



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 № 3**

**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.....	9
1.7 Выводы по заданию №1.....	9
2 Задание №2.....	10
2.1 Цель работы.....	10
2.2 Постановка задачи.....	10
2.3 Описание хода выполнения.....	10
2.4 Блок-схема алгоритма решения задачи.....	10
2.5 Исходный код полученного программного решения.....	12
2.6 Тестирование.....	15
2.7 Выводы по заданию №2.....	15
3 Задание №3.....	16
3.1 Цель работы.....	16
3.2 Постановка задачи.....	16
3.3 Описание хода выполнения.....	16
3.4 Блок-схема алгоритма решения задачи.....	16
3.5 Исходный код полученного программного решения.....	18
3.6 Тестирование.....	20
3.7 Выводы по заданию №3.....	20

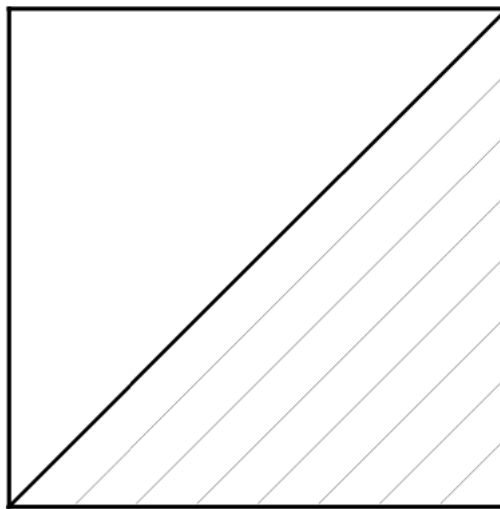
## 1 Задание №1

### 1.1 Цель работы

Динамическое распределение памяти.

### 1.2 Постановка задачи

Даны числа  $n$  и  $m$ , действительная матрица  $A_{n \times m}$ . Найти и вывести наименьшее  $E_{mn}$  и наибольшее  $E_{mx}$  из значений элементов, расположенных в заштрихованной части матрицы:



Вывести полученную матрицу  $A$ .

### 1.3 Описание хода выполнения

Для выполнения поставленной задачи была использована структура, выполняющая функции двумерного массива — матрицы.

### 1.4 Блок-схема алгоритма решения задачи

Блок-схема алгоритма №3.1:

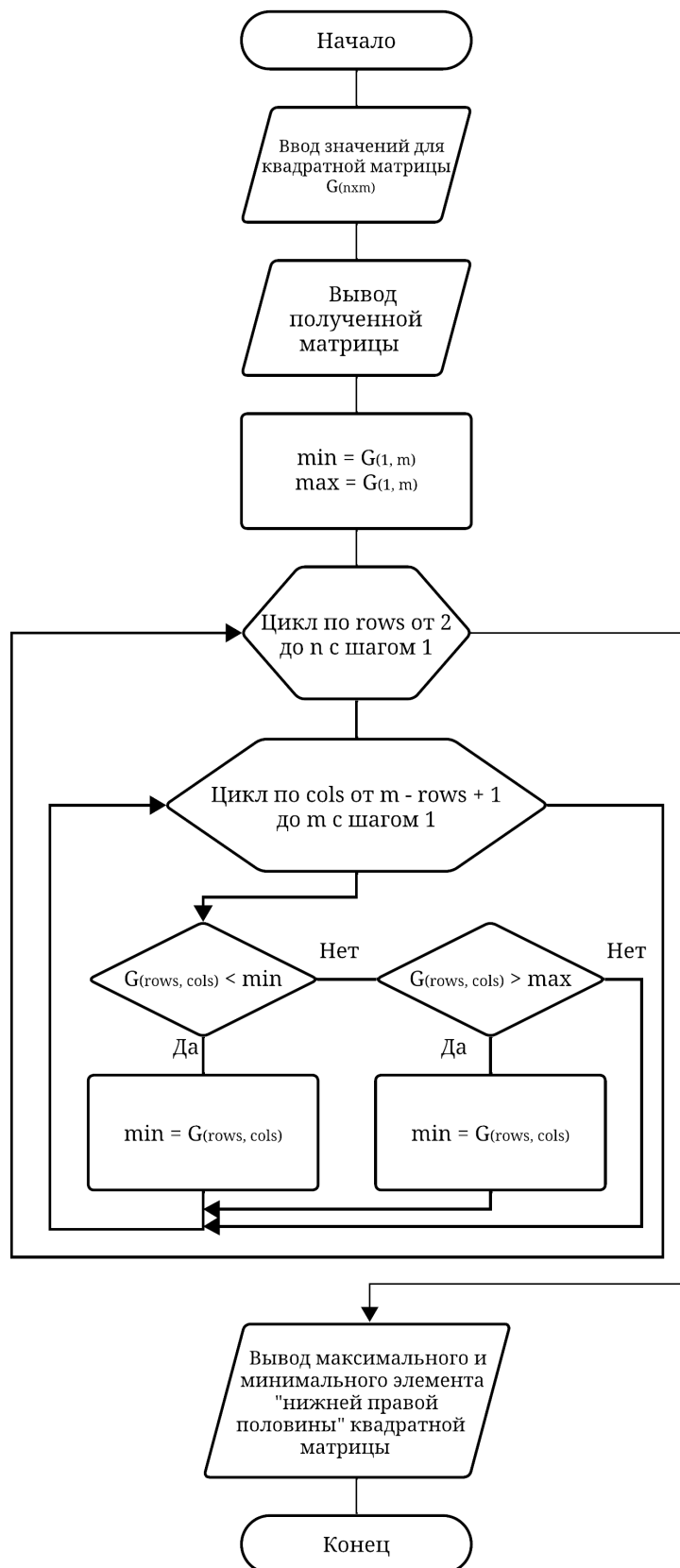


Рисунок 1 – Блок-схема алгоритма решения задачи №1

## 1.5 Исходный код полученного программного решения

Код файла **first\_task.c**:

```
#include <stdlib.h>
#include <time.h>
#include "first_algorithm.h"

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

int main(void)
{
    srand(time(NULL));
    int options = 1, loop_indicator = 1;
    int rows = 0, cols = 0;
    Array array;
    printf(" Vsevolod Rybnik test 3 task 1 var 26\n");
    while (loop_indicator)
    {
        int* result;
        puts("Specify matrix size: \n Rows: ");
        rows = get_int();
        puts(" Cols:");
        cols = get_int();
        puts(" 1 - Manual input\n 2 - Random input\n 3 - Quit");
        options = get_int();
        switch (options)
        {
            default:
                puts("Dis value is not akceptabele\n");
            case Quit:
                puts("Bye, see you later");
                return EXIT_SUCCESS;
            case ManualInput:
                array = make_array(rows, cols, false);
                output(array);
                result = max_min_finder(array);
                printf("Min Element: %d\nMax Element: %d\n", result[0],
result[1]);
                clean_space(&array);
                continue;
            case RandomInput:
                array = make_array(rows, cols, true);
                output(array);
                result = max_min_finder(array);
                printf("Min Element: %d\nMax Element: %d\n", result[0],
result[1]);
                clean_space(&array);
                continue;
        }
    }
}
```

**Код файла first\_struct.c:**

```
#include <math.h>
#include <locale.h>
#include <stdlib.h>
#include "first_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 aside_border 179

struct TwoDimensionalArray {
    int rows;
    int cols;
    int **data;
};

typedef struct TwoDimensionalArray Array;

Array make_array(int rows, int cols, bool rand_man_indicator)
{
    Array array;
    array.rows = rows;
    array.cols = cols;
    array.data = (int **) malloc(array.rows * sizeof(int *)); //
https://ufchgu.ru/blog/realloc-malloc-calloc-chem-zapolnjaet#:~:text=Функция
%20malloc%20принимает%20один%20аргумент,количество%20элементов%20и%20их
%20размер.
    for (int row = 0; row < array.rows; row++) {
        array.data[row] = (int *) malloc(array.cols * sizeof(int));
    }
    if (rand_man_indicator){
        for (int row = 0; row < array.rows; row++) {
            for (int col = 0; col < array.cols; col++) {
                array.data[row][col] = -99 + rand()%(100 + 98);
            }
        }
    }
    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] = get_double();
            }
        }
    }
    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);
    int underline_amount = 6 * array.cols + 2;
    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 файла first\_interface.c:**

