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 5

Выполнил: ст. гр. ТКИ-141

Буланый Сергей Евгеньевич

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

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

Москва 2023

Оглавление

[1 Решение задачи 1-1 3](#_Toc151483877)

[1.1 Формулировка задачи 4-1 3](#_Toc151483878)

[1.2 Блок-схема алгоритма задание 4–1 4](#_Toc151483879)

[1.3 Текст программы на языке С задание 4–1 12](#_Toc151483880)

[1.4 Результаты выполнения программы 4–1 19](#_Toc151483881)

[1.5 Отметка о выполнении задания в веб-хостинге системы контроля версий задание 4–1 21](#_Toc151483882)

[2 Решение задачи 2-2 22](#_Toc151483883)

[2.1 Формулировка задачи 4-2 22](#_Toc151483884)

[2.2 Блок-схема алгоритма задания 4-2 23](#_Toc151483885)

[2.3 Текст программы на языке С задание 4–2 31](#_Toc151483886)

[2.4 Результаты выполнения программы 4–2 38](#_Toc151483887)

[2.5 Отметка о выполнении задания в веб-хостинге системы контроля версий задание 4–2 40](#_Toc151483889)

[3 Решение задачи 1-3 41](#_Toc151483890)

[3.1 Формулировка задачи 4-3 41](#_Toc151483891)

[3.2 Блок-схема алгоритма задания 4-3 42](#_Toc151483892)

[3.3 Текст программы на языке С задание 4–3 52](#_Toc151483893)

[3.4 Результаты выполнения программы 4–3 61](#_Toc151483894)

[3.5 Отметка о выполнении задания в веб-хостинге системы контроля версий задание 4–3 63](#_Toc151483896)

1. Решение задачи 4–1

**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 |

|  |  |  |
| --- | --- | --- |
| 5 | 1. Заменить последние k элементов массива на противоположные по знаку. 2. Вывести индексы тех элементов, значения которых кратны 3. 3. Определить, есть ли пара соседних элементов с суммой, равной заданному числу. | [-100;100] |

1.2 Блок-схема алгоритма задание 4–1

Блок-схема основного алгоритма представлена ниже (Рисунок 1). Блок-схемы функций расчета значений представлены ниже (Рисунок 2-11).

