

МИНОБРНАУКИ РОССИИ

федеральное государственное бюджетное образовательное учреждение высшего образования

«Санкт-Петербургский государственный технологический институт (технический университет)»

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| УГСН | | 09.00.00 | Информатика и вычислительная техника | | |
| Уровень образования | |  | Высшее образование – бакалавриат | | |
| Форма обучения | |  | Очная | | |
| Факультет | |  | Информационных технологий  и управления | | |
| Кафедра | |  | Систем автоматизированного проектирования и управления | | |
| Учебная дисциплина | |  | Информационные технологии  и программирование | | |
| Курс | I | | | Группа | 4304 |

Отчёт по контрольной работе № 2

Вариант № 26

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Исполнитель: |  |  |  |  |
| обучающийся группы 4304 |  |  |  | Рыбник Всеволод Сергеевич |
|  |  | (дата, подпись) |  |  |
|  |  |  |  |  |
| Проверил: |  |  |  | Корниенко Иван Григорьевич |
|  |  | (дата, подпись) |  | Макарук Роман Валерьевич |
|  |  |  |  | Федин Алексей Константинович |

# СОДЕРЖАНИЕ

[1 Задание №1 3](#_Toc51078351)

[1.1 Цель работы 3](#_Toc51078352)

[1.2 Постановка задачи 3](#_Toc51078353)

[1.3 Описание хода выполнения 3](#_Toc51078354)

[1.4 Блок-схема алгоритма решения задачи](#_Toc51078355) 3

[1.5 Исходный код полученного программного решения 5](#_Toc51078356)

[1.6 Тестирование](#_Toc51078357) 6

[1.7 Выводы по заданию №1](#_Toc51078358) 6

[2 Задание №2](#_Toc51078359) 7

[2.1 Цель работы](#_Toc51078360) 7

[2.2 Постановка задачи](#_Toc51078361) 7

[2.3 Описание хода выполнения](#_Toc51078362) 7

[2.4 Блок-схема алгоритма решения задачи](#_Toc51078363) 7

[2.5 Исходный код полученного программного решения](#_Toc51078364) 9

[2.6 Тестирование](#_Toc51078365) 13

[2.7 Выводы по заданию №2](#_Toc51078366) 13

[3 Задание №3](#_Toc51078367) 15

[3.1 Цель работы](#_Toc51078368) 15

[3.2 Постановка задачи](#_Toc51078369) 15

[3.3 Описание хода выполнения](#_Toc51078370) 15

[3.4 Блок-схема алгоритма решения задачи](#_Toc51078371) 15

[3.5 Исходный код полученного программного решения 1](#_Toc51078372)7

[3.6 Тестирование](#_Toc51078373) 20

[3.7 Выводы по заданию №3](#_Toc51078374) 21

# 1 Задание №1

## 1.1 Цель работы

Реализовать рекуррентные последовательности.

## 1.2 Постановка задачи

Разработать программу вычисления рекуррентной последовательности

и вывода результата на экран, с учётом дополнительных условий варианта

задания. Программное решение поставленной задачи не должно использовать

массивы, т.е. массивами пользоваться запрещено.

|  |  |
| --- | --- |
| λ1 = 1.5  λ2 = 2 | (1) |

## 1.3 Описание хода выполнения

## Для решения данной задачи было необходимо разобраться с математическим значение Сигмы, способами переноса ее логики в среду разработки.

## 1.4 Блок-схема алгоритма решения задачи

Блок схема задачи №2.1:

