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 6

Выполнил: ст. гр. ТКИ-141

Винтфельд Рина Дмитриевна

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

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

Москва 2023

Оглавление

[1 Задача 4.1 3](#_Toc153788792)

[1.1 Формулировка задания 4.1 3](#_Toc153788793)

[1.2 Блок-схема алгоритма 4.1 4](#_Toc153788794)

[1.3 Текст программы на языке C 4.1 5](#_Toc153788795)

[1.4 Результаты выполнения программы 4.1 15](#_Toc153788796)

[1.5 Выполнение тестовых примеров 4.1 16](#_Toc153788797)

[1.6 Отметка о выполнении задания в веб-хостинге системы контроля версий 4.1 17](#_Toc153788798)

[2 Задача 4.2 18](#_Toc153788799)

[2.1 Формулировка задания 4.2 18](#_Toc153788800)

[2.2 Блок-схема алгоритма 4.2 19](#_Toc153788801)

[2.3 Текст программы на языке C 4.2 23](#_Toc153788802)

[2.4 Результаты выполнения программы 4.2 36](#_Toc153788803)

[2.5 Выполнение тестовых примеров 4.2 37](#_Toc153788804)

[2.6 Отметка о выполнении задания в веб-хостинге системы контроля версий 4.2 38](#_Toc153788805)

[3 Задача 4.3 39](#_Toc153788806)

[3.1 Формулировка задания 4.3 39](#_Toc153788807)

[3.2 Блок-схема алгоритма 4.3 40](#_Toc153788808)

[3.3 Текст программы на языке C 4.3 44](#_Toc153788809)

[3.4 Результаты выполнения программы 4.3 47](#_Toc153788810)

[3.5 Выполнение тестовых примеров 4.3 48](#_Toc153788811)

[3.6 Отметка о выполнении задания в веб-хостинге системы контроля версий 4.3 49](#_Toc153788812)

1. Задача 4.1
   1. Формулировка задания 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 |
| 6 | 1. Найти сумму отрицательных элементов, значения которых кратно10. 2. Заменить первые k элементов массива на те же элементы в обратном порядке. 3. Определить, есть ли пара соседних элементов с произведением, равным заданному числу. | [-1000;1000] |

* 1. Блок-схема алгоритма 4.1

Блок-схема основного алгоритма представлена ниже (Рисунок 1). Блок-схемы используемых функций представлены ниже ().