Изображение выглядит как текст, чек, черно-белый, документ

Автоматически созданное описание

Рисунок  ­ Блок-схема основного алгоритма

Изображение выглядит как текст, диаграмма, линия, Шрифт

Автоматически созданное описание

Рисунок  – Блок-схема используемых функций

Изображение выглядит как текст, диаграмма, линия, зарисовка

Автоматически созданное описание

Рисунок 3 – Блок-схема используемых функций

Изображение выглядит как текст, зарисовка, диаграмма, линия

Автоматически созданное описание

Рисунок 4 – Блок-схема используемых функций

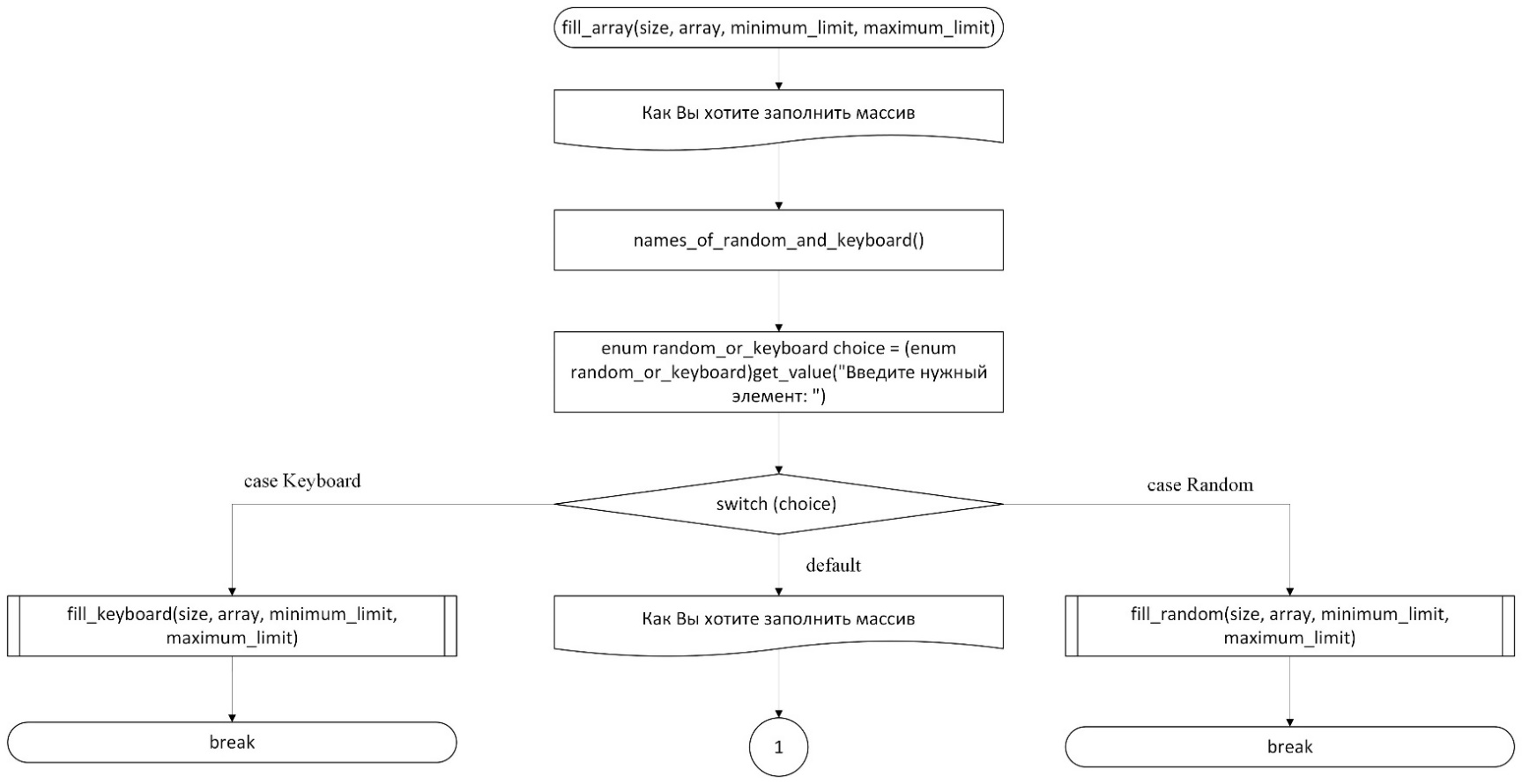


Рисунок 5 – Блок-схема используемых функций

Изображение выглядит как текст, чек, Шрифт, снимок экрана

Автоматически созданное описание

Рисунок 6 – Блок-схема используемых функций

Изображение выглядит как текст, снимок экрана, Шрифт, диаграмма

Автоматически созданное описание

Рисунок 7– Блок-схема используемых функций

Изображение выглядит как текст, диаграмма, линия, снимок экрана

Автоматически созданное описание

Рисунок 8– Блок-схема используемых функций

Изображение выглядит как текст, диаграмма, линия, Параллельный

Автоматически созданное описание

Рисунок 9– Блок-схема используемых функций

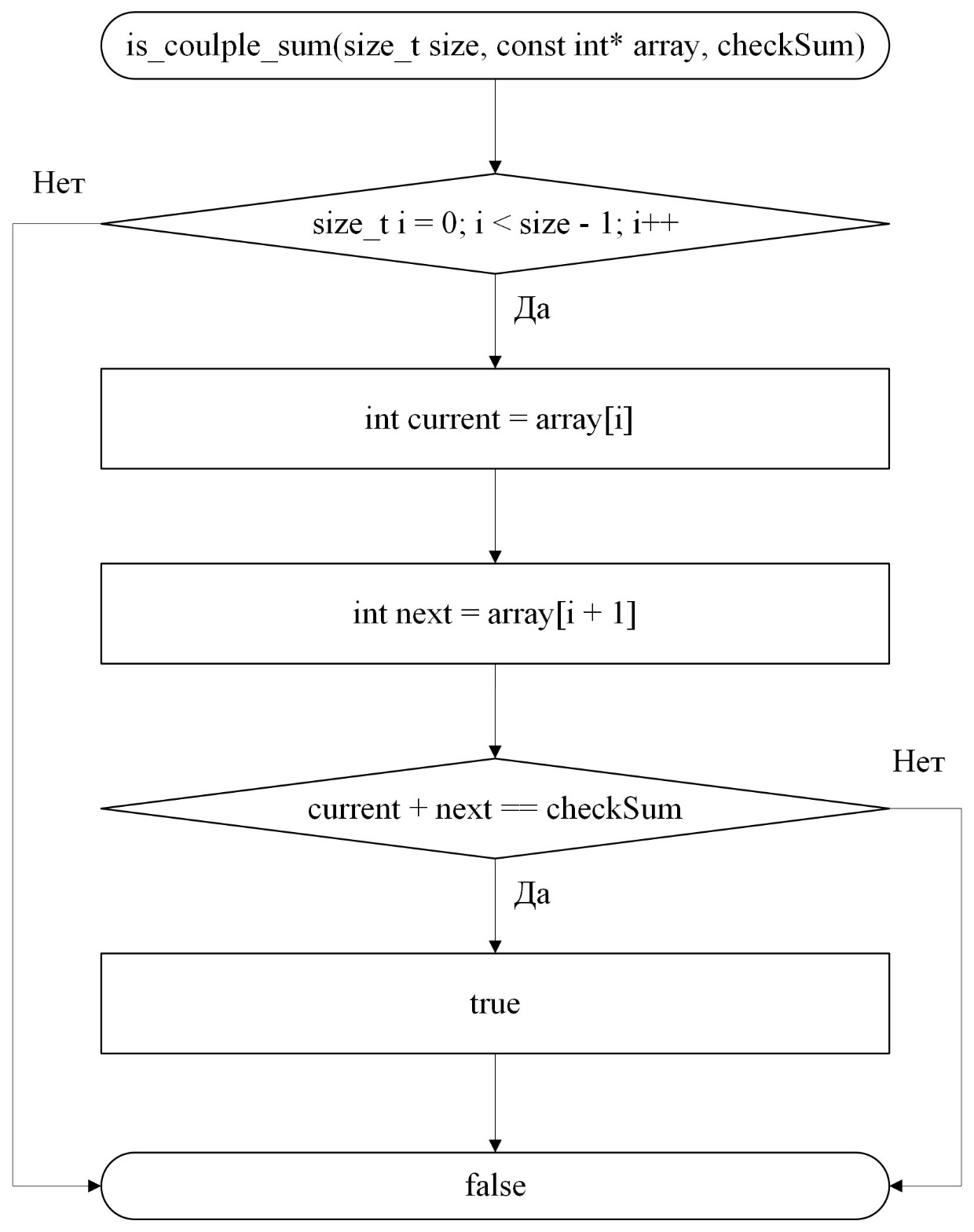


Рисунок 10– Блок-схема используемых функций

Изображение выглядит как текст, снимок экрана, Шрифт, линия

Автоматически созданное описание

Рисунок 11– Блок-схема используемых функций

1.3 Текст программы на языке С задание 4–1

#include <memory.h>

#include <stdlib.h>

#include <errno.h>

#include <time.h>

#include <stdio.h>

#include <stdbool.h>

#include <locale.h>

/\*\*

\* @brief Функция находящая индекс элементов, значения которых кратны 3

\* @param size - размер массива

\* @param array - массив

\*/

void print\_indexes\_3(const int\* const arr, size\_t size);

/\*\*

\* @brief Функция находящая последние k элементов массива и заменяющего их на противоположные

\* @param size - размер массива

\* @param array - массив

\* @return Последние k элементов изменёного массива

\*/

int\* replace\_last\_k(const int\* const arr, size\_t size);

/\*\*

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

\* @param message - текст сообщения для пользователя

\* @return Значение размера

\*/

size\_t get\_size(char const\* message);

/\*\*

\* @brief Функция выделяющая память под массив

\* @param size - размер массива

\* @return Массив

\*/

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

/\*\*

\* @brief Функция считывающая ввод с клавиатуры

\* @param size - размер массива

\* @param array - массив

\*/

void fill\_keyboard(const size\_t size, int\* array, int minimum\_limit, int maximum\_limit);

/\*\*

\* @brief Функция заполняющая массив рандомными числами

\* @param size - размер массива

\* @param array - массив

\* @param minimum\_limit - граница снизу

\* @param maximum\_limit - граница сверху

\*/

void fill\_random(const size\_t size, int\* array, int minimum\_limit, int maximum\_limit);

/\*\*

\* @brief Функция выводящая заполненный массив

\* @param size - размер массива

\* @param array - массив

\*/

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

/\*\*

\* @brief Функция принимающая и проверяющая значение на ввод

\* @param message - текст сообщения для пользователя

\* @return Значение

\*/

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

/\*\*

\* @brief Функция проверяющая массив есть ли пара соседних элементов с суммой, равной заданному числу

\* @param size - размер массива

\* @param array - массив

\* @param checkSum - сумма которой равны два соседних элемента

\* @return Возвращает true если сумма элементов в паре равна checkSum, иначе false

\*/

bool is\_coulple\_sum(const size\_t size, const int\* array, int checkSum);

/\*\*

\* @brief Функция печатает есть ли пары массива

\* @param size - размер массива

\* @param array - массив

\*/

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

/\*\*

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

\*/

void names\_of\_random\_and\_keyboard();

/\*\*

\* @brief Пользовательский ввод

\*/

enum random\_or\_keybord

{

Keyboard = 1,

Random = 2

};

/\*\*

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

\* @param size - размер массива

\* @param array - массив

\* @param minimum\_limit - граница снизу

\* @param maximum\_limit - граница сверху

\*/

void fill\_array(const size\_t size, int\* array, int minimum\_limit, int maximum\_limit);

/\*\*

\* @brief Функция проверяет что верхняя граница больше нижней

\* @param minimum\_limit - нижняя граница

\* @param maximum\_limit - верхняя граница

\* @return Возвращает true если верхняя граница больше нижней, иначе false

\*/

bool checklimit(int minimum\_limit, int maximum\_limit);

/\*\*

\* @brief Функция удаляющая массив

\* @param array - массив

\*/

void delete\_array(int\*\* arr);

/\*\*

\* @brief Точка входа в программу

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

\*/

int main()

{

setlocale(LC\_ALL, "RU");

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

int\* arr = init\_array(size);

const int minimum\_limit = get\_value("Введите нижнюю границу массива: ");

const int maximum\_limit = get\_value("Введите верхнюю границу массива: ");

checklimit(minimum\_limit, maximum\_limit);

fill\_array(size, arr, minimum\_limit, maximum\_limit);

puts("Исходный массив: \n");

print\_array(size, arr);

int\* second\_array = replace\_last\_k(arr, size);

print\_array(size, second\_array);

delete\_array(second\_array);

print\_indexes\_3(arr, size);

int checkSum = get\_value("Введите сумму элементов: ");

print\_couples(size, arr);

delete\_array(arr);

return 0;

}

void names\_of\_random\_and\_keyboard()

{

printf\_s("Keyboard - %d\n", (int)Keyboard);

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

}

int\* replace\_last\_k(const int\* const arr, size\_t size)

{

size\_t k = get\_size("Введите k: ");

int\* arr\_k = init\_array(size);

memcpy(arr\_k, arr, size \* sizeof(int));

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

{

arr\_k[i] = -arr\_k[i];

}

return arr\_k;

}

size\_t get\_size(char const\* message)

{

int size = get\_value(message);

if (size < 0)

