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 14

Выполнил: ст. гр. ТКИ-142

Кузнецов Денис Константинович

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

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

Москва 2024

1. Формулировка задания

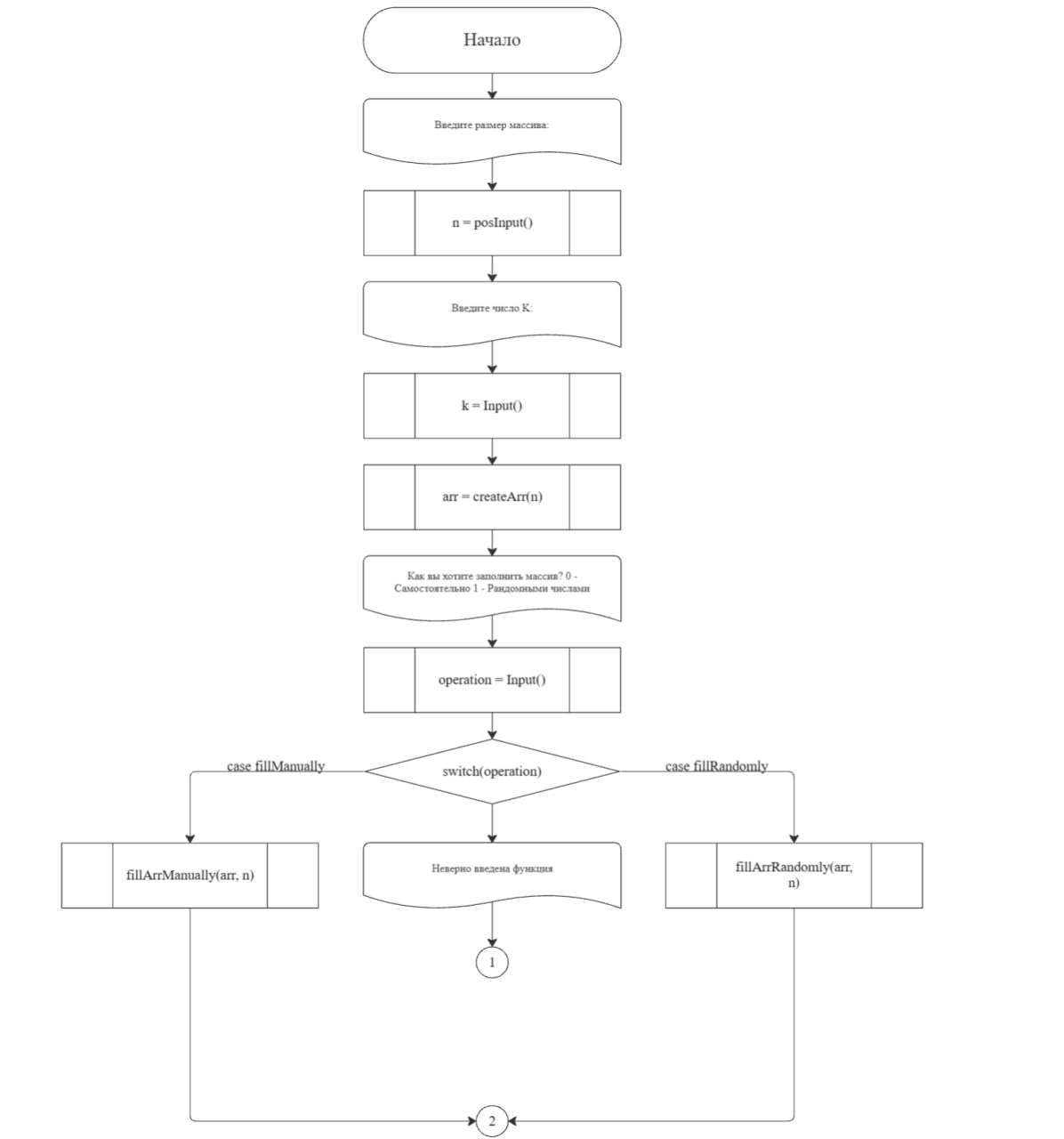
Создать одномерный массив из *n* целых чисел. Реализовать возможность заполнения массива, как случайными числами, так и с помощью клавиатуры по желанию пользователя. Пункты задания организовать в виде функций (методов). Вывести массив на экран. Составить блок-схему.

