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

По дисциплине «Языки программирования»

ВАРИАНТ 1

Выполнил: ст. гр. ТКИ - 142

Шабин Константин Олегович

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

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

Москва 2023

Оглавление

[1. Задание 4.1 3](#_Toc156863526)

[1.1. Формулировка задания 3](#_Toc156863527)

[1.2. Блок-схема алгоритма 4](#_Toc156863528)

[1.3. Код задания 4.1 15](#_Toc156863529)

[1.4. Решение тестового примера 21](#_Toc156863530)

[1.5. Зачёт задания в GitHub 24](#_Toc156863531)

[2. Задание 4.2 25](#_Toc156863532)

[2.1. Формулировка задания 25](#_Toc156863533)

[2.2. Блок-схема алгоритма 26](#_Toc156863534)

[2.3. Код задания 4.2 38](#_Toc156863535)

[2.4. Решение тестового примера 45](#_Toc156863536)

[2.5. Зачёт задания в GitHub 49](#_Toc156863537)

# Задание 4.1

* 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 1 –Формулировка задания 4.1

|  |  |  |
| --- | --- | --- |
| Вариант | Задачи | Интервал |
| 1 | 1. Найти сумму элементов, имеющих нечетное значение. 2. Вывести индексы тех элементов, значения которых больше заданного числа А. 3. Заменить второй элемент массива на максимальный среди отрицательных. | [-10;10] |

* 1. Блок-схема алгоритма

Блок-схема алгоритма представлена на рисунках (Рисунок 1, Рисунок 2, Рисунок 3, Рисунок 4, Рисунок 5, Рисунок 6, Рисунок 7, Рисунок 8, Рисунок 9, Рисунок 10, Рисунок 11, Рисунок 12, Рисунок 13, Рисунок 14, Рисунок 15, Рисунок 16)

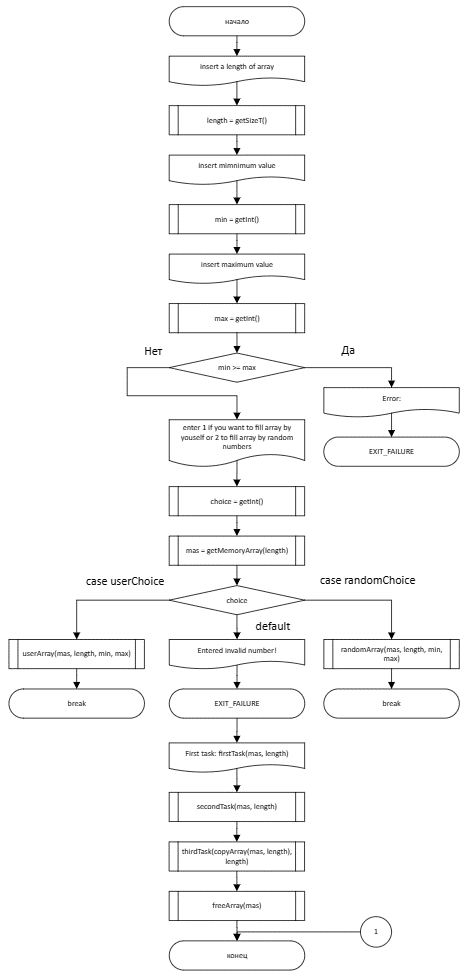


Рисунок 1 – Блок-схема функции main()

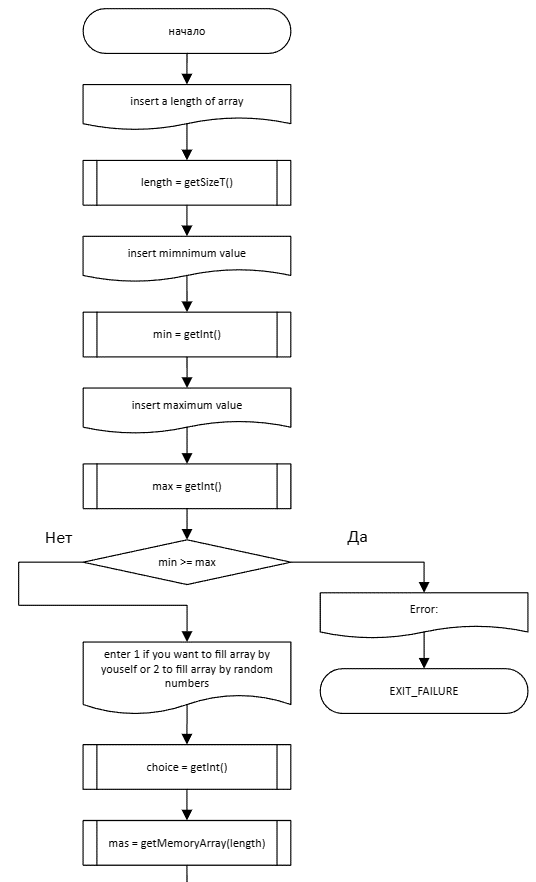


Рисунок 2 – Блок-схема функции main()(1 часть)

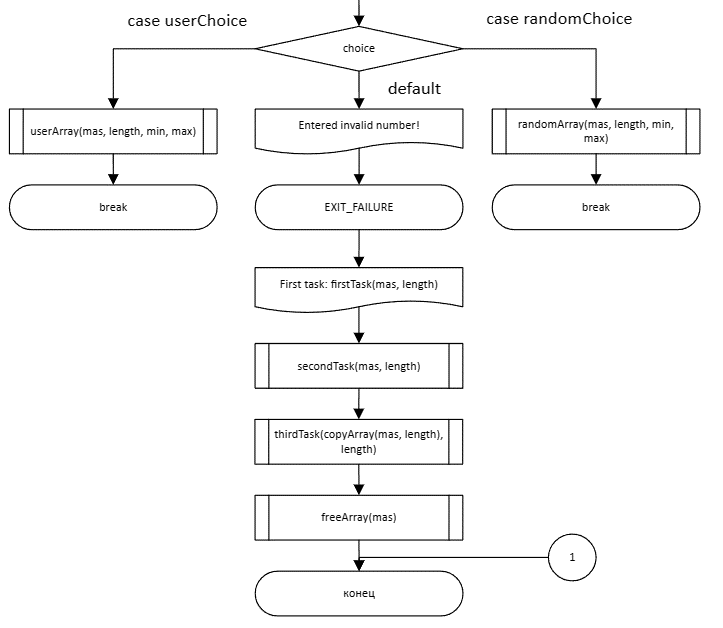


Рисунок 3 – Блок-схема функции main()(2 часть)