```

#include <stdio.h>
#include <stdbool.h>

double get_double(void)
{
    char temprem, tempclear; // временный остаток
    double input = 0;
    while(true)
    {
        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;
    }
}

int get_int(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 файла first\_algorithm.c:**

```
#include "first_struct.h"
```

```

int* max_min_finder(Array array)
{
    int* result = (int*)malloc(2 * sizeof(int)); // 0 элемент - min, 1 - max
    result[0] = array.data[array.rows-1][array.cols-1];
    result[1] = array.data[array.rows-1][array.cols-1];
    for (int rows = 1; rows < array.rows; rows++)
    {
        for (int cols = array.cols - (rows + 1); cols < array.cols; cols++)
        {
            result[0] = (array.data[rows][cols] < result[0])?
array.data[rows][cols] : result[0];
            result[1] = (array.data[rows][cols] > result[1])?
array.data[rows][cols] : result[1];
        }
    }
    return result;
}

```

**Код файла first\_algorithm.h:**

```
#include "first_struct.h"
```

```
int* max_min_finder(Array array);
```

**Код файла first\_interface.h:**

```
#include <stdio.h>
#include <stdbool.h>
```

```
double get_double(void);
int get_int(void);
```

**Код файла first\_struct.h:**

```
#include <stdlib.h>
#include "first_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, bool rand_man_indicator);

void clean_space(Array *array);

void output(Array array);

```

## 1.6 Тестирование

Результат тестирования приведён на рисунке 2.

The screenshot shows a console window titled "Console program output". The text inside is as follows:

```

Vsevolod Rybnik test 3 task 1 var 26
Specify matrix size:
Rows:
2
Cols:
20
1 - Manual input
2 - Random input
3 - Quit
2

```

70	-88	-26	-68	32	-34	-72	2	-30	56	-48	44	-54	-64	60	30	-65	-33	-45	85
-69	-78	-32	-13	87	-17	-99	45	-73	-75	-51	-35	-49	-91	-75	27	-33	-39	23	-67

```

Min Element: -67
Max Element: 23
Specify matrix size:
Rows:

```

Рисунок 2 – Экранная копия результата работы разработанной программы задания №1

## 1.7 Выводы по заданию №1

В ходе выполнения поставленной задачи была улучшена функция вывода значений матрицы из предыдущей контрольной так, чтобы она поддерживала матрицы любого формата, также был освоен метод возвращения нескольких значений из метода без использования хэш-таблицы.

## **2 Задание №2**

### **2.1 Цель работы**

Работа с битами.

### **2.2 Постановка задачи**

Дана последовательность из 8 символов. Сравнить их младший и старший биты. Если они равны, то заменить старший нулём, младший – единицей, иначе заменить старший бит единицей, младший – нулём.

### **2.3 Описание хода выполнения**

Для выполнения данного задания была изучена новая тема «Кодировки Символов», освоены методы работы с битами в Objective C.

### **2.4 Блок-схема алгоритма решения задачи**

На рисунке 3 представлена блок-схема алгоритма решения задачи №2.

