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 №4-1

По дисциплине «Языки программирования»

ВАРИАНТ 1

Выполнил: ст. гр. ТКИ - 141

Абузина Елизавета Юрьевна

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

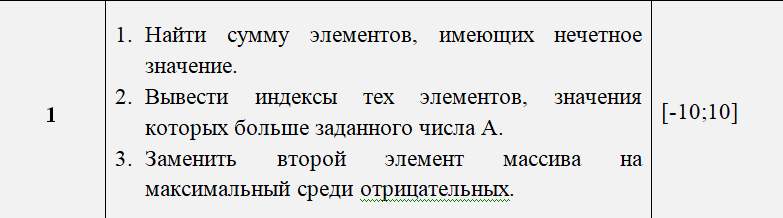
(Проверил: к.т.н, доц. Балакина Е. П.)

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). Составить блок-схему.

Таблица  – Исходные данные



1. Блок-схема алгоритма

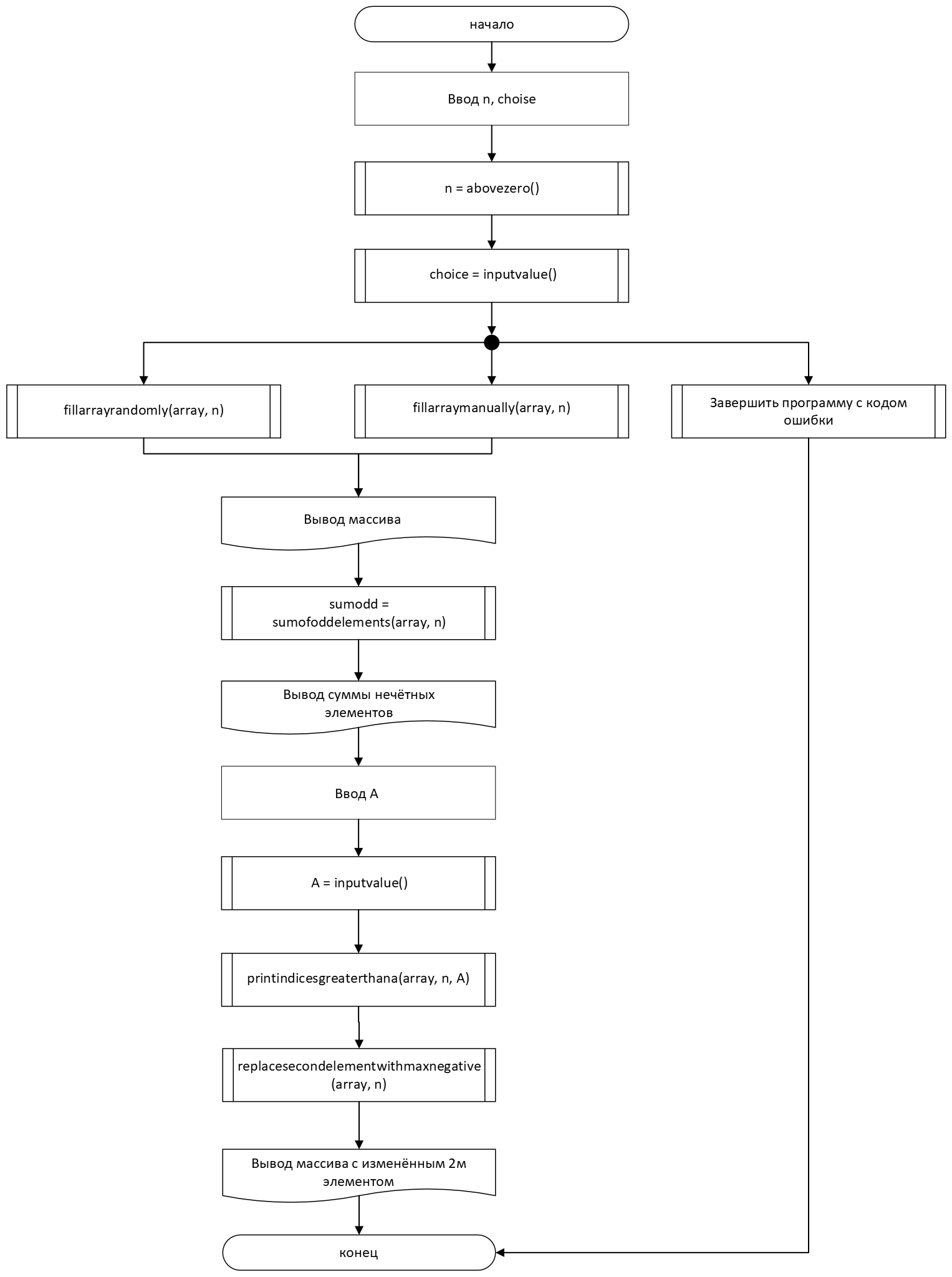


Рисунок  ­ Блок-схема основного алгоритма

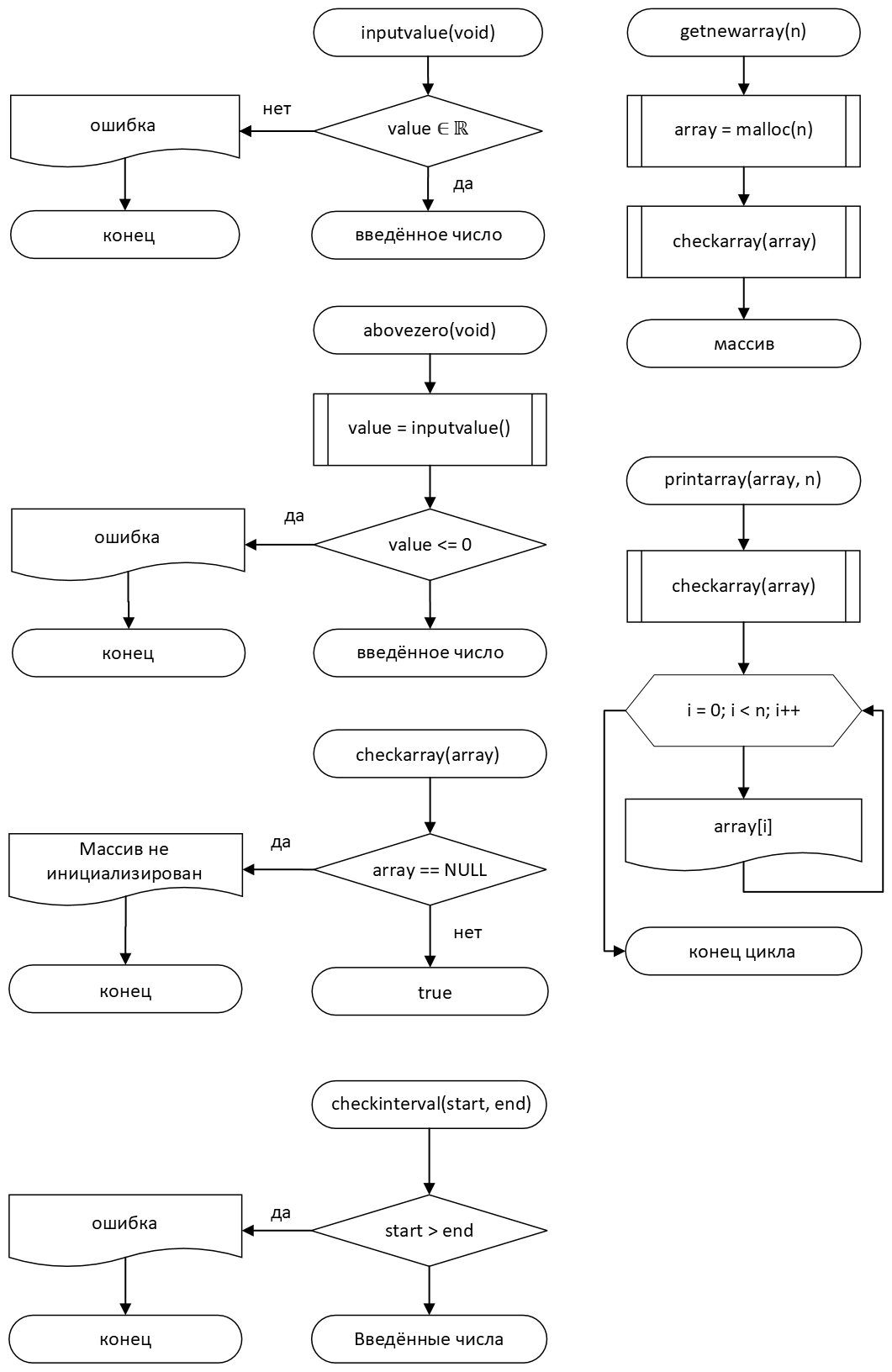


Рисунок  – Блок-схема используемых функций