{

perror("Неверный размер!\n");

abort();

}

return (size\_t)size;

}

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

{

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

if (arr == NULL)

{

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

abort();

}

return arr;

}

void fill\_array(const size\_t size, int\* array, int minimum\_limit, int maximum\_limit)

{

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

names\_of\_random\_and\_keyboard();

enum random\_or\_keyboard choice = (enum random\_or\_keyboard)get\_value("Введите нужный элемент: ");

switch (choice)

{

case Keyboard:

fill\_keyboard(size, array, minimum\_limit, maximum\_limit);

break;

case Random:

fill\_random(size, array, minimum\_limit, maximum\_limit);

break;

default:

perror("Неверный выбор!\n");

abort();

}

}

void fill\_keyboard(const size\_t size, int\* array,int minimum\_limit,int maximum\_limit)

{

puts("Введите массив: ");

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

{

int value = get\_value("");

if (value < minimum\_limit || value > maximum\_limit)

{

errno = EIO;

perror("Неверное значение");

abort();

}

array[i] = value;

}

}

void fill\_random(const size\_t size, int\* array,int minimum\_limit,int maximum\_limit)

{

unsigned int ttime = time(NULL);

srand(ttime);

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

{

array[i] = minimum\_limit + rand() % (maximum\_limit - minimum\_limit + 1);

}

}

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

{

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

{

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

}

}

int get\_value(const char\* message)

{

int value;

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

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

if (res != 1)

{

errno = EIO;

perror("Ошибка ввода");

abort();

}

return value;

}

void print\_indexes\_3(const int\* const arr, size\_t size)

{

puts("Индексы элементов, кратных 3:\n");

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

{

if (arr[i] % 3 == 0)

{

printf\_s("%d ", i);

}

}

puts("\n");

}

bool is\_coulple\_sum(const size\_t size, const int\* array, int checkSum)

{

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

{

int current = array[i];

int next = array[i + 1];

if (current + next == checkSum)

{

return true;

}

}

return false;

}

void print\_couples(const size\_t size, const int\* array)

{

int checkSum = get\_value("Введите сумму элементов: ");

if (is\_coulple\_sum( size, array, checkSum))

{

puts("Присутствуют\n");

}

else

{

puts("Отсутствуют\n");

}

}

bool checklimit(int minimum\_limit, int maximum\_limit)

{

return minimum\_limit >= maximum\_limit;

}

void delete\_array(int\*\* arr)

{

if (arr != NULL)

{

free(arr);

}

arr = NULL;

}

Результаты выполнения программы 4–1

Результаты выполнения программы в C представлена ниже (Рисунок 12 и 13).

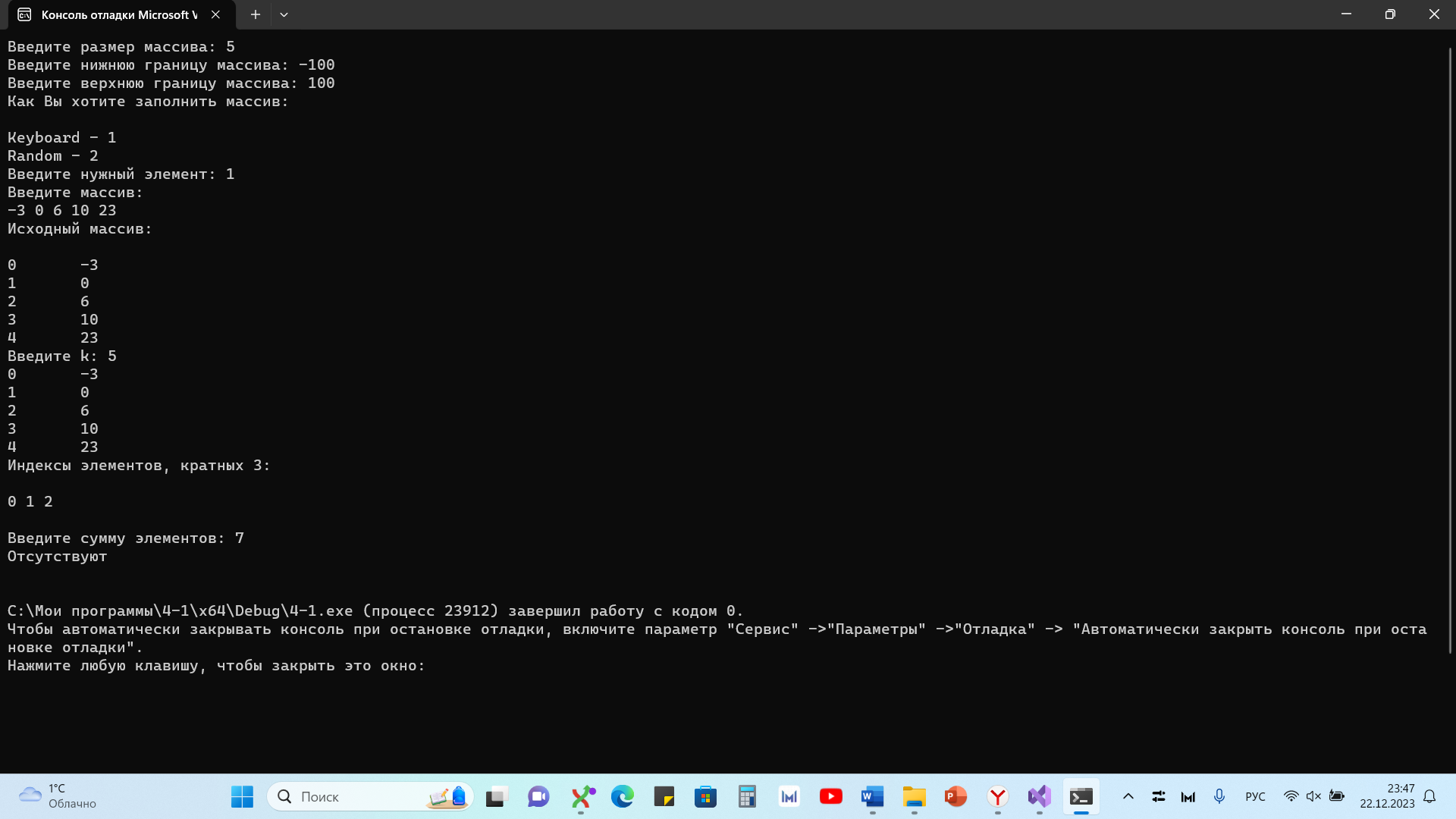


Рисунок 12 – Результаты выполнения программы

Изображение выглядит как текст, снимок экрана, дисплей, программное обеспечение

Автоматически созданное описание

Рисунок 13 – Результаты выполнения программы

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

Ниже представлено доказательство того, что задание 4–1 было принято. (Рисунок 14)

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 14 – Approve задачи 4–1

1. Решение задачи 4–2

**2.1 Формулировка** **задачи 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 |

|  |  |  |
| --- | --- | --- |
| 5 | 1. Заменить первый отрицательный элемент массива нулем.  2. Вставить число К после всех элементов, кратных своему номеру.  3. Из элементов массива D сформировать массив A той же размерности по правилу: если номер четный, то значение элемента находится по формуле Ai = (i-1)\*Di, а если нечетный, то по формуле Ai = Di\*i\*2. | [-100;100] |

2.2 Блок-схема алгоритма задание 4–2

Блок-схема основного алгоритма представлена ниже (Рисунок 1). Блок-схемы функций расчета значений представлены ниже (Рисунок 2-11).