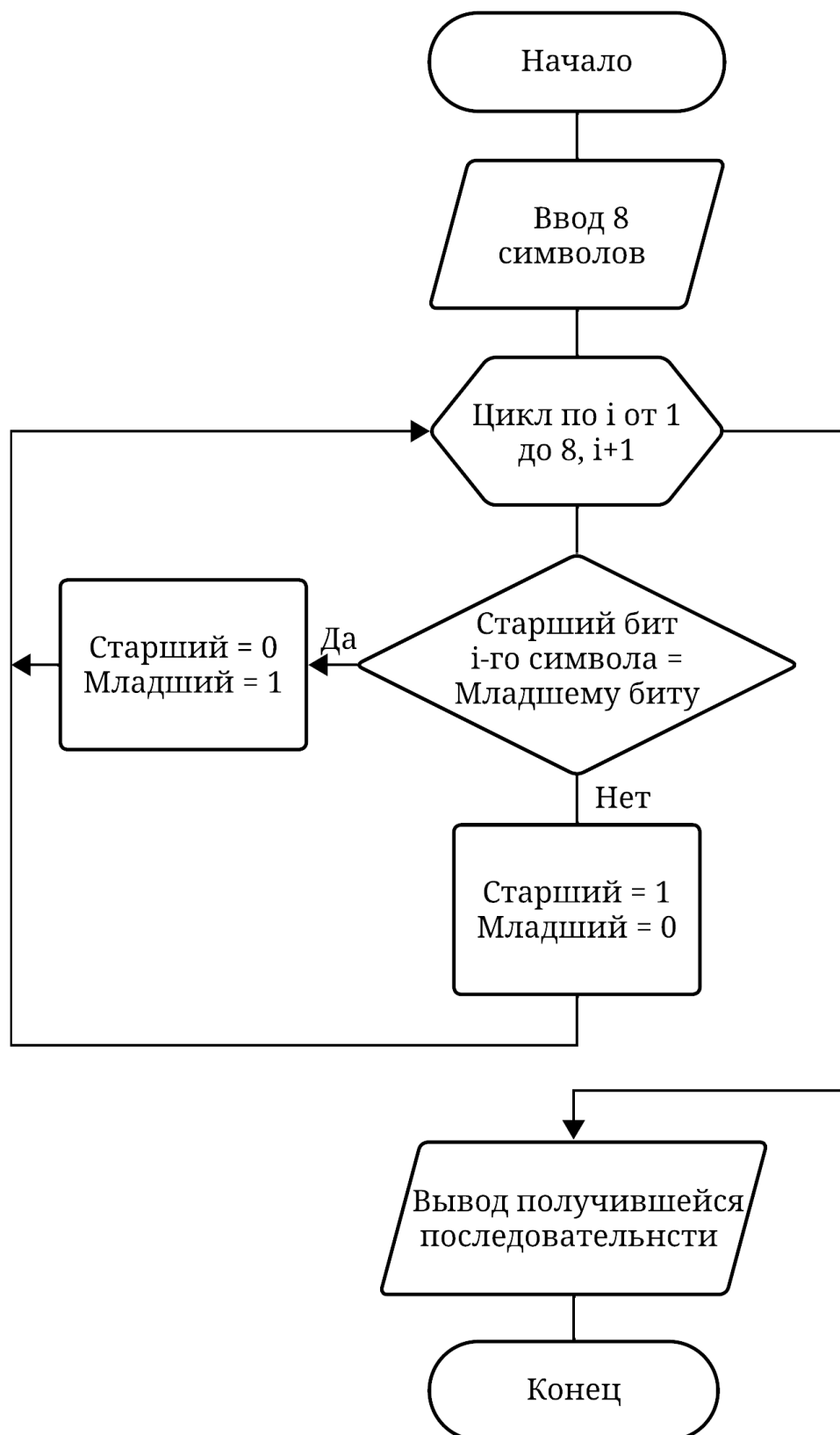


Рисунок 3 – Блок-схема алгоритма решения задачи №2

## 2.5 Исходный код полученного программного решения

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

```
#include <locale.h>
#include <time.h>
#include "second_interface.h"
#include "second_algorithm.h"

#define EL_AMOUNT 8

enum MENU {Man=1, Rand=2, Quit=3};

int main(void)
{
    setlocale (LC_CTYPE, "RU");
    srand(time(NULL));
    int loop_indicator = 1, options = 1;
    printf(" Vsevolod Rybnik test 3 task 2 var 26\n");
    while (loop_indicator)
    {
        char sequence[EL_AMOUNT];
        char* bits;
        puts(" 1 - Manual input\n 2 - Random input\n 3 - Quit");
        options = get_int();
        switch (options)
        {
            case Man:
                puts("Specify your symbols sequence: ");
                get_sequence(sequence, EL_AMOUNT);
                bits = algorithm(sequence, EL_AMOUNT);
                printf("...And result is -> %s\n", bits);
                printf("In ASCII - %s\n", back_to_char(bits, EL_AMOUNT));
                free(bits);
                continue;
            case Rand:
                printf(
                    "Your random symbol sequence is %s\n",
                    get_random_sequence(sequence, EL_AMOUNT)
                );
                bits = algorithm(sequence, EL_AMOUNT);
                printf("...And result is -> %s\n", bits);
                printf("In ASCII - %s\n", back_to_char(bits, EL_AMOUNT));
                free(bits);
                continue;
            case Quit:
                puts(" Bye, see you later!");
                return EXIT_SUCCESS;
            default:
                puts(" Dis value is not akceptabele\n");
        }
    }
}
```