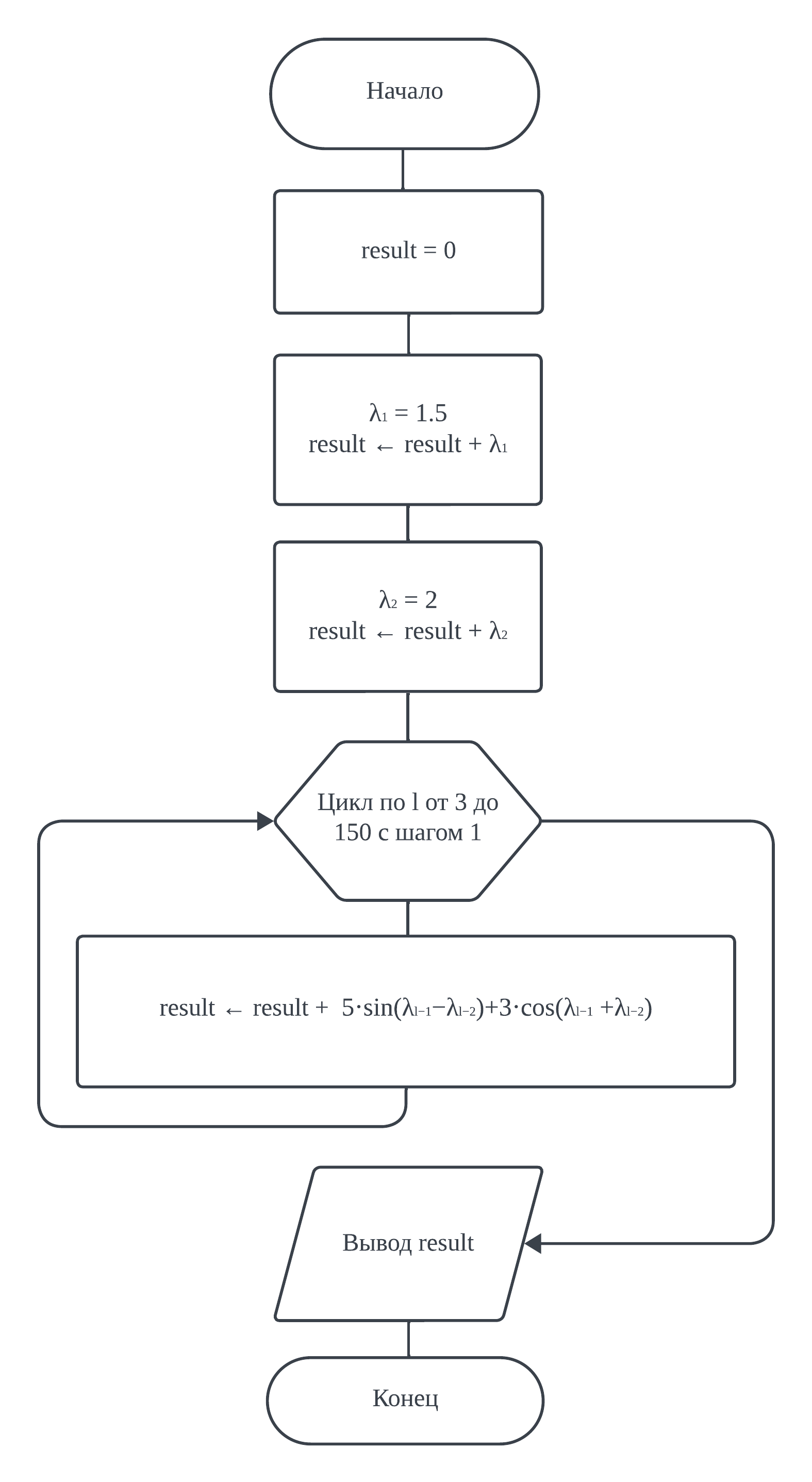


Рисунок 1 – Блок-схема алгоритма решения задачи №2.1

## 1.5 Исходный код полученного программного решения

**Код основного файла first\_task.c:**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Название: first\_task.c \*

\* Задание: Четвертая программа в осеннем семестре \*

\* Автор: в.с. рыбник, СПбГТИ (ТУ), 2023 \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include <stdlib.h>

#include <stdio.h>

#include "first\_module.h"

#define SIGMA\_AMOUNT 150

int main(void)

{

int loop\_indicator = 1;

while (loop\_indicator)

{

printf(" Vsevolod Rybnik Test 2 task 1 var. 26\n");

printf(" - Result: %f \n", formula(SIGMA\_AMOUNT));

printf(" - Wanna see output again? (`any num` - yep, 0 - nope): ");

scanf("%d", &loop\_indicator);

}

return EXIT\_SUCCESS;

}

**Код доп. файла — first\_formula.c:**

#include <math.h>

double formula(int iteration\_max)

{

double result = 3.5, result\_past\_2\_moves = 1.5, result\_past\_1\_moves = 2, dif\_move = 0.0;

int iteration = 1;

for (iteration = 3; iteration <= iteration\_max; iteration++)

{

dif\_move = 2.5 \* sin(result\_past\_1\_moves - result\_past\_2\_moves) + 3 \* cos(result\_past\_1\_moves + result\_past\_2\_moves); // dif\_move - иной ход(different)

result += dif\_move;

result\_past\_2\_moves = result\_past\_1\_moves;

result\_past\_1\_moves = dif\_move;

}

return result;

}

**Код хедера — first\_module.h:**

double formula(int iteration\_max);

## 1.6 Тестирование

Результат тестирования приведён на рисунке 2.