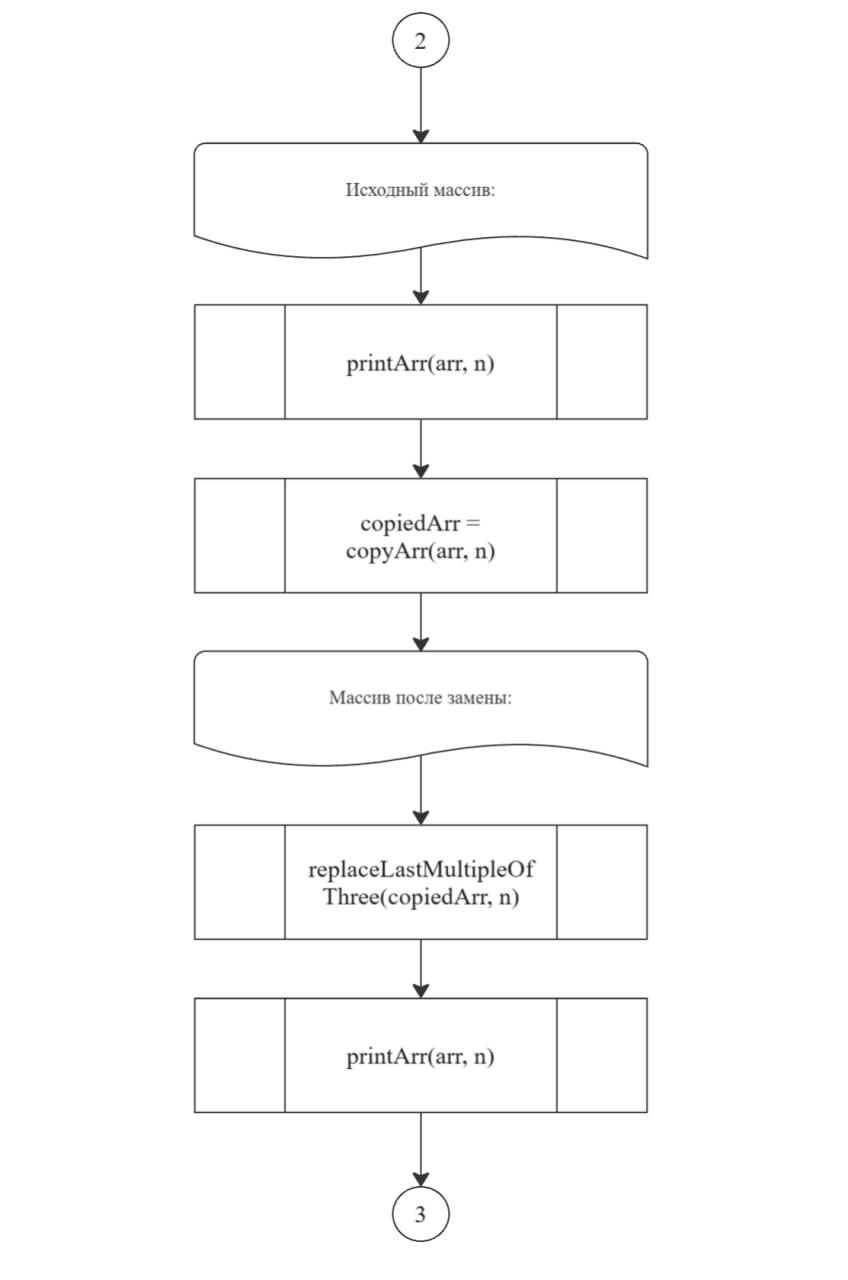
Таблица 1 – Формулировка задания 4.2

|  |  |  |
| --- | --- | --- |
| Вариант | Задачи | Интервал |
| 14 | 1. Заменить последний элемент массива, кратный 3, нулем. 2. Вставить число К после всех четных элементов. 3. Из элементов массива P сформировать массив M той же размерности по правилу: первый и последний элементы равны нулю, каждый четвертый элемент по формуле Mi=4\*abs(Pi), а все остальные по формуле Mi=-Pi\*(i+1). | [-100;100] |

1. Блок-схема алгоритма

Блок-схема основного алгоритма представлена ниже (Рисунок 1). Блок-схемы остальных функций представлены ниже (Рисунок 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13).





