МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ

«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ
ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ им. В. Г. ШУХОВА»
(БГТУ им. В.Г. Шухова)

Кафедра программного обеспечения вычислительной техники и автоматизированных систем

Учебно-ознакомительная практика

по дисциплине: Основы программирования

Выполнил: ст. группы ПВ-223
Пахомов Владислав Андреевич

Проверили: Новожен Никита Викторович
Черников Сергей Викторович

Белгород 2023 г.

Оглавление

Вариант 10

1 Линейные алгоритмы

1.1 Задание варианта

Найти корни квадратного уравнения, заданного своими коэффициентами, с положительным дискриминантом; подстановкой в уравнение убедиться в погрешности вычислений.

1.2 Обоснование

Корни уравнения находим по формуле $x_1 = \frac{-b+\sqrt{D}}{2 \cdot a}$ и $x_2 = \frac{-b-\sqrt{D}}{2 \cdot a}$, где $D = b^2 - 4 \cdot a \cdot c$

1.3 Спецификация

1. Заголовок: `void solveSquareEquation(float a, float b, float c, float *x1, float *x2)`
2. Назначение: принимает коэффициенты уравнения вида `a` $\cdot x^2 +$ `b` $\cdot x +$ `c` $= 0$, сохраняет его корни в `x1`, `x2`.

1.4 Блок-схема

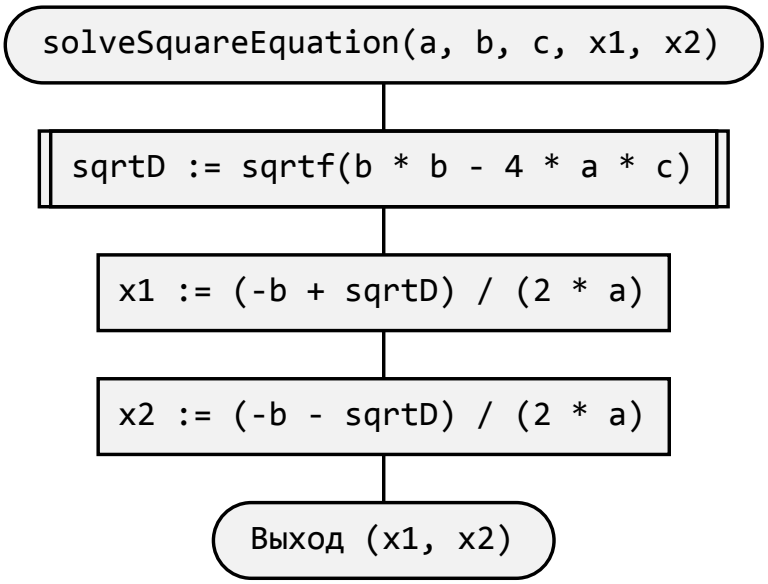

Рисунок 1: Функция solveSquareEquation

1.5 Код программы

```
#include <math.h>

void solveSquareEquation(float a, float b, float c, float *x1, float *x2) {
    float sqrtD = sqrtf(b * b - 4 * a * c);
    *x1 = (-b + sqrtD) / (2 * a);
    *x2 = (-b - sqrtD) / (2 * a);
}
```

Таблица 1

Результаты выполнения функции

a	b	c	x_1 (факт.)	x_2 (факт.)	$\lvert F(x_1)\rvert$	$\lvert F(x_2)\rvert$
2	-12	16	2.000000	4.000000	0	0
2	$-6\sqrt{2}$	16	1.414214	2.828427		
2	4	-16	2.000000	-4.000000	0	0
1.5	-6	$-\frac{14}{3}$	4.666667	-0.666667	$2.666667 \cdot 10^{-6}$	$2.666667 \cdot 10^{-6}$
2	6	0	-3.000000	0.000000	0	0
3	1	0	-0.333333	0.000000	$3.333333 \cdot 10^{-7}$	0
2	-6	0	3.000000	0.000000	0	0
3	-1	0	0.333333	0.000000	$3.333333 \cdot 10^{-7}$	0
2	12	16	-2.000000	-4.000000	0	0
2	$6\sqrt{2}$	8	-1.414214	-2.828427	$1.237795 \cdot 10^{-6}$	$3.528354 \cdot 10^{-6}$
-2	-12	-16	-2.000000	-4.000000	0	0
-2	$-6\sqrt{2}$	-8	1.414214	2.828427	$1.237795 \cdot 10^{-6}$	$3.528354 \cdot 10^{-6}$
-2	4	16	-2.000000	4.000000	0	0
-1.5	-6	$\frac{14}{3}$	-4.666667	0.666667	$2.666667 \cdot 10^{-6}$	$2.666667 \cdot 10^{-6}$
-2	-6	0	-3.000000	0.000000	0	0
-3	-1	0	-0.333333	0.000000	$3.333333 \cdot 10^{-7}$	0
-2	6	0	3.000000	0.000000	0	0
-3	1	0	0.333333	-0.000000	$3.333333 \cdot 10^{-7}$	0
-2	12	-16	2.000000	4.000000	0	0
2	$-6\sqrt{2}$	8	1.414214	2.828427	$1.237795 \cdot 10^{-6}$	$3.528354 \cdot 10^{-6}$

1.6 Тестовые данные

Таблица 2

Тестовые данные solveSquareEquation

a	b	c	x_1	x_2
2	-12	16	2	4
2	$-6\sqrt{2}$	16	$\sqrt{2}$	$2\sqrt{2}$
2	4	-16	2	-4
1.5	-6	$-\frac{14}{3}$	$4\frac{2}{3}$	$-\frac{2}{3}$
2	6	0	-3	0
3	1	0	$-\frac{1}{3}$	0
2	-6	0	3	0
3	-1	0	$\frac{1}{3}$	0
2	12	16	-2	-4
2	$6\sqrt{2}$	8	$-\sqrt{2}$	$-2\sqrt{2}$
-2	-12	-16	-2	-4
-2	$-6\sqrt{2}$	-8	$\sqrt{2}$	$2\sqrt{2}$
-2	4	16	-2	4
-1.5	-6	$\frac{14}{3}$	$-4\frac{2}{3}$	$\frac{2}{3}$
-2	-6	0	-3	0
-3	-1	0	$-\frac{1}{3}$	0
-2	6	0	3	0
-3	1	0	$\frac{1}{3}$	0
-2	12	-16	2	4
2	$-6\sqrt{2}$	8	$\sqrt{2}$	$2\sqrt{2}$

1.7 Результат выполнения тестов

```
Test project /Users/vlad/Desktop/C/programming-and-algorithmization-basics/cmake-build-debug
      Start  1: testTask1APositivePositiveIntRoots
 1/72 Test  #1: testTask1APositivePositiveIntRoots ..............   Passed    0.46 sec
      Start  2: testTask1APositivePositiveRealRoots
 2/72 Test  #2: testTask1APositivePositiveRealRoots .............   Passed    0.19 sec
      Start  3: testTask1APositivePositiveAndNegativeIntRoots
 3/72 Test  #3: testTask1APositivePositiveAndNegativeIntRoots ....   Passed    0.19 sec
      Start  4: testTask1APositivePositiveAndNegativeRealRoots
 4/72 Test  #4: testTask1APositivePositiveAndNegativeRealRoots ...   Passed    0.19 sec
      Start  5: testTask1APositiveZeroAndNegativeIntRoots
 5/72 Test  #5: testTask1APositiveZeroAndNegativeIntRoots ........   Passed    0.19 sec
      Start  6: testTask1APositiveZeroAndNegativeRealRoots
 6/72 Test  #6: testTask1APositiveZeroAndNegativeRealRoots .......   Passed    0.19 sec
      Start  7: testTask1APositivePositiveAndZeroIntRoots
 7/72 Test  #7: testTask1APositivePositiveAndZeroIntRoots ........   Passed    0.19 sec
      Start  8: testTask1APositivePositiveAndZeroRealRoots
 8/72 Test  #8: testTask1APositivePositiveAndZeroRealRoots .......   Passed    0.19 sec
      Start  9: testTask1APositiveNegativeIntRoots
 9/72 Test  #9: testTask1APositiveNegativeIntRoots ..............   Passed    0.19 sec
      Start 10: testTask1APositiveNegativeRealRoots
10/72 Test #10: testTask1APositiveNegativeRealRoots ..............   Passed    0.19 sec
      Start 11: testTask1ANegativeNegativeIntRoots
11/72 Test #11: testTask1ANegativeNegativeIntRoots ..............   Passed    0.19 sec
      Start 12: testTask1ANegativeNegativeRealRoots
12/72 Test #12: testTask1ANegativeNegativeRealRoots ..............   Passed    0.19 sec
      Start 13: testTask1ANegativePositiveAndNegativeIntRoots
13/72 Test #13: testTask1ANegativePositiveAndNegativeIntRoots ....   Passed    0.19 sec
      Start 14: testTask1ANegativePositiveAndNegativeRealRoots
14/72 Test #14: testTask1ANegativePositiveAndNegativeRealRoots ...   Passed    0.19 sec
      Start 15: testTask1ANegativeZeroAndNegativeIntRoots
15/72 Test #15: testTask1ANegativeZeroAndNegativeIntRoots ........   Passed    0.19 sec
      Start 16: testTask1ANegativeZeroAndNegativeRealRoots
16/72 Test #16: testTask1ANegativeZeroAndNegativeRealRoots .......   Passed    0.19 sec
      Start 17: testTask1ANegativePositiveAndZeroIntRoots
17/72 Test #17: testTask1ANegativePositiveAndZeroIntRoots ........   Passed    0.19 sec
      Start 18: testTask1ANegativePositiveAndZeroRealRoots
18/72 Test #18: testTask1ANegativePositiveAndZeroRealRoots .......   Passed    0.19 sec
      Start 19: testTask1ANegativePositiveIntRoots
19/72 Test #19: testTask1ANegativePositiveIntRoots ..............   Passed    0.19 sec
      Start 20: testTask1ANegativePositiveRealRoots
20/72 Test #20: testTask1ANegativePositiveRealRoots ..............   Passed    0.19 sec
```

Рисунок 2: Результат выполнения тестов функции solveSquareEquation

2 Разветвляющиеся алгоритмы

2.1 Задание варианта

Можно ли на прямоугольном участке застройки размером a на b метров разместить два дома размером в планер q на p и r на s метров? Дома можно располагать только параллельно сторонам участка.