Изображение выглядит как текст, чек, черно-белый

Автоматически созданное описание

Рисунок  ­ Блок-схема основного алгоритма

Изображение выглядит как текст, диаграмма, линия, зарисовка

Автоматически созданное описание

Рисунок  – Блок-схема используемой функции

Изображение выглядит как текст, диаграмма, зарисовка, линия

Автоматически созданное описание

Рисунок 3 – Блок-схема используемой функции

Изображение выглядит как текст, диаграмма, линия, снимок экрана

Автоматически созданное описание

Рисунок 4 – Блок-схема используемой функции

Изображение выглядит как текст, диаграмма, снимок экрана, чек

Автоматически созданное описание

Рисунок 5 – Блок-схема используемой функции

Изображение выглядит как диаграмма, текст, линия, зарисовка

Автоматически созданное описание

Рисунок 6 – Блок-схема используемой функции

Изображение выглядит как текст, Шрифт, снимок экрана, чек

Автоматически созданное описание

Рисунок 7 – Блок-схема используемой функции

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Рисунок 8 – Блок-схема используемой функции

Изображение выглядит как текст, чек, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 9 – Блок-схема используемой функции

Изображение выглядит как текст, чек, Шрифт, число

Автоматически созданное описание

Рисунок 10 – Блок-схема используемой функции

Изображение выглядит как текст, чек, Шрифт, документ

Автоматически созданное описание

Рисунок 11 – Блок-схема используемой функции

2.3 Текст программы на языке С задание 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 names\_of\_random\_and\_keyboard();

/\*\*

\* @brief Пользовательский ввод

\*/

enum random\_or\_keybord

{

Keyboard = 1,

Random = 2

};

/\*\*

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

\* @param message - текст сообщения для пользователя

\* @return Значение размера

\*/

size\_t get\_size(char const\* message);

/\*\*

\* @brief Функция выделяющая память под массив

\* @param size - размер массива

\* @return Массив

\*/

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

/\*\*

\* @brief Функция считывающая ввод с клавиатуры

\* @param size - размер массива

\* @param array - массив

\*/

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

/\*\*

\* @brief Функция заполняющая массив рандомными числами

\* @param size - размер массива

\* @param array - массив

\*/

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

/\*\*

\* @brief Функция выводящая заполненный массив

\* @param size - размер массива

\* @param array - массив

\*/

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

/\*\*

\* @brief Функция принимающая и проверяющая значение на ввод

\* @param message - текст сообщения для пользователя

\* @return Значение

\*/

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

/\*\*

\* @brief Функция заменяющая первый отрицательный элемент нулём

\* @param array - целочисленный массив

\* @param size - размер

\* @return Возвращает изменёный массив

\*/

int\* replace\_first\_negative(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 - элемен

\* @return Возвращает изменёный массив

\*/

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

/\*\*

\* @brief Функция ищет индекс последнего элемента кратного своему номеру

\* @param array - целочисленный массив

\* @param size - размер

\* @return Возвращает индекс

\*/

size\_t find\_index\_last\_divisible(const int\* const array, size\_t size);

/\*\*

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

\* @param array - целочисленный массив

\* @param size - размер

\* @return Возвращает новый массив A

\*/

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

/\*\*

\* @brief Функция удаляющая массив

\* @param array - массив

\*/

void delete\_array(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 = get\_size("Введите размер массива: ");

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

fill\_array(size, array);

puts("Исходный массив:\n");

print\_array(size, array);

int\* array\_with\_replaced\_first\_negative = replace\_first\_negative(array, size);

puts("Массив с заменённым первым отрицательным элементом на 0:\n");

print\_array(size, array\_with\_replaced\_first\_negative);

int k = get\_value("Введите k: ");

int\* array\_with\_inserted\_k = insert\_k(array, size, k);

puts("Массив со вставленным числом после элементов, кратных своему номеру (начиная с 1):\n");

print\_array(size + 1, array\_with\_inserted\_k);

int\* A\_array = get\_array\_A(array, size);

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

print\_array(size, A\_array);

delete\_array(array);

delete\_array(array\_with\_inserted\_k);

delete\_array(A\_array);

delete\_array(array\_with\_replaced\_first\_negative);

return 0;

}

void names\_of\_random\_and\_keyboard()

{

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

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

}

size\_t get\_size(char const\* message)

{

int size = get\_value(message);

if (size < 0)

{

perror("Неверный массив!\n");

abort();

}

return (size\_t)size;

}

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

{

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

if (arr == NULL)

{

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

}

return arr;

}

void fill\_array(const size\_t size, int\* array)

{

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

names\_of\_random\_and\_keyboard();

enum random\_or\_keybord choice = get\_value("Выберите нужный вариант: ");

switch (choice)

{

case Keyboard:

fill\_keyboard(size, array);

break;

case Random:

fill\_random(size, array);

break;

default:

perror("Неверный выбор!!\n");

}

}

void fill\_keyboard(const size\_t size, int\* array)

{

const int minimum\_limit = get\_value("Введите нижнюю границу массива: ");

const int maximum\_limit = get\_value("Введите верхнюю границу массива: ");

if (minimum\_limit >= maximum\_limit)

{

errno = EIO;

perror("Неверные границы!\n");

abort();

}

puts("Введите массив: ");

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

{

int c = get\_value("");

if (c < minimum\_limit || c > maximum\_limit)

{

errno = EIO;

perror("Ошибка ввода!\n");

abort();

}

array[i] = c;

}

}

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

{

const int minimum\_limit = get\_value("Введите нижнюю границу массива: ");

const int maximum\_limit = get\_value("Введите верхнюю границу массива: ");

unsigned int ttime = time(NULL);

srand(ttime);

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

{

array[i] = minimum\_limit + rand() % (maximum\_limit - minimum\_limit + 1);

}

}

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

{

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

{

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

}

}

int get\_value(const char\* message)

{

int value;

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

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

if (res != 1)

{

errno = EIO;

perror("Неверное значение\n");

abort();

}

return value;

}

int\* replace\_first\_negative(const int\* const array, size\_t size)

{

int\* new\_array = init\_array(size);

bool is\_first = true;

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

{

new\_array[i] = array[i];

if (is\_first && array[i] < 0)

{

new\_array[i] = 0;

is\_first = false;

}

}

return new\_array;

}

size\_t find\_index\_last\_divisible(const int\* const array, size\_t size)

{

size\_t index = 0;

bool is\_divisible\_elements = false;

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

{

if (array[i] % (int)(i + 1) == 0)

{

is\_divisible\_elements = true;

index = i;

}

}

if (!is\_divisible\_elements)

{

perror("Нет элементов, кратных своим номерам!\n");

abort();

}

return index;

}

int\* insert\_k(const int\* const array, size\_t size, int k)

{

int\* new\_array = init\_array(size + 1);

size\_t last\_divisible\_index = find\_index\_last\_divisible(array, size);

if (last\_divisible\_index >= size)

{

perror("Индекс последнего кратного элемента массива больше размера массива!\n");

abort();

}

for (size\_t i = 0; i < last\_divisible\_index + 1; ++i)

{

new\_array[i] = array[i];

}

new\_array[last\_divisible\_index + 1] = k;

for (size\_t i = last\_divisible\_index + 1; i < size - last\_divisible\_index - 1; ++i)

{

new\_array[i + 1] = array[i];

}

return new\_array;

}

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

{

int\* new\_array = init\_array(size);

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

{

if ((i + 1) % 2 == 0)

{

new\_array[i] = (i - 1) \* array[i];

}

else

{

new\_array[i] = array[i] \* i \* 2;

}

}

return new\_array;

}

void delete\_array(int\*\* array)

{

if (array != NULL)

{

free(array);

}

array = NULL;

}

Результаты выполнения программы 4–2

Результаты выполнения программы в C представлена ниже (Рисунок 12 и13)

Изображение выглядит как текст, снимок экрана

Автоматически созданное описаниеРисунок 12 – Результаты выполнения программы

Изображение выглядит как текст, снимок экрана, программное обеспечение, дисплей

Автоматически созданное описание

Изображение выглядит как снимок экрана, текст, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

Рисунок 13 – Результаты выполнения программы

Отметка о выполнении задания в веб-хостинге системы контроля версий задание 4–2

Ниже представлено доказательство того, что задание 4–2 было принято. (Рисунок 14)

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 14– Approve задачи 4–2

1. Решение задачи 4–3

**3.1 Формулировка** **задачи 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.

Таблица  – Исходные данные

|  |  |
| --- | --- |
| Вариант | Задачи |

|  |  |
| --- | --- |
| 5 | 1. Заменить максимальный элемент каждой строки на противоположный по знаку.  2. Вставить после всех столбцов, содержащих максимальный элемент столбец из нулей. |

3.2 Блок-схема алгоритма задание 4–3

Блок-схема основного алгоритма представлена ниже (Рисунок 1). Блок-схемы функций расчета значений представлены ниже (Рисунок 2-16).



