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 10

Выполнил: ст. гр. ТКИ-142

Левин Всеволод Максимович

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

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

Москва 2023

Оглавление

[1 Решение задачи 4–1 3](#_Toc156871751)

[1.1 Формулировка задачи 4–1 3](#_Toc156871752)

[1.2 Блок-схема алгоритма задание 4–1 4](#_Toc156871753)

[1.3 Текст программы на языке С задание 4–1 7](#_Toc156871754)

[1.4 Результаты выполнения программы 4–1 13](#_Toc156871755)

[Отметка о выполнении задания в веб-хостинге системы контроля версий задание 4–1 17](#_Toc156871756)

1. 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  – Исходные данные

|  |  |  |
| --- | --- | --- |
| Вариант | Задачи | Интервал |
| 10 | **1**. Найти сумму отрицательных элементов.  **2.** Найти количество тех элементов, значения которых положительны и не превосходят заданного числа А.  **3.** Найти номер последней пары соседних элементов с разными знаками. | [-100;200] |

1.2 Блок-схема алгоритма задание 4–1

Блок-схема основного алгоритма представлена ниже (Рисунок 1). Блок-схемы функций рис. (2-7)

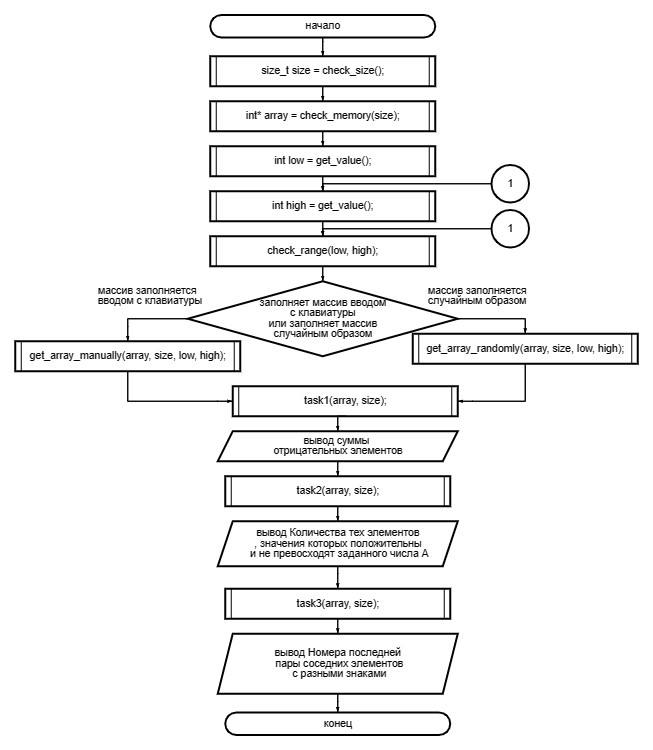


Рисунок  ­ Блок-схема основного алгоритма

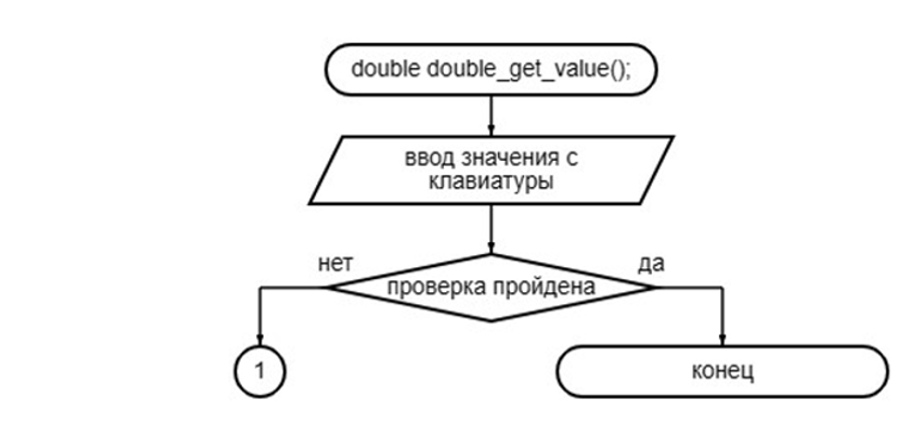


Рисунок  – Блок-схема используемых функций

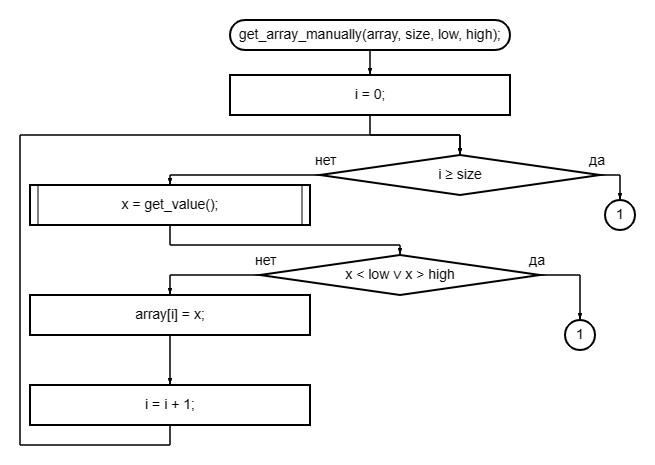


Рисунок 3 – Блок-схема используемых функций