2.2 Обоснование

Обоснование:

Легче всего данную задачу представить в графическом формате. Возможно 8 способов расстановки домов:

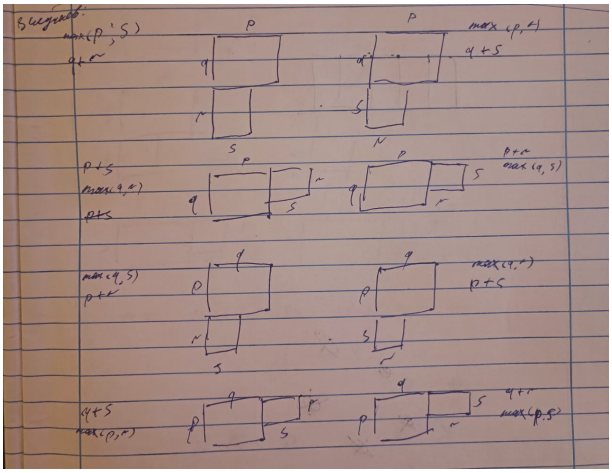

Рисунок 3: Все возможные перестановки домов

сли одна из расстановок помещается в a на b - значит дома разместить можно, иначе - нет.

2.3 Спецификация

1. Заголовок: `static int max(int a, int b)`
2. Назначение: Возвращает максимальное из `a` и `b`

---

1. Заголовок: `void solveSquareEquation(float a, float b, float c, float *x1, float *x2)`
2. Назначение: принимает коэффициенты уравнения вида `a` $\cdot x^2 +$ `b` $\cdot x +$ `c` $= 0$, сохраняет его корни в `x1`, `x2`.

2.4 Блок-схема

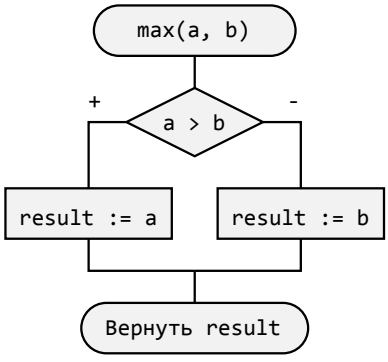

Рисунок 4: Функция max

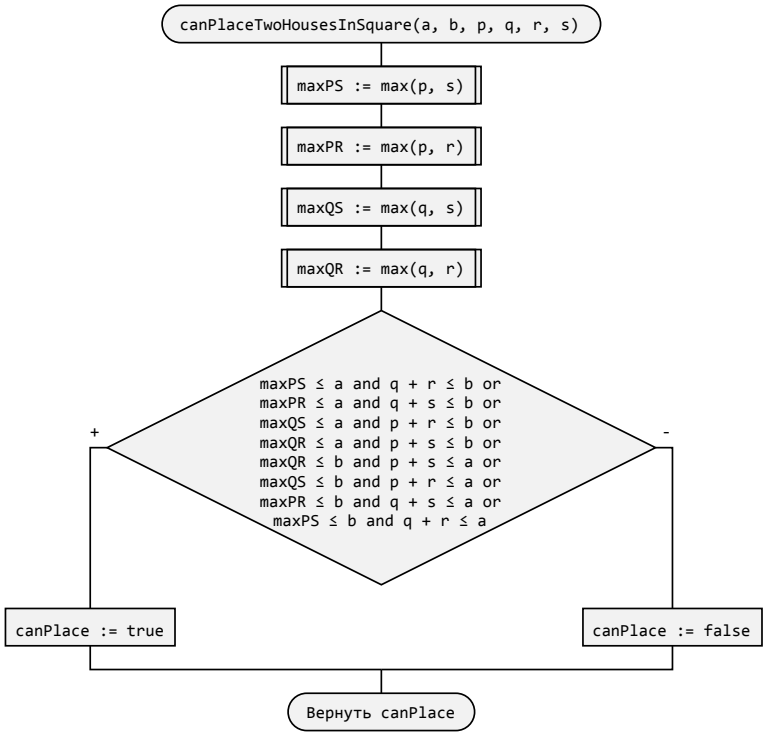

Рисунок 5: Функция canPlaceTwoHousesInSquare

2.5 Код программы

```c
#include <stdbool.h>

static int max(int a, int b) {
    return a > b ? a : b;
}

bool canPlaceTwoHousesInSquare(int a, int b, int p, int q, int r, int s) {
    return max(p, s) <= a && q + r <= b ||
           max(p, r) <= a && q + s <= b ||
           max(q, s) <= a && p + r <= b ||
           max(q, r) <= a && p + s <= b ||
           p + s <= a && max(q, r) <= b ||
           p + r <= a && max(q, s) <= b ||
           q + s <= a && max(p, r) <= b ||
           q + r <= a && max(p, s) <= b;
}
```

2.6 Тестовые данные

Таблица 3

Тестовые данные canPlaceTwoHousesInSquare

Тестовые данные	Ожидаемый результат
a = 0, b = 0, p = 2, q = 3, r = 1, s = 5	false
a = 2, b = 5, p = 2, q = 3, r = 1, s = 5	false
a = 1, b = 500, p = 2, q = 3, r = 1, s = 5	false
a = 4, b = 6, p = 4, q = 2, r = 4, s = 4	true
a = 4, b = 8, p = 2, q = 5, r = 3, s = 3	true

2.7 Результат выполнения тестов

```
      Start 21: testTask2EmptySpace
21/72 Test #21: testTask2EmptySpace ..............................   Passed    0.19 sec
      Start 22: testTask2NotEnoughSpace
22/72 Test #22: testTask2NotEnoughSpace ..........................   Passed    0.19 sec
      Start 23: testTask2EnoughSpaceCannotPlace
23/72 Test #23: testTask2EnoughSpaceCannotPlace ..................   Passed    0.19 sec
      Start 24: testTask2ShouldFitPerfectly
24/72 Test #24: testTask2ShouldFitPerfectly ......................   Passed    0.19 sec
      Start 25: testTask2ShouldFitExtraSpaceLeft
25/72 Test #25: testTask2ShouldFitExtraSpaceLeft .................   Passed    0.19 sec
```

Рисунок 6: Результат выполнения тестов функции canPlaceTwoHousesInSquare

3 Циклические и итерационные алгоритмы

3.1 Задание варианта

Известно время начала и окончания (например, 6:00 и 24:00) работы некоторого пригородного автобусного маршрута с одним автобусом на линии, а также протяжённость маршрута в минутах (в один конец) и время отдыха на конечных остановках. Составить суточное расписание этого маршрута (моменты отправления с конечных пунктов) без учета времени на обед и пересменку.

3.2 Обоснование

Будем определять время начала работы в минутах и общее время работы. В цикле инициализирующее выражение - инициализируем счётчик *scheduleSize = 0, на каждой итерации цикла будет его постепенно увеличивать. В условии возобновлении цикла будем проверять, что время старта + время работы (работник вряд-ли будет счастлив, если выедет ровно во время окончания рабочего дня и будет работать дополнительно, например, 2 часа) не будет превышать общее время работы. Пока это условие выполняется, будем сохранять время выезда в массив. Также в условии возобновления цикла проверим, что мы не превысим максимальное количество записей в расписании, иначе можем получить ошибку во время выполнения программы.

3.3 Спецификация

1. Заголовок: `getBusSchedule(Clock start, Clock end, int restTime, int travelTime, Clock *schedule, int *scheduleSize, int maxScheduleSize)`

2. Назначение: сохраняет в массив `schedule` максимального размера `maxScheduleSize` расписание автобуса при условии, что он начинает работу в `start`, оканчивает в `end`, отдыхает на остановках `restTime` имеет длительность маршрута `travelTime`. Сохраняет количество записей в `scheduleSize`.

3.4 Блок-схема

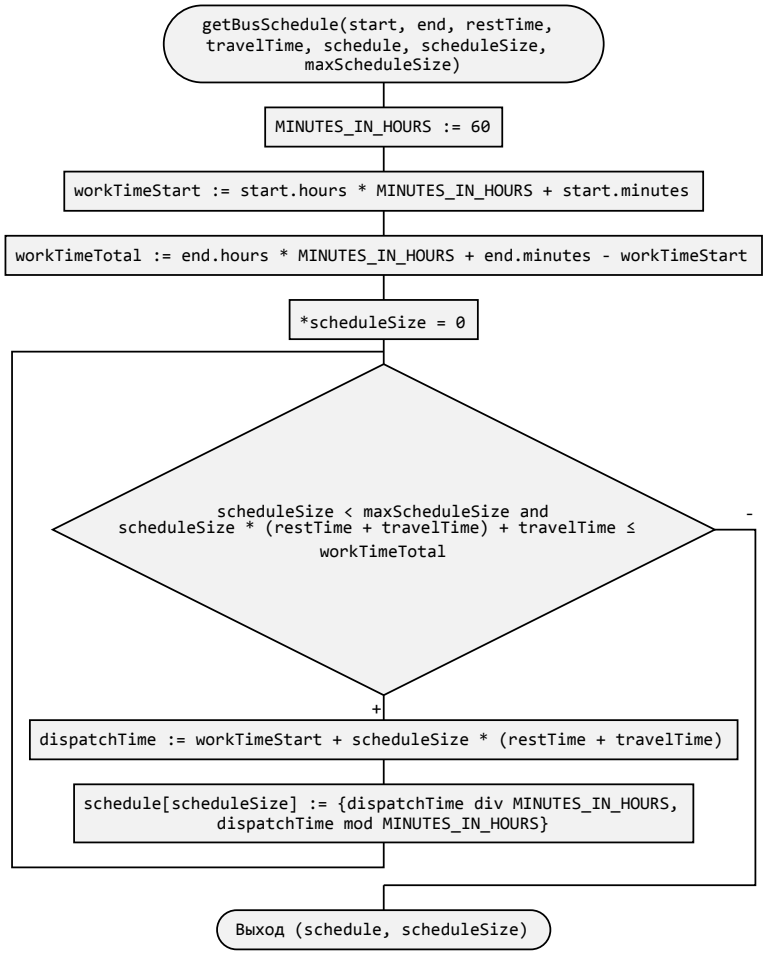

Рисунок 7: Функция getBusSchedule

3.5 Код программы

```
#include <stddef.h>

#include "Tasks.h"

#define MINUTES_IN_HOURS 60

// Содержит время - hours (часы) и minutes (минуты)
typedef struct Clock {
    int hours;
    int minutes;
} Clock;

void getBusSchedule(Clock start, Clock end, int restTime, int travelTime, Clock *schedule, int
↪  *scheduleSize, int maxScheduleSize) {
    // Определяем время начала работы и её окончания
```

```
    int workTimeStart = start.hours * MINUTES_IN_HOURS + start.minutes;
    int workTimeTotal = end.hours * MINUTES_IN_HOURS + end.minutes - workTimeStart;

    for (*scheduleSize = 0; *scheduleSize < maxScheduleSize &&
    *scheduleSize * (restTime + travelTime) + travelTime <= workTimeTotal;
    (*scheduleSize)++) {
        // Определяем время отправки в минутах
        int dispatchTime = workTimeStart + (*scheduleSize) * (restTime + travelTime);
        // И сохраняем это время в расписание
        schedule[*scheduleSize] = (Clock) {dispatchTime / MINUTES_IN_HOURS,
                                           dispatchTime % MINUTES_IN_HOURS};
    }
}
```