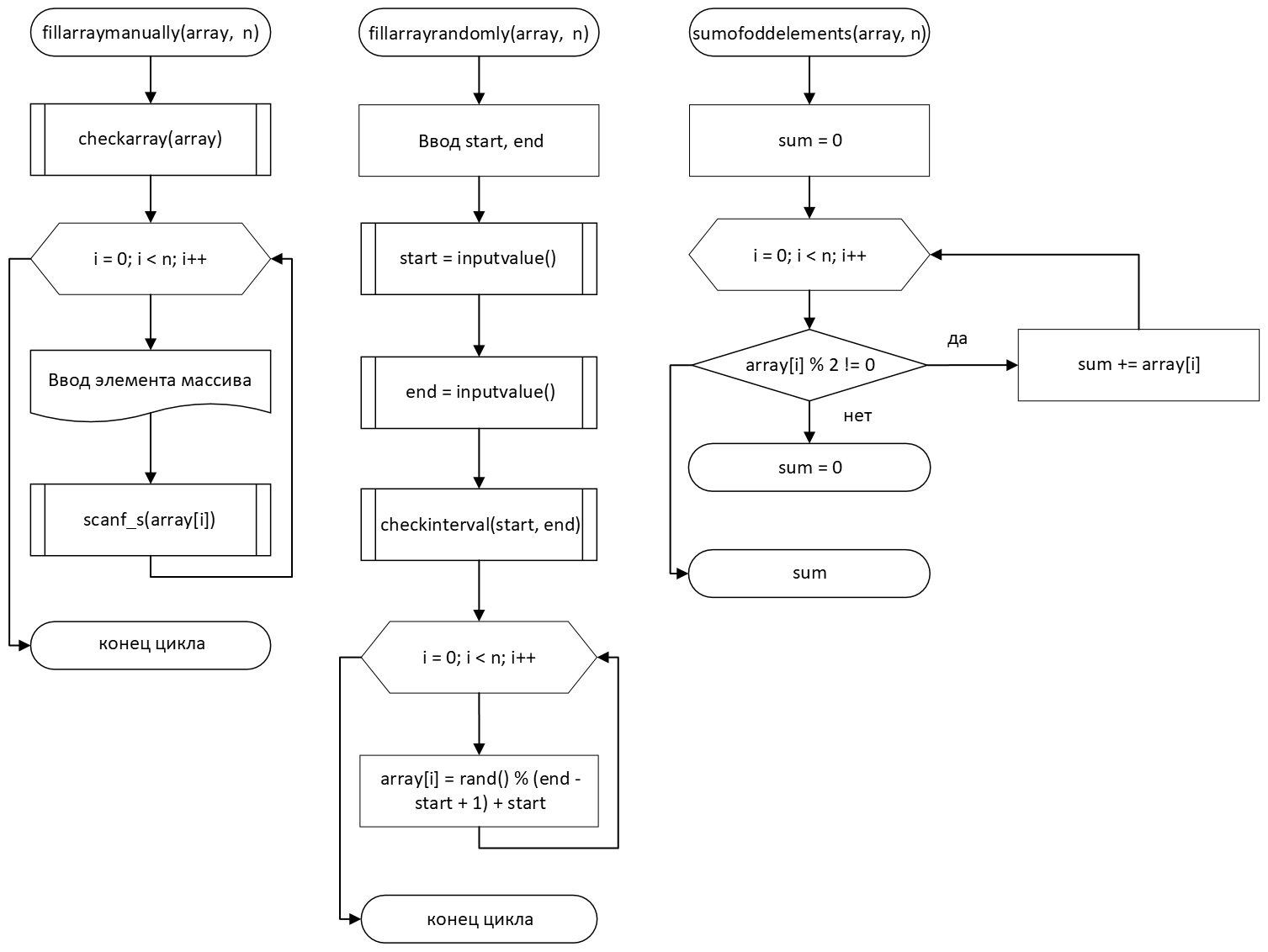


Рисунок 3 – Блок-схема используемых функций

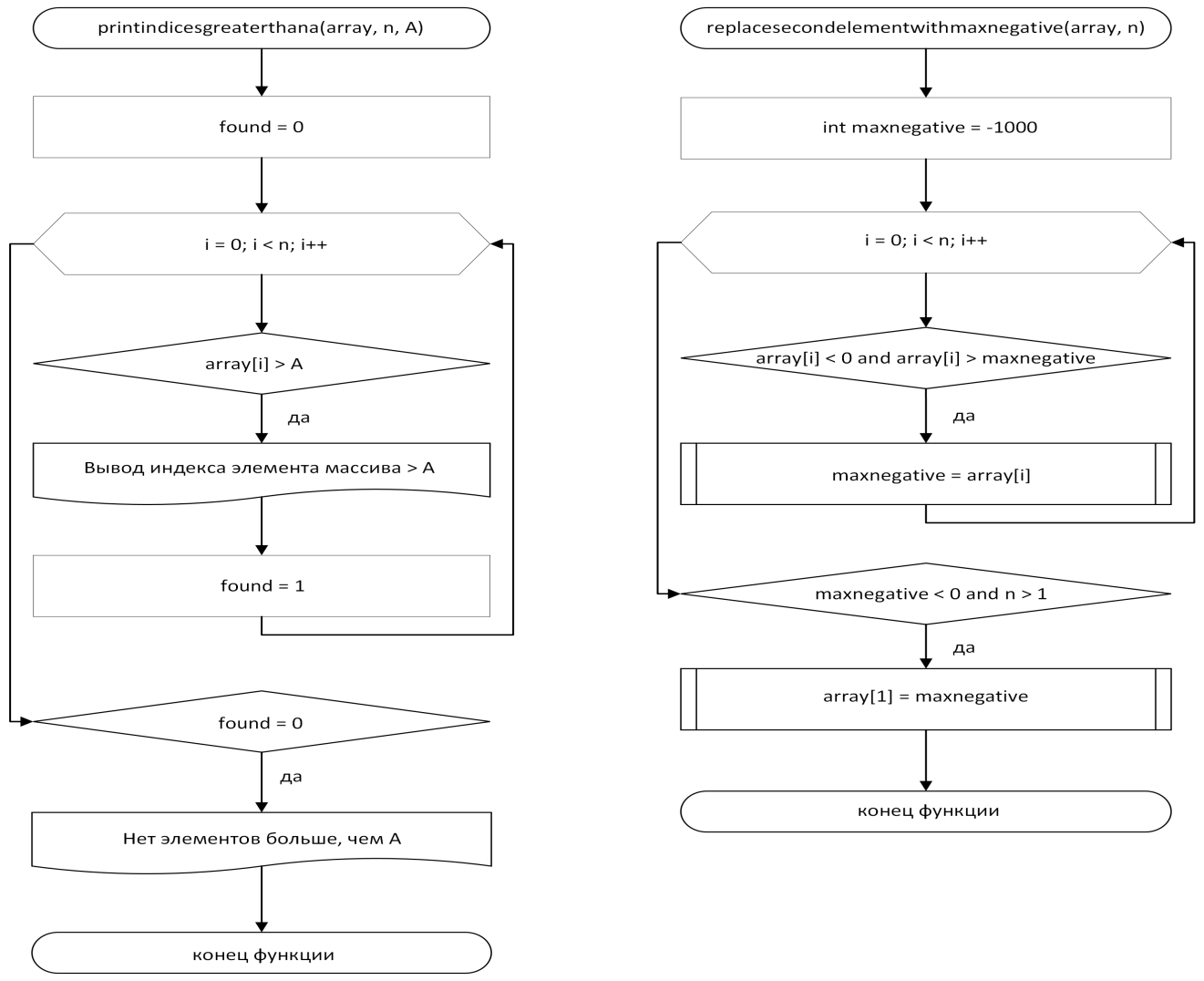


Рисунок 4 – Блок-схема используемых функций

1. Текст программы на языке C

#include <stdio.h>

#include <locale.h>

#include <math.h>

#include <stdlib.h>

#include <errno.h>

#include <time.h>

#include <stdbool.h>

/\*\*

\* @brief Вычисляет сумму нечетных элементов массива

\* @param array - указатель на массив целых чисел

\* @param n - размер массива

\* @return Сумма нечетных элементов массива

\*/

int sumofoddelements(int\* array, const size\_t n);