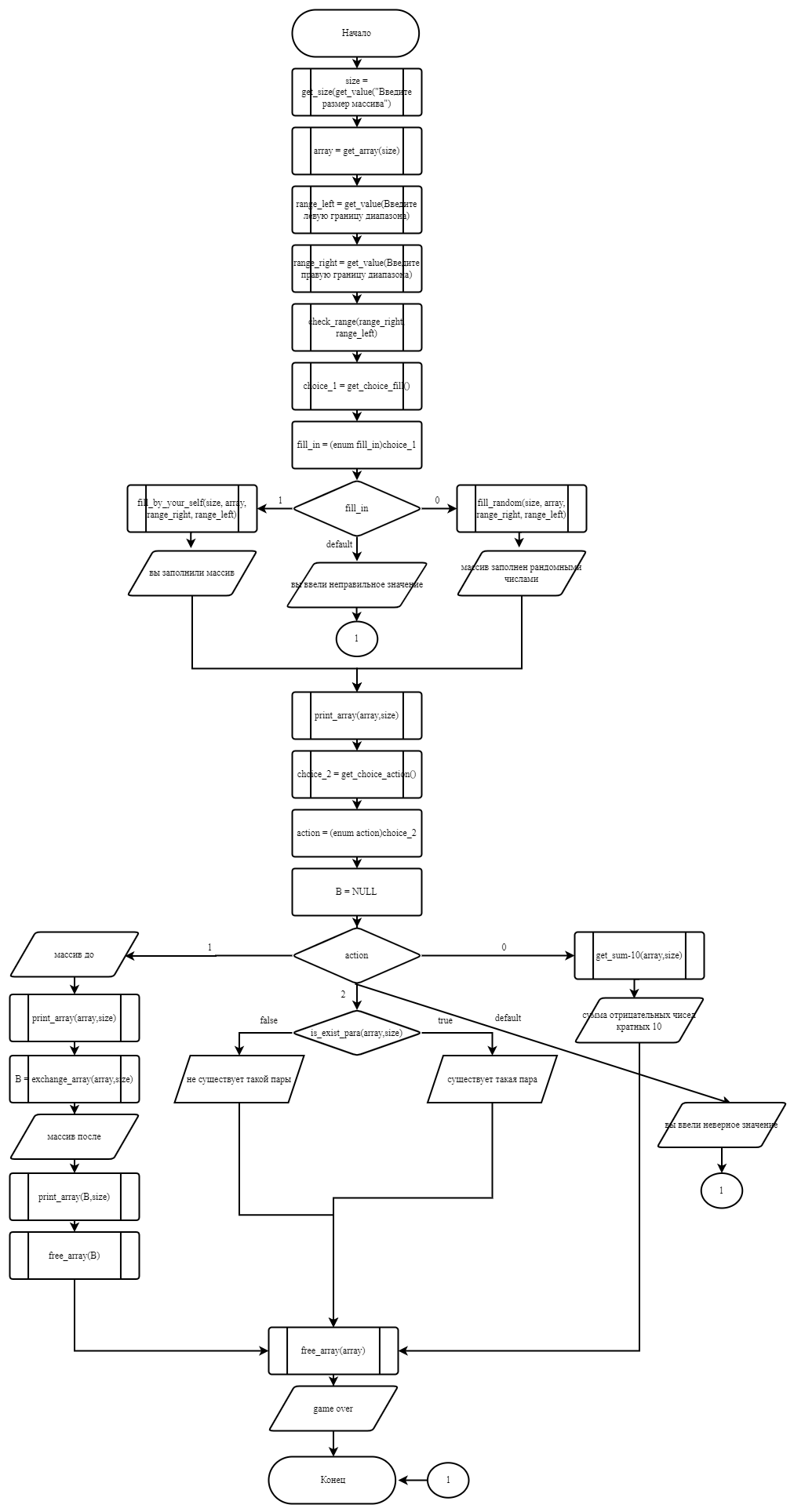
****

Рисунок – Блок-схема основного алгоритма

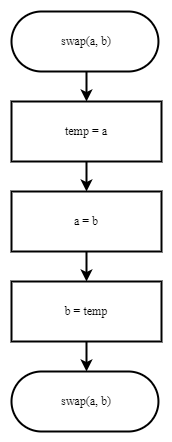


Рисунок – Блок-схема используемой функции

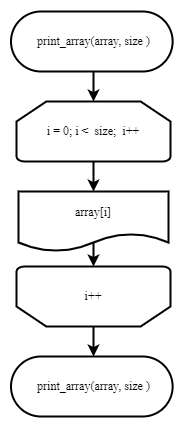


Рисунок - Блок-схема используемой функции

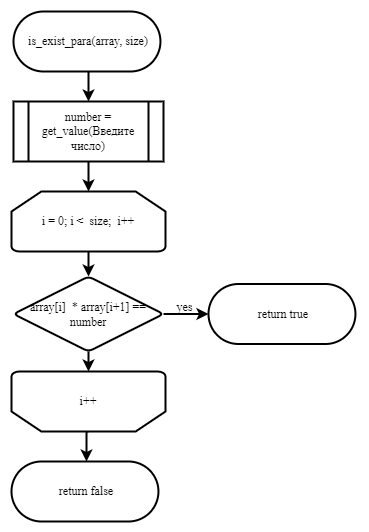


Рисунок - Блок-схема используемой функции



Рисунок - Блок-схема используемой функции

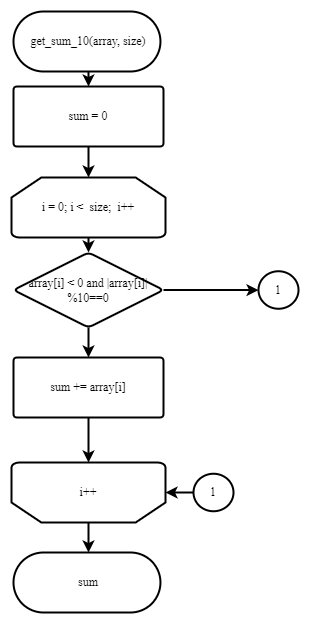


Рисунок - Блок-схема используемой функции



Рисунок - Блок-схема используемой функции



Рисунок - Блок-схема используемой функции



Рисунок - Блок-схема используемой функции

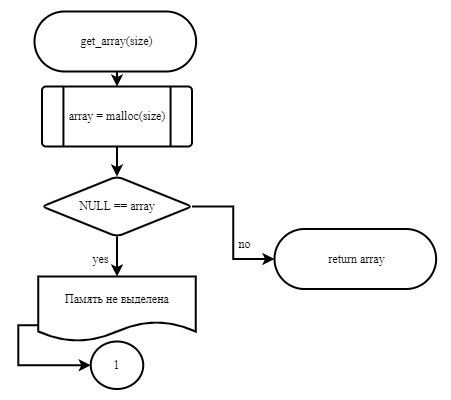


Рисунок - Блок-схема используемой функции

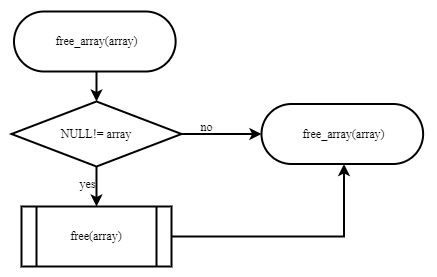


Рисунок - Блок-схема используемой функции

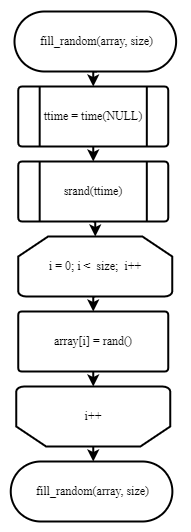


Рисунок - Блок-схема используемой функции

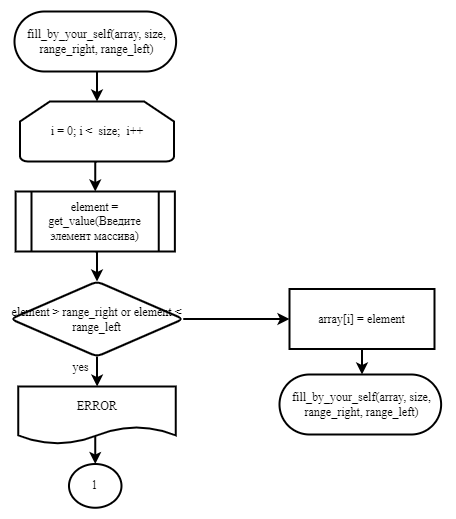


Рисунок - Блок-схема используемой функции

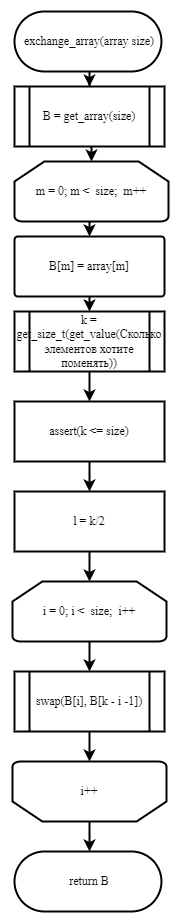


Рисунок - Блок-схема используемой функции

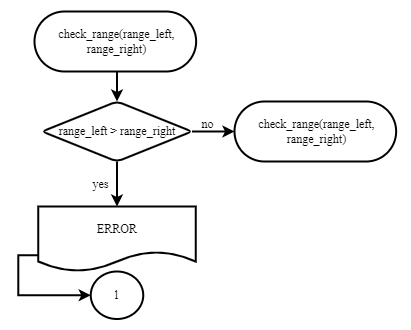


Рисунок - Блок-схема используемой функции

* 1. Текст программы на языке C 4.1

#include <stdio.h>

#include <stdlib.h>

#include <errno.h>

#include <time.h>

#include <stdbool.h>

#include <assert.h>

/\*

\* @brief Выбор пользователя

\* @brief fill\_random - Заполнение массива пользователем

\* @brief fill\_by\_my\_self - Заполнение массива рандомными числами

\*/

enum fill\_in{fill\_randomm, fill\_by\_my\_self};

/\*

\* @brief Выбор пользователя

\* @brief sum\_10 - Найти сумму отрицательных элементов, значения которых кратно 10

\* @brief exchange - Заменить первые k элементов массива на те же элементы в обратном порядке

\* @brief para - Определить, есть ли пара соседних элементов с произведением, pавным заданном числу

\*/

enum action {sum\_10, exchange, para};

/\*

\* @brief Считает сумму отрицательных чисел, кратных 10

\* @param array - массив

\* @param size - размер массива

\* @return sum - сумма отрицательных чисел, кратных 10

\*/

int get\_sum\_10(int\* array, size\_t size);

/\*

\* @brief Заняет первые k элементов на те же элементы, записанные в обратном порядке

\* @param array - массив

\* @param size - размер массива

\* @return изменённый массив

\*/

int\* exchange\_array(int\* array,size\_t size);

/\*

\* @brief меняет переменные

\* @param a - переменная

\* @param b - переменная

\*/

void swap(int\* a, int\* b);

/\*

\* @brief определяет существует ли такая пара элементов с произедением, равным заданному числу

\* @param array - массив

\* @param size - размер массива

\* @return True/False

\*/

bool is\_exist\_para(int\* array, size\_t size);

/\*

\* @brief предлагает выбор пользователю