3.6 Тестовые данные

Таблица 4

Тестовые данные getBusSchedule

Тестовые данные	Ожидаемый результат
begin = 12:00, end = 12:00, restTime = 12, travelTime = 12, MAX_SCHEDULE_SIZE = 1000	schedule = {}
begin = 12:00, end = 10:00, restTime = 12, travelTime = 12, MAX_SCHEDULE_SIZE = 1000	schedule = {}
begin = 12:00, end = 12:11, restTime = 12, travelTime = 12, MAX_SCHEDULE_SIZE = 1000	schedule = {}
begin = 12:00, end = 12:12, restTime = 12, travelTime = 12, MAX_SCHEDULE_SIZE = 1000	schedule = {12:00}
begin = 10:00, end = 12:12, restTime = 10, travelTime = 15, MAX_SCHEDULE_SIZE = 1000	schedule = {10:00, 10:25, 10:50, 11:15, 11:40}
begin = 10:00, end = 11:00, restTime = 10, travelTime = 0, MAX_SCHEDULE_SIZE = 1000	schedule = {10:00, 10:10, 10:20, 10:30, 10:40, 10:50, 11:00}
begin = 10:00, end = 11:00, restTime = 0, travelTime = 10, MAX_SCHEDULE_SIZE = 1000	schedule = {10:00, 10:10, 10:20, 10:30, 10:40, 10:50}
begin = 10:00, end = 11:00, restTime = 0, travelTime = 0, MAX_SCHEDULE_SIZE = 1000	scheduleSize = 1000

3.7 Результат выполнения тестов

```
      Start 26: testTask3NoWorkingTime
26/72 Test #26: testTask3NoWorkingTime ..........................   Passed    0.19 sec
      Start 27: testTask3NegativeWorkingTime
27/72 Test #27: testTask3NegativeWorkingTime ....................   Passed    0.19 sec
      Start 28: testTask3NotEnoughWorkingTime
28/72 Test #28: testTask3NotEnoughWorkingTime ...................   Passed    0.19 sec
      Start 29: testTask3TimeEnoughJustForOneTravel
29/72 Test #29: testTask3TimeEnoughJustForOneTravel .............   Passed    0.19 sec
      Start 30: testTask3RegularScenario
30/72 Test #30: testTask3RegularScenario ........................   Passed    0.19 sec
      Start 31: testTask3Teleportation
31/72 Test #31: testTask3Teleportation ..........................   Passed    0.19 sec
      Start 32: testTask3NoRest
32/72 Test #32: testTask3NoRest .................................   Passed    0.19 sec
      Start 33: testTask3ShouldNotOverfill
33/72 Test #33: testTask3ShouldNotOverfill ......................   Passed    0.20 sec
```

Рисунок 8: Результат выполнения тестов функции getBusSchedule

4 Простейшие операции над массивами

4.1 Задание варианта

Найти среднее арифметическое элементов каждой строки матрицы $Q(l, m)$ и вычесть его из элементов этой строки.

4.2 Обоснование

Сначала вычислим среднее для каждой строчки матрицы, потом вычислим среднее строчки из каждого элемента строчки.

4.3 Спецификация

1. Заголовок: `void subtractMiddleOfLineFromEveryLine(double **matrix, int l, int m)`
2. Назначение: из каждого элемента строчки матрицы `matrix` размером `l` $\times$ `m` вычитает среднее строчки.

4.4 Блок-схема

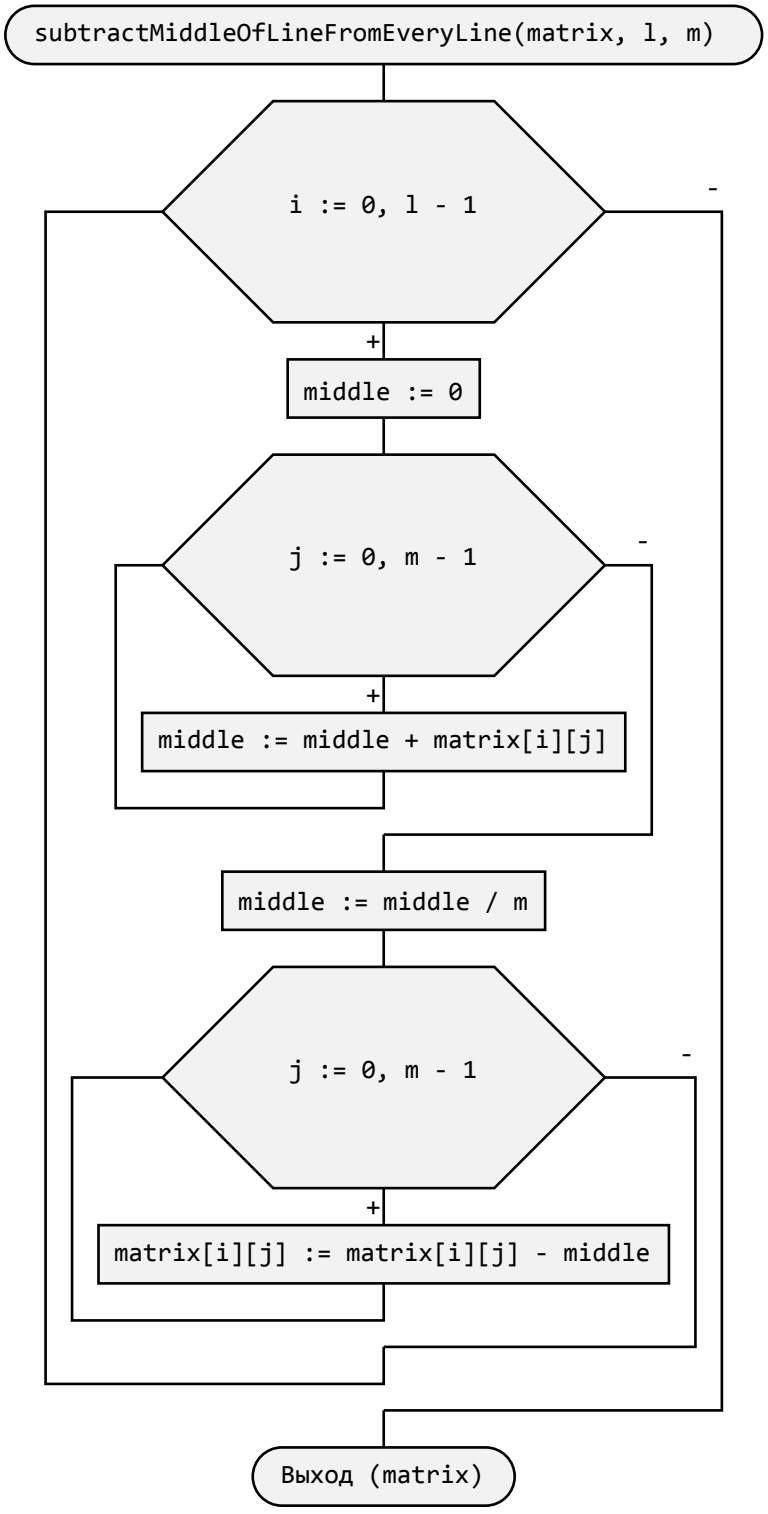

Рисунок 9: Функция subtractMiddleOfLineFromEveryLine

4.5 Код программы

```
#include "Tasks.h"

void subtractMiddleOfLineFromEveryLine(double **matrix, int l, int m) {
    for (int i = 0; i < l; i++) {
        // Для каждой строчки высчитаем среднее.
        // Сначала высчитаем сумму
        double middle = 0;

        for (int j = 0; j < m; j++)
            middle += matrix[i][j];

        // После этого можем поделить сумму на m - кол-во элементов - и получить среднее
        middle /= m;

        // Остаётся только вычесть её из каждого элемента в строке
        for (int j = 0; j < m; j++)
            matrix[i][j] -= middle;
    }
}
```

4.6 Тестовые данные

Таблица 5

Тестовые данные subtractMiddleOfLineFromEveryLine

Тестовые данные	Ожидаемый результат
Пустая матрица	Пустая матрица
$\begin{pmatrix} 5 & 1 & 4 & 2 & 3 \end{pmatrix}$	$\begin{pmatrix} 2 & -2 & 1 & -1 & 0 \end{pmatrix}$
$\begin{pmatrix} 5 \\ 1 \\ 4 \\ 2 \\ 3 \end{pmatrix}$	$\begin{pmatrix} 2 \\ -2 \\ 1 \\ -1 \\ 0 \end{pmatrix}$
$\begin{pmatrix} 5 & 1 & 4 & 2 & 3 \\ 5 & 5 & 5 & 5 & 5 \\ 3 & 1 & 1 & 1 & 3 \\ 0 & 0 & 5 & 0 & 0 \\ 5 & 4 & 3 & 4 & 5 \end{pmatrix}$	$\begin{pmatrix} 2 & -2 & 1 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 1.2 & -0.8 & -0.8 & -0.8 & 1.2 \\ -1 & -1 & 4 & -1 & -1 \\ 0.8 & -0.2 & -1.2 & -0.2 & 0.8 \end{pmatrix}$

4.7 Результат выполнения тестов

```
      Start 34: testTask4Empty
34/72 Test #34: testTask4Empty ..................................   Passed    0.19 sec
      Start 35: testTask4OneLine
35/72 Test #35: testTask4OneLine ................................   Passed    0.19 sec
      Start 36: testTask4OneColumn
36/72 Test #36: testTask4OneColumn ..............................   Passed    0.19 sec
      Start 37: testTask4Regular
37/72 Test #37: testTask4Regular ................................   Passed    0.18 sec
```

Рисунок 10: Результат выполнения тестов функции subtractMiddleOfLineFromEveryLine

5 Векторы и матрицы

5.1 Задание варианта

По кругу располагаются n человек. Ведущий считает по кругу, начиная с первого, и выводит («казнит») m-го человека. Круг смыкается, счёт возобновляется со следующего после «казнённого»; так продолжается, пока «в живых» останется только один человек. Найти номер оставшегося «в живых» человека, а также для заданного найти такое $m > 1$, при котором «в живых» останется первый.

