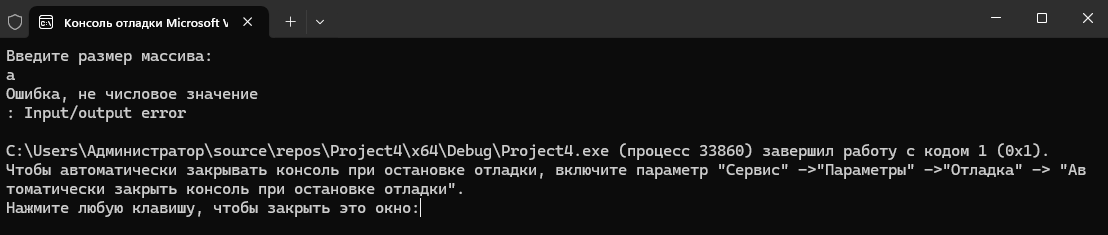


Рисунок 19 – Ввод символьного значения при выборе размера массива

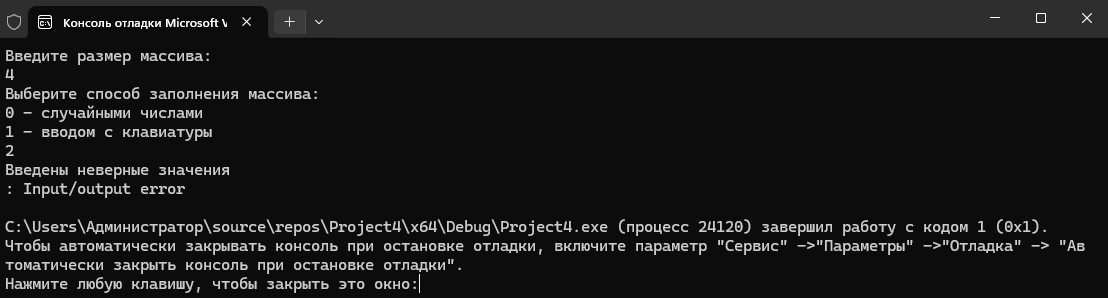


Рисунок 20 – Ввод отличного от заданных значение при выборе метода заполнения массива