/\*\*

\* @brief Печатает индексы элементов массива, которые больше заданного значения A

\* @param array - указатель на массив целых чисел

\* @param n - размер массива

\* @param A - значение, по которому производится сравнение

\*/

void printindicesgreaterthana(int\* array, const size\_t n, int A);

/\*\*

\* @brief Заменяет второй элемент массива максимальным отрицательным элементом

\* @param array - указатель на массив целых чисел

\* @param n - размер массива

\*/

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

/\*\*

\* @brief Массив

\* @param array - массив

\* @param n - размер массива

\* @return Выводит пользователю массив

\*/

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

/\*\*

\* @brief Заполнение массива n - количеством рандомных элементов в интервале.

\* @param array - массив.

\* @param n - размер массива.

\* @return Заполненый массив.

\*/

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

/\*\*

\* @brief Заполнение массива n - количеством элементов полученных от пользователя.

\* @param array - массив

\* @param n - размер массива

\* @return Заполненый массив

\*/

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

/\*\*

\* @brief Проверка корректности ввода интервала.

\* @param start - начало интервала.

\* @param end - конец интервала.

\* @return Значение "true" в случае усепшной проверки.

\*/

void checkinterval(int const start, int const end);

/\*\*

\* @brief Проверка иницилизации массива.

\* @param array - массив

\* @return Значение "true" в случае усепшной проверки.

\*/

void checkarray(const int\* array);

/\*\*

\* @brief Массив

\* @param n - размер массива

\* @return Создание нового массива

\*/

int\* getnewarray(const int n);

/\*\*

\* @brief Считывает целочисленное значение.

\* @remarks При неправильном вводе программа завершает выполнение.

\* @return Целочисленное значение.

\*/

int inputvalue(void);

/\*\*

\* @brief Считывает целочисленное значение.

\* @remarks Проверяет полученное значение на ноль, при соответствии программа завершает выполнение.

\* @return Целочисленное значение.

\*/

int abovezero(void);

/\*\*

\* @brief Выбор исполняемой функции

\* @brief Заполнить массив рандомными числами

\* @brief Заполнить массив самостоятельно

\*/

enum Choise

{

Fillrandomly = 1,

Fillmanually = 2,

};

/\*\*

\* @brief Точка входа в программу.

\* @return 0, в случае успеха.

\*/

int main(void)

{

setlocale(LC\_ALL, "Russian");

printf("Введите размер массива: ");

size\_t n = (size\_t)abovezero();

printf("Если вы хотите заполнить массив рандомными числами, то введите (%d), иначе (%d): ", Fillrandomly, Fillmanually);

int choice = inputvalue();

int\* array = getnewarray(n);

switch ((enum Choise)choice)

{

case Fillrandomly:

fillarrayrandomly(array, n);

break;

case Fillmanually:

fillarraymanually(array, n);

break;

default:

puts("Ошибка ввода номера операции.");

free(array);

return 1;

}

printf("Исходный массив: \n");

printarray(array, n);

int sumodd = sumofoddelements(array, n);

printf("Сумма нечетных элементов: %d\n", sumodd);

printf("Введите число A: \n");

int A = inputvalue();

printindicesgreaterthana(array, n, A);

replacesecondelementwithmaxnegative(array, n);

printf("Массив после замены второго элемента:\n");

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

{

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

}

printf("\n");

free(array);

return 0;

}

int sumofoddelements(int\* array, const size\_t n)

{

int sum = 0;

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

{

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

{

sum += array[i];

}

}

return sum;

}

void printindicesgreaterthana(int\* array, const size\_t n, int A)