5.2 Обоснование

Логично предположить что в круге из одного человека при любом шаге он будет победителем. Теперь рассмотрим задачу из двух людей. Вычеркнем из этого круга человека в зависимости от m - мы получим оставшегося победителя. Рассмотрим задачу с большим количество людей, например 5. Будем проходиться по кругу с шагом m, вычеркнем из этого круга m человека. У нас останется 4 человека, мы уже знаем решение этой задачи, поэтому на основе этих данных мы можем определить текущего победителя для n и m по формуле
$res[n][m] = (m + res[n-1][m] - 1)\%n + 1$
на основе этой формулы можем составить матрицу, где элемент с индексом n и m содержит ответ на задачу.

5.3 Спецификация

1. Заголовок: `static JosephProblemMatrix createJosephProblemMatrix(int nn)`
2. Назначение: возвращает структуру, содержащую указатель на матрицу шириной и длиной `nn` и её длину. Ячейка матрицы в i ряду и j столбце является ответом задача Иосифа Флавия, где (i + 1) - количество людей в круге, (j + 1) - шаг с которым казнят людей в кругу.

---

1. Заголовок: `void freeJosephProblemMatrix(JosephProblemMatrix matrix)`
2. Назначение: освобождает память выделенную на объект `matrix`.

---

1. Заголовок: `int josephProblemFindSurvivor(int n, int m)`

2. Назначение: возвращает номер выжившего в задаче Иосифа Флавия в кругу из `n` человек с шагом `m`.

---

1. Заголовок: `int josephProblemFindMForSurvivorOne(int n)`
2. Назначение: возвращает шаг при котором в задаче Иосифа Флавия в кругу из `n` людей выжившим является человек с 1 номером.