\* @return введенный номер действия

\*/

int get\_choice\_fill();

/\*

\* @brief предлагает выбор пользователю

\* @return введенный номер действия

\*/

int get\_choice\_action();

/\*

\* @brief проверяет введенное значение

\* @param massage - сообщение для пользователя

\* @return переменную

\*/

int get\_value(const char\* massage);

/\*

\* @brief проверяет или введенное значение > 0

\* @param value введенное значение

\* @return размер массива

\*/

size\_t get\_size\_t(int value);

/\*

\* @brief проверяет или введенный диапазон соответствует действительности

\* @param range\_right - правая граница диапазона

\* @param range\_left - левая граница диапазона

\*/

void check\_range(int range\_right, int range\_left);

/\*

\* @brief создает массив

\* @param size - размер массива

\* @return массив

\*/

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

/\*

\* @brief Заполняет массив рандомными числами

\* @param size - размер массива

\* @param array - массив

\* @param range\_right - правая граница диапазона

\* @param range\_left - левая граница диапазона

\* @return 0 - в случае успеха

\*/

int fill\_random(const size\_t size, int\* array, int range\_right, int range\_left);

/\*

\* @brief Пользователь заполняет массив

\* @param size - размер массива

\* @param array - массив

\* @param range\_right - правая граница диапазона

\* @param range\_left - левая граница диапазона

\* @return 0 - в случае успеха

\*/

int fill\_by\_your\_self(const size\_t size, int\* array, int range\_right, int range\_left);

/\*

\* @brief чистит использованную память

\* @param array - массив

\*/

void free\_array(int\* array);

/\*

\* @brief печатает элементы массива

\* @param array - массив

\* @param size - размер массива

\*/

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

/\*

\* @brief Точка входа в программу

\* @return 0, в случае успешного завершения программы

\*/

int main() {

const size\_t size = get\_size\_t(get\_value("Введите размер массива\t"));

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

int range\_left = get\_value("\nВведите левую границу диапозона\t");

int range\_right = get\_value("\nВведите правую границу диапозона\t");

check\_range(range\_right, range\_left);

int choice\_1 = get\_choice\_fill();

enum fill\_in fill\_in = (enum fill\_in)choice\_1;

switch (fill\_in)

{

case (fill\_randomm):

fill\_random(size, array, range\_right, range\_left);

puts("\nМассив заполнен рандомными числами:\n");

break;

case (fill\_by\_my\_self):

fill\_by\_your\_self(size, array, range\_right, range\_left);

puts("\nВы заполнили массив:\n");

break;

default:

printf\_s("Вы ввели неправильное значение\n");

errno = EIO;

return 1;

