



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

\_\_\_\_\_  
(дата, подпись)

Рыбник Всеволод Сергеевич

Проверил:

\_\_\_\_\_  
(дата, подпись)

Корниенко Иван Григорьевич

Макарук Роман Валерьевич

Федин Алексей Константинович

Санкт-Петербург  
2023

## СОДЕРЖАНИЕ

1 Задание №1.....	3
1.1 Цель работы.....	3
1.2 Постановка задачи.....	3
1.3 Описание хода выполнения.....	3
1.4 Блок-схема алгоритма решения задачи.....	3
1.5 Исходный код полученного программного решения.....	5
1.6 Тестирование.....	6
1.7 Выводы по заданию №1.....	6
2 Задание №2.....	7
2.1 Цель работы.....	7
2.2 Постановка задачи.....	7
2.3 Описание хода выполнения.....	7
2.4 Блок-схема алгоритма решения задачи.....	7
2.5 Исходный код полученного программного решения.....	9
2.6 Тестирование.....	13
2.7 Выводы по заданию №2.....	13
3 Задание №3.....	15
3.1 Цель работы.....	15
3.2 Постановка задачи.....	15
3.3 Описание хода выполнения.....	15
3.4 Блок-схема алгоритма решения задачи.....	15
3.5 Исходный код полученного программного решения.....	17
3.6 Тестирование.....	20
3.7 Выводы по заданию №3.....	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.

$S = \sum_{l=1}^{150} \lambda_l$ $\lambda_1 = 1.5$ $\lambda_2 = 2$ $\lambda_l = 5 \cdot \sin(\lambda_{l-1} - \lambda_{l-2}) + 3 \cdot \cos(\lambda_{l-1} + \lambda_{l-2})$	(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:

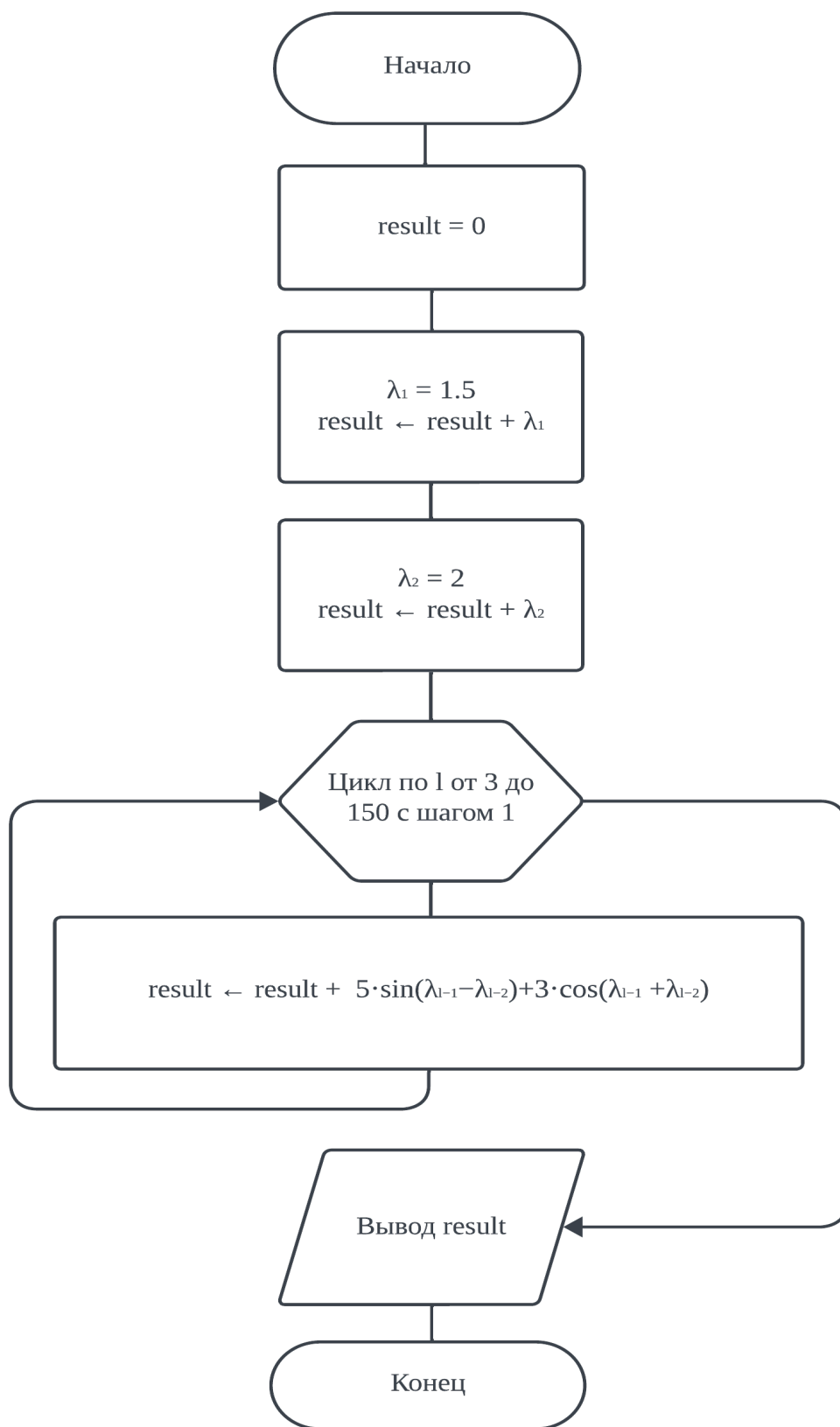


Рисунок 1 – Блок-схема алгоритма решения задачи №2.1

## 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>

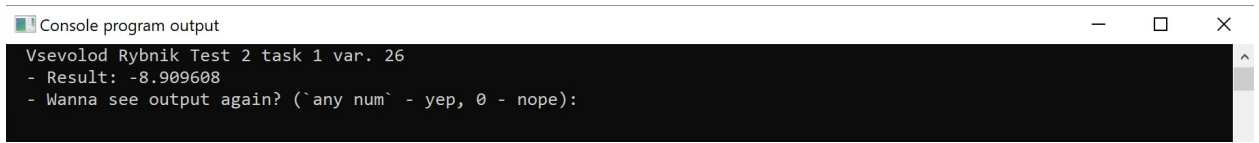
double formula(int iteration_max)
{
    double result = 3.5, result_past_2_moves = 1.5, result_past_1_moves = 2,
    dif_move = 0.0;
    int iteration = 1;
    for (iteration = 3; iteration <= iteration_max; iteration++)
    {
        dif_move = 2.5 * sin(result_past_1_moves - result_past_2_moves) + 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.

A screenshot of a console window titled "Console program output". The window has standard Windows window controls (minimize, maximize, close) in the top right corner. The output text is as follows:

```
Vsevolod Rybnik Test 2 task 1 var. 26  
- Result: -8.909608  
- Wanna see output again? (`any num` - yes, 0 - nope):
```

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 соответствии с данным условием:

«Даны действительные числа  $(C_i)_{i=1}^N = C_1, C_2, \dots, C_N$ , где  $N=25$ . Получить последовательности  $(X_j)_{j=1}^p = X_1, \dots, X_p$  и  $(Y_z)_{z=p+1}^N = y_1, \dots, y_N$ , где  $p$  – порядковый номер члена  $(C_i)_{i=1}^N$  полученный путём разбиения пополам интервала между порядковыми номерами  $C_{\min} = \min((C_i)_{i=1}^N)$  и  $C_{\max} = \max((C_i)_{i=1}^N)$ . Последовательность  $(C_i)_{i=1}^N$  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.