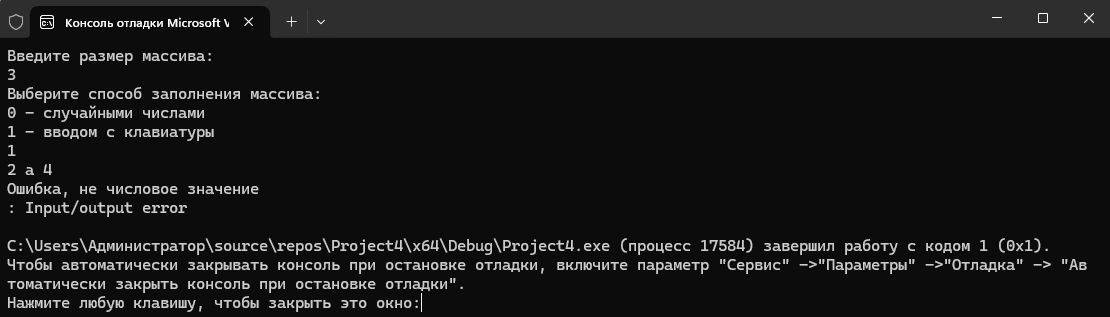


Рисунок 21 – Ввод символьного значения в качестве элементов массива

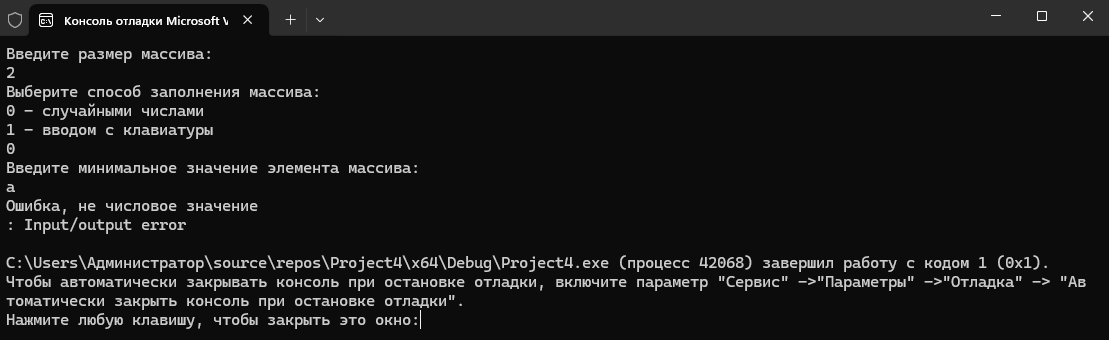


Рисунок 22 – Ввод символьного значения в качестве минимального значения

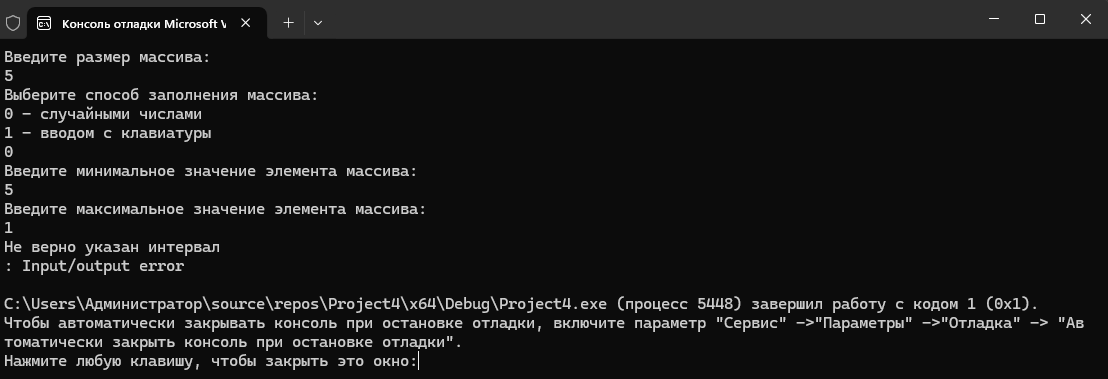
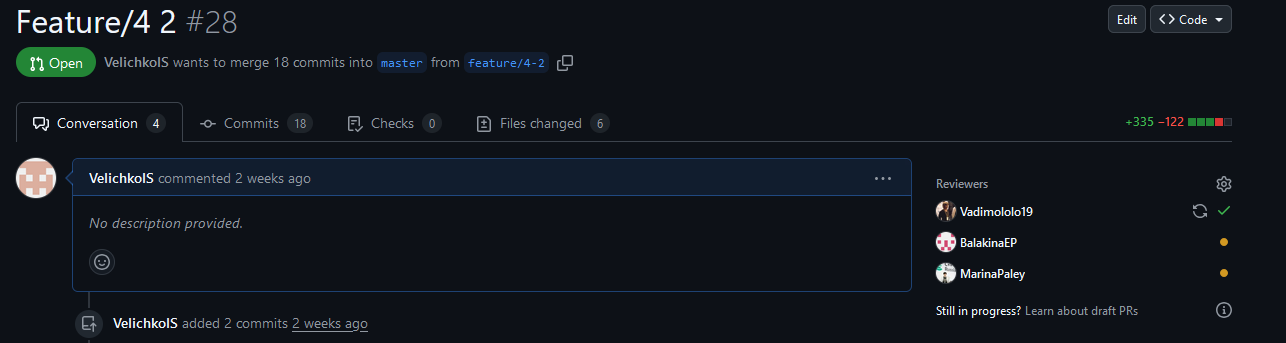


Рисунок 23 – Ввод неправильного интервала

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



Отметка о выполнение задания представлена ниже (Рисунок 22)