}

print\_array(array, size);

int choice\_2 = get\_choice\_action();

enum action action = (enum action)choice\_2;

int\* B = NULL;

switch (action)

{

case (sum\_10):

printf\_s("Сумма отрицательных чисел, кратных 10 = %d\n", get\_sum\_10(array, size));

break;

case (exchange):

printf\_s("\nМассив до:\t");

print\_array(array, size);

B = exchange\_array(array, size);

printf\_s("\nМассив после:\t");

print\_array(B, size);

free\_array(B);

break;

case (para):

if (is\_exist\_para(array, size))

{

puts("\nСуществует такая пара элементов с произедением, равным заданному числу\n");

}

else

{

puts("\nНе существует такой пары элементов с произедением, равным заданному числу\n");

}

break;

default:

errno = EIO;

perror("Вы ввели неверное значение");

return 1;

}

free\_array(array);

puts("\nGame over");

return 0;

}

int get\_sum\_10(int\* array, size\_t size)

{

int sum = 0;

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

{

if (array[i] < 0 && abs(array[i]) % 10 == 0)

{

sum += array[i];

}

}

return sum;

}

int\* exchange\_array(int\* array, size\_t size)

{

int\* B = get\_array(size);

for (int m = 0; m < size; m++)

{

B[m] = array[m];

}

size\_t k = get\_size\_t(get\_value("\nСколко элементов хотите поменять?\t"));

assert(k <= size);

size\_t l = k / 2;

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

{

swap(&(B[i]), &(B[k - i - 1]));

}

return B;

}

void swap(int\* a, int\* b)

{

int temp = \*a;

\*a = \*b;

\*b = temp;

}

bool is\_exist\_para(int\* array, size\_t size)

{

int number = get\_value("Введите число\t");

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

{

if (array[i] \* array[i + 1] == number)

{

return true;

}

}

return false;

}

int get\_choice\_fill()

{

printf\_s("\n\n%d - Заполнить массив рандомными числами\n%d - заполнить массив самостоятельно\n\n", fill\_randomm,fill\_by\_my\_self);

int choice = get\_value("Выберите действие:\t");

return choice;

}

int get\_choice\_action()

{

printf\_s("\n\n%d - Найти сумму отрицательных элементов, значения которых кратно 10\n%d - Заменить первые k элементов массива на те же элементы в обратном порядке\n%d - Определить, есть ли пара соседних элементов с произведением, pавным заданном числу\n\n", sum\_10, exchange, para);

int choice = get\_value("Выберите действие:\t");

return choice;

}

int get\_value(const char\* massage)

{

int value = 0;

printf\_s("%s", massage);

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

if (res != 1)

{

errno = EIO;

perror("ERROR");

abort();

}

return value;

}

size\_t get\_size\_t(int value)

{

if (value <= 0)

{

errno = ENOMEM;

perror("wrong size");

abort();

}

return (size\_t)value;

}

void check\_range(int range\_right, int range\_left)

{

if (range\_left > range\_right)

{

errno = EIO;

perror("ERROR");

abort();

}

}

int\* get\_array(const size\_t size)

{

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

if (NULL == array)

{

errno = ENOMEM;

perror("Память не выделена\n");

abort();

}

return array;

}

int fill\_random(const size\_t size, int\* array, int range\_right, int range\_left)

{

unsigned int ttime = (unsigned int)time(NULL);

srand(ttime);

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

array[i] = rand() % (range\_right - range\_left + 1) + range\_left;

}

return 0;

}

int fill\_by\_your\_self(const size\_t size, int\* array, int range\_right, int range\_left)

{

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

{

int element = get\_value("Введите элемент массива в введенном диапазоне:\t");

if (element > range\_right || element < range\_left)

{

perror("ERROR");

errno = EIO;

abort();

}

array[i] = element;

}

return 0;

}

void free\_array(int\* array)

{

if (NULL != array)

{

free(array);

}

}

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

{

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

{

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

}

}

* 1. Результаты выполнения программы 4.1

Результаты выполнения программы представлены ниже ().

* 1. Выполнение тестовых примеров 4.1

В выполнен тестовый пример. Результат его выполнения представлен ниже ().

Тестовый пример

* 1. Отметка о выполнении задания в веб-хостинге системы контроля версий 4.1

Approved task 4.1

1. Задача 4.2
   1. Формулировка задания 4.2

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 - Исходные данные

|  |  |  |
| --- | --- | --- |
| Вариант | Задачи | Интервал |
| 6 | 1. Заменить минимальный элемент массива на средний (количество элементов – нечетно). 2. Удалить из него все элементы, в записи которых есть цифра 5. 3. Из элементов массива C сформировать массив A той же размерности по правилу: если номер i элемента четный, то Ai=Ci2, если нечетный, то Ai=2Ci. | [-1000;1000] |

* 1. Блок-схема алгоритма 4.2

Блок-схема основного алгоритма представлена ниже (). Блок-схемы используемых функций представлены ниже ().