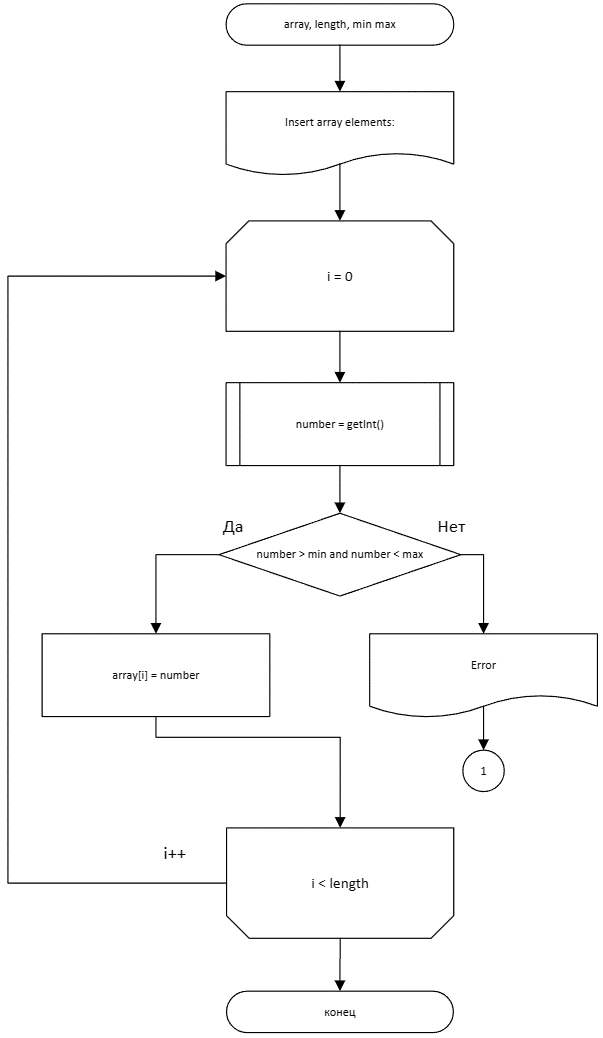
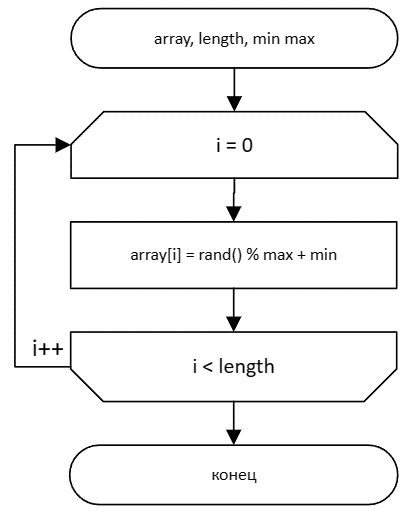
Рисунок 4 – Блок-схема функции userArray(array,length,min,max) 

Рисунок 5 – Блок-схема функции randomArray (array, length, min, max)

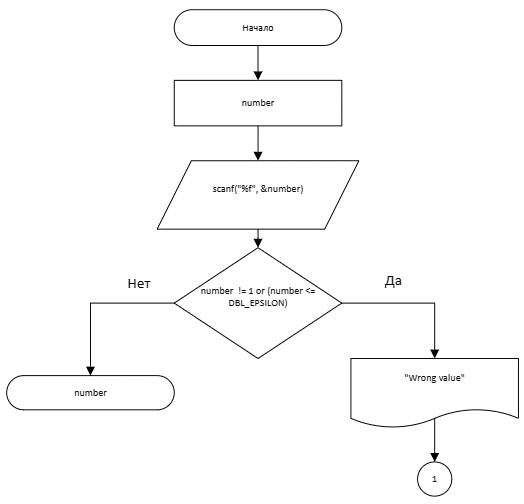


Рисунок 6 – Блок-схема функции getInt()

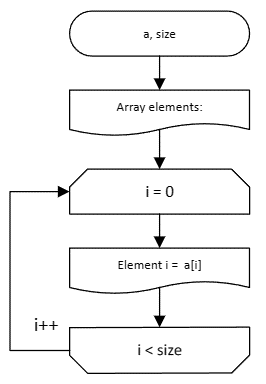


Рисунок 7– Блок-схема функции printArray(a, size)

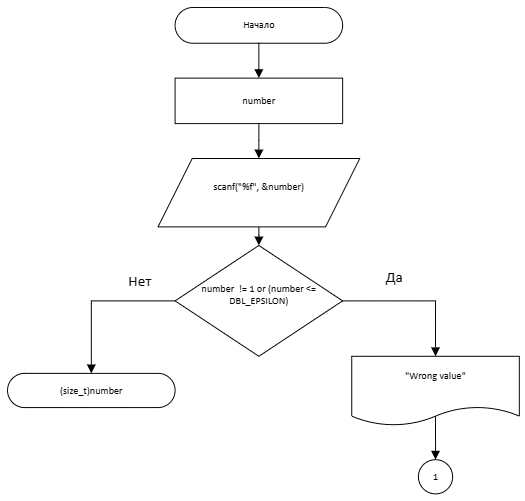


Рисунок 8 – Блок-схема функции getSizet()

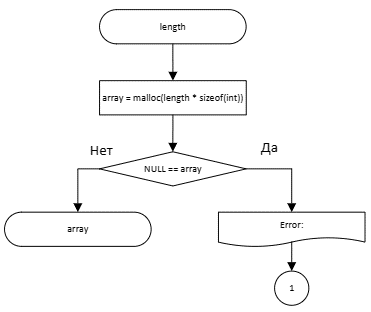


Рисунок *9* – Блок-схема функции getMemoryArray(length)

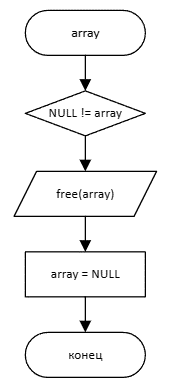


Рисунок *10* – Блок-схема функции freeArray(array)

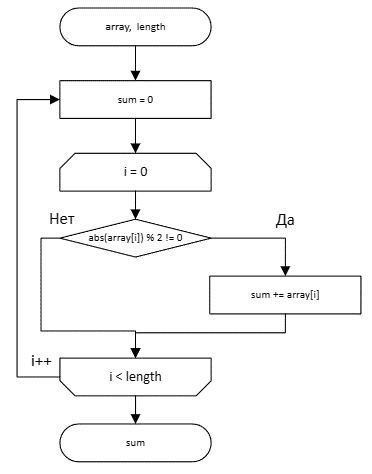


Рисунок *11* – Блок-схема функции firstTask(array,length)

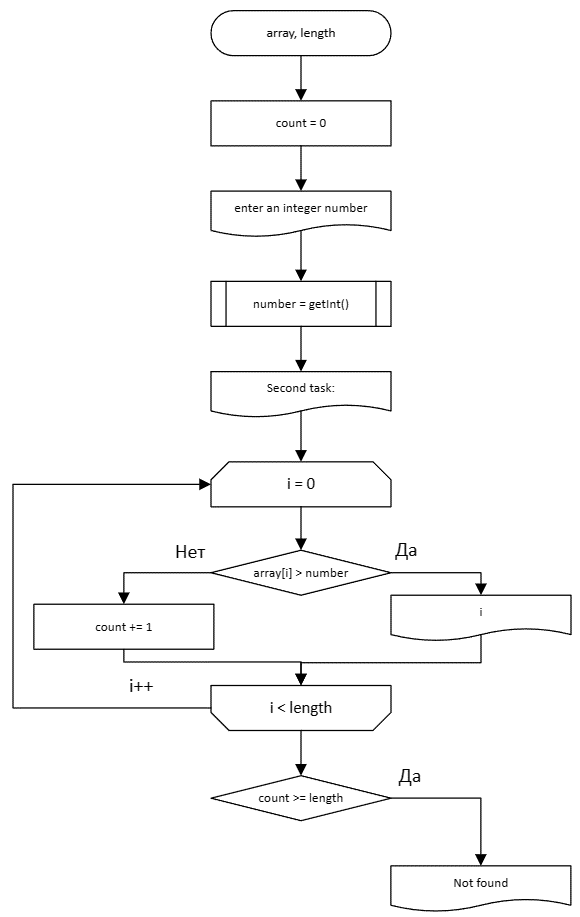


Рисунок *12* – Блок-схема функции secondTask(array,length)