**Код файла second\_algorithm.c:**

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
```

```

char* algorithm(char* sequence, int seq_len) {
    char* bits = (char*)malloc(64); // массив для хранения битов

    for (int i = 0; i < seq_len; i++) {
        char c = sequence[i];
        sprintf(bits + i * 8, "%08d", c);
    }

    for (int i = 0; i < seq_len; i++) {
        int firstBit = bits[i * 8] - '0';
        int lastBit = bits[i * 8 + 7] - '0';
        if (firstBit == lastBit) {
            bits[i * 8] = '0';
            bits[i * 8 + 7] = '1';
        } else {
            bits[i * 8] = '1';
            bits[i * 8 + 7] = '0';
        }
    }

    return bits;
}

```

**Код файла second\_interface.c:**

```

#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>
#include <string.h>

int get_int(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;
    }
}

char* get_sequence(char* sequence, int length){
    char temprem, tempclear;
    while(true)
    {
        temprem=0;
        tempclear=0;
        sequence[0]='\0';
        if((!scanf("%s",sequence))|| (int)strlen(sequence) != length)
        {
            printf(" - Error: Invalid value for 8 length string sequence.\n
One more time: ");

```

```

        while(tempclear != '\n')
            scanf("%c",&tempclear);
    }
    else
        return sequence;
}
}

char* get_random_sequence(char* sequence, int length)
{
    for (int i = 0; i < length; i++) {
        sequence[i] = rand() % 26 + 'A';
    }
    sequence[8] = '\0';
    return sequence;
}

char* back_to_char(char* bits, int seq_len)
{
    char* result = (char*) malloc(seq_len + 1);
    for (int i = 0; i < seq_len; i++) {
        char c = 0;
        for (int j = 0; j < 8; j++) {
            c = (c << 1) | (bits[i * 8 + j] - '0');
        }
        result[i] = c;
    }
    result[seq_len] = '\0';
    return result;
}

```

**Код файла second\_interface.h:**

```

#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>

int get_int(void);
char* get_random_sequence(char* sequence, int length);
char* back_to_char(char* bits, int seq_len);
char* get_sequence(char* sequence, int length);

```

**Код файла second\_algorithm.h:**

```

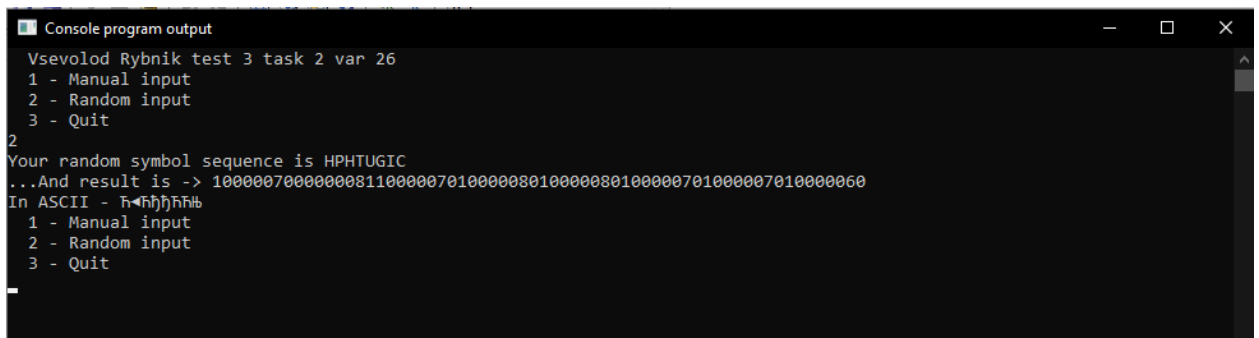
#include <stdio.h>
#include <string.h>

char* algorithm(char* sequence, int seq_len);

```

## 2.6 Тестирование

Тестирование было проведено на рандомных значениях:



```
Console program output
Vsevolod Rybnik test 3 task 2 var 26
1 - Manual input
2 - Random input
3 - Quit
2
Your random symbol sequence is HPHTUGIC
...And result is -> 10000070000000811000007010000080100000701000007010000060
In ASCII - HPHHTUGIC
1 - Manual input
2 - Random input
3 - Quit
```

Рисунок 4 – Экранная копия результата работы разработанной программы задания 2

## 2.7 Выводы по заданию №2

В ходе выполнения задания были изучены новые темы и понятия в программировании, способы работы с битами символов в Objective C.

### **3 Задание №3**

#### **3.1 Цель работы**

Разобраться в функционале указателей на функции.

#### **3.2 Постановка задачи**

Функция `fold` суммирует все числа в массиве целочисленных чисел с использованием функции `sum`, получает в качестве аргументов указатель на исходный массив, размер массива, указатель на функцию `sum` от двух аргументов и возвращает сумму всех элементов массива. Функция `sum` производит сложение двух аргументов и возвращает их сумму.

#### **3.3 Описание хода выполнения**

Для реализации поставленной задачи были изучены указатели на функции в Objective C.

#### **3.4 Блок-схема алгоритма решения задачи**

Блок схема к задаче №3:



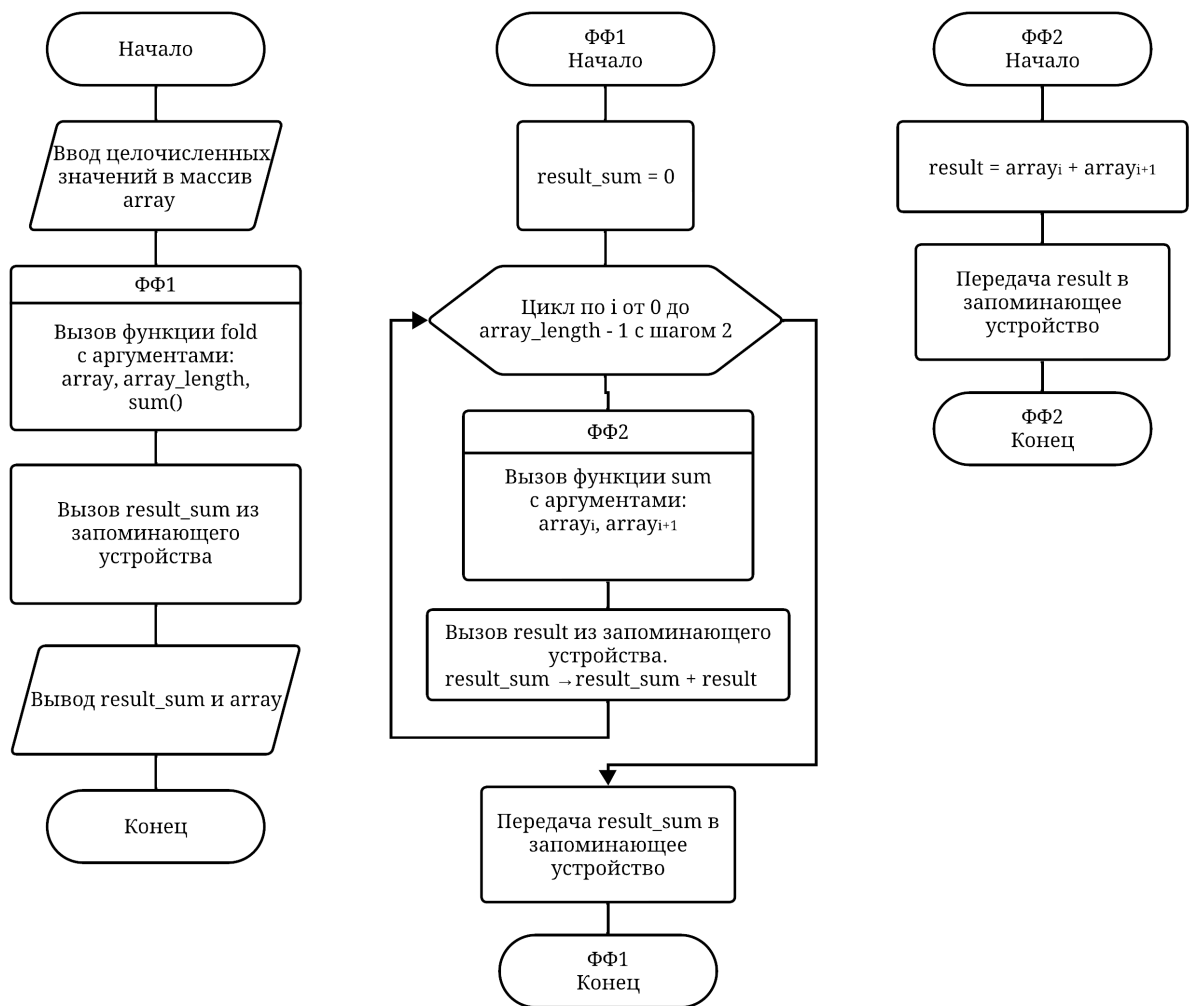


Рисунок 5 – Блок-схема алгоритма решения задачи №3

### 3.5 Исходный код полученного программного решения

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