5.4 Блок-схема

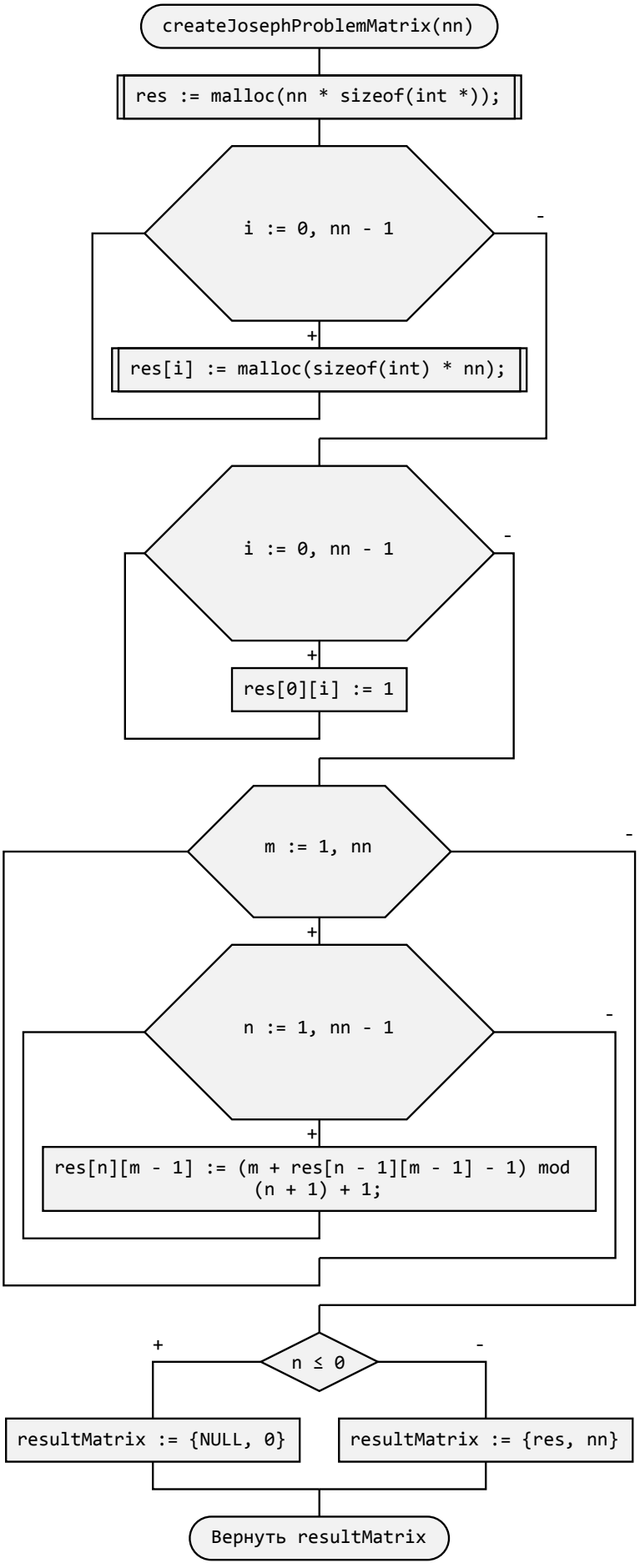

Рисунок 11: Функция createJosephProblemMatrix

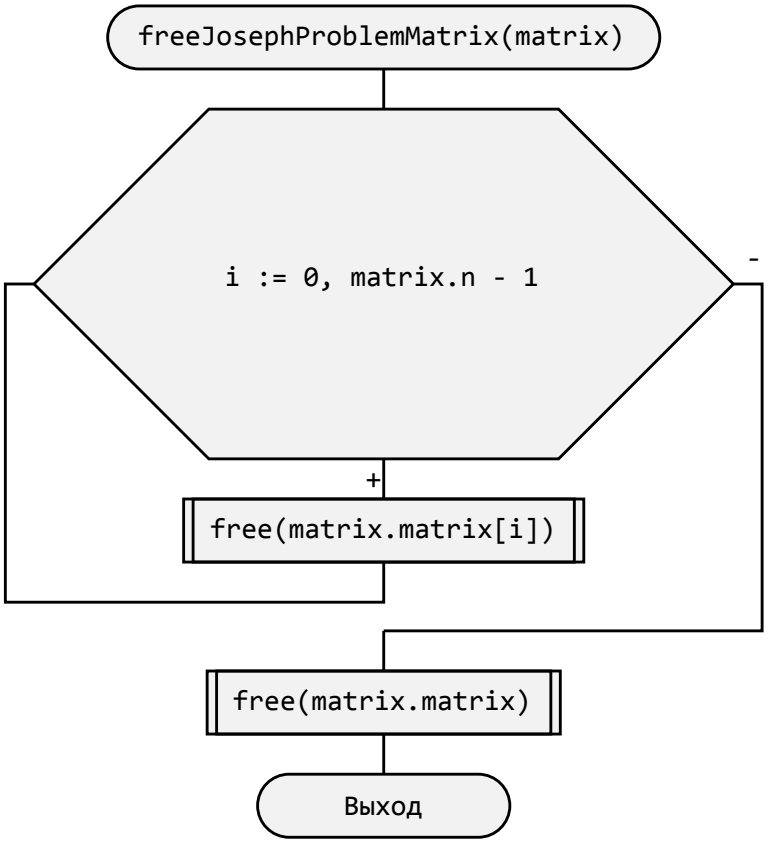

Рисунок 12: Функция freeJosephProblemMatrix

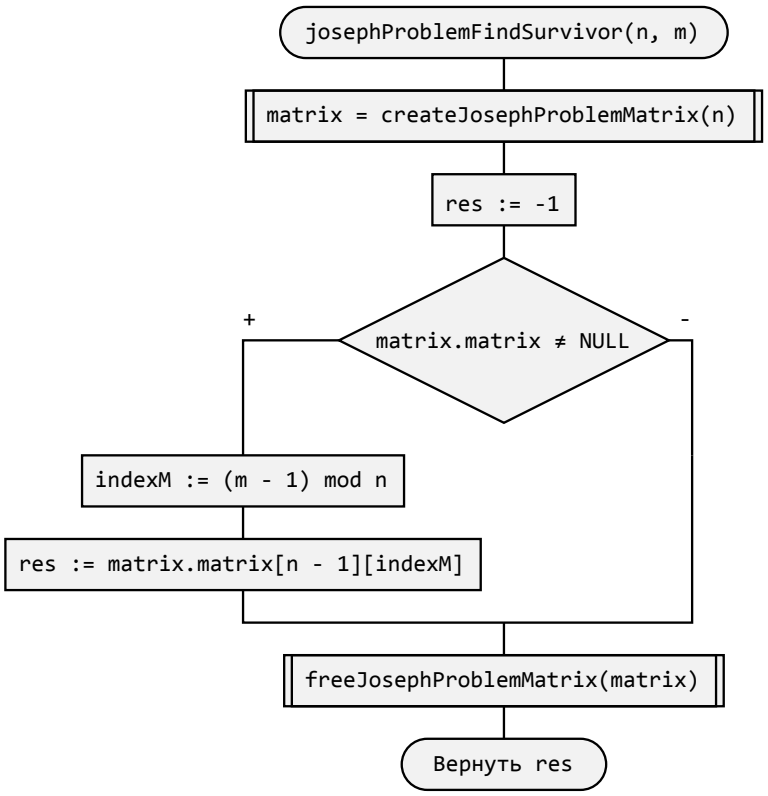

Рисунок 13: Функция josephProblemFindSurvivor

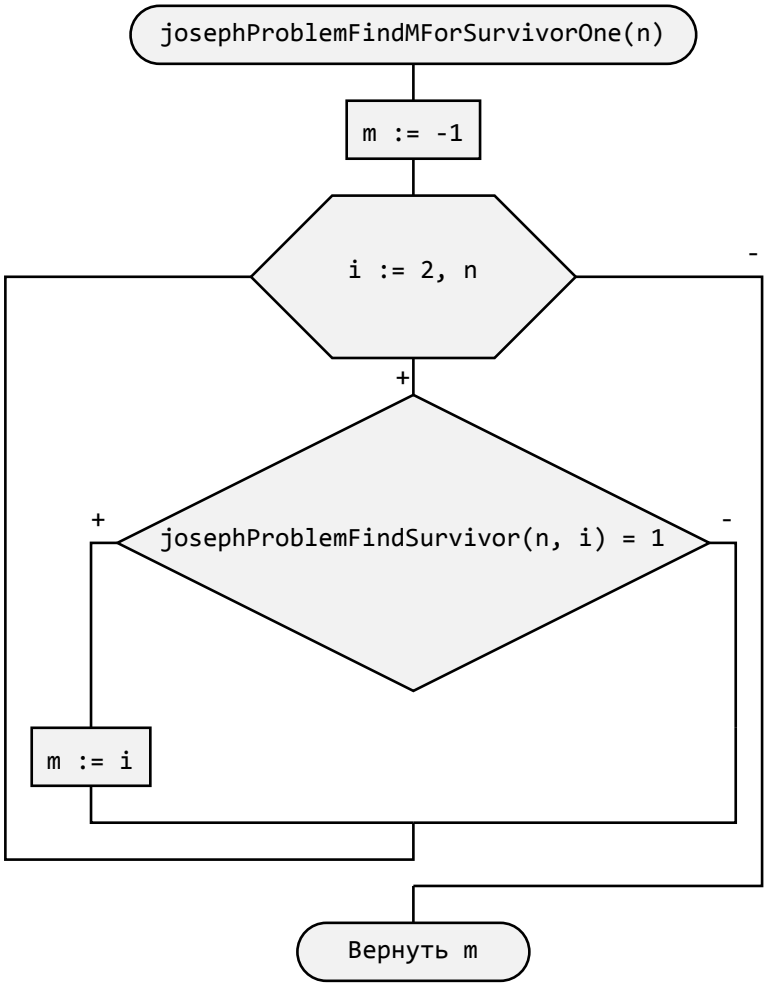

Рисунок 14: Функция josephProblemFindMForSurvivorOne

5.5 Код программы

```
#include <stddef.h>
#include <stdlib.h>

#include "Tasks.h"

typedef struct JosephProblemMatrix {
    int **matrix;
    int n;
} JosephProblemMatrix;

static JosephProblemMatrix createJosephProblemMatrix(int nn) {
    // Сформируем матрицу таким образом, что res[i][j] содержит номер "выжившего"
    // При этом i - изначальное количество людей в круге, j - шаг с которым будут
    // Казнить людей.

    int **res = malloc(nn * sizeof(int *));
```

```
    for (int i = 0; i < nn; i++)
        res[i] = malloc(sizeof(int) * nn);

    if (nn <= 0)
        return (JosephProblemMatrix) {NULL, 0};

    // Логично предположить что в круге из одного человека при любом шаге
    // он будет победителем. Так и делаем, всю первую строку матрицы
    // инициализируем единицей
    for (int i = 0; i < nn; i++)
        res[0][i] = 1;

    for (int m = 1; m <= nn; m++)
        // Теперь попробуем рассчитать победителя для конкретной ячейки
        // Допустим, мы казним одного человека. В итоге в круге останется
        // n - 1 человек. Шаг мы тоже знаем - m, значит нам остаётся только
        // вычислить победителя на основе информации о номере человека выжившем при кол-ве
        // людей меньше на 1 с тем же шагом.
        for (int n = 1; n < nn; n++) {
            res[n][m - 1] = (m + res[n - 1][m - 1] - 1) % (n + 1) + 1;
        }

    // Возвращаем матрицу
    return (JosephProblemMatrix) {res, nn};
}

void freeJosephProblemMatrix(JosephProblemMatrix matrix) {
    for (int i = 0; i < matrix.n; i++)
        free(matrix.matrix[i]);

    free(matrix.matrix);
}

int josephProblemFindSurvivor(int n, int m) {
    // Получаем матрицу
    JosephProblemMatrix matrix = createJosephProblemMatrix(n);

    int res = -1;
    if (matrix.matrix != NULL) {
        int indexM = (m - 1) % n;
        // Получаем результат для конкретной ячейки
        res = matrix.matrix[n - 1][indexM];
    }

    // Удаляем матрицу
    freeJosephProblemMatrix(matrix);

    // Возвращаем рзультат
    return res;
}
```

```
int josephProblemFindMForSurvivorOne(int n) {
    int m = -1;

    // Здесь просто переберём все возможные шаги
    for (int i = 2; i <= n; i++)
        if (josephProblemFindSurvivor(n, i) == 1)
            m = i;

    return m;
}
```

5.6 Тестовые данные

Таблица 6

Тестовые данные josephProblemFindSurvivor

Тестовые данные	Ожидаемый результат
n = 5, m = 1	5
n = 5, m = 2	3
n = 5, m = 3	4
n = 5, m = 4	1
n = 5, m = 5	2
n = 5, m = 6	5
n = 5, m = 7	3
n = 5, m = 8	4
n = 5, m = 9	1
n = 5, m = 10	2
n = 0, m = 0	-1

Таблица 7

Тестовые данные josephProblemFindMForSurvivorOne

Тестовые данные	Ожидаемый результат
n = 7	-1
n = 0	-1
n = 10	8

5.7 Результат выполнения тестов

```
      Start 38: testTask5FindSurvivorRegularScenario
38/72 Test #38: testTask5FindSurvivorRegularScenario .............   Passed    0.19 sec
      Start 39: testTask5FindSurvivorMBiggerThanN
39/72 Test #39: testTask5FindSurvivorMBiggerThanN ................   Passed    0.26 sec
      Start 40: testTask5FindSurvivorNZero
40/72 Test #40: testTask5FindSurvivorNZero .......................   Passed    0.19 sec
      Start 41: testTask5FindSurvivorOneNoResultsFound
41/72 Test #41: testTask5FindSurvivorOneNoResultsFound ...........   Passed    0.19 sec
      Start 42: testTask5FindSurvivorOneNZero
42/72 Test #42: testTask5FindSurvivorOneNZero ....................   Passed    0.19 sec
      Start 43: testTask5FindSurvivorOneRegularScenario
43/72 Test #43: testTask5FindSurvivorOneRegularScenario ..........   Passed    0.19 sec
```

Рисунок 15: Результат выполнения тестов функции josephProblemFindMForSurvivorOne и josephProblemFindSurvivor

6 Линейный поиск

6.1 Задание варианта

Матрица $L(n, k)$ состоит из нулей и единиц. Найти в ней самую длинную цепочку подряд стоящих нулей по горизонтали, вертикали или диагонали.

6.2 Обоснование

Чтобы найти самую длинную последовательность, необходимо пройтись по всем линиям, столбцам и диагоналям матрицы и найти в ней самую длинную последовательность. Для определения наибольшей длины последовательности создадим два счётчика - максимальная длина и текущая. Дальше переберём элементы. Если элемент - ноль, увеличим счётчик на 1,

сравнивая с максимальной длиной. Если она больше максимальной - присваиваем текущую длину. Иначе если элемент 1 - сбрасываем счётчик. С проходом по линиям и столбцам всё довольно легко. Для прохода по всем диагоналям будем выполнять следующие действия. Для каждого элемента крайнего левого столбца будем перебирать элементы диагонали начинающейся с текущей точки проходящей сверху слева - вправо вниз. Для каждого элемента крайнего правого столбца будем перебирать элементы диагонали начинающейся с текущей точки проходящей сверху справа - влево вниз. Для верхней границы будем перебирать все элементы двух видов диагонали, как слева вверх направо вниз, так и справа вверх налево вниз.

6.3 Спецификация

1. Заголовок: `static void findLongestZeroSubsequence(bool *arr, int size, int *begin, int *end)`
2. Назначение: находит начало и конец наидлиннейшей подпоследовательности последовательности `arr` размером `size` из нулей и сохраняет её начало в `begin` и конец в `end`

---

1. Заголовок: `static void getLine(bool **matrix, int k, int lineIndex, bool *line)`
2. Назначение: сохраняет в `line` ряд по индексу `lineIndex` матрицы `matrix` с количеством столбцов `k`

---

1. Заголовок: `static void getColumn(bool **matrix, int n, int columnIndex, bool *column)`
2. Назначение: сохраняет в `column` столбец по индексу `columnIndex` матрицы `matrix` с количеством рядов `n`

---

1. Заголовок: `static void getDiagonalTopLeftToBottomRight(bool **matrix, int n, int k, int iStart, int jStart, bool *diagonal, int *diagonalSize)`
2. Назначение: сохраняет в `diagonal` диагональ и её размер в `diagonalSize` по индексу (`iStart`; `jStart`) матрицы `matrix` размером `n` $\times$ `k`. Диагональ проходит сверху слева - справа вниз.

---

1. Заголовок: `static void getDiagonalTopRightToBottomLeft(bool **matrix, int n, int k, int iStart, int jStart, bool *diagonal, int *diagonalSize)`

2. Назначение: сохраняет в `diagonal` диагональ и её размер в `diagonalSize` по индексу (`iStart`; `jStart`) матрицы `matrix` размером `n` × `k`. Диагональ проходит сверху справа - влево вниз.