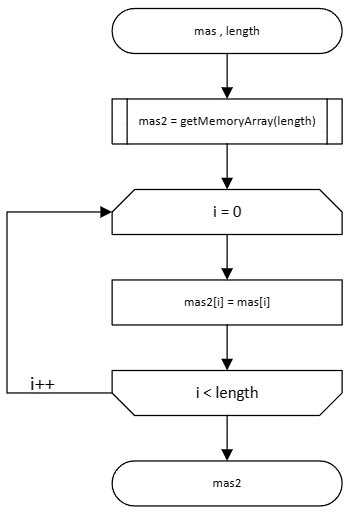


Рисунок *13* – Блок-схема функции copyArray(mas, length)

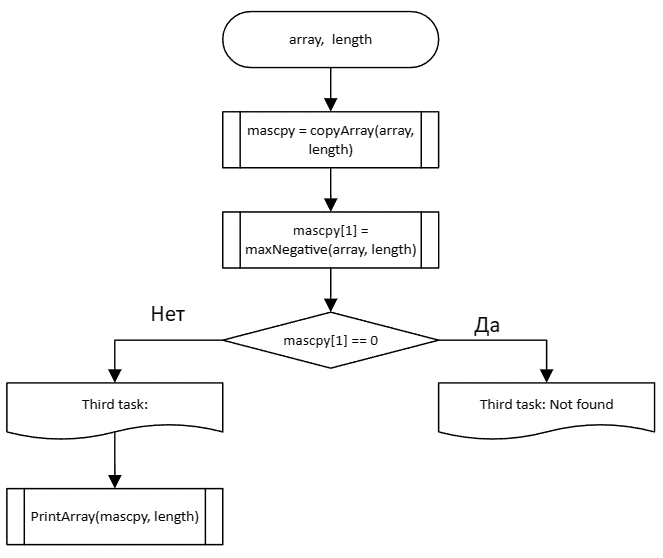


Рисунок *14* – Блок-схема функции thirdTask(array, length)

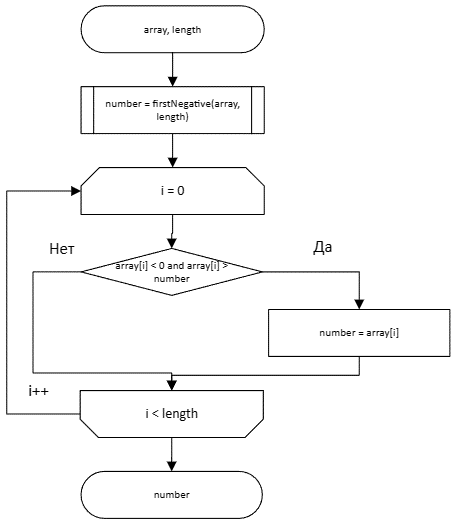


Рисунок *15* – Блок-схема функции maxNegative (array, length)

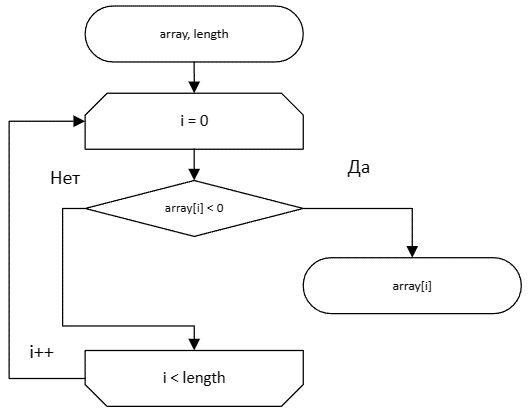


Рисунок *16* – Блок-схема функции firstNegative(array, length)

* 1. Код задания 4.1

#include <stdio.h>

#include <stdlib.h>

#include <malloc.h>

#include <errno.h>

/\*\*

\* @brief присваивает целочисленное значение переменной

\* @return целочисленная цифра

\*/

int getInt();

/\*\*

\* @brief присваивает переменной целочисленное значение и проверяет его на положителность

\* @return number положительное число

\*/

size\_t getSizeT();

/\*\*

\* @brief хранит константы, указывающие выбор пользователя для заполнения массива

\* @brief userChoice хранит значение, вызывающее ввод массива вручную

\* @brief randomChoice хранит значение, вызывающее ввод массива случайными числами

\*/

enum Choices

{

userChoice = 1,

randomChoice = 2,

};

/\*\*

\* @brief выделяет память под массив

\* @param length длина массива

\* @return array указатель на пустой массив

\*/

int\* getMemoryArray(const int length);

/\*\*

\* @brief заполняет массив, элементы вводит пользователь

\* @param array указатель на заполняемый массив

\* @param length длина массива

\* @param min минимальное значение массива

\* @param max максимальное значение массива

\*/

void userArray(int\* array, const size\_t length, int min, int max);

/\*\*

\* @brief заполняет массив случайными числами в диапазоне [-10:10]

\* @param array указатель на заполняемый массив

\* @param length длина массива

\* @param min минимальное значение массива

\* @param max максимальное значение массива

\*/

void randomArray(int\* array, const size\_t length, int min, int max);

/\*\*

\* @brief функция копирует массив

\* @param mas - копия массива

\* @param length - размер массива

\* @return Возвращает копию массива

\*/

int\* copyArray(int\* mas , const int length);

/\*\*

\* @brief освобождение массива

\* @param array указатель на массив

\*/

void freeArray(int\* array);

/\*\*

\* @brief находит сумму элементов, имеющих нечетное значение

\* @param array указатель на заполняемый массив

\* @param length длина массива

\*/

const int firstTask(int\* array, size\_t length);

/\*\*

\* @brief выводит индексы элементов, значения которых больше заданного числа

\* @param array указатель на заполняемый массив

\* @param length длина массива

\*/

const int secondTask(int\* array, size\_t length);

/\*\*

\* @brief заменяет второй элемент массива на максимальный среди отрицательных и выводит его

\* @param array указатель на заполняемый массив

\* @param length длина массива

\*/

void thirdTask(int\* array, const int length);

/\*\*

\* @brief находит наибольшее отрицательное значение в массиве

\* @param length длина массива

\* @param array указатель на заполняемый массив

\* @return array указатель на пустой массив

\*/

const int maxNegative (int\* array, const int length);

/\*\*

\* @brief функция печатает массив

\* @param size - размер массива

\* @param a - массив

\*/

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

/\*\*

\* @brief находит первый отрицательный элемент массива

\* @param array указатель на заполняемый массив

\* @param length длина массива

\* @return возвращает первое отрицательное значение массива если оно есть

\* @return возвращает 0 если в массиве нет отрицательных чисел

\*/

const int firstNegative(int\* array, const int length);

/\*\*

\* @brief Точка входа в программу

\* @return Возвращает 0 если программа работет исправно, иначе 1

\*/

int main()

{

printf("insert a length of array \n");

size\_t length = getSizeT();

printf("insert mimnimum value \n");

int min = getInt();

printf("insert maximum value \n");

int max = getInt();

if (min >= max)

{

errno = EIO;

perror("Error: ");

return 1;

}

printf("enter %d if you want to fill array by youself or %d to fill array by random numbers \n", userChoice, randomChoice);

int choice = getInt();

int\* mas = getMemoryArray(length);

switch ("%d", choice)

{

case randomChoice:

randomArray(mas, length, min, max);

break;

case userChoice:

userArray(mas, length, min, max);

break;

default:

puts("Entered invalid number!\n");

return 1;

}

printf("First task: %d\n", firstTask(mas, length));

secondTask(mas, length);

thirdTask(copyArray(mas, length), length);

freeArray(mas);

return 0;

}

int getInt()