```
#include <time.h>
#include "third_interface.h"
#include "third_algorithm.h"

enum MENU {Man=1, Rand=2, Quit=3};

int main(void)
{
    srand(time(NULL));
    int ARRAY_LEN = 0;
    int loop_indicator = 1, options = 1;
    int* array = (int*)malloc(sizeof(int));
    printf(" Vsevolod Rybnik test 3 task 3 var 26\n");
    while (loop_indicator)
    {
        puts("Specfiy array length: ");
        ARRAY_LEN = get_int();
        puts(" 1 - Manual input\n 2 - Random input\n 3 - Quit");
        options = get_int();
        switch (options)
        {
            case Man:
                bind_values(array, false, ARRAY_LEN);
                printf("Result: %d \n", fold(array, ARRAY_LEN, algorithm));
                output(array, ARRAY_LEN);
                continue;
            case Rand:
                bind_values(array, true, ARRAY_LEN);
                output(array, ARRAY_LEN);
                printf("Result: %d \n", fold(array, ARRAY_LEN, algorithm));
                continue;
            case Quit:
                puts(" Bye, see you later!");
                return EXIT_SUCCESS;
            default:
                puts(" Dis value is not akceptabele\n");
        }
    }
}
```

**Код файла third\_interface.c:**

```
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>

int get_int(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;
}
}

void output(int array[], int ARRAY_LEN)
{
    printf("Array: [ %d", array[0]);
    for (int i = 1; i < ARRAY_LEN; i++)
    {
        printf(", ");
        printf("%d", array[i]);
    }
    printf("]\n");
}

int* bind_values(int array[], bool rand_indicator, int ARRAY_LEN)
{
    for (int i = 0; i < ARRAY_LEN; i++){
        if (rand_indicator){
            array[i] = -99 + rand()%(100+98);
        } else {
            printf("    - Specify %d element of Array: ", i+1);
            array[i] = get_int();
        }
    }
    return array;
}

```

#### **Код файла third\_algorithm.c:**

```

int algorithm(int first_value, int second_value)
{
    return first_value + second_value;
}

int fold(int array[], int array_len, int (*sum)(int, int))
{
    int result_sum = 0;
    for (int i = 0; i < array_len - 1; i+=2)
    {
        result_sum += sum(array[i], array[i+1]);
    }
    return result_sum;
}

```

#### **Код файла third\_algorithm.h:**

```

int fold(int array[], int array_len, int (*sum)(int, int));

```

```
int algorithm(int first_value, int second_value);
```

**Код файла third\_interface.h:**

```
#include <stdbool.h>
#include <stdlib.h>
#include <stdio.h>
```

```
int get_int(void);
```

```
void output(int array[], int ARRAY_LEN);
```

```
int* bind_values(int array[], bool rand_indicator, int ARRAY_LEN);
```

### 3.6 Тестирование

Тестирование было проведено на случайных значениях:



Рисунок 6 – Экранная копия результата работы разработанной программы задания №3

### 3.7 Выводы по заданию №3

В ходе выполнения задания были изучены способы передачи функции в качестве аргумента в другом методе.