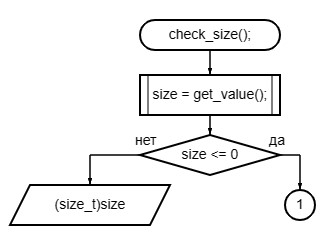


Рисунок 4 – Блок-схема используемых функций

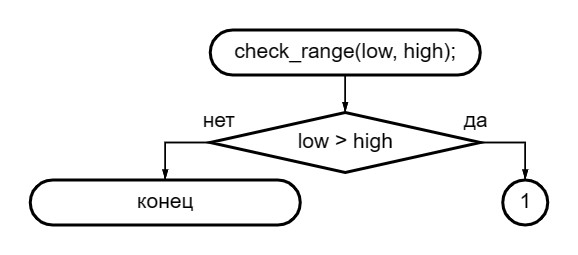


Рисунок 5 – Блок-схема используемых функций

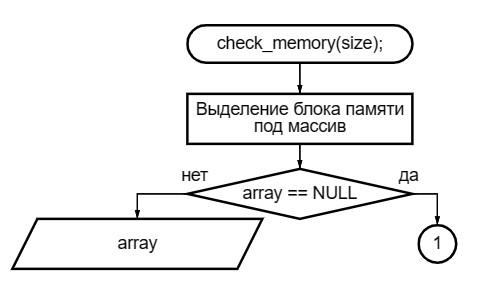


Рисунок 6 – Блок-схема используемых функций

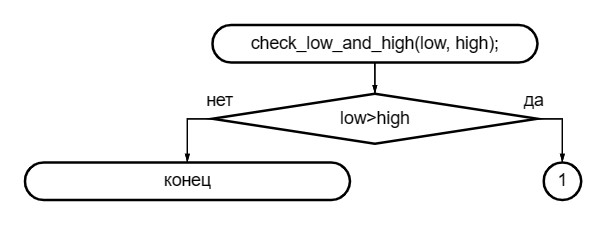


Рисунок 7 – Блок-схема используемых функций

1.3 Текст программы на языке С задание 4–1

#include <stdio.h>

#include <errno.h>

#include <stdlib.h>

#include <time.h>

#include <locale.h>

/\*\*

\* @brief Способы заполнения массива.

\*/

enum array\_input

{

/\*\*

\* @brief Заполнение массива вручную.

\*/

MANUAL = 1,

/\*\*

\* @brief Заполнение массива случайными числами.

\*/

RANDOM = 2

};

/\*\*

\* @breaf проверяет правильность ввода данных

\* @return возвращает результат в случае успеха

\*/

int get\_value();

/\*\*

\* @breaf проверяет правильность ввода размера массива

\* @return возвращает размер массива в случае успеха

\*/

int check\_size();

/\*\*

\* @brief Функция для считывания элементов массива с клавиатуры.

\* @param array - массив

\* @param size - длинна массива

\* @param low - нижняя граница массива

\* @param high - верхняя граница массива

\*/

void get\_array\_manually(int\* array, const size\_t size, const int low, const int high);

/\*\*

\* @brief Функция для заполнения массива случайными числами.