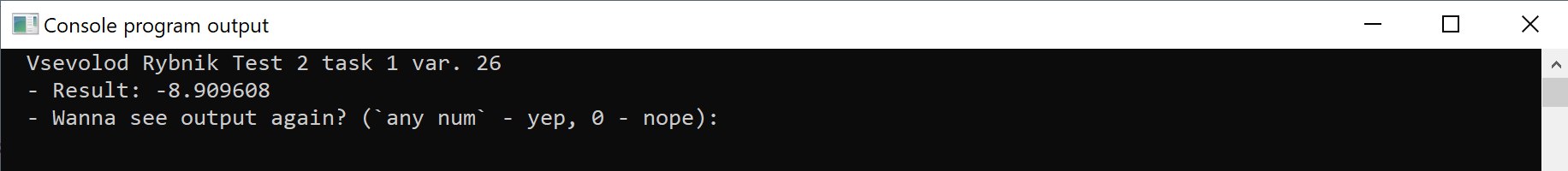


Рисунок 2 – Экранная копия результата работы разработанной программы задания №2.1

## 1.7 Выводы по заданию №1

В ходе выполнения был изучен математический смысл Сигмы, а также различные способы ее реализации в программировании.

# 2 Задание №2

## 2.1 Цель работы

Разработка и операции над одномерными массивами.

## 2.2 Постановка задачи

Разработать программу в соответсвии с данным условием:

«Даны действительные числа (Ci)Ni=1=C1, C2,…, CN, где N=25. Получить последовательности (Xj)pj=1=X1,…, Xp и (Yz)Nz=p+1=y1,…, yN , где p – порядковый номер члена (Сi)Ni=1 полученный путём разбиения пополам интервала между порядковыми номерами Сmn=min((Ci)N i=1) и Сmx=max((Ci)Ni=1). Последовательность (Сi)Ni=1 не сортировать.»

## 2.3 Описание хода выполнения

Сначала была составлена блок-схема будущего алгоритма для более четкого понимания требований задачи. Затем были изучены способы реализации. Впервые мной был использован оператор switch-case для элементов меню, разработана структура — Хэш-таблица. А также мною были изучены способы передачи методов в качестве аргументов функции в Objective C.

## 2.4 Блок-схема алгоритма решения задачи

На рисунке 3 представлена блок-схема алгоритма решения задачи №2.2.

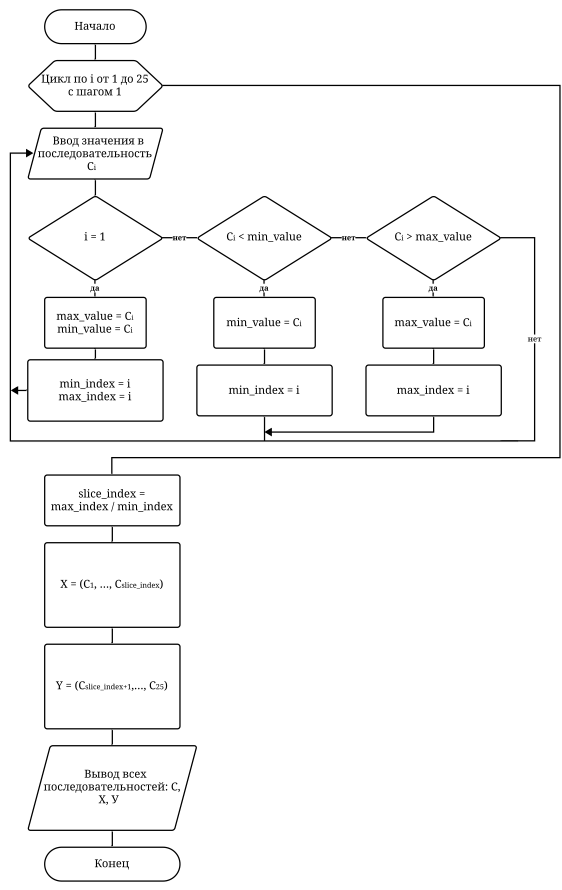


Рисунок 3 – Блок-схема алгоритма решения задачи №2.2

## 2.5 Исходный код полученного программного решения

**Код основного файла — second\_task.c:**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Название: second\_task.c \*