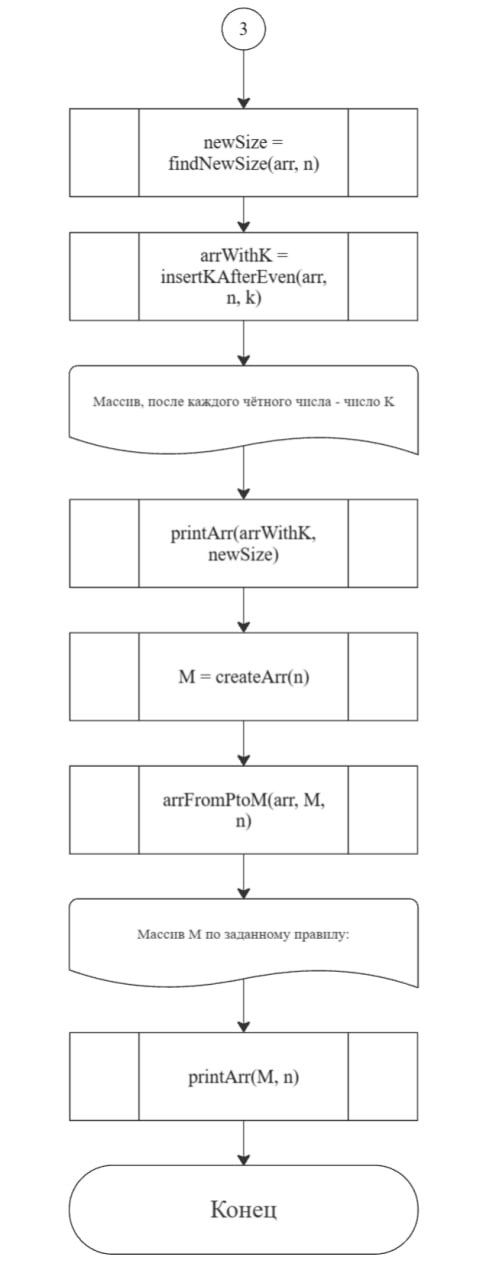


Рисунок  -­ Блок-схема основного алгоритма



Рисунок 2 – Блок-схема функции input()



Рисунок 3 – Блок-схема функции createArr(n)



Рисунок 4 – Блок-схема функции copyArr(arr, n)



Рисунок 5 – Блок-схема функции posInput()



Рисунок 6 – Блок-схема функции checkArr(arr)



Рисунок 7 – Блок-схема функции fillArrRandomly(arr, n)



Рисунок 8 – Блок-схема функции fillArrManually (arr, n)



Рисунок 9 – Блок-схема функции printArr(arr, n)

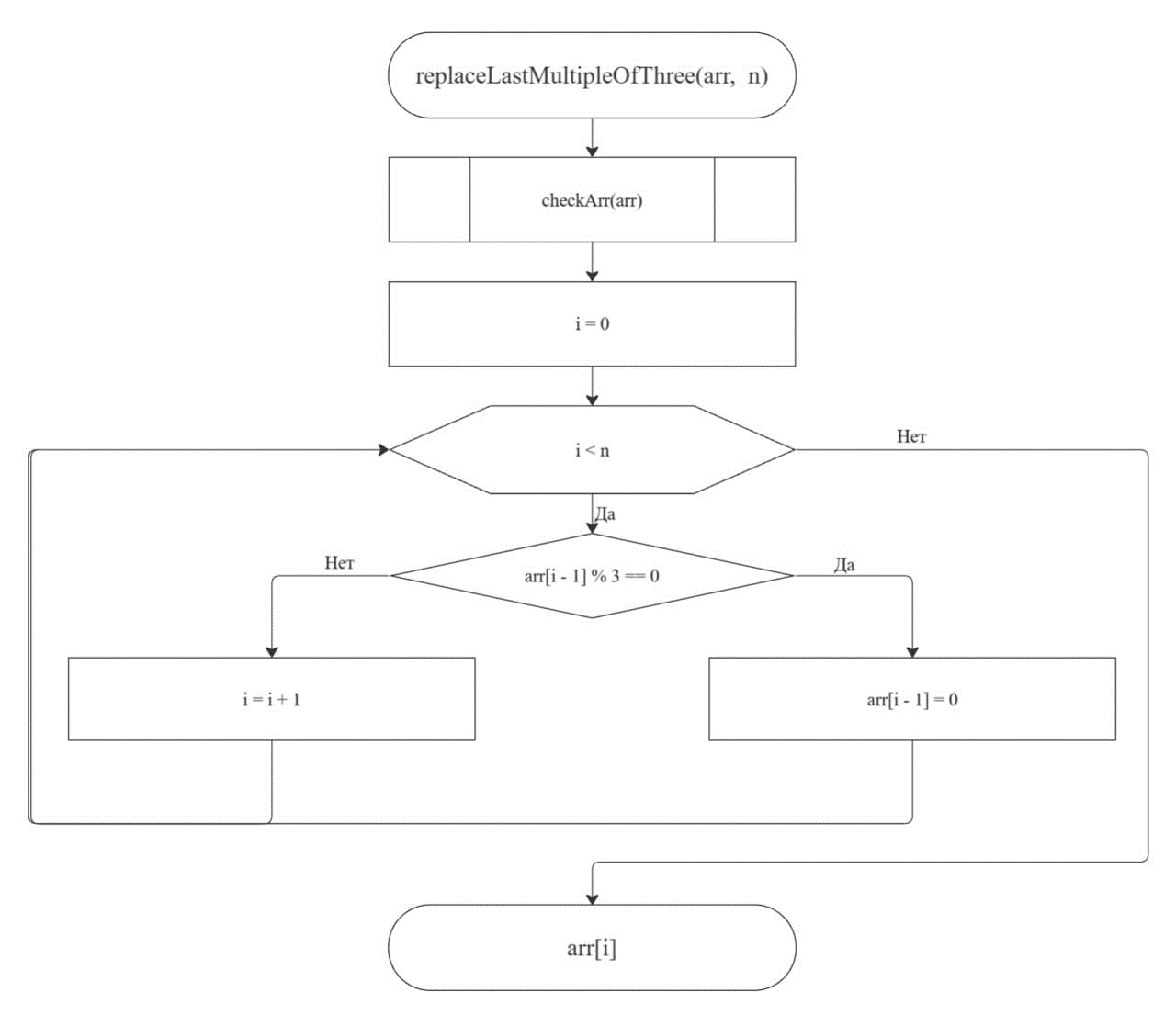


Рисунок 10 – Блок-схема функции replaceLastMultipleOfThree(arr, n)