{

int number;

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

{

errno = EIO;

perror("Error: ");

abort();

}

return number;

}

size\_t getSizeT()

{

int number = getInt();

if (number <= 0)

{

errno = ERANGE;

perror("Error: \n");

abort();

}

return (size\_t)number;

}

int\* getMemoryArray(const int length)

{

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

if (NULL == array)

{

errno = ENOMEM;

perror("Error: ");

abort();

}

return array;

}

void userArray(int\* array, const size\_t length, int min, int max)

{

printf("Insert array elements:");

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

{

int number = getInt();

if (number > min && number < max)

{

array[i] = number;

}

else

{

errno = EIO;

perror("Error: \n");

abort();

}

}

}

void randomArray(int\* array, const size\_t length, int min, int max)

{

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

{

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

}

}

void freeArray(int\* array)

{

if (NULL != array)

{

free(array);

array = NULL;

}

}

const int firstTask(int\* array, size\_t length)

{

int sum = 0;

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

{

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

{

sum += array[i];

}

}

return sum;

}

const int secondTask(int\* array, size\_t length)

{

size\_t count = 0;

printf("enter an integer number \n");

int number = getInt();

printf("Second task: ");

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

{

if (array[i] > number)

{

printf("%u ", i);

}

else

{

count += 1;

}

}

if (count >= length)

{

printf("Not found");

}

}

int\* copyArray(int\* mas , const int length)

{

int\* mas2 = getMemoryArray(length);

for (int i = 0; i < length; i++)

{

mas2[i] = mas[i];

}

return mas2;

}

void thirdTask(int\* array, const int length)

{

int\* mascpy = copyArray(array, length);

mascpy[1] = maxNegative(array, length);

if (mascpy[1] == 0)

{

printf("\nThird task: Not found");

}

else

{

puts("\nThird task:");

PrintArray(mascpy, length);

}

}

const int maxNegative (int\* array, const int length)

{

int number = firstNegative(array, length);

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

{

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

{

number = array[i];

}

}

return number;

}

const int firstNegative(int\* array, const int length)

{

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

{

if (array[i] < 0)

{

return array[i];

}

}

}

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

{

puts("Array elements: ");

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

{

printf("Element %zu = %d\n", i, a[i]);

}

}

* 1. Решение тестового примера

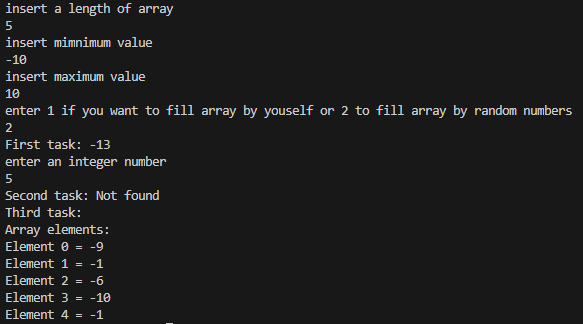


Рисунок 17 – Решение тестового примера рандомным заполнением массива

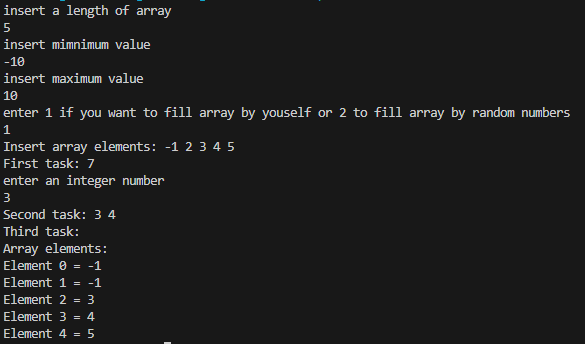


Рисунок 18 – Решение тестового примера пользовательским заполнением массива



Рисунок 19 – Вывод программы, когда размер массива – буква

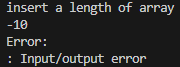


Рисунок 20 – Вывод программы, когда размер массива – отрицательное число

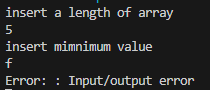


Рисунок 21 – Вывод программы, когда минимум массива – буква

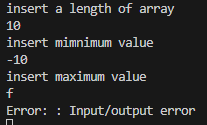


Рисунок 22 – Вывод программы, когда максимум массива – буква

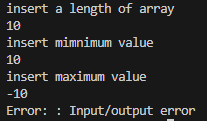


Рисунок 23 – Вывод программы, когда минимум массива больше максимума

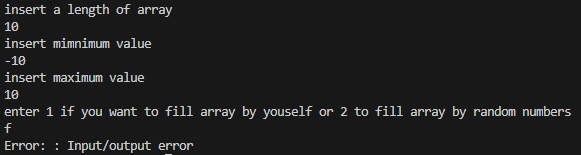


Рисунок 24 – Вывод программы, когда выбор способа заполнения массива – буква

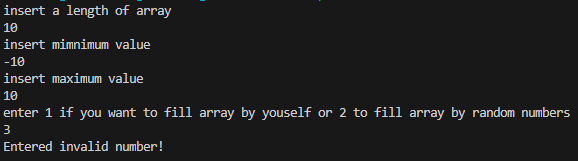


Рисунок 25 – Вывод программы, когда выбор способа заполнения массива не входит в указанные значения

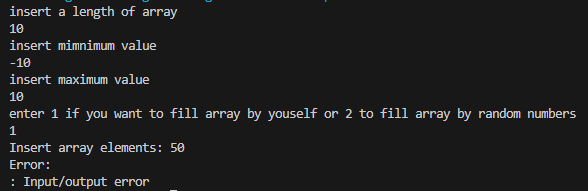


Рисунок 26 – Вывод программы, когда пользовательский ввод элемента массива не входит в указанный промежуток

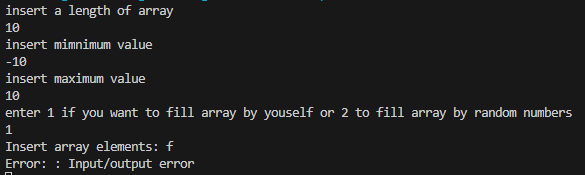


Рисунок 27 – Вывод программы, когда пользовательский ввод элемента массива – буква

* 1. Зачёт задания в GitHub

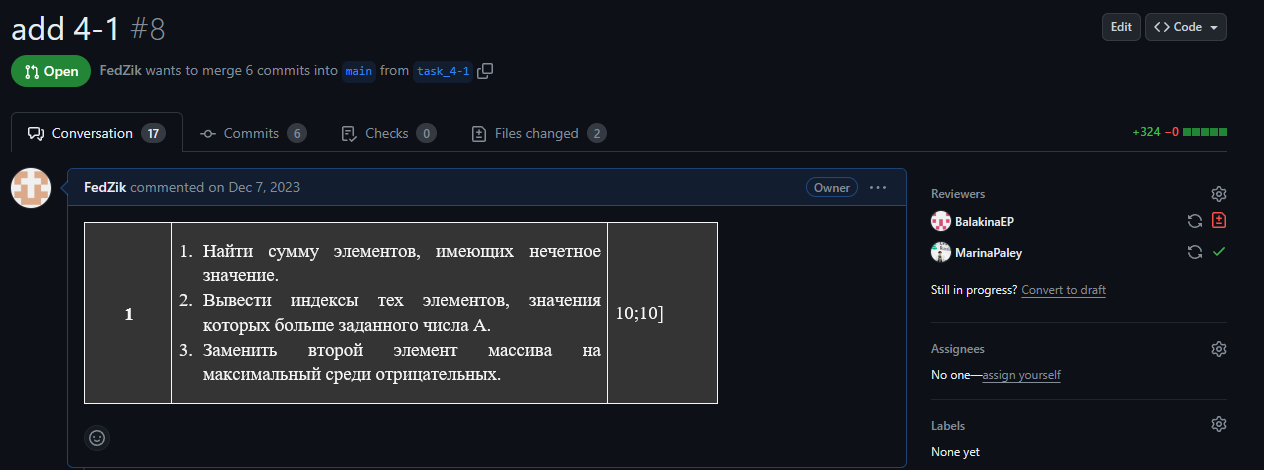


Рисунок 28 – Зачёт задания в GitHub

# Задание 4.2

* 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.