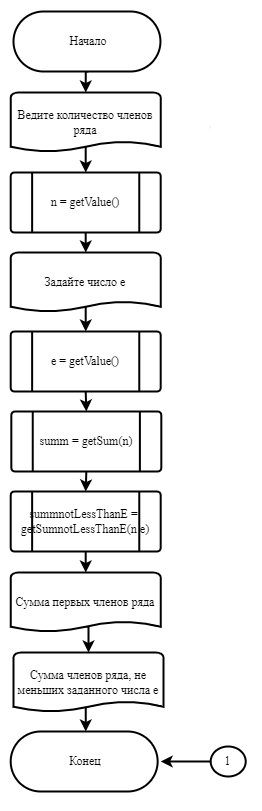


Рисунок - Блок-схема основного алгоритма

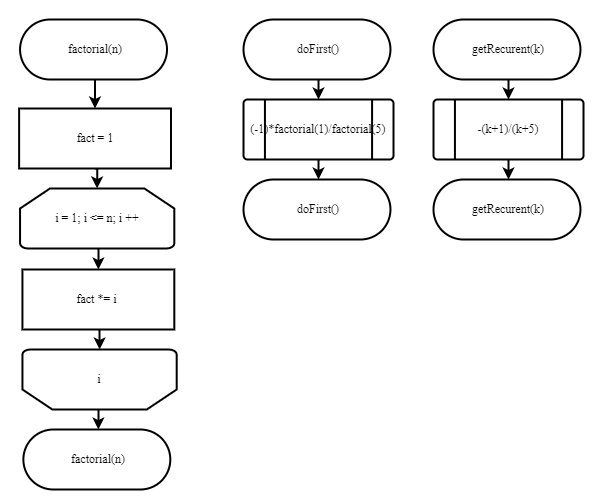


Рисунок - Блок-схема используемых функций

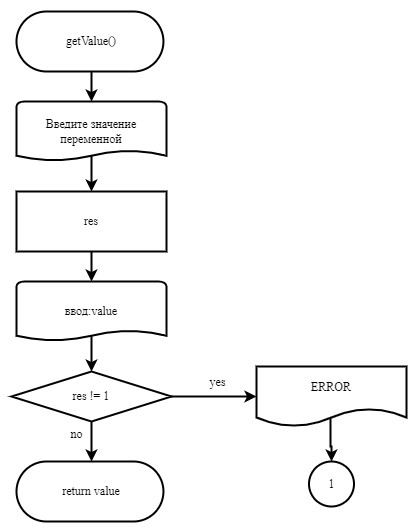


Рисунок - Блок-схема используемой функции

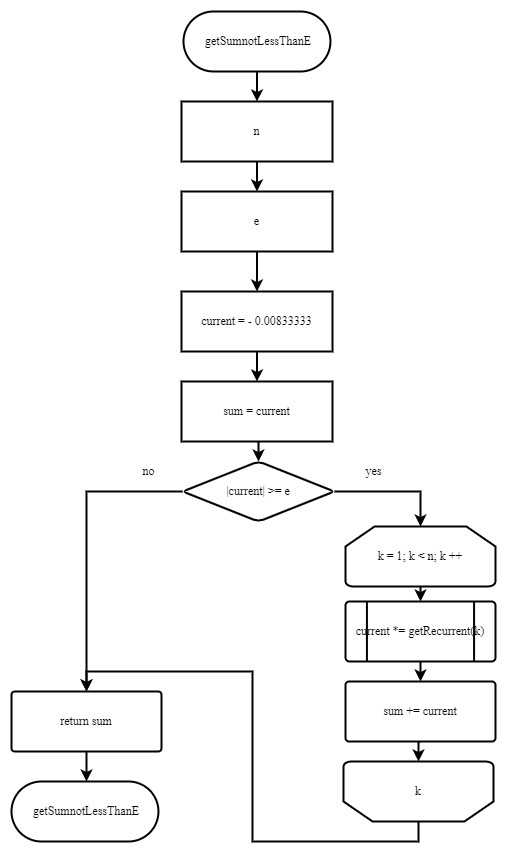


Рисунок - Блок-схема используемой функции

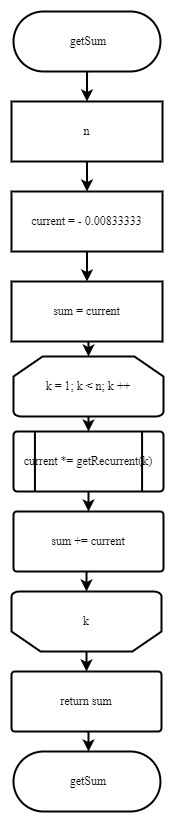


Рисунок - Блок-схема используемой функции

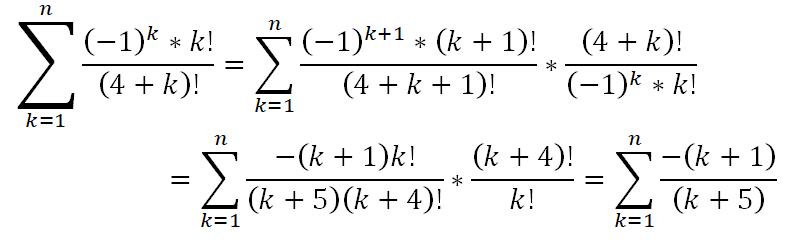


Рисунок - Расчётная формула

* 1. Текст программы на языке C 4.2

#include <stdio.h>

#include <stdlib.h>

#include <errno.h>

#include <time.h>

#include <stdbool.h>

/\*

\* @brief Выбор пользователя

\* @brief fill\_random - Заполнение массива пользователем

\* @brief fill\_by\_my\_self - Заполнение массива рандомными числами

\*/

enum fill\_in {fill\_randomm, fill\_by\_my\_self};

/\*

\* @brief Выбор пользователя

\* @brief min\_middle\_change Заменить минимальный элемент массива на средний (количество элементов – нечетно)

\* @brief delete\_element\_with\_5 Удалить из массива все элементы, в записи которых есть цифра 5

\* @brief new\_array Из элементов массива C сформировать массив A той же размерности по правилу: если номер i элемента четный, то Ai=Ci2, если нечетный, то Ai=2Ci.