---

1. Заголовок: `static void processLine(bool **matrix, int k, int lineIndex, MatrixIndex *longestZeroSubsequenceBegin, MatrixIndex *longestZeroSubsequenceEnd, int *maxSubsequenceSize, bool *buf)`

2. Назначение: находит в ряду по индексу `lineIndex` в матрице `matrix` с количеством столбцов `k` наидлиннейшую подпоследовательность из нулей и сохраняет её начало в `longestZeroSubsec` конец в `longestZeroSubsequenceEnd`, размер подпоследовательности в `maxSubsequenceSize`. Принимает ссылку на буфер `buf` для хранения временных значений.

---

1. Заголовок: `static void processColumn(bool **matrix, int n, int columnIndex, MatrixIndex *longestZeroSubsequenceBegin, MatrixIndex *longestZeroSubsequenceEnd, int *maxSubsequenceSize, bool *buf)`

2. Назначение: находит в столбце по индексу `columnIndex` в матрице `matrix` с количеством рядов `n` наидлиннейшую подпоследовательность из нулей и сохраняет её начало в `longestZeroSubsequenceBegin`, конец в `longestZeroSubsequenceEnd`, размер подпоследовательности в `maxSubsequenceSize`. Принимает ссылку на буфер `buf` для хранения временных значений.

---

1. Заголовок: `static void processDiagonalTopLeftToBottomRight(bool **matrix, int n, int k, int row, int column, MatrixIndex *longestZeroSubsequenceBegin, MatrixIndex *longestZeroSubsequenceEnd, int *maxSubsequenceSize, bool *buf)`

2. Назначение: находит в диагонале проходящей сверху слева - вправо вниз по индексу (`row`; `column`) в матрице `matrix` размером `n` × `k` наидлиннейшую подпоследовательность из нулей и сохраняет её начало в `longestZeroSubsequenceBegin`, конец в `longestZeroSubsequenceEnd`, размер подпоследовательности в `maxSubsequenceSize`. Принимает ссылку на буфер `buf` для хранения временных значений.

---

1. Заголовок: `static void processDiagonalTopRightToBottomLeft(bool **matrix, int n, int k, int row, int column, MatrixIndex *longestZeroSubsequenceBegin, MatrixIndex *longestZeroSubsequenceEnd, int *maxSubsequenceSize, bool *buf)`

2. Назначение: находит в диагонале проходящей сверху справа - влево вниз по индексу (`row`; `column`) в матрице `matrix` размером `n` × `k` наидлиннейшую подпоследовательность из нулей и сохраняет её начало в `longestZeroSubsequenceBegin`, конец в `longestZeroSubsequenceEnd`, размер подпоследовательности в `maxSubsequenceSize`. Принимает ссылку на буфер `buf` для хранения временных значений.

---

1. Заголовок: `static int max(int a, int b)`
2. Назначение: Возвращает максимальное из `a` и `b`

---

1. Заголовок: `void findLongestZeroLineInMatrix(bool **matrix, int n, int k, MatrixIndex *longestZeroSubsequenceBegin, MatrixIndex *longestZeroSubsequenceEnd, int *maxSubsequenceSize)`
2. Назначение: Находит в матрице `matrix` размером `n` × `k` наидлиннейшую подпоследовательность из нулей и сохраняет её начало в `longestZeroSubsequenceBegin`, конец в `longestZeroSubseque` размер подпоследовательности в `maxSubsequenceSize`.