|  |  |  |
| --- | --- | --- |
| Вариант | Задачи | Интервал |
| 1 | 1. Заменить второй элемент массива на максимальный среди отрицательных. 2. Вставить число К перед всеми элементами, в которых есть цифра 1. 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 = Di / i. | [-10;10] |

* 1. Блок-схема алгоритма

Блок-схема алгоритма представлена на рисунках (Рисунок 29, Рисунок 30, Рисунок 31, Рисунок 32, Рисунок 33, Рисунок 34, Рисунок 35, Рисунок 36, Рисунок 37, Рисунок 38, Рисунок 39, Рисунок 40, Рисунок 41, Рисунок 42, Рисунок 43, Рисунок 44, Рисунок 45, Рисунок 46, Рисунок 47)

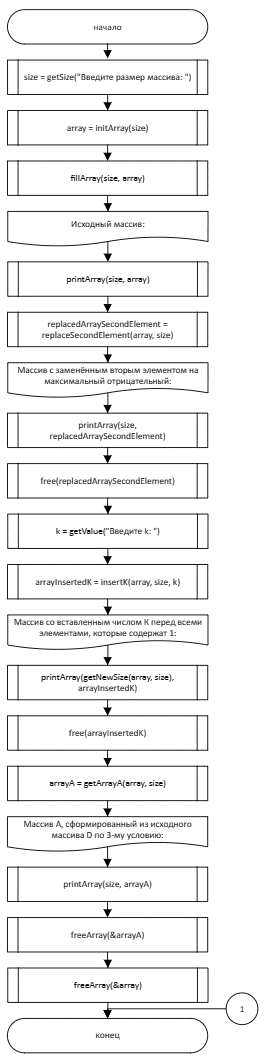


Рисунок 29 – Блок-схема функции main()

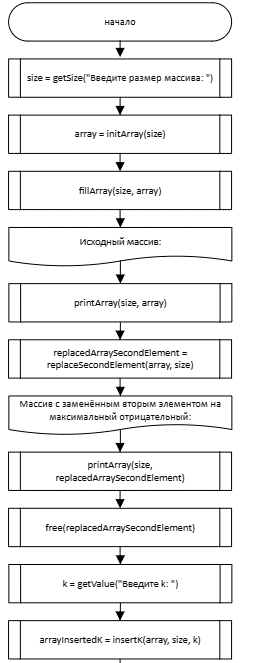


Рисунок 30 – Блок-схема функции main()(1 часть)

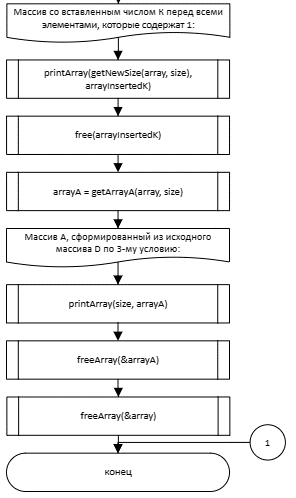


Рисунок 31 – Блок-схема функции main()(2 часть)

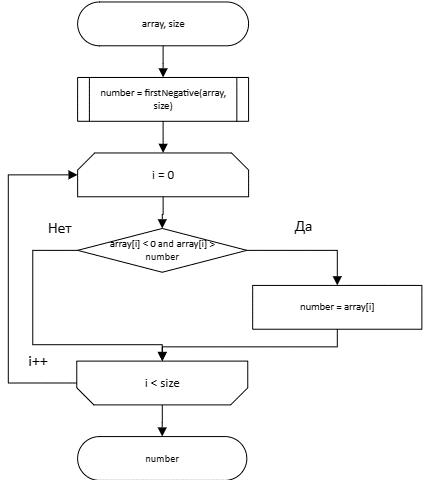


Рисунок 32 – Блок-схема функции maxNegative(array, size)

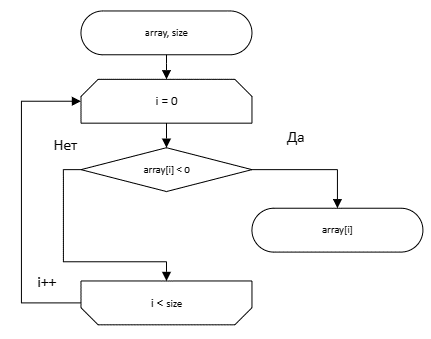


Рисунок 33 – Блок-схема функции firstNegative(array, size)

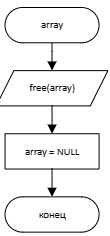


Рисунок 34 – Блок-схема функции freeArray(array)

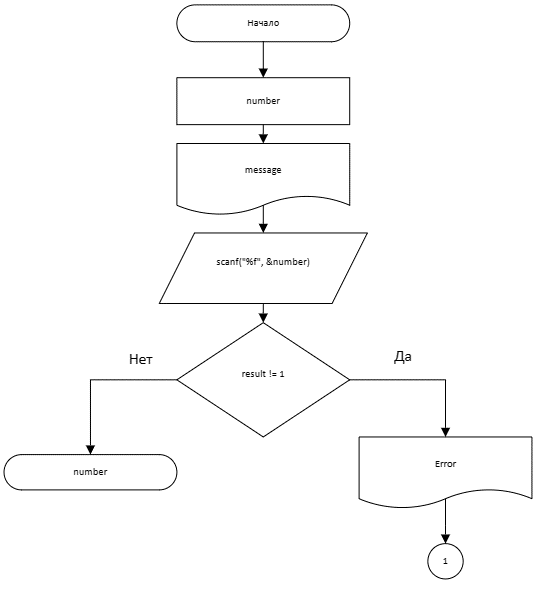


Рисунок 35 – Блок-схема функции getValue(message)

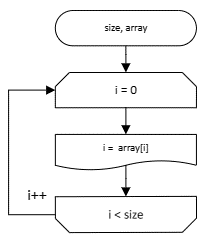


Рисунок 36– Блок-схема функции printArray(size, array)

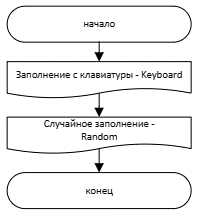


Рисунок 37– Блок-схема функции userChoice()

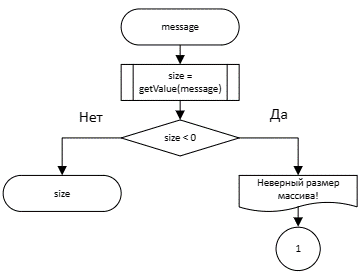


Рисунок 38– Блок-схема функции getSize(message)

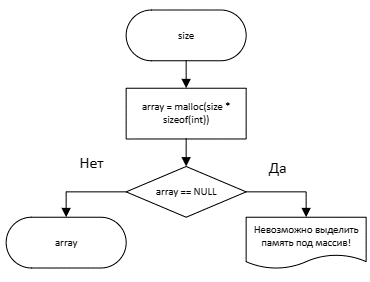


Рисунок 39– Блок-схема функции initArray(size)