{

printf("Индексы элементов больше %d: \n", A);

int found = 0;

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

{

if (array[i] > A)

{

printf("Индекс элемента, большего чем %d: %zu\n", A, i);

found = 1;

}

}

if (!found)

{

printf("Нет элементов, больших чем %d.\n", A);

}

}

void replacesecondelementwithmaxnegative(int\* array, const size\_t n)

{

int maxnegative = -1000;

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

{

if (array[i] < 0 && array[i] > maxnegative)

{

maxnegative = array[i];

}

}

if (maxnegative < 0 && n > 1)

{

array[1] = maxnegative;

}

}

void printarray(const int\* array, const size\_t n)

{

checkarray(array);

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

{

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

}

printf("\n");

}

void fillarrayrandomly(int\* array, const size\_t n)

{

checkarray(array);

srand(time(NULL));

printf("Введите начало интервала: ");

int start = inputvalue();

printf("Введите конец интервала: ");

int end = inputvalue();

checkinterval(start, end);

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

{

array[i] = rand() % (end - start + 1) + start;

}

}

void fillarraymanually(int\* array, const size\_t n)

{

checkarray(array);

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

{

printf("Введите элемент %zu: ", i + 1);

scanf\_s("%d", &array[i]);

}

}

void checkinterval(int const start, int const end)

{

if (start > end)

{

errno = EIO;

perror("Ошибка ввода интервала: ");

exit(EXIT\_FAILURE);

}

}

void checkarray(const int\* array)

{

if (array == NULL)

{

errno = EIO;

perror("Массив не инициализирован: ");

exit(EXIT\_FAILURE);

}

}

int\* getnewarray(const int n)

{

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

checkarray(array);

return array;

}

int inputvalue(void)

{

int value = 0;

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

{

errno = EIO;

perror("Ошибка ввода числового значения: ");

exit(EXIT\_FAILURE);

}

return value;

}

int abovezero(void)

{

int value = inputvalue();

if (value <= 0)

{

errno = EIO;

perror("Ошибка ввода числового значения (оно не может быть меньше или равным нулю): ");

exit(EXIT\_FAILURE);

}

return value;

}

1. Результаты выполнения программы

Результаты выполнения программы представлены ниже (Рисунок 5,Рисунок 6, Рисунок 7, Рисунок 8, Рисунок 9, Рисунок 10, Рисунок 11).