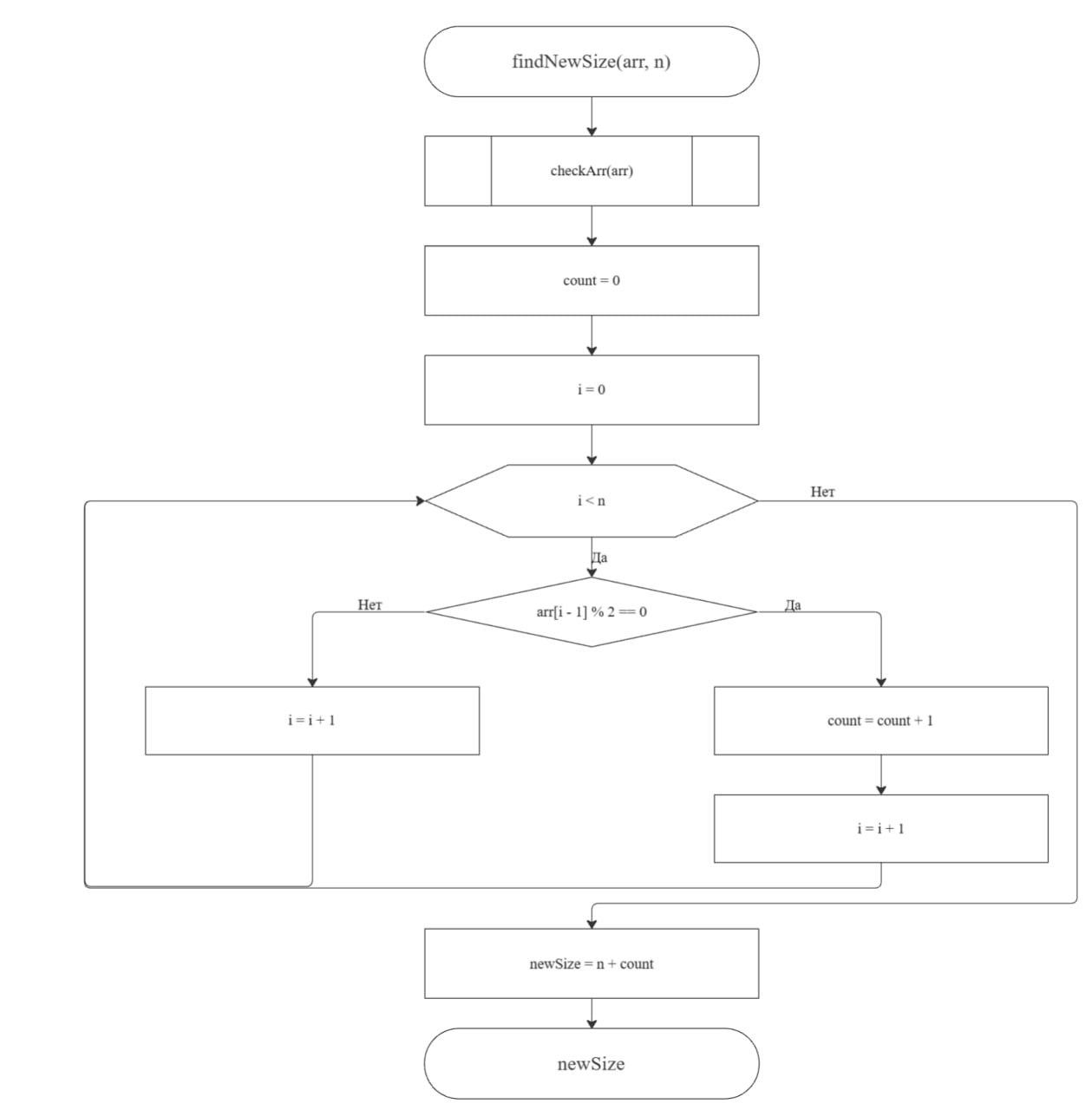


Рисунок 11 – Блок-схема функции findNewSize(arr, n)