\* @param array - массив

\* @param size - длинна массива

\* @param low - нижняя граница массива

\* @param high - верхняя граница массива

\*/

void get\_array\_randomly(int\* array, const size\_t size, const int low, const int high);

/\*\*

\* @brief Функция для вычисления задания 1

\* @param array - массив

\* @param size - длинна массива

\* @return Возвращает ответ на задание 1

\*/

int task1(const int\* array, const size\_t size);

/\*\*

\* @brief Функция для вычисления задания 2

\* @param array - массив

\* @param size - длинна массива

\* @return Возвращает ответ на задание 2

\*/

int task2(const int\* array, const size\_t size);

/\*\*

\* @brief Функция для вычисления задания 3

\* @param array - массив

\* @param size - длинна массива

\* @return Возвращает ответ на задание 3

\*/

size\_t task3(const int\* array, const size\_t size);

/\*\*

\* @brief Выводит сообщение об ошибке и завершает программу.

\*/

void errors();

/\*\*

\* @brief - проверка на правильность ввода границ массива

\* @param low - нижняя граница массива

\* @param high -верхняя граница массива

\*/

void check\_range(const int low, const int high);

/\*\*

\* @brief - проверка на выделение памяти для динамического массива

\* @param size – размер массива

\*/

int\* check\_memory(const size\_t size);

/\*\*

\* @brief Точка входа в программу.

\* @return Возвращает 0 в случае успеха.

\*/

int main()

{

setlocale(LC\_ALL, "RU");

printf("Введите размер массива: ");

size\_t size = check\_size();

int\* array = check\_memory(size);

printf("Введите нижнюю границу массива: ");

int low = get\_value();

printf("Введите верхнюю границу массива: ");

int high = get\_value();

check\_range(low, high);

printf("Как Вы хотите заполнить массив?\n");

printf("%d - Самостоятельно\n", MANUAL);

printf("%d - Случайным образом\n", RANDOM);

int choose = get\_value();

enum array\_input user\_input = (enum array\_input)choose;

switch(user\_input)

{

case MANUAL:

get\_array\_manually(array, size, low, high);

break;

case RANDOM:

get\_array\_randomly(array, size, low, high);

break;

default:

errors();

return 1;

break;

}

printf("1) Сумма отрицательных элементов равна: ");

printf("%d\n", task1(array, size));

printf("Введите значение А для задания 2: ");

printf("2) Количество тех элементов, значения которых положительны и не превосходят заданного числа А: %d", task2(array, size));

if (task3(array, size) == 0)

{

printf("\n3) Требуемых пар не нашлось.");

}

else

{

printf("\n3) Номер последней пары соседних элементов с разными знаками: %zu", task3(array, size));

}

if (array != NULL)

{

free(array);

}

return 0;

}

int get\_value()

{

int result;

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

if (scanf\_result!=1)

{

errors();

}

return result;

}

int check\_size()

{

int size = get\_value();

if (size <= 0)

{

errors();

}

return (size\_t) size;

}

void get\_array\_manually(int\* array, const size\_t size, const int low, const int high)

{

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

{

printf("Введите число: ");

int x = get\_value();

if (x < low || x > high)

{

errors();

}

array[i] = x;

}

}

void get\_array\_randomly(int\* array, const size\_t size, const int low, const int high)

{

srand(time(NULL));

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

{

array[i] = rand() % (high - low + 1) + low;

}

}

int task1(const int\* array, const size\_t size)

{

int sum = 0;

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

{

if (array[i] < 0)

{

sum += array[i];

}

}

return sum;

}

int task2(const int\* array, const size\_t size)

{

int a = get\_value();

int count = 0;

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

{

if ((array[i] > 0) && (a >= array[i]))

{

count++;

}

}

return count;

}

size\_t task3(const int\* array, const size\_t size)

{

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

{

if ((array[i] \* array[i - 1]) < 0)

{

return i;

}

}

return 0;

}

void errors()

{

errno = EIO;

perror("Error");

abort();

}

void check\_range(const int low, const int high)

{

if (low > high)

{

errors();

}

}

int\* check\_memory(const size\_t size)

{

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

if (array == NULL)

{

perror("Ошибка выделения памяти");

exit(errno);

}

return array;

}

* 1. Результаты выполнения программы 4–1

Результаты выполнения программы в C представлена ниже (Рисунок 8-15).