\* Задание: Пятая программа в осеннем семестре \*

\* Автор: в.с. рыбник, СПбГТИ (ТУ), 2023 \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include <string.h>

#include <time.h>

#include <math.h>

#include "second\_interface\_modules.h"

#define ARRAY\_SIZE 25

enum MENU {ManualInput=1, RandomInput=2, Quit =3};

void Calculation(double order[], HashTable\* min\_max\_table)

{

output("Original", 0, ARRAY\_SIZE, order);

int slice = (((int) get(min\_max\_table, "min\_index")) + ((int) get(min\_max\_table, "max\_index"))) /2;

output("X", 0, slice, order);

output("Y", slice, ARRAY\_SIZE, order);

}

double random\_value(char \*name, int i)

{

return round(-9999 + rand()%(10000 + 9800))/100;

}

int main(void)

{

srand(time(NULL));

int options = 1, loop\_indicator = 1;

double order[ARRAY\_SIZE];

printf(" Vsevolod Rybnik test 2 task 3 var 26\n");

while (loop\_indicator)

{

HashTable\* min\_max\_table = createHashTable();

puts(" 1 - Manuale inpute\n 2 - Randome inpute\n 3 - Quite");

options = GetInt();

switch (options)

{

case ManualInput:

input(order, GetDouble, min\_max\_table, "Specify array element №", ARRAY\_SIZE);

Calculation(order, min\_max\_table);

destroyHashTable(min\_max\_table);

continue;

case RandomInput:

input(order, random\_value, min\_max\_table, NULL, ARRAY\_SIZE);

Calculation(order, min\_max\_table);

destroyHashTable(min\_max\_table);

continue;

case Quit:

return EXIT\_SUCCESS;

default:

puts("Dis value is not akceptabele\n");

}

}

}

**Код дополнительного файла с методами структуры — хэш-таблицы — second\_hash\_table.c:**

#include <stdint.h>

#include <stdlib.h>

#include <stdio.h>

#include <string.h>

#define TABLE\_SIZE 4

#define FNV\_OFFSET 1052960442348

#define FNV\_PRIME 1099511628234

typedef struct {

char\* key;

double value;

} Entry;

typedef struct {

Entry\* entries;

int size;

} HashTable;

int hash(char\* key) {

int hashValue = 0;

for (int i = 0; i < (int)strlen(key); i++) {

hashValue ^= (int)(unsigned char) key[i];

hashValue += key[i];

}

return hashValue;

}

HashTable\* createHashTable(void) {

HashTable\* hashTable = (HashTable\*) malloc(sizeof(HashTable));

hashTable->entries = (Entry\*) malloc(TABLE\_SIZE \* sizeof(Entry));

hashTable->size = TABLE\_SIZE;

for (int i = 0; i < TABLE\_SIZE; i++) {

hashTable->entries[i].key = NULL;

}

return hashTable;

}

void insert(HashTable\* hashTable, char\* key, double value) {

int index = hash(key);

while (hashTable->entries[index].key!= NULL) {

if (strcmp(hashTable->entries[index].key, key) == 0) {

hashTable->entries[index].value = value;

return;

}

index = (index + 1) % hashTable->size;

}

hashTable->entries[index].key = \_strdup(key);

hashTable->entries[index].value = value;

}

double get(HashTable\* hashTable, char\* key) {

int index = hash(key);

while (hashTable->entries[index].key!= NULL) {

if (strcmp(hashTable->entries[index].key, key) == 0) {

return hashTable->entries[index].value;

}

index = (index + 1) % hashTable->size;

}

return 0;

}

HashTable\* max\_min\_finder(double element, HashTable\* min\_max\_table, int i)