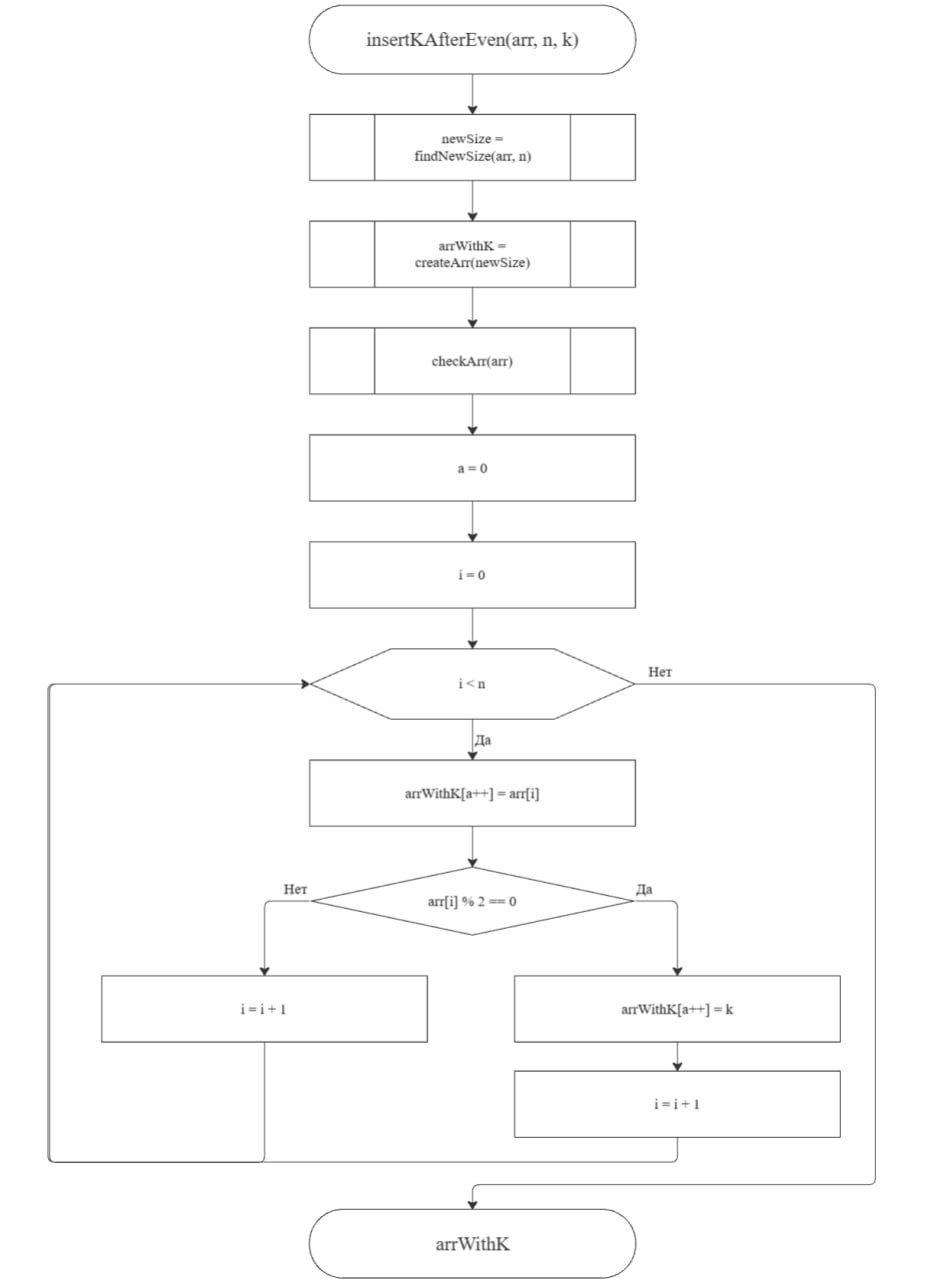


Рисунок 12 – Блок-схема функции insertKAfterEven(arr, n, k)