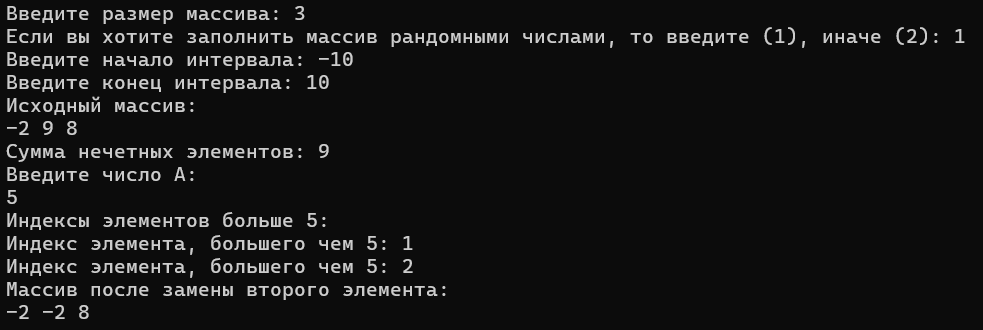


Рисунок 5 – Результаты выполнения программы

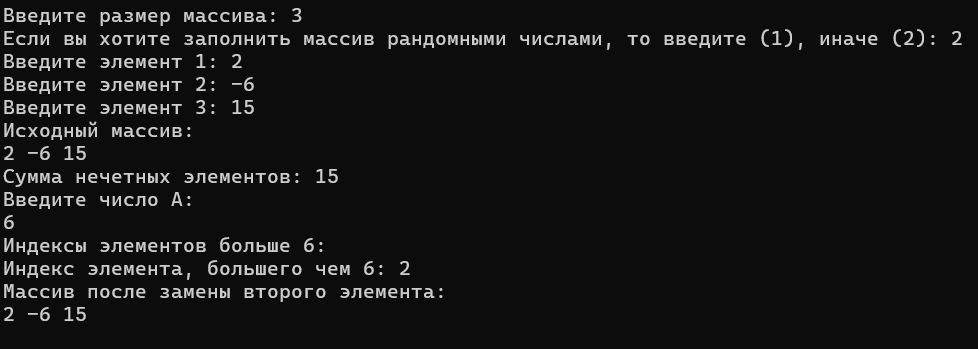


Рисунок 6 – Результаты выполнения программы при альтернативном вводе



Рисунок 7 – Результаты выполнения программы при альтернативном вводе

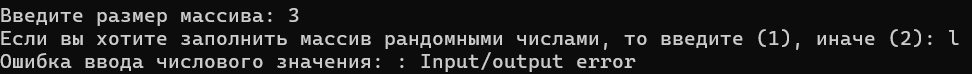


Рисунок 8 – Результаты выполнения программы при альтернативном вводе

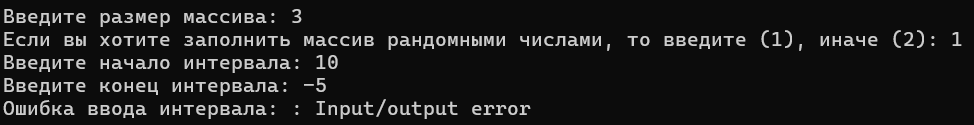


Рисунок 9 – Результаты выполнения программы при альтернативном вводе

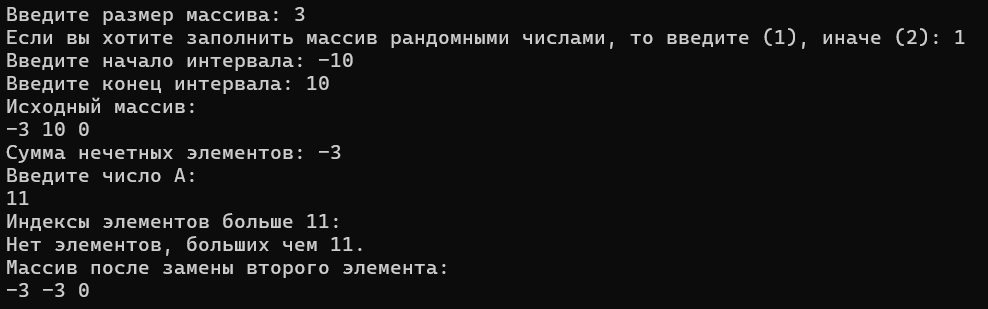


Рисунок 10 – Результаты выполнения программы при альтернативном вводе

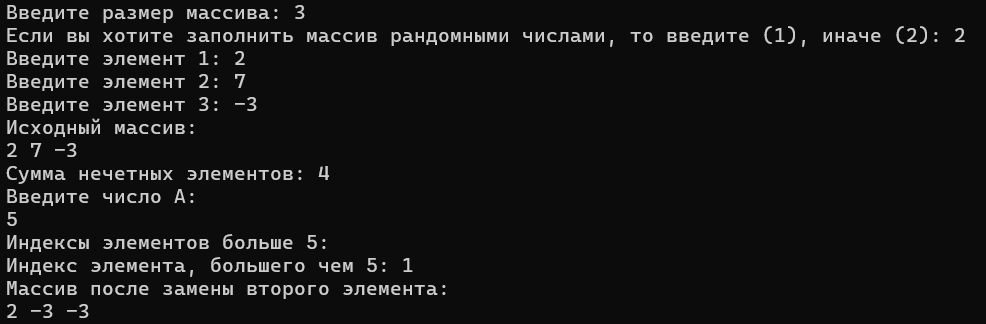


Рисунок 11 – Результаты выполнения программы при альтернативном вводе

1. Выполнение тестовых примеров

В программе IDLE выполнен тестовый пример. Результаты его выполнения представлен ниже (Рисунок 12, Рисунок 13, Рисунок 14, Рисунок 15, Рисунок 16, Рисунок 17, Рисунок 18).