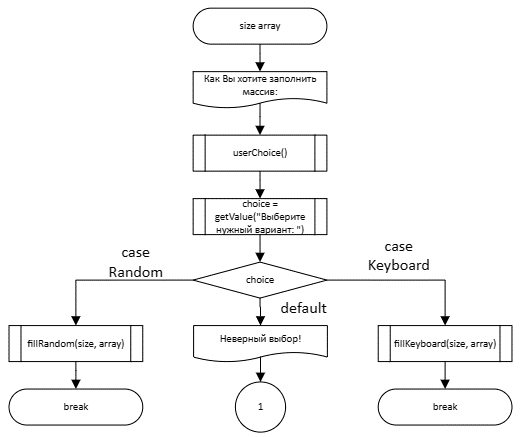


Рисунок 40– Блок-схема функции fillArray(size, array)

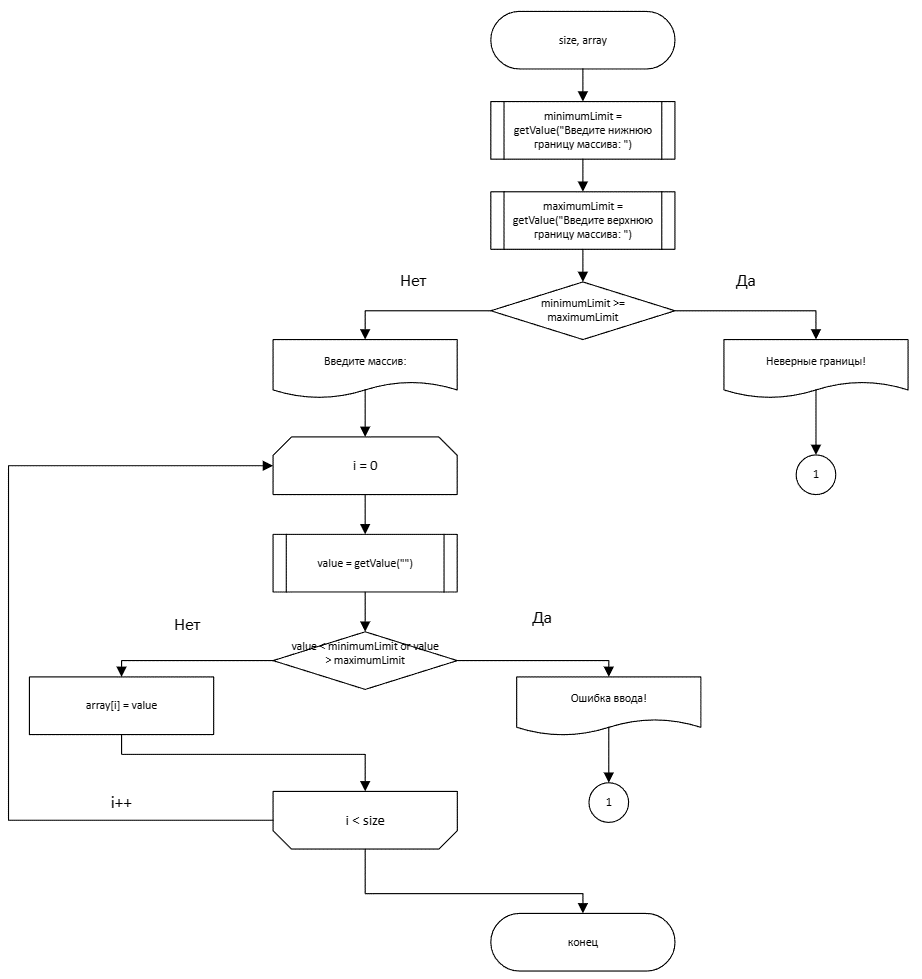


Рисунок 41– Блок-схема функции fillKeyboard(size, array)

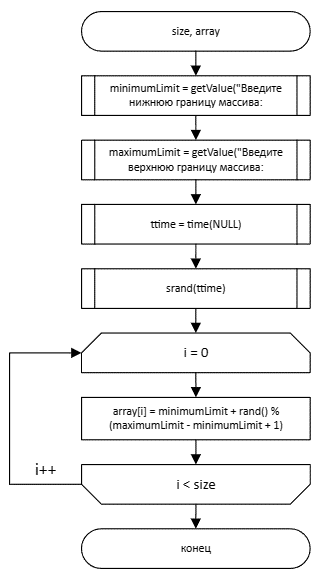


Рисунок 42– Блок-схема функции fill Random (size, array)

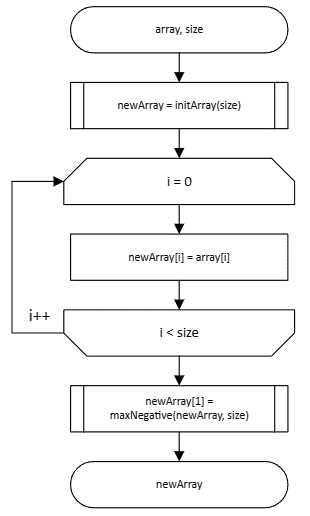


Рисунок 43– Блок-схема функции replaceSecondElement(array, size)

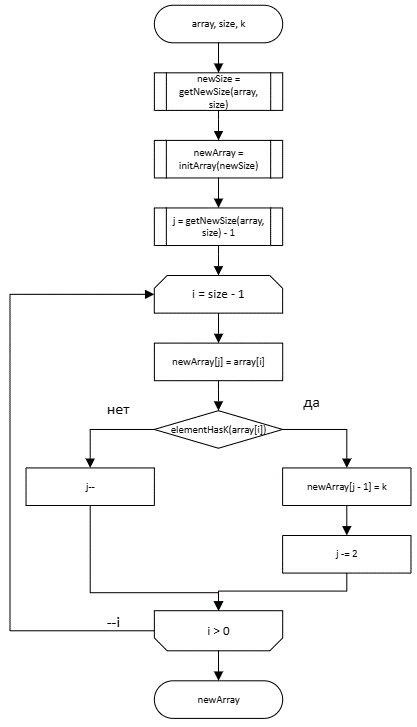


Рисунок 44– Блок-схема функции insertK(array, size, k)

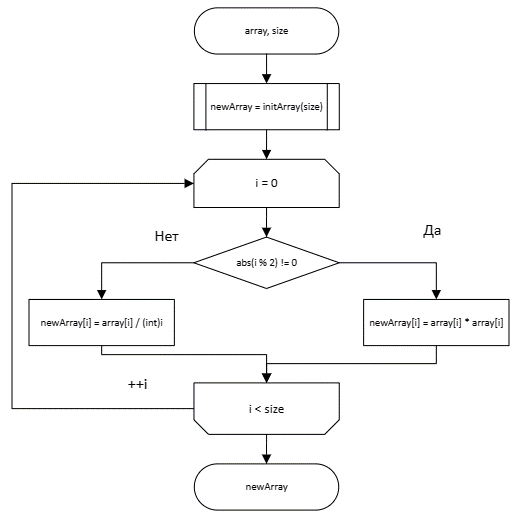


Рисунок 45– Блок-схема функции getArrayA(array, size)