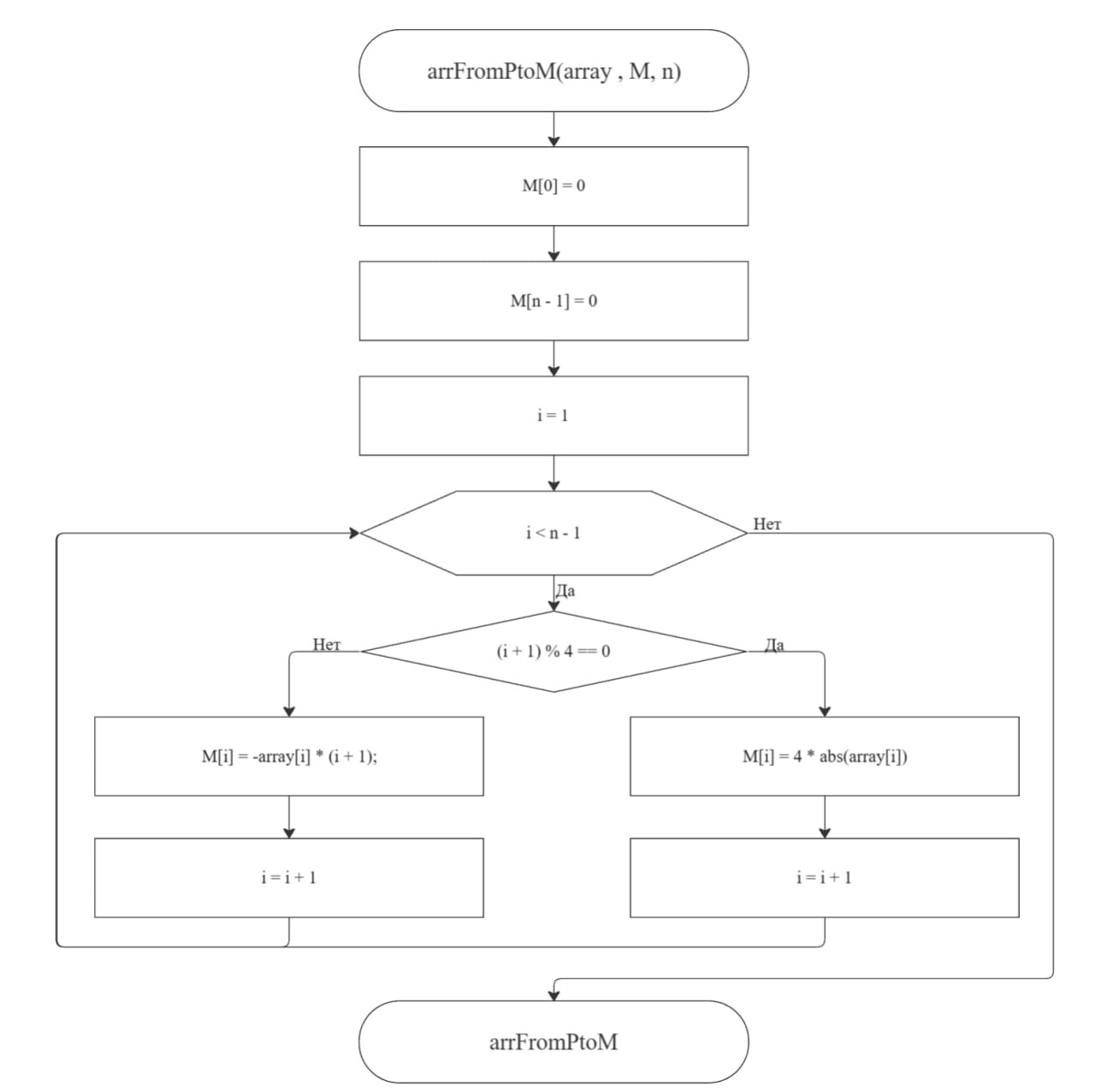


Рисунок 13 - Блок-схема функции arrFromMToP(arr, M, n)

1. Текст программы на языке C

#include <stdio.h>

#include <stdlib.h>

#include <errno.h>

#include <stdbool.h>

#include <time.h>

/\*\*

\*@param manual реализация способа заполнения массива числами, которые вводит пользователь

\* @param random реализация способа заполнения массива рандомными(случайными) числами

\*/

enum opetation {

fillManually,

fillRandomly

};

/\*\*

\* @brief функция, считывающая вводимое значение

\* @return введенное значение

\*/

int input(void);

/\*\*

\* @brief функция, проверяющая ввод на положительность

\* @param value вводимое число

\* @return Возвращает: число, если число положительно; ошибку, если число отрицательно

\*/

size\_t posInput(void);

/\*\*

\* @brief функция, которая создаёт массив размером n

\* @param n размер массива

\* @return созданный массив

\*/

int\* createArr(const size\_t n);

/\*\*

\* @brief функция, проверяющая корректность адреса указателя массива

\* @param arr массив

\* @return Возвращает ошибку, если потерялась память

\*/

void checkArr(const int\* arr);

/\*\*

\* @brief функция, которая выводит массив пользователю

\* @param arr указатель на наш массив

\* @param n размер массива

\* @return наш массив

\*/

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

/\*\*

\* @brief функция, которая позволяет пользователю заполнить массив самостоятельно

\* @param arr массив

\* @param n количество элементов в массиве

\* @return массив, заполненный пользователем самостоятельно

\*/

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

/\*\*

\* @brief функция, которая позволяет пользователю заполнить массив с помощью случайных чисел

\* @param arr массив

\* @param n количество элементов в массиве

\* @return массив, заполненный случайными числами

\*/

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

/\*\*

\* @brief функция, которая копирует элементы из одного массива в новый

\* @param arr исходный массив

\* @param n размер массива

\* @return новый копированный массив, который повторяет исходный

\*/

int\* copyArr(const int\* arr, const size\_t n);

/\*\*

\* @brief функция, заменяющая последний элемент, который делится на 3, нулём

\* @param arr исходный массив

\* @param n размер массива

\*/

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

/\*\*

\* @brief функция, которая вычисляет новый размер массива

\* @param arr исходный массив

\* @param n количество элементов в массиве

\* @return новый размер массива

\*/

size\_t findNewSize(const int\* arr, const size\_t n);

/\*\*

\* @brief функция, которая вставляет число K после всех четных элементов

\* @param arr массив

\* @param n количество элементов в массиве

\* @param k число, которое нужно вставлять

\* @return измененный массив

\*/

int\* insertKAfterEven(const int\* arr, size\_t n, const int k);

/\*\*

\* @brief Из элементов массива P формирует массив M

\* @param array массив P (исходный)