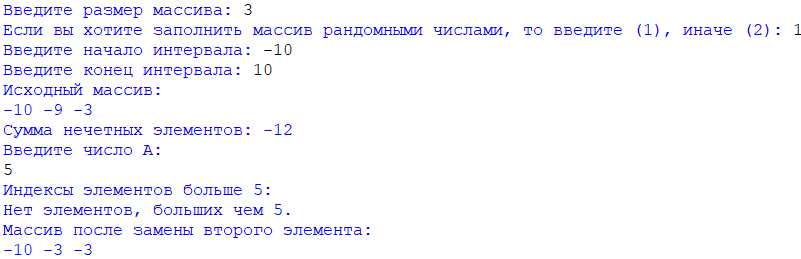


Рисунок 12 – Результат работы программы

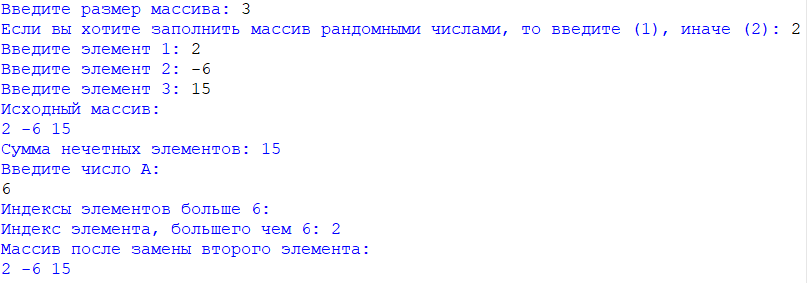


Рисунок 13 – Результат работы программы при альтернативном вводе



Рисунок 14 – Результат работы программы при альтернативном вводе



Рисунок 15 – Результат работы программы при альтернативном вводе

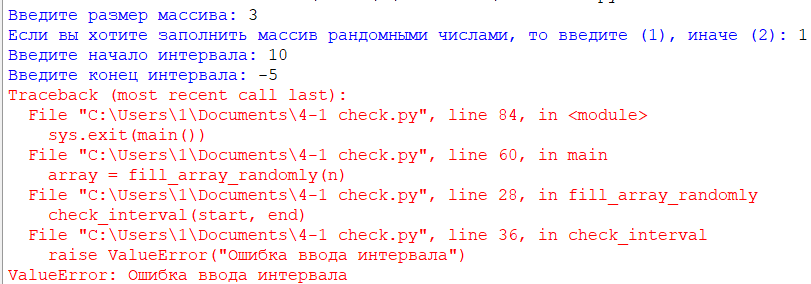


Рисунок 16 – Результат работы программы при альтернативном вводе

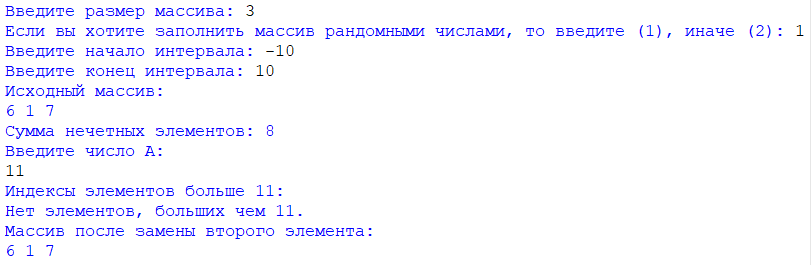


Рисунок 17 – Результат работы программы при альтернативном вводе

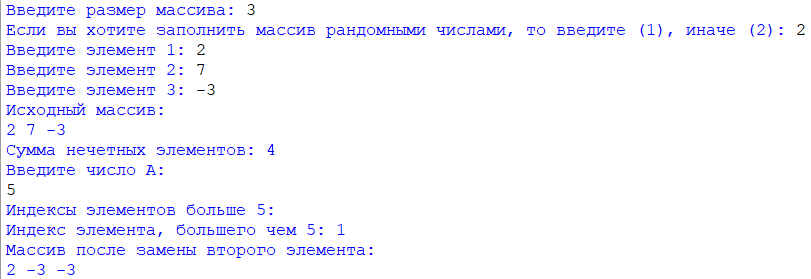


Рисунок 18 – Результат работы программы при альтернативном вводе

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