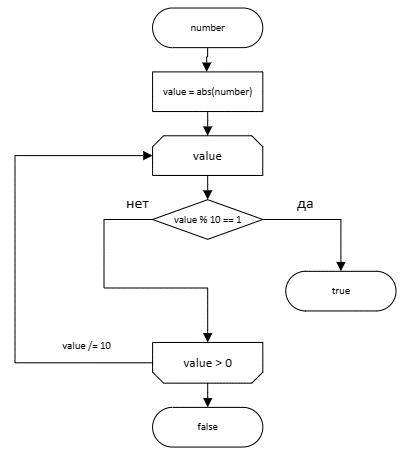


Рисунок 46– Блок-схема функции elementHasK(number)

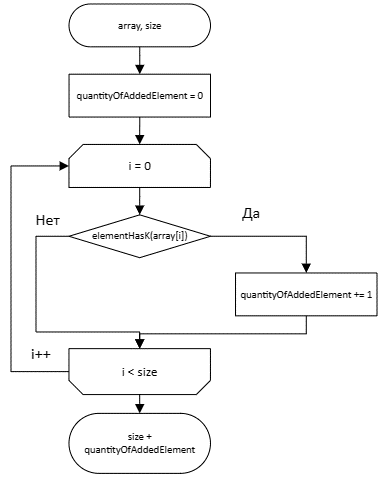


Рисунок 47– Блок-схема функции getNewSize(array, size)

* 1. Код задания 4.2

#include <memory.h>

#include <stdlib.h>

#include <errno.h>

#include <time.h>

#include <stdio.h>

#include <stdbool.h>

#include <locale.h>

/\*\*

\* @brief Функция выводит вариант выбора: Клавиатура или Рандом

\*/

void userChoise();

/\*\*

\* @brief Пользовательский ввод

\*/

enum randomOrKeyboard

{

Keyboard = 1,

Random = 2

};

/\*\*

\* @brief Функция принимающая и проверяющая значение размера на ввод

\* @param message - текст сообщения для пользователя

\* @return Значение размера

\*/

size\_t getSize(char const\* message);

/\*\*

\* @brief Функция выделяющая память под массив

\* @param size - размер массива

\* @return Массив

\*/

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

/\*\*

\* @brief Функция запалняет массив случайными числами или числами с клавиатуры в зависимости от выбора пользователя

\* @param size - размер массива

\* @param array - массив

\*/

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

/\*\*

\* @brief Функция считывающая ввод с клавиатуры

\* @param size - размер массива

\* @param array - массив

\*/

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

/\*\*

\* @brief Функция заполняющая массив рандомными числами

\* @param size - размер массива

\* @param array - массив

\* @return 1 если все хорошо

\*/

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

/\*\*

\* @brief Функция выводящая заполненный массив

\* @param size - размер массива

\* @param array - массив

\*/

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

/\*\*

\* @brief Функция принимающая и проверяющая значение на ввод

\* @param message - текст сообщения для пользователя

\* @return Значение

\*/

int getValue(const char\* message);

/\*\*

\* @brief Функция заменяющая второй элемент на максимальный отрицательный

\* @param array - целочисленный массив

\* @param size - размер

\* @return Возвращает изменёный массив

\*/

int\* replaceSecondElement(const int\* const array, size\_t size);

/\*\*

\* @brief Функция вставляющающая k после всех элементов кратную своему номеру

\* @param array - целочисленный массив

\* @param size - размер

\* @param k - элемент

\* @return Возвращает изменёный массив

\*/

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

/\*\*

\* @brief Функция формирует массив из исходного по условию 3 задания

\* @param array - целочисленный массив

\* @param size - размер

\* @return Возвращает новый массив A

\*/

int\* getArrayA(const int\* const array, size\_t size);

/\*\*

\* @brief Поиск максимального отрицательного элемента

\* @param array - целочисленный массив

\* @param size - размер

\* @return Возвращает максимальный отрицательный элемент

\*/

int maxNegative(const int\* const array, size\_t size);

/\*\*

\* @brief Поиск первого отрицательного элемента

\* @param array - целочисленный массив

\* @param size - размер

\* @return Возвращает первый отрицательный элемент

\*/

int firstNegative(const int\* const array, size\_t size);

/\*\*

\* @brief получение размера нового массива

\* @param array - целочисленный массив

\* @param size - размер

\* @return Возвращает размер нового массива

\*/

size\_t getNewSize(const int\* array, size\_t size);

/\*\*

\* @brief проверка числа на наличие 1

\* @param array - целочисленный массив

\* @param size - размер

\* @return Возвращает истину в случае успешной проверки

\*/

bool elementHasK(int number);

/\*\*

\* @brief освобождение массива

\* @param array указатель на массив

\*/

void freeArray(int\*\* array);

/\*\*

\* @brief Точка входа в программу

\* @return Возврящает 0, если программа работает верно, иначе 1

\*/

int main()

{

setlocale(LC\_ALL, "RU");

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

int\* array = initArray(size);

fillArray(size, array);

puts("Исходный массив:\n");

printArray(size, array);

int\* replacedArraySecondElement = replaceSecondElement(array, size);

puts("Массив с заменённым вторым элементом на максимальный отрицательный:\n");

printArray(size, replacedArraySecondElement);

free(replacedArraySecondElement);

int k = getValue("Введите k: ");

int\* arrayInsertedK = insertK(array, size, k);

puts("Массив со вставленным числом К перед всеми элементами, которые содержат 1:\n");

printArray(getNewSize(array, size), arrayInsertedK);

free(arrayInsertedK);

int\* arrayA = getArrayA(array, size);

puts("Массив A, сформированный из исходного массива D по 3-му условию:\n");

printArray(size, arrayA);

freeArray(&arrayA);

freeArray(&array);

return 0;

}

void userChoise()

{

printf("Заполнение с клавиатуры - %d\n", (int)Keyboard);

printf("Случайное заполнение - %d\n", (int)Random);

}

size\_t getSize(char const\* message)

{

int size = getValue(message);

if (size < 0) {

perror("Неверный размер массива!\n");

abort();

}

return (size\_t)size;

}

int\* initArray(const size\_t size)

{

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

if (array == NULL)

{

perror("Невозможно выделить память под массив!\n");

}

return array;

}

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

{

puts("Как Вы хотите заполнить массив:\n");

userChoise();

enum randomOrKeyboard choice = getValue("Выберите нужный вариант: ");

switch (choice)

{

case Keyboard:

fillKeyboard(size, array);

break;

case Random:

fillRandom(size, array);

break;

default:

perror("Неверный выбор!!\n");

}

}

void fillKeyboard(const size\_t size, int\* array)

{

const int minimumLimit = getValue("Введите нижнюю границу массива: ");

const int maximumLimit = getValue("Введите верхнюю границу массива: ");

if (minimumLimit >= maximumLimit)

{

errno = EIO;

perror("Неверные границы!\n");

abort();

}

puts("Введите массив: ");

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

{

int value = getValue("");

if (value < minimumLimit || value > maximumLimit)

{

errno = EIO;

perror("Ошибка ввода!\n");

abort();

}

array[i] = value;

}

}

void fillRandom(const size\_t size, int\* array)

{

const int minimumLimit = getValue("Введите нижнюю границу массива: ");

const int maximumLimit = getValue("Введите верхнюю границу массива: ");

unsigned int ttime = time(NULL);

srand(ttime);

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

{

array[i] = minimumLimit + rand() % (maximumLimit - minimumLimit + 1);

}

}

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

{

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

{

printf("%lu\t%d\n", i, array[i]);

}

}

int getValue(const char\* message)

{

int value;

printf("%s", message);

int result = scanf\_s("%d", &value);

if (result != 1)

{

errno = EIO;

perror("Неверное значение\n");

abort();

}

return value;

}

int\* replaceSecondElement(const int\* const array, size\_t size)