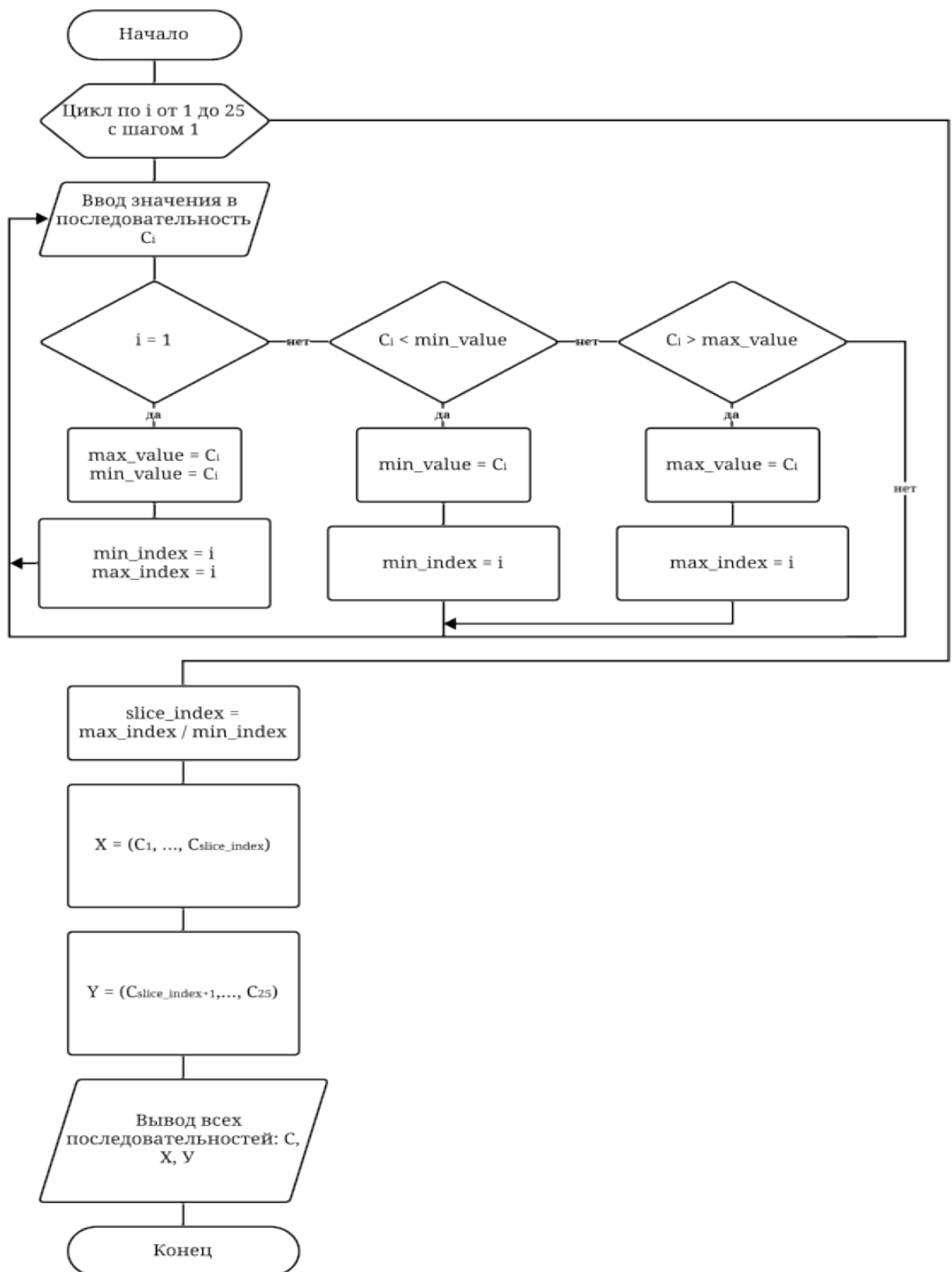


Рисунок 3 – Блок-схема алгоритма решения задачи №2.2



## 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 случайных значениях:

```

user@users-MacBook-Pro second_exam % ./second_task
Vsevolod Rybnik test 2 task 3 var 26
1 - Manual input
2 - Random input
3 - Quit
2
Original order: (-10.54, -28.87, -95.34, 68.43, -63.62, 15.55, -48.92, -39.09, -49.86, -98.64, 34.64, -52.85, 30.62, 9.83, -40.97, -72.03, -5.81, -81.74, -34.84, 92.56, -71.78, -52.52, 90.67, -72.06, 85.87, )
X order: (-10.54, -28.87, -95.34, 68.43, -63.62, 15.55, -48.92, -39.09, -49.86, -98.64, 34.64, -52.85, 30.62, 9.83, )
Y order: (-40.97, -72.03, -5.81, -81.74, -34.84, 92.56, -71.78, -52.52, 90.67, -72.06, 85.87, )
1 - Manual input
2 - Random input
3 - Quit
3

```

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 С — и успешно применены на практике (хоть и понадобилась она лишь для замены 8 одинаковых строчек в коде....). При разработке Хэш-Таблицы были изучены различные способы хэширования значений (Jenkins hash function, 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  $G_{N \times M}$ , где  $N=6$ ;  $M=6$ . Элемент главной диагонали в каждой строке  $G(i,)$  заменить суммой элементов  $S_i$ , расположенных за ним (если элемент на главной диагонали не равен нулю). Элементы главной диагонали сохранить в векторе  $a = (g_{1,1}, g_{2,2}, \dots, g_{i,i})$ . Вывести исходную и преобразованную матрицы  $G$ , полученный вектор  $a$ .  $i=1, 2, \dots, N$ , если  $N \leq M$  и  $i=1, 2, \dots, M$ , 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:

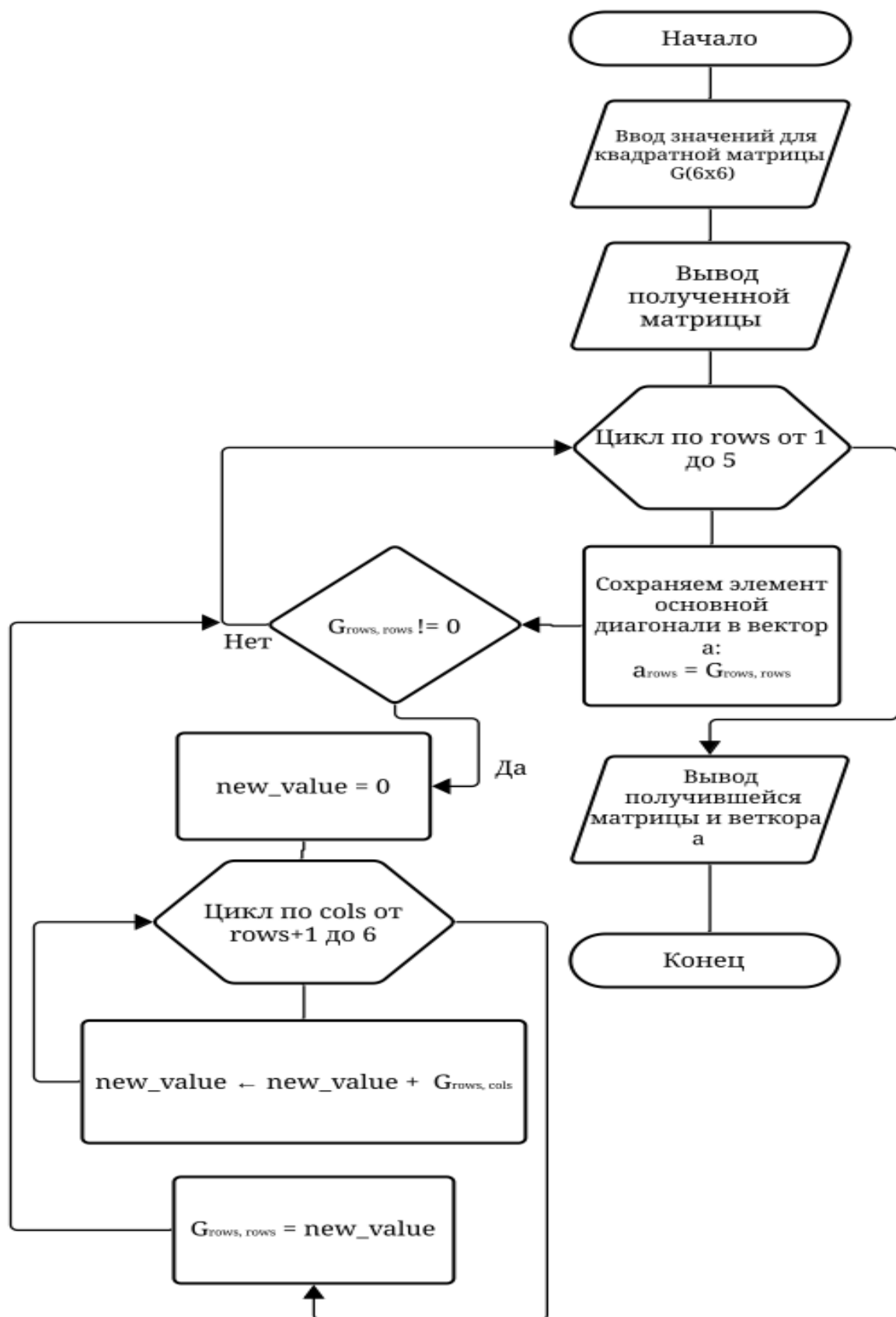


Рисунок 5 – Блок-схема алгоритма решения задачи №2.3



### 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(" - Modified 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 случайных значениях:

```

Console program output
Vsevolod Rybnik test 2 task 3 var 26
- Wanna specify values yourself or get random ones? (`any num` - random, 0 - manually): 1
- Original Array:
  0  -32  -66  -56  68  -70
48  -68  -4   16  10  64
-28  -4  -60   2   61  53
-1   77  -77  -9   15  59
-57  36  -88  36  -19  77
-27  61  -35  91  -89  -95
- Modified Array:
  0  -32  -66  -56  68  -70
48  86   -4   16  10  64
-28  -4  116   2   61  53
-1   77  -77  74   15  59
-57  36  -88  36  77  77
-27  61  -35  91  -89  -95
- Vector A (here placed old values of main diagonal): ( 0, -68, -60, -9, -19, -95,)
- Wanna new data input? (`any num` - yep, 0 - nope): _

```

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 С — ее выделение и высвобождение после окончания всех манипуляций.