\* @param M массив M

\* @param n количество элементов в массиве

\* @return массив M

\*/

void arrFromPtoM(const int\* arr, int\* M, const size\_t n);

/\*\*

\* @brief точка входа в программу

\* @return 0 в случае успеха

\*/

int main(void) {

printf("Введите размер массива: ");

size\_t n = posInput();

printf("Введите число K: ");

const int k = input();

int\* arr = createArr(n);

printf("Как вы хотите заполнить массив? \n%d - Самостоятельно \n%d - Рандомными числами\n", fillManually, fillRandomly);

int operation = input();

switch(operation) {

case fillManually:

fillArrManually(arr, n);

break;

case fillRandomly:

fillArrRandomly(arr, n);

break;

default:

free(arr);

errno = ERANGE;

perror("Неверно введена функция");

exit(EXIT\_FAILURE);

break;

}

printf("Исходный массив: ");

printArr(arr, n);

int\* copiedArr = copyArr(arr, n);

printf("\nМассив после замены: ");

replaceLastMultipleOfThree(copiedArr, n);

printArr(copiedArr, n);

free(copiedArr);

size\_t newSize = findNewSize(arr, n);

int\* arrWithK = insertKAfterEven(arr, n, k);

printf("\nМассив, после каждого чётного числа - число K ");

printArr(arrWithK, newSize);

free(arrWithK);

int\* M = createArr(n);

arrFromPtoM(arr, M, n);

printf("\nМассив M по заданному правилу: ");

printArr(M, n);

free(M);

free(arr);

return 0;

}

int input(void) {

int value = 0;

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

if (res != 1) {

errno = EIO;

perror("Ошибка ввода (число должно быть целым)");

exit(EXIT\_FAILURE);

}

return value;

}

size\_t posInput(void) {

int value = input();

if (value <= 0) {

errno = EINVAL;

perror("Ошибка ввода (число должно быть положительным)");

exit(EXIT\_FAILURE);

}

return (size\_t)value;

}

int\* createArr(const size\_t n) {

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

checkArr(arr);

return arr;

}

void checkArr(const int\* arr) {

if (arr == NULL) {

errno = ENOMEM;

perror("Ошибка памяти, адрес указателя на массив оказался нулевым");

exit(EXIT\_FAILURE);

}

}

void printArr(const int\* arr, const size\_t n) {

checkArr(arr);

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

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

if (i < n - 1) {

printf(", ");

}

}

}

void fillArrManually(int\* arr, const size\_t n) {

checkArr(arr);

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

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

arr[i] = input();

}

}

void fillArrRandomly(int\* arr, const size\_t n) {

checkArr(arr);

printf("Введите минимальное значение интервала \n");

int min = input();

printf("Введите максимальное значение интервала \n");

int max = input();

if (min > max) {

errno = EINVAL;

perror("У вас неправильно заданы границы интервала: конец должен быть меньше начала");

exit(EXIT\_FAILURE);

}

srand(time(NULL));

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

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

}

}

int\* copyArr(const int\* arr, const size\_t n) {

checkArr(arr);

int\* copiedArr = createArr(n);

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

copiedArr[i] = arr[i];

}

return copiedArr;

}

void replaceLastMultipleOfThree(int\* arr, const size\_t n) {

checkArr(arr);

for (size\_t i = n; i > 0; i--) {

if (arr[i - 1] % 3 == 0) {

arr[i - 1] = 0;

break;

}

}

}

size\_t findNewSize(const int\* arr, const size\_t n) {

checkArr(arr);

size\_t count = 0;

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

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

count++;

}

}

size\_t newSize = n + count;

return newSize;

}

int\* insertKAfterEven(const int\* arr, size\_t n, const int k) {

size\_t newSize = findNewSize(arr, n);

int\* arrWithK = createArr(newSize);

checkArr(arr);

size\_t a = 0;

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

arrWithK[a++] = arr[i];

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

arrWithK[a++] = k;

}

}

return arrWithK;

}

void arrFromPtoM(const int\* array, int\* M, const size\_t n) {

M[0] = 0;

M[n - 1] = 0;

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

if ((i + 1) % 4 == 0) {

M[i] = 4 \* abs(array[i]);

}

else {

M[i] = -array[i] \* (i + 1);

}

}

}

1. Результаты выполнения программы

Результаты выполнения программы представлены ниже (Рисунок 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25)

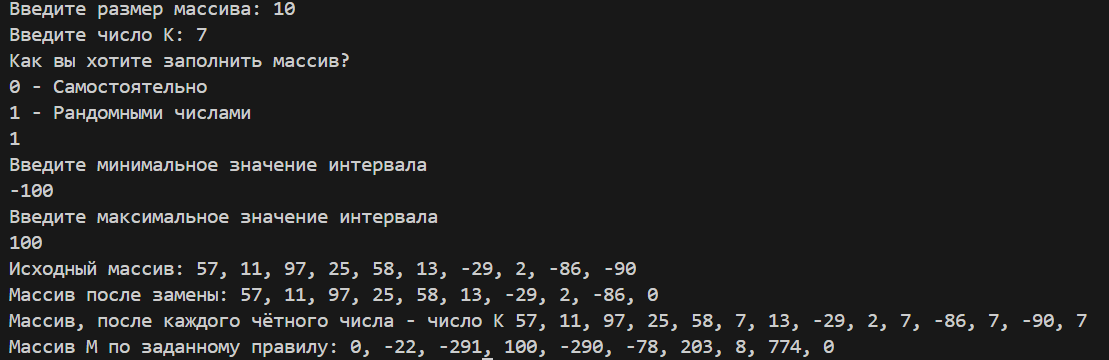


Рисунок 14 – Результаты выполнения программы



Рисунок 15 – Результаты выполнения программы при вводе отрицательного числа в величину размера массива.