Рисунок  ­ Блок-схема основного алгоритма

Изображение выглядит как текст, диаграмма, линия, зарисовка

Автоматически созданное описание

Рисунок  – Блок-схема используемой функции

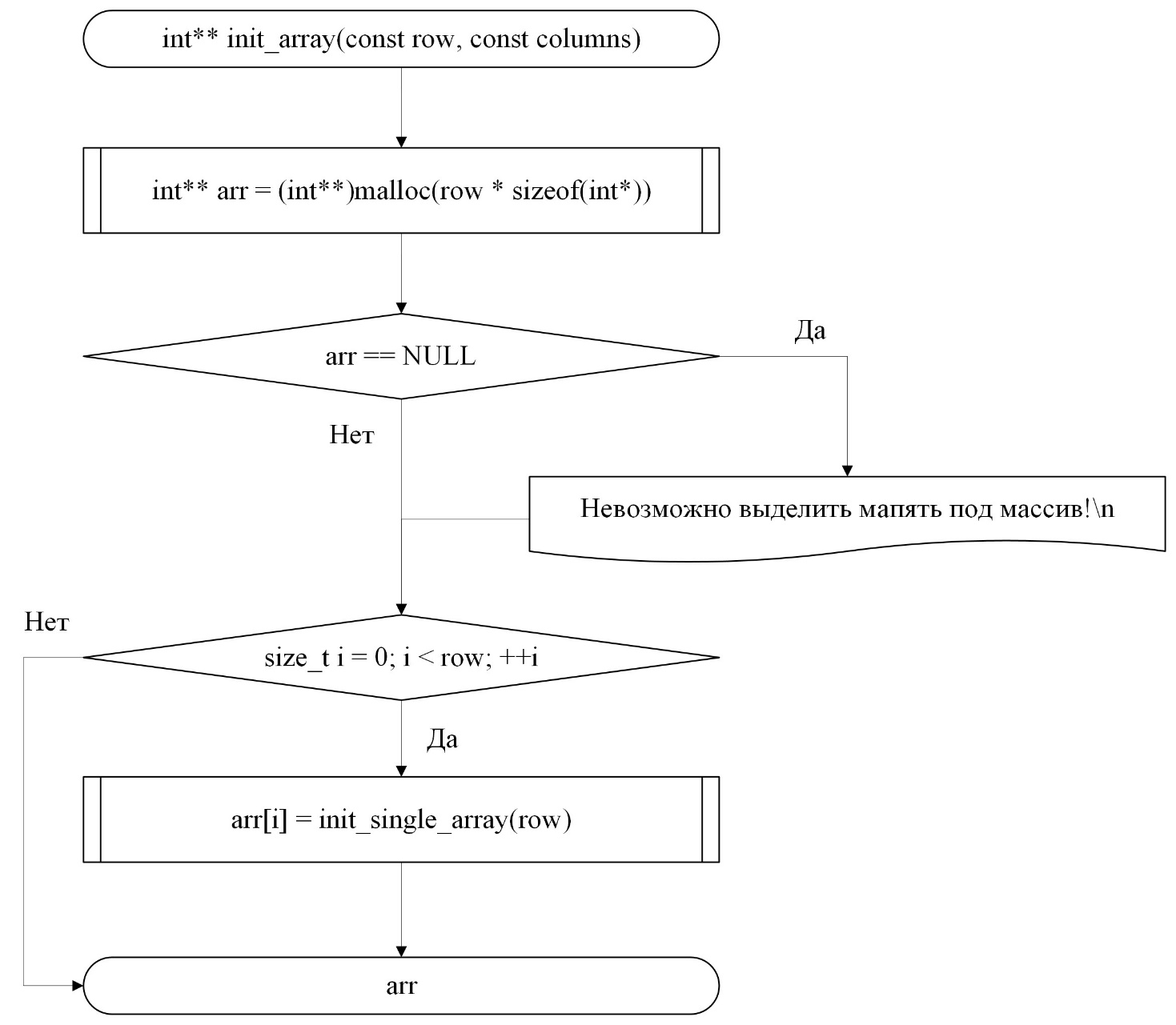


Рисунок 3 – Блок-схема используемой функции

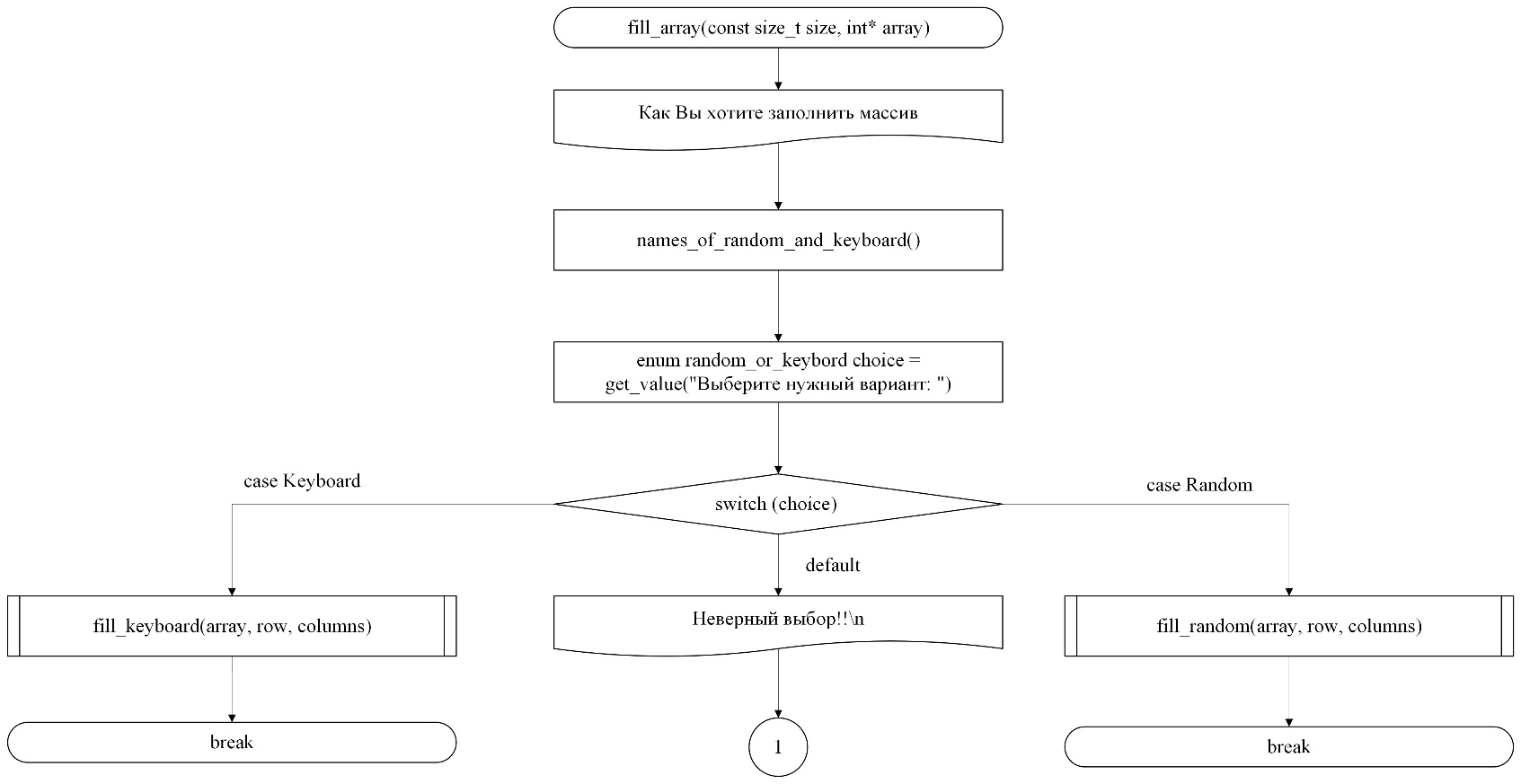


Рисунок 4 – Блок-схема используемой функции

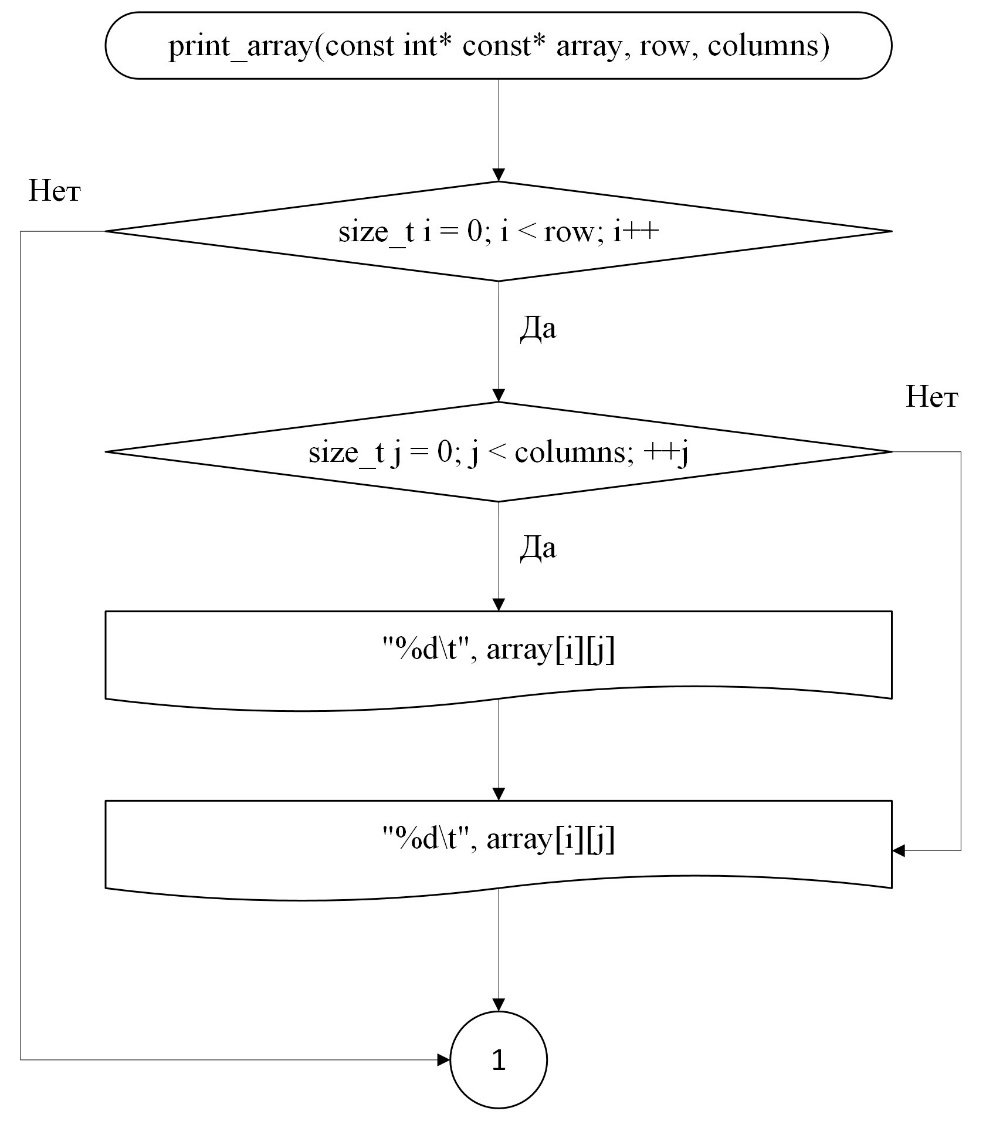


Рисунок 5 – Блок-схема используемой функции

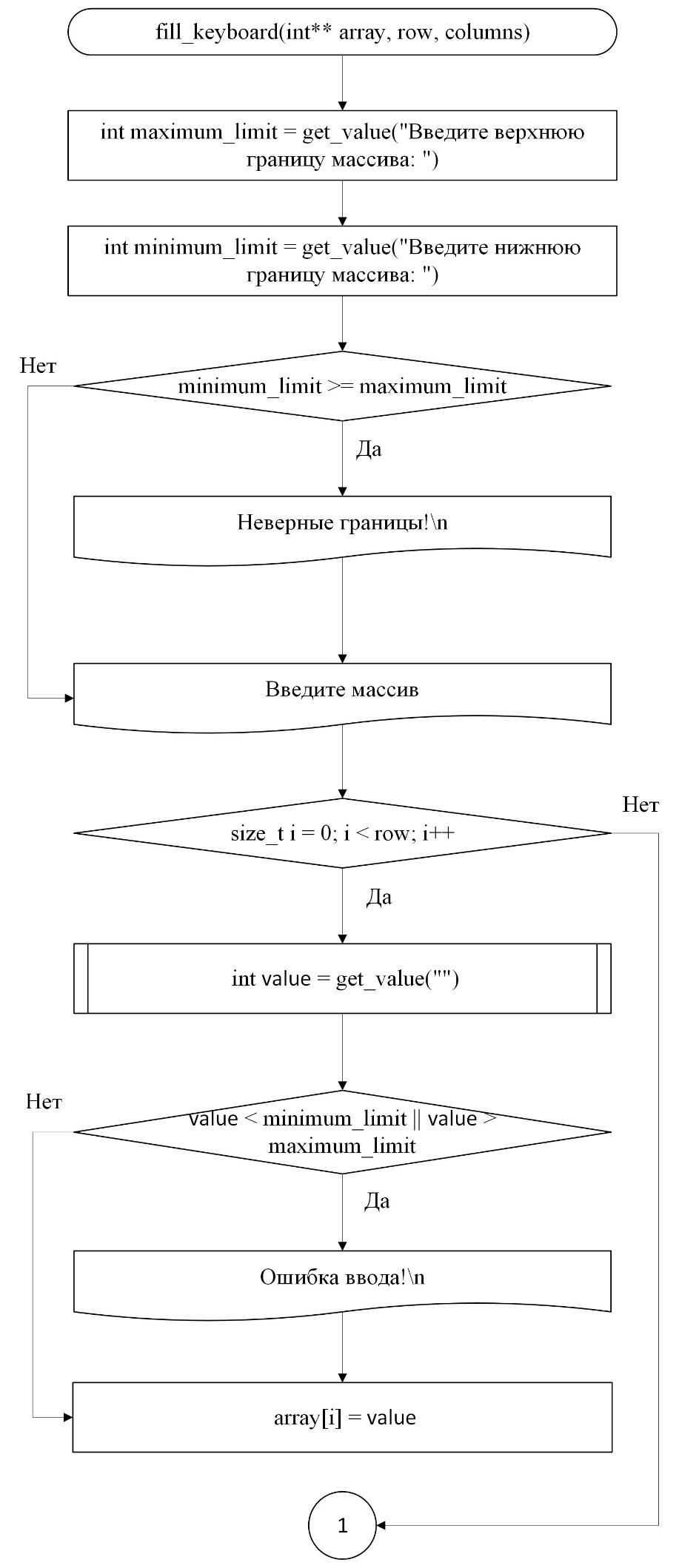


Рисунок 6 – Блок-схема используемой функции

Изображение выглядит как текст, снимок экрана, Шрифт, диаграмма

Автоматически созданное описание

Рисунок 7 – Блок-схема используемой функции

Изображение выглядит как текст, снимок экрана, Шрифт, линия

Автоматически созданное описание

Рисунок 8 – Блок-схема используемой функции

Изображение выглядит как текст, чек, Шрифт, снимок экрана

Автоматически созданное описание

Рисунок 9 – Блок-схема используемой функции

Изображение выглядит как текст, снимок экрана, диаграмма, линия

Автоматически созданное описание

Рисунок 10 – Блок-схема используемой функции

Изображение выглядит как текст, снимок экрана, Шрифт, линия

Автоматически созданное описание

Рисунок 11 – Блок-схема используемой функции

Изображение выглядит как текст, снимок экрана, Шрифт, линия

Автоматически созданное описание

Рисунок 12 – Блок-схема используемой функции

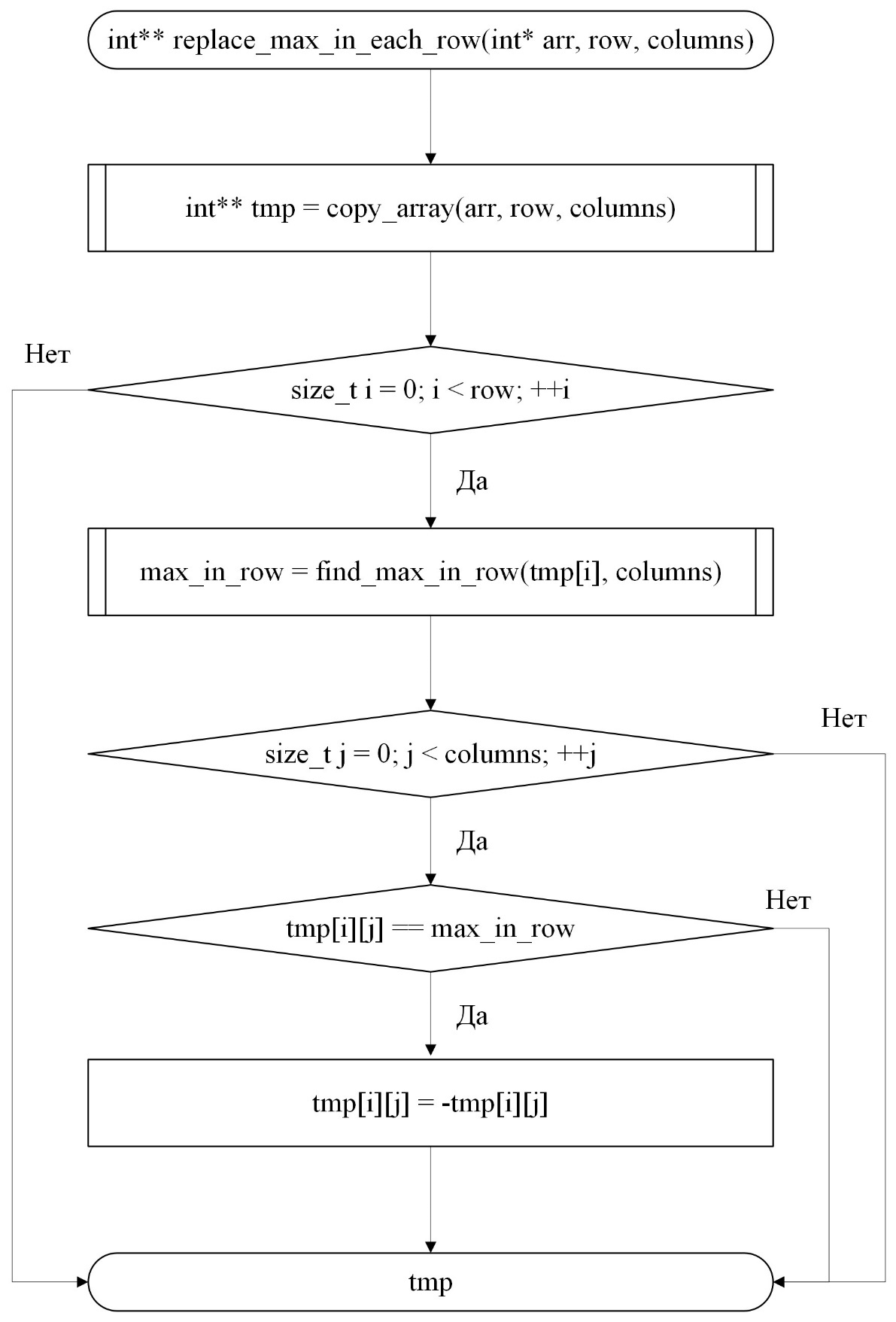


Рисунок 13 – Блок-схема используемой функции

Изображение выглядит как текст, чек, Шрифт, документ

Автоматически созданное описание

Рисунок 14 – Блок-схема используемой функции