\*/

enum action {min\_middle\_change, delete\_element\_with\_5, new\_array};

/\*

\* @brief сортирует массив

\* @param array - массив

\* @param size - размер массива

\*/

void bubble\_sort(int\* array, size\_t size);

/\*

\* @brief меняет два элемента местами

\* @param a - элемент массива

\* @param b - элемент массива

\*/

void swap(int\* a, int\* b);

/\*

\* @brief меняет минимальный элемент на средний

\* @param array - массив

\* @param size - размер массива

\* @return изменённый массив

\*/

int\* min\_middle(int\* array, size\_t size);

/\*

\* @brief Ищет минимальный элемент массива

\* @param array - массив

\* @param size - размер массива

\* @return минимальный элемент

\*/

int min\_element(int\* array, size\_t size);

/\*

\* @brief Ищет средний элемент массива

\* @param array - массив

\* @param size - размер массива

\* @return средний элемент

\*/

int middle\_element(int\* array, size\_t size);

/\*

\* @brief копирует массив

\* @param array - массив

\* @param size - размер массива

\* @return скопироанный массив

\*/

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

/\*

\* @brief считает сколько элементов с цифрой 5 в числе

\* @param array - массив

\* @param size - размер массива

\* @return кол-во элементов с цифрой 5 в числе

\*/

int amount\_5(int\* array, size\_t size);

/\*

\* @brief Определяет, есть ли цифра пять в числе

\* @param а - элемент массива(число)

\* @return true, если цифра пять есть в числе

\*/

bool has\_five(int a);

/\*

\* @brief создает и заполняет массив элементами без цифры 5

\* @param array - массив

\* @param size - размер массива

\* @param k - кол-во элементов с цифрой 5 в числе

\* @return заполненный массив

\*/

int\* get\_array\_5(int\* array, size\_t k, size\_t size);

/\*

\* @brief предлагает выбор пользователю

\* @return введенный номер действия

\*/

int get\_choice\_fill();

/\*

\* @brief предлагает выбор пользователю

\* @return введенный номер действия

\*/

int get\_choice\_action();

/\*

\* @brief проверяет введенное значение

\* @param massage - сообщение для пользователя

\* @return переменную

\*/

int get\_value(const char\* massage);

/\*

\* @brief проверяет чтобы введенное значение было > 0 и нечетное

\* @param value введенное значение

\* @return размер массива

\*/

size\_t get\_size(int value);

/\*

\* @brief создает массив

\* @param size - размер массива

\* @return массив

\*/

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

/\*

\* @brief создает и заполняет массив по правилу: если номер i элемента четный, то Ai=Ci2, если нечетный, то Ai=2Ci

\* @param array - массив

\* @param size - размер массива

\* @return новый массив

\*/

int\* get\_new\_array(int\* array, const size\_t size);

/\*

\* @brief Заполняет массив рандомными числами

\* @param size - размер массива

\* @param array - массив

\* @return 0 - в случае успеха

\*/

int fill\_random(const size\_t size, int\* array);

/\*

\* @brief Проверяет правильность введеного диапазона

\* @param left - леая граница диапазона

\* @param right - правая граница диапазона

\*/

void check\_range(int left, int right);

/\*

\* @brief Пользователь заполняет массив

\* @param size - размер массива

\* @param array -массив

\*/

void fill\_by\_your\_self(const size\_t size, int\* array);

/\*

\* @brief чистит использованную память

\* @param array - массив

\*/

void free\_array(int\* array);

/\*

\* @brief печатает элементы массива

\* @param array - массив

\* @param size - размер массива

\*/

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

/\*

\* @brief Точка входа в программу

\* @return 0, в случае успешного завершения программы

\*/

int main() {

const size\_t size = get\_size(get\_value("Введите размер массива\t"));

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

enum fill\_in fill\_in = (enum fill\_in)get\_choice\_fill();

switch (fill\_in)

{

case (fill\_randomm):

fill\_random(size, array);

puts("\nМассив заполнен рандомными числами:\n");

break;

case (fill\_by\_my\_self):

fill\_by\_your\_self(size, array);

puts("\nВы заполнили массив:\n");

break;

default:

errno = EIO;

perror("Вы ввели неверное значение");

return 1;

}

print\_array(array, size);

enum action action = (enum action)get\_choice\_action();

int\* new = NULL;

switch (action)

{

case (min\_middle\_change):

printf\_s("\n\n%d - Минимальный элемент\n%d - Средний элемент\n", min\_element(array, size), middle\_element(array, size));

puts("\nМассив после:\t");

new = min\_middle(array, size);

print\_array(new, size);

free\_array(new);

break;

case (delete\_element\_with\_5):

puts("\nМассив до:\t");

print\_array(array, size);

puts("\nМассив после:\t");

new = get\_array\_5(array, amount\_5(array, size), size);

print\_array(new, (size - amount\_5(array, size)));

free\_array(new);

break;

case (new\_array):

new = get\_new\_array(array, size);

print\_array(new, size);

free\_array(new);

break;

default:

errno = EIO;

perror("Вы ввели неверное значение");

return 1;