Рисунок 16 – Результаты выполнения программы если размер массива – буква



Рисунок 17 – Результаты выполнения программы если размер массива – ноль

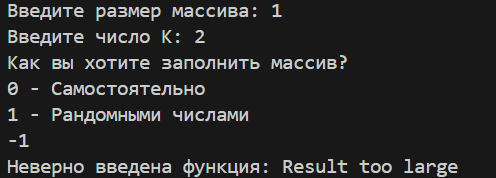


Рисунок 18 - Результаты выполнения программы, если число выбора отрицательно

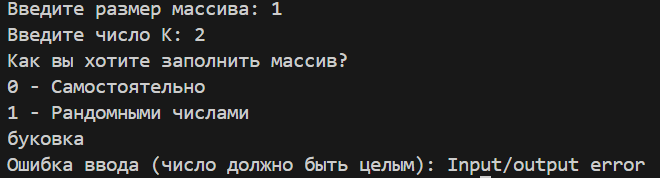


Рисунок 19 – Результаты выполнения программы, если введенный выбор – буква

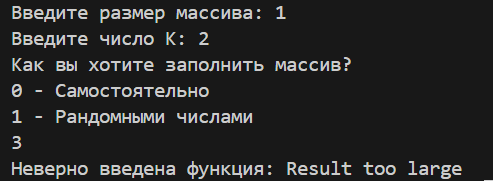
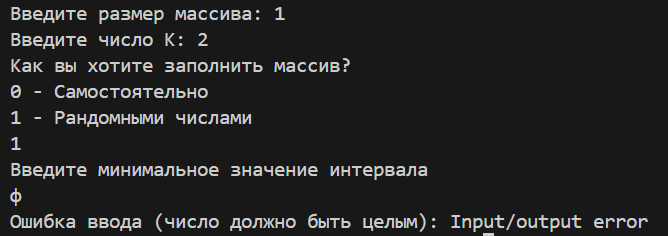


Рисунок 20 – Результат выполнения программы, если выбор больше диапазона

  
Рисунок 21 – Результат выполнения программы, если минимальное значения интервала – буква

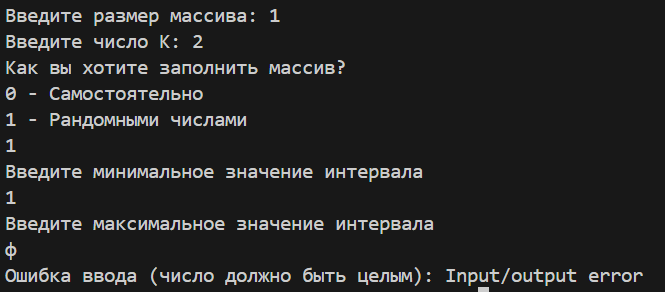


Рисунок 22 – Результат выполнения программы, если введенный максимум интервала – буква

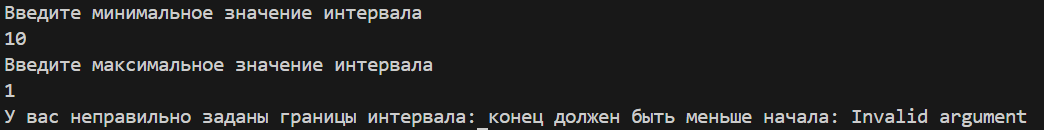


Рисунок 23 – Результат выполнения программы, если минимум интервала больше максимума

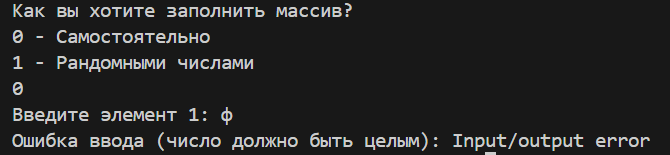


Рисунок 24 – Результат выполнения программы, если введенный элемент массива – буква

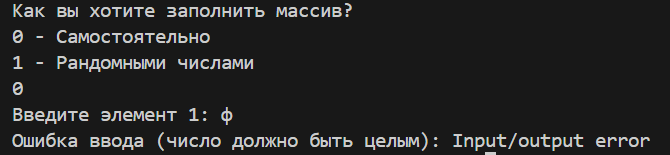


Рисунок 25 – Результат выполнения программы, если введенное значение k - буква

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