6.4 Блок-схема

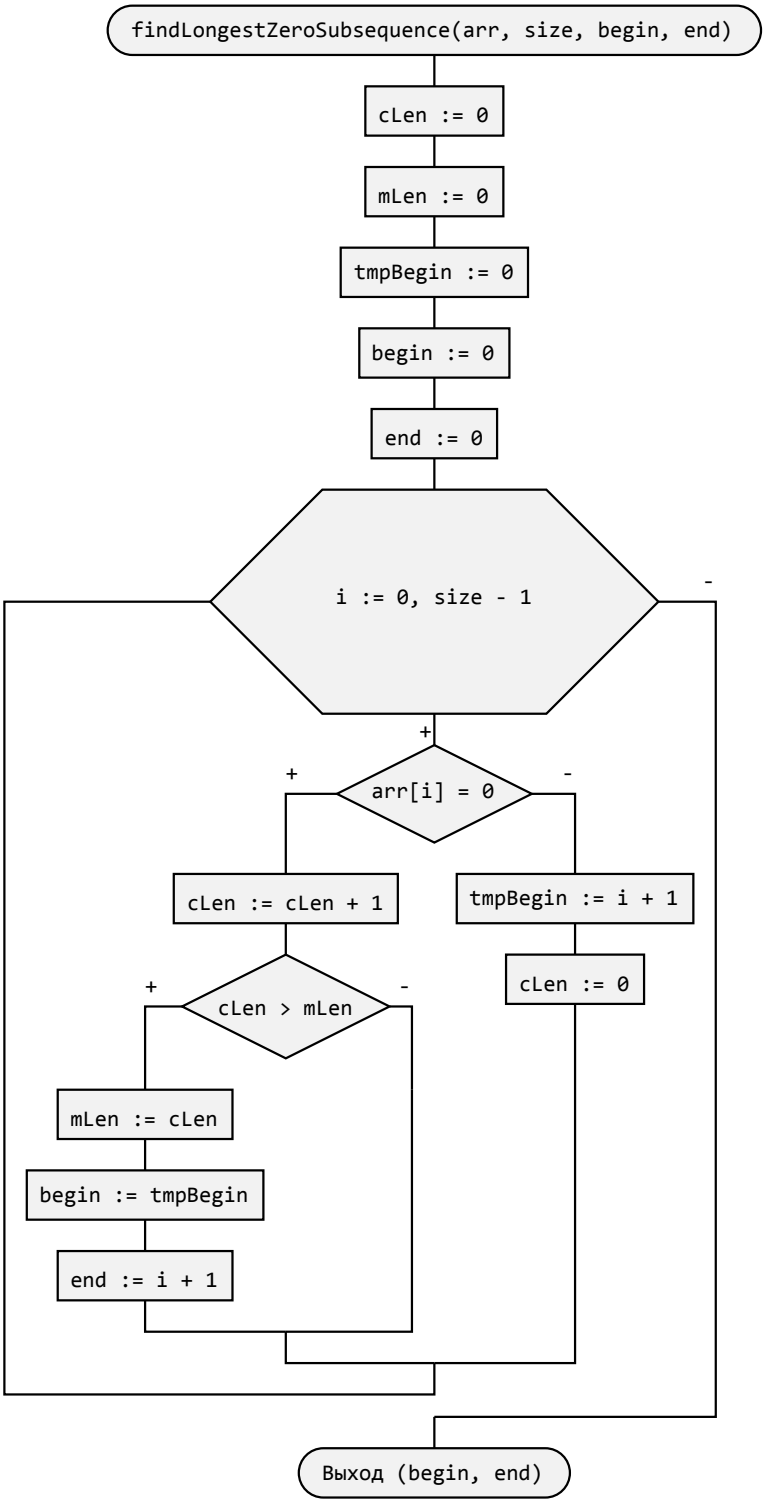

Рисунок 16: Функция findLongestZeroSubsequence

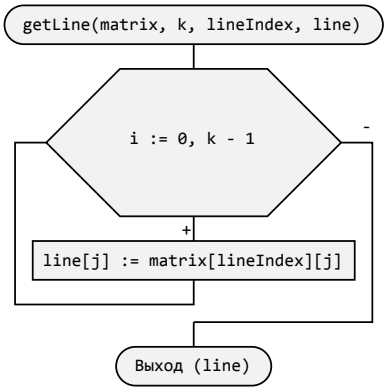

Рисунок 17: Функция getLine

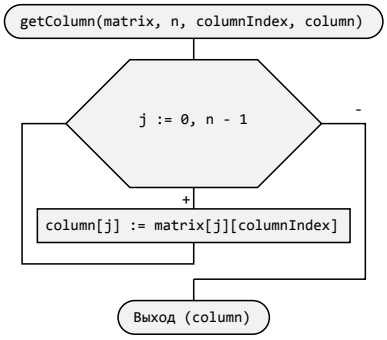

Рисунок 18: Функция getColumn

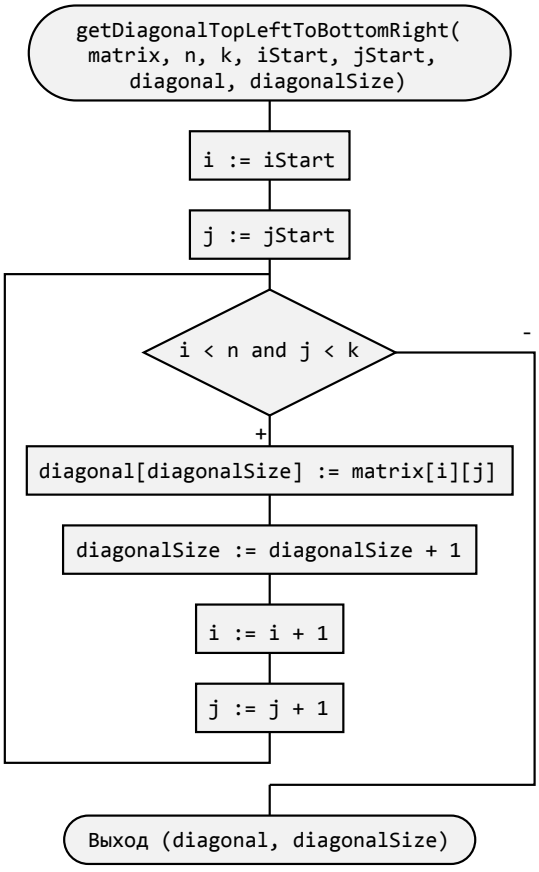

Рисунок 19: Функция getDiagonalTopLeftToBottomRight

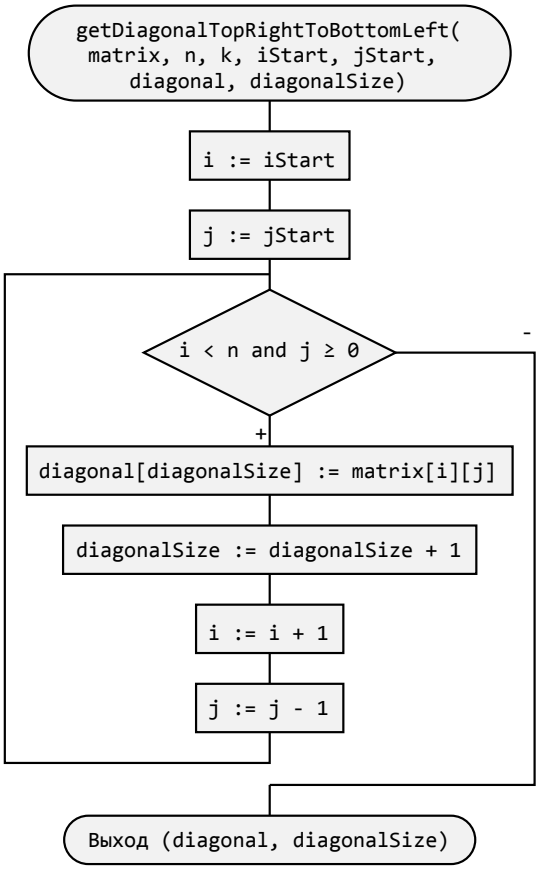

Рисунок 20: Функция getDiagonalTopRightToBottomLeft

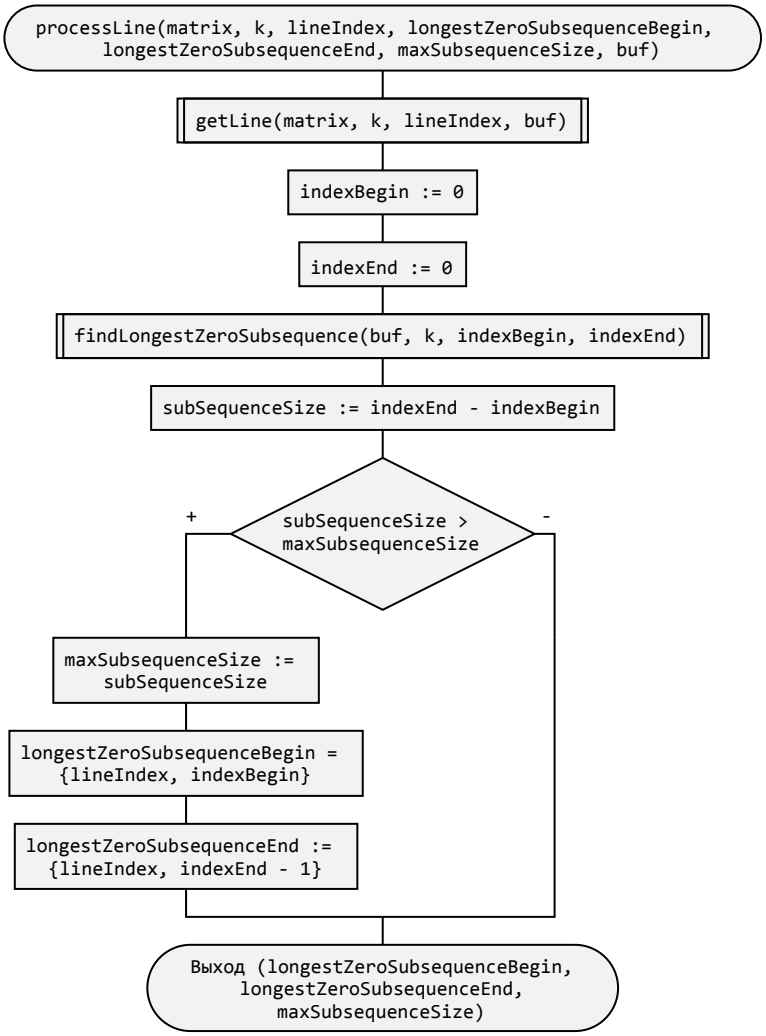

Рисунок 21: Функция processLine

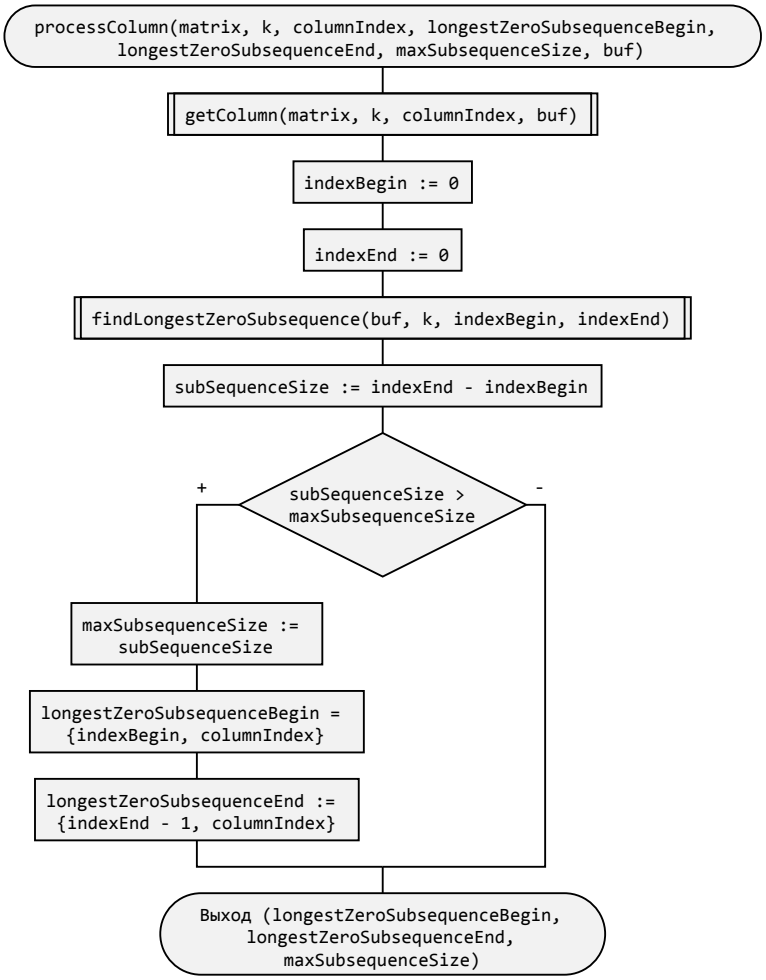

Рисунок 22: Функция processColumn

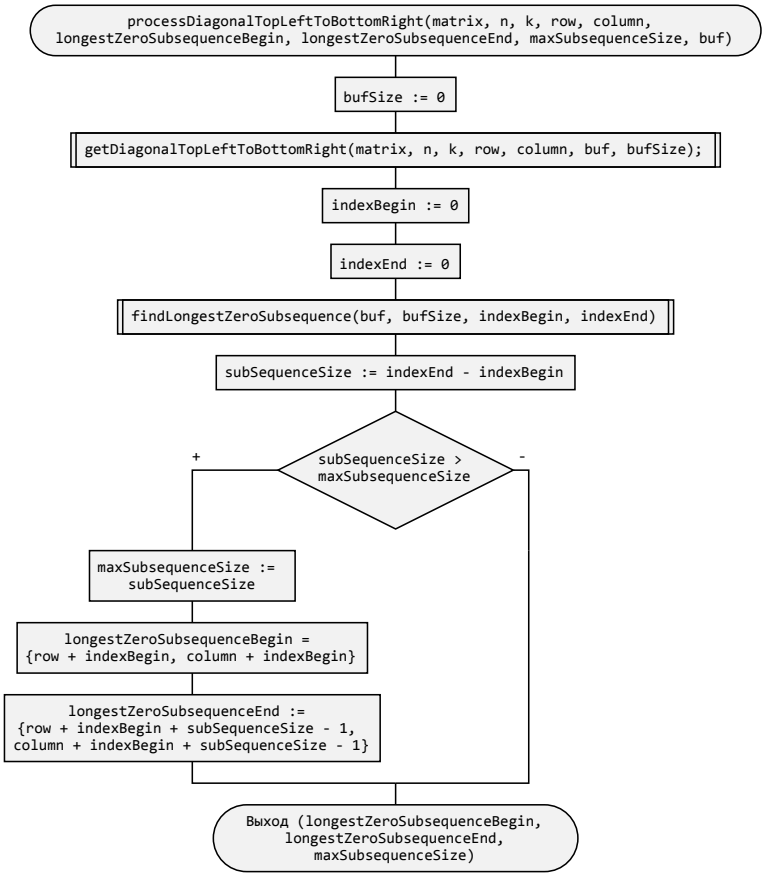

Рисунок 23: Функция processDiagonalTopLeftToBottomRight

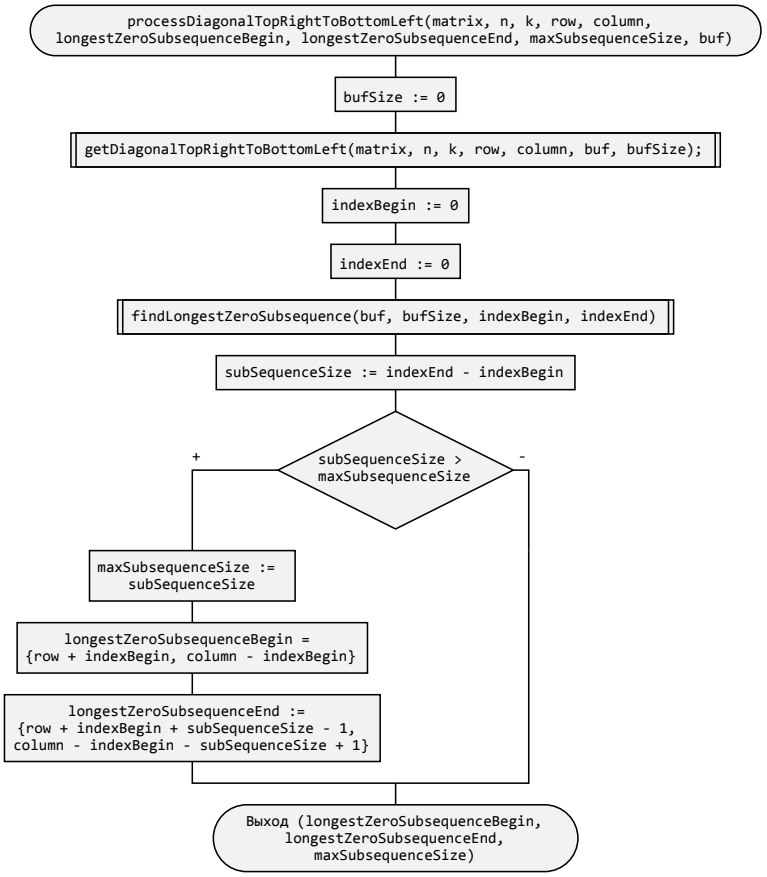

Рисунок 24: Функция processDiagonalTopRightToBottomLeft

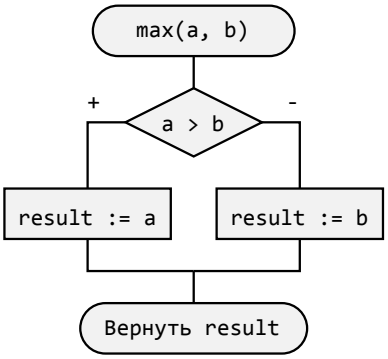

Рисунок 25: Функция max

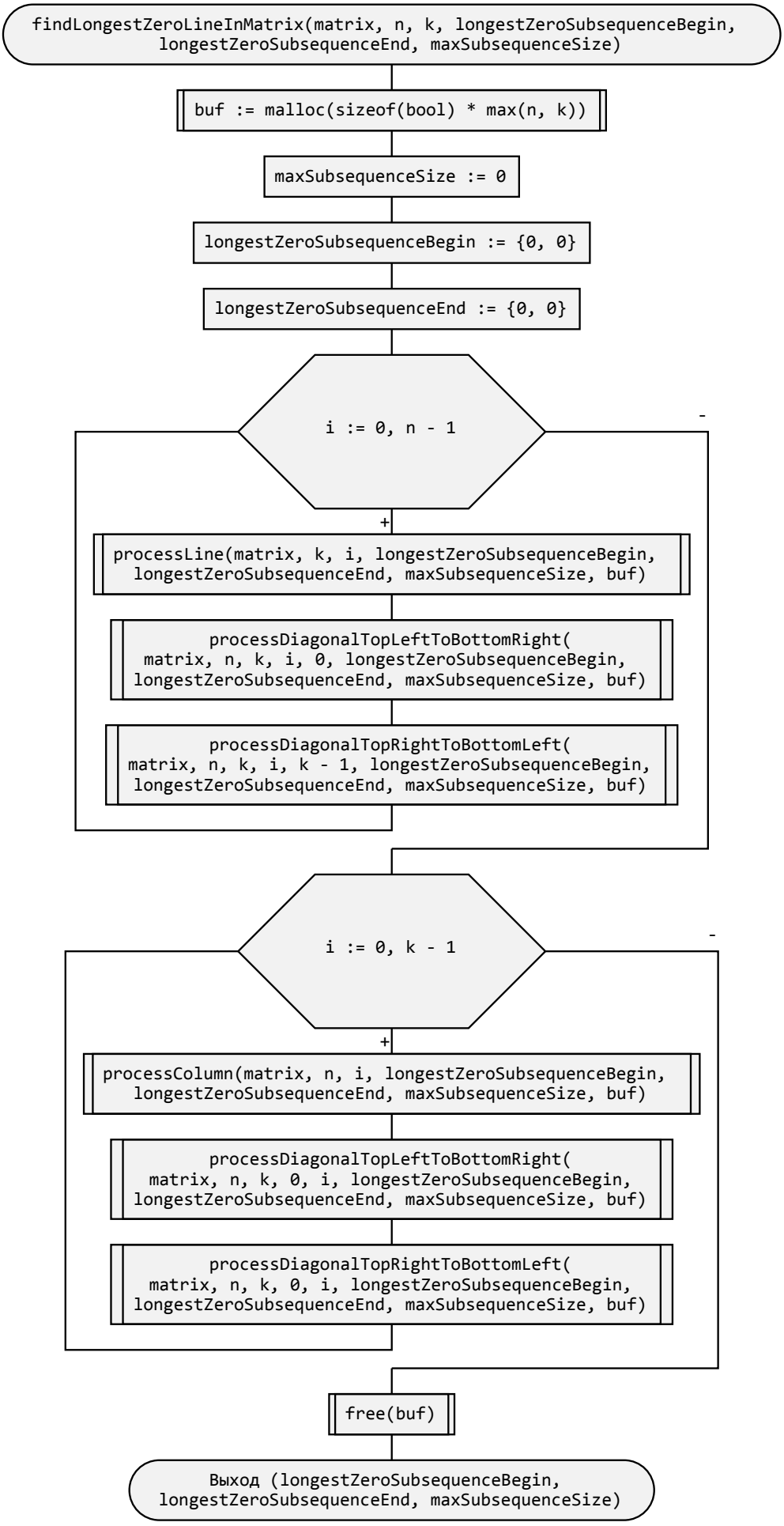
findLongestZeroLineInMatrix(matrix, n, k, longestZeroSubsequenceBegin, longestZeroSubsequenceEnd, maxSubsequenceSize)
buf := malloc(sizeof(bool) * max(n, k))
maxSubsequenceSize := 0
longestZeroSubsequenceBegin := {0, 0}
longestZeroSubsequenceEnd := {0, 0}
i := 0, n - 1
-
+
processLine(matrix, k, i, longestZeroSubsequenceBegin, longestZeroSubsequenceEnd, maxSubsequenceSize, buf)
processDiagonalTopLeftToBottomRight(matrix, n, k, i, 0, longestZeroSubsequenceBegin, longestZeroSubsequenceEnd, maxSubsequenceSize, buf)
processDiagonalTopRightToBottomLeft(matrix, n, k, i, k - 1, longestZeroSubsequenceBegin, longestZeroSubsequenceEnd, maxSubsequenceSize, buf)
i := 0, k - 1
-
+
processColumn(matrix, n, i, longestZeroSubsequenceBegin, longestZeroSubsequenceEnd, maxSubsequenceSize, buf)
processDiagonalTopLeftToBottomRight(matrix, n, k, 0, i, longestZeroSubsequenceBegin, longestZeroSubsequenceEnd, maxSubsequenceSize, buf)
processDiagonalTopRightToBottomLeft(matrix, n, k, 0, i, longestZeroSubsequenceBegin, longestZeroSubsequenceEnd, maxSubsequenceSize, buf)
free(buf)
Выход (longestZeroSubsequenceBegin, longestZeroSubsequenceEnd, maxSubsequenceSize)

Рисунок 26: Функция findLongestZeroLineInMatrix

6.5 Код программы

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

#include "Tasks.h"

static void findLongestZeroSubsequence(bool *arr, int size, int *begin, int *end) {
    int cLen = 0;
    int mLen = 0;
    int tmpBegin = 0;
    *begin = 0;
    *end = 0;

    for (int i = 0; i < size; i++) {