{

int\* newArray = initArray(size);

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

{

newArray[i] = array[i];

}

newArray[1] = maxNegative(newArray, size);

return newArray;

}

int\* insertK(const int\* const array, size\_t size, int k)

{

const size\_t newSize = getNewSize(array, size);

int\* newArray = initArray(newSize);

size\_t j = getNewSize(array, size) - 1;

for (size\_t i = size - 1 ; i > 0; --i)

{

newArray[j] = array[i];

if (elementHasK(array[i]))

{

newArray[j - 1] = k;

j -= 2;

}

else

{

j--;

}

}

return newArray;

}

int\* getArrayA(const int\* const array, size\_t size)

{

int\* newArray = initArray(size);

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

{

if (abs(i % 2) != 0)

{

newArray[i] = array[i] \* array[i];

}

else

{

newArray[i] = array[i] / (int)i;

}

}

return newArray;

}

int maxNegative(const int\* const array, size\_t size)

{

int number = firstNegative(array, size);

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

{

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

{

number = array[i];

}

}

return number;

}

int firstNegative(const int\* array, size\_t size)

{

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

{

if (array[i] < 0)

{

return array[i];

}

}

return 0;

}

bool elementHasK(int number)

{

int value = abs(number);

while (value > 0)

{

if (value % 10 == 1)

{

return true;

}

value /= 10;

}

return false;

}

size\_t getNewSize(const int\* array, size\_t size)

{

size\_t quantityOfAddedElement = 0;

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

{

if (elementHasK(array[i]))

{

quantityOfAddedElement += 1;

}

}

return size + quantityOfAddedElement;

}

void freeArray(int\*\* array)

{

free(array);

array = NULL;

}

* 1. Решение тестового примера

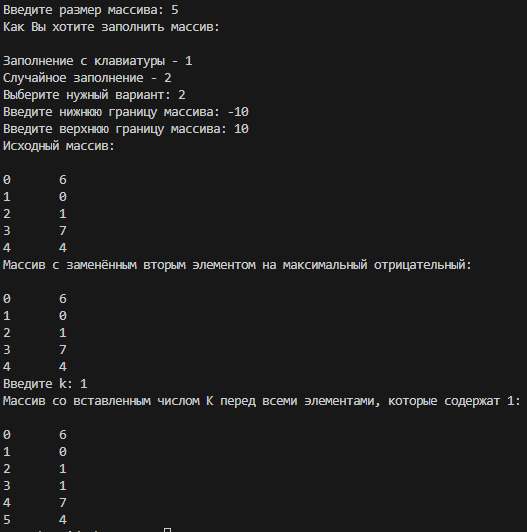


Рисунок 48 – Решение тестового примера рандомным заполнением массива

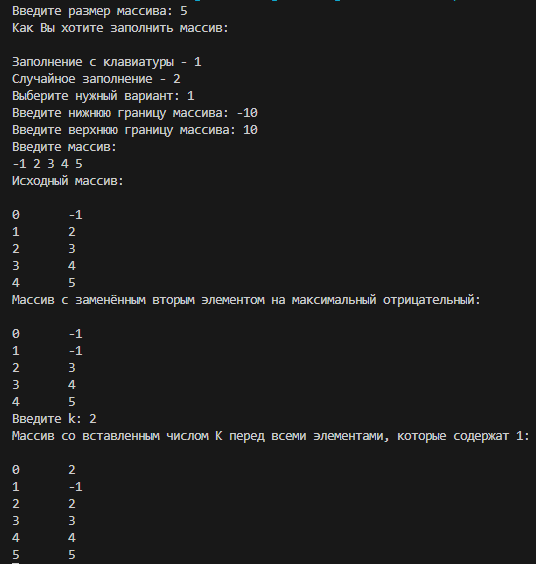


Рисунок 49 – Решение тестового примера пользовательским заполнением массива



Рисунок 50 – Вывод программы, когда размер массива отрицательный



Рисунок 51 – Вывод программы, когда размер массива – буква

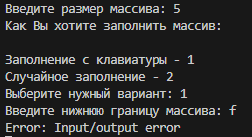


Рисунок 52 – Вывод программы, когда минимум массива – буква

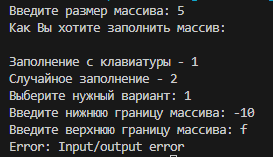


Рисунок 53– Вывод программы, когда максимум массива – буква

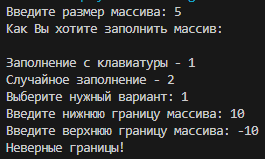


Рисунок 54– Вывод программы, когда максимум массива меньше минимума

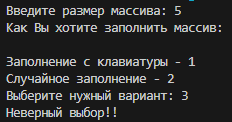


Рисунок 55 – Вывод программы, когда выбор способа заполнения массива не соответствует указанным значениям

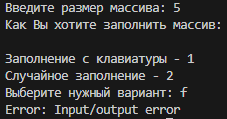


Рисунок 56 – Вывод программы, когда выбор способа заполнения массива – буква

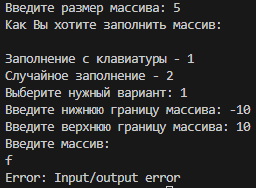


Рисунок 57 – Вывод программы, когда элемент массива при пользовательском вводе – буква

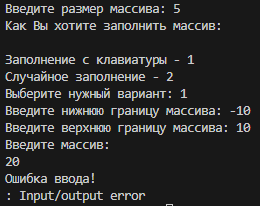


Рисунок 58 – Вывод программы, когда элемент массива при пользовательском вводе не входит промежуток

* 1. Зачёт задания в GitHub

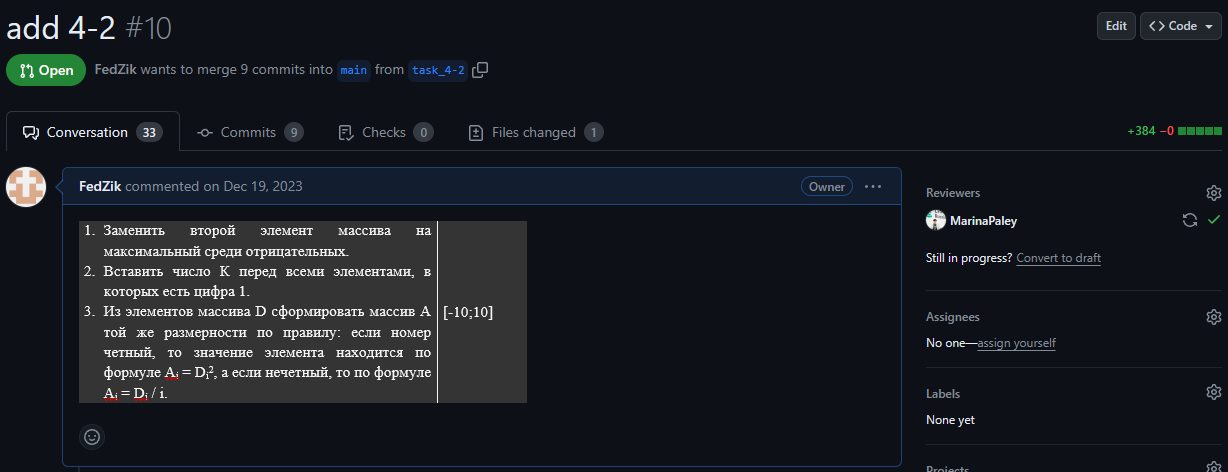


Рисунок 59 – Зачёт задания в GitHub