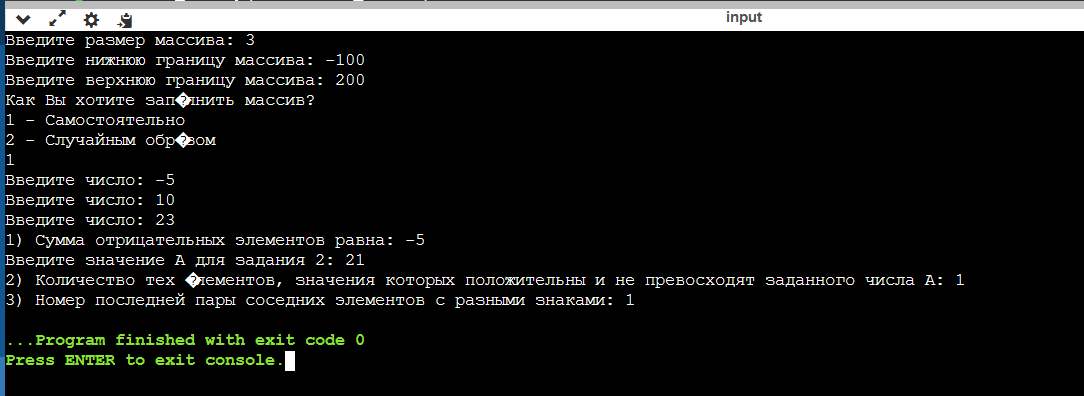


Рисунок 8 – Результаты выполнения программы

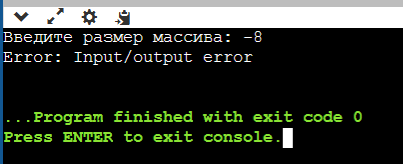


Рисунок 9 – Результаты выполнения программы если размер массива = -8

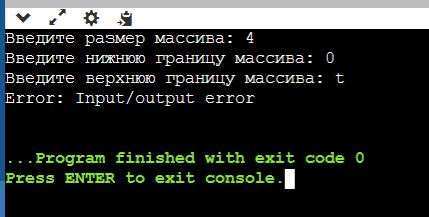


Рисунок 10 – Результаты выполнения программы если верхняя граница массива = t

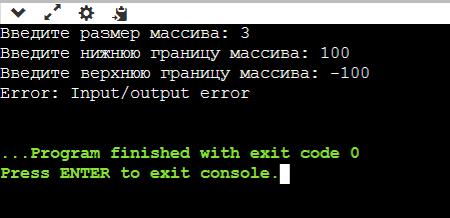


Рисунок 11 – Результаты выполнения программы если (нижняя граница массива) > (верхняя граница массива)

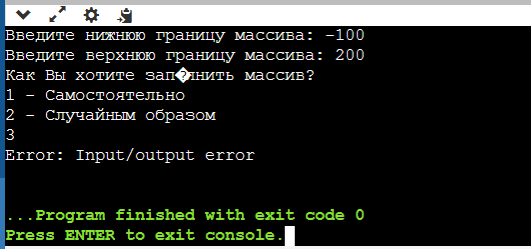


Рисунок 12 – Результаты выполнения программы при неправильном вводе пути заполнения массива

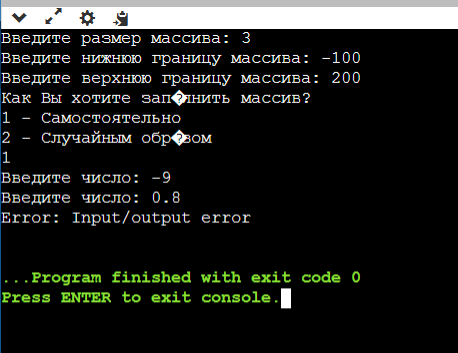


Рисунок 13 – Результаты выполнения программы если элемент массива = 0.8

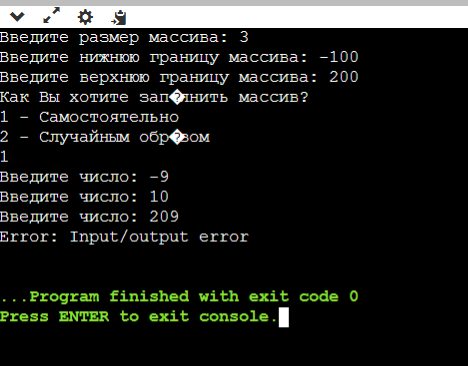


Рисунок 14 – Результаты выполнения программы если (элемент массива) > (верхней границы массива)