{

insert(min\_max\_table, "min", ((element < get(min\_max\_table, "min"))? element : (i == 1)? element : get(min\_max\_table, "min")));

insert(min\_max\_table, "max", ((element > get(min\_max\_table, "max"))? element : (i == 1)? element : get(min\_max\_table, "max")));

insert(min\_max\_table, "min\_index", ((element == get(min\_max\_table, "min"))? i : get(min\_max\_table, "min\_index")));

insert(min\_max\_table, "max\_index", ((element == get(min\_max\_table, "max"))? i : get(min\_max\_table, "max\_index")));

return min\_max\_table;

}

void destroyHashTable(HashTable\* hashTable) {

for (int i = 0; i < hashTable->size; i++) {

if (hashTable->entries[i].key!= NULL) {

free(hashTable->entries[i].key);

hashTable->entries[i].key = NULL;

}

}

free(hashTable->entries);

hashTable->entries = NULL;

free(hashTable);

hashTable = NULL;

}

**Код доп. файла с кодом различных интерфейсов — second\_interface.h:**

#include <stdio.h>

#include <stdbool.h>

#include "second\_struct.h"

int GetInt(void)

{

char temprem, tempclear;

int input = 0;

while(true)

{

temprem=0;

tempclear=0;

if((!scanf("%d%c",&input ,&temprem))|| temprem != '\n')

{

printf(" - Error: Invalid value for int variables.\nOne more time: ");

while(tempclear != '\n')

scanf("%c",&tempclear);

}

else

return input;

}

}

double GetDouble(char \*label, int position)

{

char temprem, tempclear; // временный остаток

double input = 0;

while(true)

{

printf("%s%d: ", label, position+1);

temprem = 0;

tempclear = 0;

if((!scanf("%lf%c",&input ,&temprem))|| temprem != '\n')

{

printf(" - Error: Invalid value for double variables.\n - One more time: ");

while(tempclear != '\n')

scanf("%c",&tempclear);

}

else

return input;

}

}

void output(char \*name, int left\_border, int right\_border, double order[])

{

printf("%s order: (", name);

for (int i = left\_border; i < right\_border; i++)

printf("%0.2f, ", order[i]);

printf(")\n");

}

double\* input(double order[], double (\*input\_type)(char \*, int), HashTable\* min\_max\_table, char \*label, int ARRAY\_SIZE)

{

for (int i = 0; i < ARRAY\_SIZE; i++)

{

order[i] = input\_type(label, i);

min\_max\_table = max\_min\_finder(order[i], min\_max\_table, i);

}

return order;

}

**Код хедера — second\_hash\_table\_module.h:**

typedef struct {

char\* key;

double value;

} Entry;

typedef struct {

Entry\* entries;

int size;

} HashTable;

HashTable\* max\_min\_finder(double element, HashTable\* min\_max\_table, int i);

void destroyHashTable(HashTable\* hashTable);

double get(HashTable\* hashTable, char\* key);

void insert(HashTable\* hashTable, char\* key, double value);

HashTable\* createHashTable(void);

**Код хедера — second\_interface\_module.h:**

#include "second\_hash\_table\_module.h"

int GetInt(void);

double GetDouble(char \*label, int position);

void output(char \*name, int left\_border, int right\_border, double order[]);

double\* input(double order[], double (\*input\_type)(char \*, int), HashTable\* min\_max\_table, char \*label, int ARRAY\_SIZE);

## 2.6 Тестирование

Тестирование было проведено на рандомных значениях:

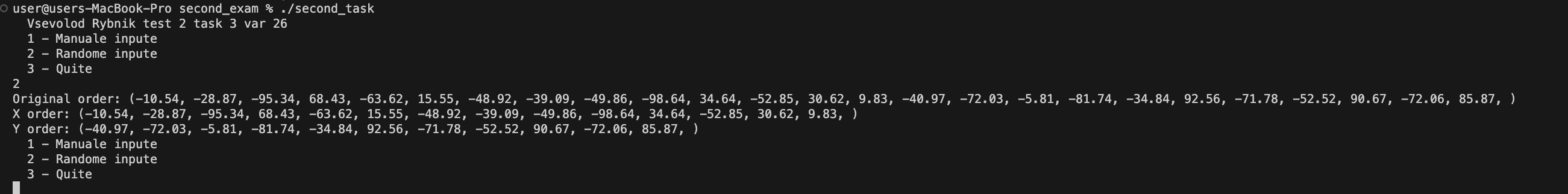


Рисунок 4 – Экранная копия результата работы разработанной программы задания 1.2

## 2.7 Выводы по заданию №2

В процессе выполнения задания было изучено много нового об устройстве хэш-таблиц в C — и успешно применены на практике (хоть и понадобилась она лишь для замены 8 одинаковых строчек в коде….). При разработке Хэш-Таблицы был изучены различные способы хэширования значений (Jenkins hash funtion, FNV-1a и др), применить их оказалось невозможно из-за использования ими сторонних библиотек и ошибок при компиляции, поэтому был создан метод на основе этих методов, но все еще простой и понятный, убедившись в том, что метод не выдает одинаковые значения для распространенных значений-коллизий — было принято решение оставить его. Также освоены иные способы реализации экранного меню с помощью значений enum и оператора switch-case.

# 3 Задание №3

## 3.1 Цель работы

Разработать двумерный массив и провести над ним определенные действия.

## 3.2 Постановка задачи

Разработать программу решения поставленной задачи (в соответствии с вариантом) и вывода результата на экран.

Текст задачи: «Дана матрица GN×M, где N=6; M=6. Элемент главной диагонали в каждой строке G(i,) заменить суммой элементов Si, расположенных за ним (если элемент на главной диагонали не равен нулю). Элементы главной диагонали сохранить в векторе a =(g1,1, g2,2,…, gi,i). Вывести исходную и преобразованную матрицы G, полученный вектор a. i=1, 2,…, N, если N≤M и i=1, 2,…, M, если N >M.»

## 3.3 Описание хода выполнения

Для разработки данного алгоритма было необходимо детально изучить структуры в Objective C, способы работы с выделением памяти и последующим ее освобождением. Дополнительно мной были изучены способы визуального оформления таблиц для вывода матриц.

## 3.4 Блок-схема алгоритма решения задачи

Блок схема к задаче №2.3:

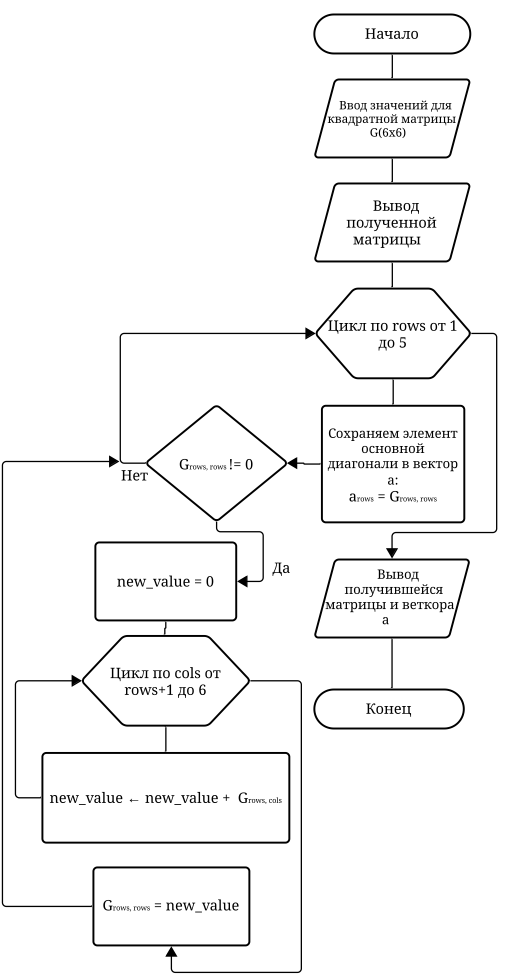


Рисунок 5 – Блок-схема алгоритма решения задачи №2.3

## 3.5 Исходный код полученного программного решения

**Код основного файла third\_task.c:**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Название: third\_task.c \*

\* Задание: Шестая программа в осеннем семестре \*

\* Автор: в.с. рыбник, СПбГТИ (ТУ), 2023 \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include <stdlib.h>

#include <time.h>

#include "third\_struct.h"

#define cols\_amount 6

#define rows\_amount 6

void main\_algorithm(Array array)

{

srand(time(NULL));

int a[cols\_amount];

for (int row = 0; row < array.rows-1; row++) {

int sum\_values = 0;

a[row] = array.data[row][row];

if (a[row] != 0){

for (int col = (row + 1); col < array.cols; col++) {

sum\_values += array.data[row][col];