}

free\_array(array);

puts("\n\nGame over");

return 0;

}

int get\_choice\_fill()

{

printf\_s("\n\n%d - Заполнить массив рандомными числами\n%d - заполнить массив самостоятельно\n", fill\_randomm, fill\_by\_my\_self);

int choice = get\_value("\nВыберите действие:\t");

return choice;

}

int get\_choice\_action()

{

printf\_s("\n%d - Заменить минимальный элемент массива на средний(для выполнения этого задания размер массива должен быть нечетным числом)\n%d - Удалить из массива все элементы, в записи которых есть цифра 5\n%d - Из элементов массива C сформировать массив A той же размерности по правилу: если номер i элемента четный, то Ai=Ci^2, если нечетный, то Ai=2Ci.\n", min\_middle\_change, delete\_element\_with\_5, new\_array);

int choice = get\_value("\nВыберите действие:\t");

return choice;

}

void bubble\_sort(int\* array, size\_t size)

{

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

{

for (size\_t j = 0; j < size - i - 1; j++)

{

if (array[j] > array[j + 1])

{

swap(&(array[j]), &(array[j + 1]));

}

}

}

}

void swap(int\* a, int\* b)

{

int temp = \*a;

\*a = \*b;

\*b = temp;

}

int\* min\_middle(int\* array, size\_t size)

{

if (size % 2 == 1)

{

int\* B = copy\_array(array, size);

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

{

if (B[j] == min\_element(B, size))

{

B[j] = middle\_element(B, size);

break;

}

}

return B;

}

else

{

errno = EIO;

perror("\nНеверный размер(должен быть нечетным числом)");

abort();

}

}

int min\_element(int\* array, size\_t size)

{

int min = array[0];

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

{

if (array[i] < min)

{

min = array[i];

}

}

return min;

}

int middle\_element(int\* array, size\_t size)

{

int\* B = copy\_array(array, size);

bubble\_sort(B, size);

int middle = B[(size / 2)];

free\_array(B);

return middle;

}

int\* copy\_array(int\* array, size\_t size)

{

int\* B = get\_array(size);

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

{

B[i] = array[i];

}

return B;

}

int amount\_5(int\* array, size\_t size)

{

int k = 0;

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

{

if (has\_five(array[i]))

{

k++;

}

}

return k;

}

bool has\_five(int a)

{

int b = abs(a);

while (b > 0)

{

if (b % 10 == 5)

{

return true;

}

b = b / 10;

}

return false;

}

int\* get\_array\_5(int\* array, size\_t k, size\_t size)

{

int\* array\_without\_5 = get\_array(size - k);

size\_t i = 0;

size\_t j = 0;

while (i < size)

{

if (has\_five(array[i]))

{

i++;

}

else

{

array\_without\_5[j] = array[i];

i++;

j++;

}

}

return array\_without\_5;

}

int get\_value(const char\* massage)

{

int value = 0;

printf\_s("%s", massage);

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

if (res != 1)

{

errno = EIO;

perror("ERROR");

abort();

}

return value;

}

size\_t get\_size(int value)

{

if (value <= 0)

{

errno = ENOMEM;

perror("Wrong size");

abort();

}

return (size\_t)value;

}

int\* get\_array(const size\_t size)

{

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

if (NULL == array)

{

errno = ENOMEM;

perror("Память не выделена\n");

abort();

}

return array;

}

int\* get\_new\_array(int\* array, const size\_t size)

{

int\* new\_array = get\_array(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];

}

}

return new\_array;

}

int fill\_random(const size\_t size, int\* array)

{

unsigned int ttime = (unsigned int)time(NULL);

srand(ttime);

int A = get\_value("\nВведите левую границу диапозона\t");

int B = get\_value("\nВведите правую границу диапозона\t");

check\_range(A, B);

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

array[i] = rand() % (B - A + 1) + A;

}

return 0;

}

void check\_range(int left, int right)

{

if (left > right)

{

errno = EIO;

perror("\nWrong number");

abort();

}

}

void fill\_by\_your\_self(const size\_t size, int\* array)

{

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

{

array[i] = get\_value("Введите элемент массива\t");

}

}

void free\_array(int\* array)

{

if (NULL != array)

{

free(array);

}

}

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

{

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

{

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

}

}

* 1. Результаты выполнения программы 4.2

Результат выполнения программы представлен ниже ().

Результат выполнения программы

* 1. Выполнение тестовых примеров 4.2

В выполнены тестовые примеры. Результаты их выполнения представлены ниже ().

Тестовый расчёт

* 1. Отметка о выполнении задания в веб-хостинге системы контроля версий 4.2

Approved task 4.2

1. Задача 4.3
   1. Формулировка задания 4.3

Создать многомерный массив nˣm 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.

|  |  |
| --- | --- |
| Вариант | Задачи |
| 6 | 1. Заменить максимальный элемент каждой строки нулем. 2. Вставить перед всеми строками, первый элемент которых делится на 3, строку из нулей. |

* 1. Блок-схема алгоритма 4.3

Блок-схема основного алгоритма представлена ниже (). Блок-схемы используемых функций представлены ниже ().