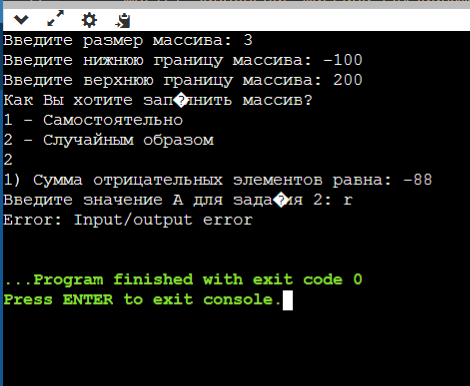


Рисунок 15 – Результаты выполнения программы если A = r

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

Ниже представлено доказательство того, что задание 4–1 было принято. (Рисунок 16)

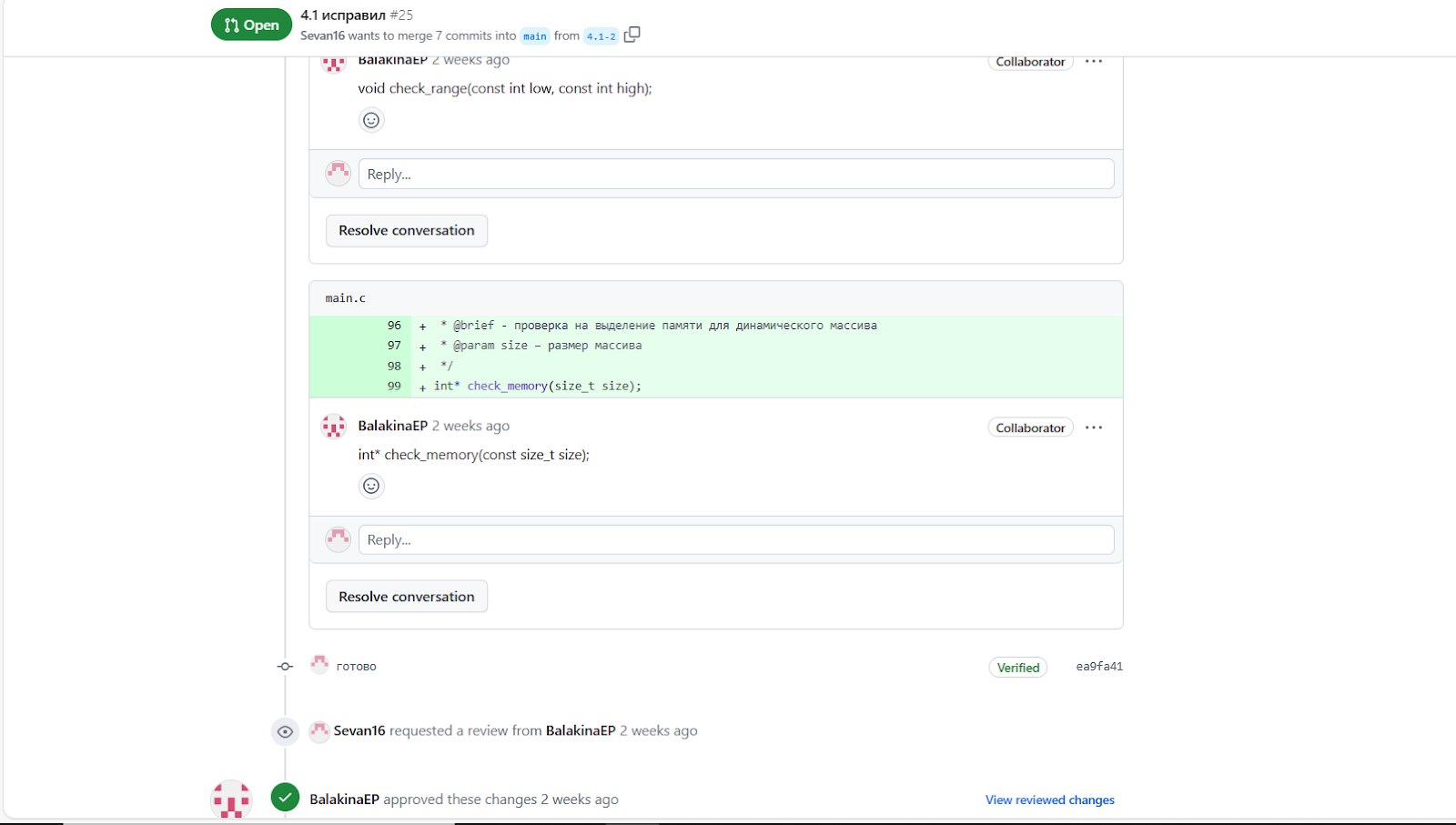


Рисунок 16 – Approve задачи 4–1