}

array.data[row][row] = sum\_values;

}

}

printf(" - Modificated Array: \n");

output(array);

printf(" - Vector A (here placed old values of main diagonal): (");

for(int i = 0; i < cols\_amount; ++i)

{

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

}

printf(")\n");

}

int main(void)

{

int loop\_indicator = 1, rand\_man\_indicator = 1;

Array array;

printf(" Vsevolod Rybnik test 2 task 3 var 26\n");

while (loop\_indicator)

{

printf(" - Wanna specify values yourself or get random ones? (`any num` - random, 0 - manually): ");

rand\_man\_indicator = GetInt();

array = make\_array(rows\_amount, cols\_amount, rand\_man\_indicator);

printf(" - Original Array: \n");

output(array);

main\_algorithm(array);

clean\_space(&array);

printf(" - Wanna new data input? (`any num` - yep, 0 - nope): ");

scanf("%d", &loop\_indicator);

}

return EXIT\_SUCCESS;

}

**Код файл доп модуля third\_struct.c:**

#include <math.h>

#include <locale.h>

#include <stdio.h>

#include "third\_interface.h"

#define upper\_left\_border 218

#define upper\_right\_border 191

#define lower\_left\_border 192

#define lower\_right\_border 217

#define underline 196

#define underline\_amount 38

#define aside\_border 179

struct TwoDimensionalArray {

int rows;

int cols;

int \*\*data;

};

typedef struct TwoDimensionalArray Array;

Array random\_values\_for\_array(Array array)

{

for (int row = 0; row < array.rows; row++) {

for (int col = 0; col < array.cols; col++) {

array.data[row][col] = -99 + rand()%(100 + 98);

}

}

return array;

}

Array make\_array(int rows, int cols, int rand\_man\_indicator)

{

Array array;

array.rows = rows;

array.cols = cols;

array.data = (int \*\*) malloc(array.rows \* sizeof(int \*));

for (int row = 0; row < array.rows; row++) {

array.data[row] = (int \*) malloc(array.cols \* sizeof(int));

}

if (rand\_man\_indicator){

return random\_values\_for\_array(array);

}

else

{

for (int row = 0; row < array.rows; row++) {

for (int col = 0; col < array.cols; col++) {

printf(" - Specify %d %d element of Matrix: ", row+1, col+1);

array.data[row][col] = GetInt();

}

}

}

return array;

}

void clean\_space(Array \*array)

{

for (int row\_index = 0; row\_index < array->rows; row\_index++) {

free(array->data[row\_index]);

array->data[row\_index] = NULL;

}

free(array->data);

array->data = NULL;

}

void output(Array array)

{

printf(" %c", upper\_left\_border);

for (int i = 1; i < underline\_amount; i++)

printf("%c", underline);

printf("%c\n", upper\_right\_border);

for (int row = 0; row < array.rows; row++) {

for (int col = 0; col < array.cols; col++) {

if (col == 0)

printf(" %c", aside\_border);

if (array.data[row][col] >= 0)

printf(" ");

double digits = floor(log10(abs(array.data[row][col]))) + 1;

printf("%\*s", digits == 2? 2 : digits == 3? 1 : 3, " ");

printf("%d ", array.data[row][col]);

}

printf(" %c\n", aside\_border);

}

printf(" %c", lower\_left\_border);

for (int i = 1; i < underline\_amount; i++)

printf("%c", underline);

printf("%c\n", lower\_right\_border);

}

**Код еще одного доп файла - third\_interace.c:**

#include <stdio.h>

#include <stdbool.h>

int GetInt(void)

{

char temprem, tempclear;

int input = 0;

while(true)

{

temprem=0;

tempclear=0;

if((!scanf("%d%c",&input ,&temprem))|| temprem != '\n')

{

printf(" - Error: Invalid value for int variables.\nOne more time: ");

while(tempclear != '\n')

scanf("%c",&tempclear);

}

else

return input;

}

}

**Код хедера — third\_interface.h:**

#include <stdio.h>

#include <stdlib.h>

int GetInt(void);

**Код хедера — third\_struct.h:**

#include "third\_interface.h"

typedef struct TwoDimensionalArray {

int rows;

int cols;

int \*\*data;