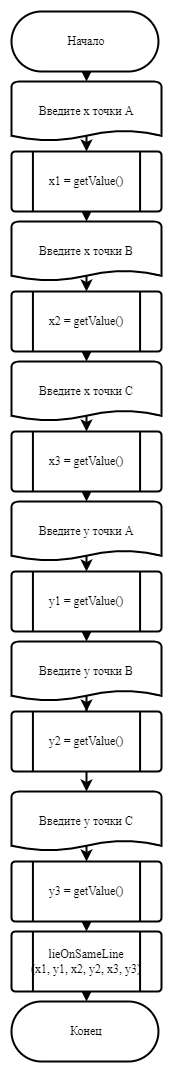


Рисунок - Блок-схема основного алгоритма

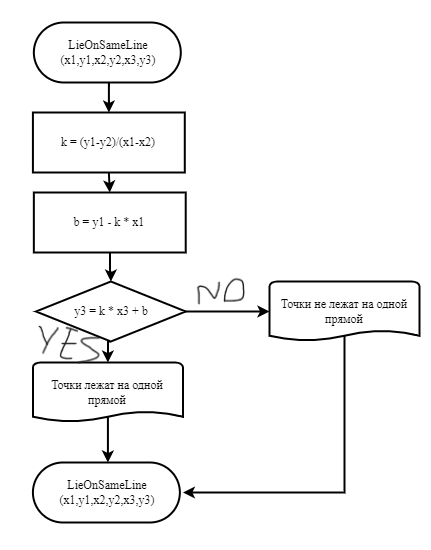


Рисунок - Блок-схема используемой функции

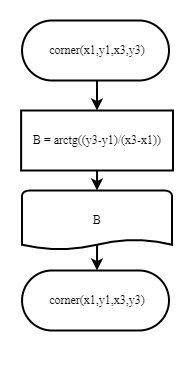


Рисунок - Блок-схема используемой функции

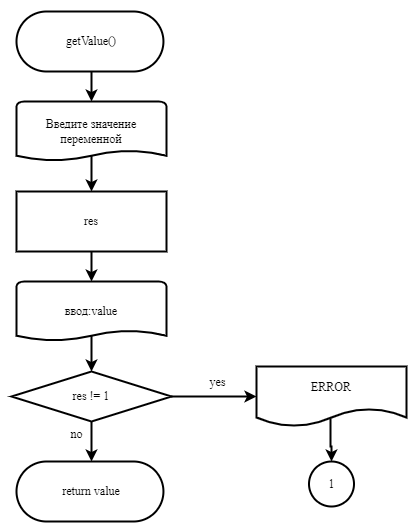


Рисунок - Блок-схема используемой функции

* 1. Текст программы на языке C 4.3

#define \_USE\_MATH\_DEFINES

#include <stdio.h>

#include <stdlib.h>

#include <math.h>

#include <errno.h>

#include <stdbool.h>

#include <float.h>

/\*

\* @brief проверяет пользовательский ввод

\* @return введеное значение

\*/

double getValue();

/\*

\* @brief Функция считает значение угла между 3 точками

\* @param Ax координата х первой точки

\* @param Cx координата х третей точки

\* @param Ay координата у первой точки

\* @param Cy координата у третей точки

\* @return величинa угла В

\*/

double corner(double x1,double y1,double x3,double y3);

/\*

\* @brief Функция определяет лежат ли 3 точки на одной линии

\* @param Ax координата х первой точки

\* @param Bx координата х второй точки

\* @param Cx координата х третей точки

\* @param Ay координата у первой точки

\* @param By координата у второй точки

\* @param Cy координата у третей точки

\* @return если точки лежат на одной прямой, то пишет ответ и завершает программу, в противном случае обращается к функции corner

\*/

double lieOnSameLine(double x1, double y1, double x2, double y2, double x3, double y3);

/\*

\* @brief является точкой входа в программу

\* @return 0, в случае успеха

\*/

int main() {

printf\_s("Введите значение x точки A\n");

double x1 = getValue();

printf\_s("Введите значение y точки A\n");

double y1 = getValue();

printf\_s("Введите значение x точки B\n");

double x2 = getValue();

printf\_s("Введите значение y точки B\n");

double y2 = getValue();

printf\_s("Введите значение x точки C\n");

double x3 = getValue();

printf\_s("Введите значение y точки C\n");

double y3 = getValue();

lieOnSameLine(x1, y1, x2, y2, x3, y3);

return EXIT\_SUCCESS;

}

double getValue() {

double value;

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

if (res != 1)

{

errno = EIO;

perror("ERROR");

abort();

}

return value;

}

double corner(double x1,double y1,double x3,double y3) {

double B = atan((double)(y3 - y1)/(x3 - x1));

B = B \* (180.0 / M\_PI); //пеевод из радиан в градусы

printf\_s("Угол В = %.2lf\n", B);

return B;

}

double lieOnSameLine(double x1, double y1, double x2, double y2, double x3, double y3) {

double k = (double)(y1 - y2) / (x1 - x2);

double b = y1 - k \* x1;

if (y3 - k \* x3 - b <= DBL\_MIN)

{

printf\_s("Точки лежат на одной прямой\n");

return 0.0;

}

else

{

printf\_s("Точки не лежат на одной прямой\n");

return corner(x1, y1, x3, y3);

}

}

* 1. Результаты выполнения программы 4.3

Результаты выполнения программы представлены ниже ().

* 1. Выполнение тестовых примеров 4.3

На выполнены тестовые примеры. Результаты их выполнения представлены ниже ().

* 1. Отметка о выполнении задания в веб-хостинге системы контроля версий 4.3

Approved task 4.3