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Рисунок 15 – Блок-схема используемой функции

Изображение выглядит как текст, снимок экрана, Шрифт, диаграмма

Автоматически созданное описание

Рисунок 16 – Блок-схема используемой функции

3.3 Текст программы на языке С задание 4–3

#include <errno.h>

#include <time.h>

#include <stdio.h>

#include <stdbool.h>

#include <locale.h>

#include <stdlib.h>

/\*\*

\* @brief Пользовательский ввод

\*/

enum random\_or\_keybord

{

Keyboard = 1,

Random = 2

};

/\*\*

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

\*/

void names\_of\_random\_and\_keyboard();

/\*\*

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

\* @param message - текст сообщения для пользователя

\* @return Значение размера

\*/

size\_t get\_size(char const\* message);

/\*\*

\* @brief Функция выделяющая память под двумерный массив

\* @param row - количество строк

\* @param columns - количество столбцов

\* @return Массив

\*/

int\*\* init\_array(const size\_t row, const size\_t columns);

/\*\*

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

\* @param row - количество строк

\* @param columns - количество столбцов

\* @param array - массив

\*/

void fill\_array(int\*\* array, size\_t row, size\_t columns);

/\*\*

\* @brief Функция считывающая ввод с клавиатуры

\* @param row - количество строк

\* @param columns - количество столбцов

\* @param array - массив

\*/

void fill\_keyboard(int\*\* array, size\_t row, size\_t columns);

/\*\*

\* @brief Функция заполняющая массив рандомными числами