} Array;

Array random\_values\_for\_array(Array array);

Array make\_array(int rows, int cols, int rand\_man\_indicator);

void clean\_space(Array \*array);

void output(Array array);

## 3.6 Тестирование

Тестирование было проведено на рандомных значениях:

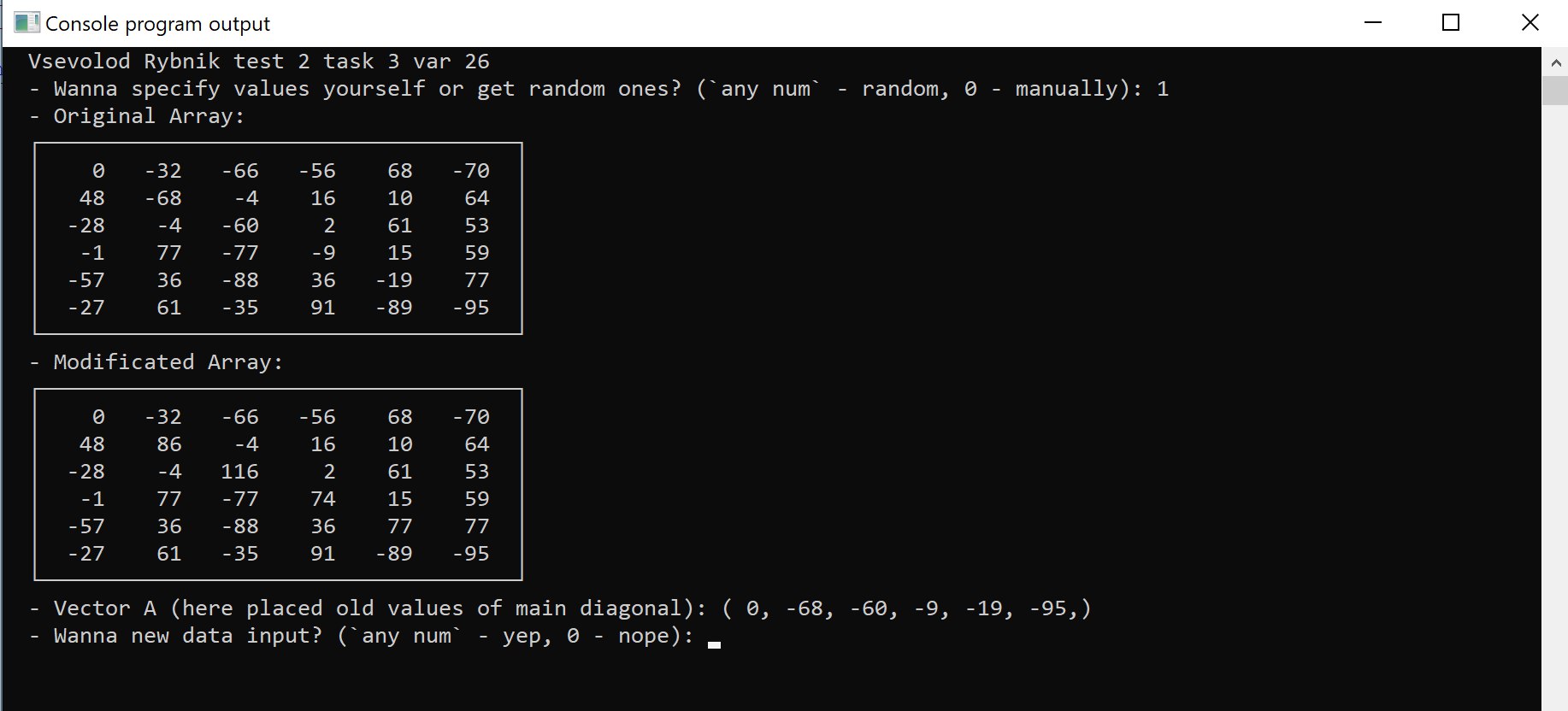


Рисунок 6 – Экранная копия результата работы разработанной программы задания №1.3

## 3.7 Выводы по заданию №3

В ходе задания были изучено много нового материала: кодировки различных символов, структуры — были изучены и впервые применены, способы работы с памятью в C — ее выделение и высвобождение после окончания всех манипуляций.