\* @param row - количество строк

\* @param columns - количество столбцов

\*/

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

/\*\*

\* @brief Функция выводящая заполненный массив

\* @param row - количество строк

\* @param columns - количество столбцов

\* @param array - массив

\*/

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

/\*\*

\* @brief Функция принимающая и проверяющая значение на ввод

\* @param message - текст сообщения для пользователя

\* @return Значение

\*/

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

/\*\*

\* @brief Функция удаляющая массив

\* @param array - массив

\* @param row - количество строк

\*/

void delete\_array(int\*\*\* arr, size\_t row);

/\*\*

\* @brief Функция которая ищет максимальное значение в строке массива

\* @param row - строка массива

\* @param columns - количество столбцов

\* @return Максимальное значение в строке массива

\*/

int find\_max\_in\_row(int const\* const row, size\_t columns);

/\*\*

\* @brief Функция копирующая массив

\* @param arr - массив

\* @param row - количество строк

\* @param columns - количество столбцов

\* @return Возвращает копию массива

\*/

int\*\* copy\_array(const int\* const\* const arr, size\_t row, size\_t columns);

/\*\*

\* @brief Заминить максимальный элемент в каждой строке на противоположный по знаку

\* @param arr - массив

\* @param row - количество строк

\* @param columns - количество столбцов

\* @return Возвращает изменённый массив

\*/

int\*\* replace\_max\_in\_each\_row(const int\* const\* const arr, size\_t row, size\_t columns);

/\*\*

\* @brief Функция создаёт одномерный массив размером m

\* @param columns - количество столбцов

\* @return Возвращает созданный одномерный массив

\*/

int\* init\_single\_array(size\_t columns);

/\*\*

\* @brief Функция ищет максимальный элемент

\* @param arr - массив

\* @param row - количество строк

\* @param columns - количество столбцов

\* @return Возвращает номер столбца максимального элемента

\*/

size\_t find\_last\_max\_column\_index(const int\* const\* const arr, size\_t row, size\_t columns);

/\*\*

\* @brief Функция вставляет элемент в строку массива

\* @param row - строка массива

\* @param elem - элемент который надо вставить

\* @param index - индекс после которго надо вставить элемент

\* @param columns - количество столбцов

\* @return Возвращает изменённую строку массива

\*/

int\* insert\_to\_row(const int\* const row, size\_t columns, int elem, size\_t index);

/\*\*

\* @brief Функция вставляющая после всех столбцов, содержащих максимальный элемент столбец из нулей

\* @param arr - массив

\* @param row - количество строк

\* @param columns - количество столбцов

\* @return Возвращает изменённый массив

\*/

int\*\* insert\_max(const int\* const\* const arr, size\_t row, size\_t columns);

/\*\*

\* @brief Функция выделяющая память под массив указателей

\* @param row - размер массива

\* @return Возвращает массив указателей

\*/

int\*\* init\_array\_of\_ptrs(size\_t row);

/\*\*

\* @brief Функция копирующая массив src в массив dst размера size

\* @param size - размер массивов

\* @param dst - массив назначения

\* @param src - исходный массив

\*/

void copy(int\* dst, int\* src, size\_t size);

/\*\*

\* @brief Точка входа в программу

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

\*/

int main() {

setlocale(LC\_ALL, "RU");

size\_t row = get\_size("Введите количество строк в массиве: ");

size\_t columns = get\_size("Введите количество столбцов в массиве: ");

int\*\* arr = init\_array(row, columns);

fill\_array(arr, row, columns);

puts("Исходный массив:\n");

print\_array(arr, row, columns);

int\*\* first\_arr = replace\_max\_in\_each\_row(arr, row, columns);

puts("Первое задание:\n");

print\_array(first\_arr, row, columns);

int\*\* second\_arr = insert\_max(arr, row, columns);

puts("Второе задание:\n");

print\_array(second\_arr, row, columns + 1);

delete\_array(&arr, row);

delete\_array(&first\_arr, row);

delete\_array(&second\_arr, row);

return 0;

}

void names\_of\_random\_and\_keyboard()

{

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

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

}

size\_t get\_size(char const\* message)

{

int size = get\_value(message);

if (size < 0)

{

perror("Неверный размер!\n");

abort();

}

return (size\_t)size;

}

int\*\* init\_array(const size\_t row, const size\_t columns)

{

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

if (arr == NULL)

{

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

}

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

{

arr[i] = init\_single\_array(row);

}

return arr;

}

void fill\_array(int\*\* array, size\_t row, size\_t columns)

{

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

names\_of\_random\_and\_keyboard();

enum random\_or\_keybord choice = get\_value("Выберите нужный вариант: ");

switch (choice)

{

case Keyboard:

fill\_keyboard(array, row, columns);

break;

case Random:

fill\_random(array, row, columns);

break;

default:

perror("Неверный выбор!!\n");

}

}

void fill\_keyboard(int\*\* array, size\_t row, size\_t columns)

{

const int minimum\_limit = get\_value("Введите нижнюю границу массива: ");

const int maximum\_limit = get\_value("Введите верхнюю границу массива: ");

if (minimum\_limit >= maximum\_limit)

{

errno = EIO;

perror("Неверные границы!\n");

abort();

}

puts("Введите массив: ");

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

{

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

{

int value = get\_value("");

if (value < minimum\_limit || value > maximum\_limit)

{

errno = EIO;

perror("Ошибка ввода!\n");

abort();

}

array[i][j] = value;

}

}

}

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

{

const int minimum\_limit = get\_value("Введите нижнюю границу массива: ");

const int maximum\_limit = get\_value("Введите верхнюю границу массива: ");

unsigned int ttime = time(NULL);

srand(ttime);

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

{

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

{

array[i][j] = minimum\_limit + rand() % (maximum\_limit - minimum\_limit + 1);

}

}

}

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

{

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

{

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

{

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

}

puts("\n");

}

}

int get\_value(const char\* message)

{

int value;

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

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

if (res != 1)

{

errno = EIO;

perror("Неверное значение\n");

abort();

}

return value;

}

void delete\_array(int\*\*\* arr, size\_t row)

{

if (arr != NULL)

{

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

{

if (arr[i] != NULL)

{

free(arr[i]);

}

}

free(arr);

arr = NULL;

}

}

int find\_max\_in\_row(int const\* const row, size\_t columns)

{

int max = row[0];

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

{

if (row[i] > max)

{

max = row[i];

}

}

return max;

}

int\*\* copy\_array(const int\* const\* const arr, size\_t row, size\_t columns)

{

int\*\* new\_arr = init\_array(row, columns);

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

{

memcpy(new\_arr[i], arr[i], columns \* sizeof(int));

}

return new\_arr;

}

int\*\* replace\_max\_in\_each\_row(const int\* const\* const arr, size\_t row, size\_t columns)

{

int\*\* tmp = copy\_array(arr, row, columns);

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

{

int max\_in\_row = find\_max\_in\_row(tmp[i], columns);

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

{

if (tmp[i][j] == max\_in\_row)

{

tmp[i][j] = -tmp[i][j];

}

}

}

return tmp;

}

int\* init\_single\_array(size\_t columns)

{

int\* array\_row = malloc(columns \* sizeof(int));

if (array\_row == NULL)

{

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

}

return array\_row;

}

size\_t find\_last\_max\_column\_index(const int\* const\* const arr, size\_t row, size\_t columns)

{

int max = arr[0][0];

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

{

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

{

if (arr[i][j] >= max)

{

max = arr[i][j];

}

}

}

size\_t column\_index = 0;

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

{

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

{

if (arr[i][j] == max && j > column\_index)

{

column\_index = j;

}

}

}

return column\_index;

}

int\*\* init\_array\_of\_ptrs(size\_t row)

{

int\*\* ptr\_array = (int\*\*)malloc(row \* sizeof(int\*));

if (ptr\_array == NULL)

{

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

}

return ptr\_array;

}

void copy(int\* dst, int\* src, size\_t size)

{

if (dst != NULL && src != NULL)

{

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

{

dst[i] = src[i];

}

}

}

int\* insert\_to\_row(const int\* const row, size\_t columns, int elem, size\_t index)

{

int\* new\_array\_row = init\_single\_array(columns + 1);

copy(new\_array\_row, row, index + 1);

new\_array\_row[index + 1] = elem;

copy(new\_array\_row + index + 2, row + index + 1, columns - index - 1);

return new\_array\_row;

}

int\*\* insert\_max(const int\* const\* const arr, size\_t row, size\_t columns)

{

int\*\* new\_arr = init\_array\_of\_ptrs(row);

size\_t index = find\_last\_max\_column\_index(arr, row, columns);

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

{

new\_arr[i] = insert\_to\_row(arr[i], columns, 0, index);

}

return new\_arr;

}

3.4 Результаты выполнения программы 4–3

Результаты выполнения программы в C представлена ниже (Рисунок 17 и 18)

Изображение выглядит как текст, снимок экрана, меню

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, Шрифт, клавиатура

Автоматически созданное описание

Рисунок 17 – Результаты выполнения программы

Изображение выглядит как снимок экрана, текст, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

Изображение выглядит как снимок экрана, текст, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

Рисунок 18 – Результаты выполнения программы

3.5 Отметка о выполнении задания в веб-хостинге системы контроля версий задание 4–3

Ниже представлено доказательство того, что задание 4–3 было принято. (Рисунок 19)

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 19 – Approve задачи 4–3