        if (!arr[i]) {
            cLen++;

            if (cLen > mLen) {
                mLen = cLen;
                *begin = tmpBegin;
                *end = i + 1;
            }
        } else {
            tmpBegin = i + 1;
            cLen = 0;
        }
    }
}

static void getLine(bool **matrix, int k, int lineIndex, bool *line) {
    for (int j = 0; j < k; j++)
        line[j] = matrix[lineIndex][j];
}

static void getColumn(bool **matrix, int n, int columnIndex, bool *column) {
    for (int j = 0; j < n; j++)
        column[j] = matrix[j][columnIndex];
}

static void getDiagonalTopLeftToBottomRight(bool **matrix, int n, int k, int iStart, int jStart, bool
↪  *diagonal, int *diagonalSize) {
    for (int i = iStart, j = jStart; i < n && j < k; i++, j++)
        diagonal[(*diagonalSize)++] = matrix[i][j];
}
```

```
static void getDiagonalTopRightToBottomLeft(bool **matrix, int n, int k, int iStart, int jStart, bool
↪  *diagonal, int *diagonalSize) {
    for (int i = iStart, j = jStart; i < n && j >= 0; i++, j--)
        diagonal[(*diagonalSize)++] = matrix[i][j];
}

static void processLine(bool **matrix, int k, int lineIndex, MatrixIndex *longestZeroSubsequenceBegin,
                MatrixIndex *longestZeroSubsequenceEnd, int *maxSubsequenceSize, bool *buf) {
    getLine(matrix, k, lineIndex, buf);

    int indexBegin, indexEnd;
    findLongestZeroSubsequence(buf, k, &indexBegin, &indexEnd);

    int subSequenceSize = indexEnd - indexBegin;
    if (subSequenceSize > *maxSubsequenceSize) {
        *maxSubsequenceSize = subSequenceSize;
        *longestZeroSubsequenceBegin = (MatrixIndex) {lineIndex, indexBegin};
        *longestZeroSubsequenceEnd = (MatrixIndex) {lineIndex, indexEnd - 1};
    }
}

static void processColumn(bool **matrix, int n, int columnIndex, MatrixIndex
↪  *longestZeroSubsequenceBegin,
                  MatrixIndex *longestZeroSubsequenceEnd, int *maxSubsequenceSize, bool *buf) {
    getColumn(matrix, n, columnIndex, buf);

    int indexBegin, indexEnd;
    findLongestZeroSubsequence(buf, n, &indexBegin, &indexEnd);

    int subSequenceSize = indexEnd - indexBegin;
    if (subSequenceSize > *maxSubsequenceSize) {
        *maxSubsequenceSize = subSequenceSize;
        *longestZeroSubsequenceBegin = (MatrixIndex) {indexBegin, columnIndex};
        *longestZeroSubsequenceEnd = (MatrixIndex) {indexEnd - 1, columnIndex};
    }
}

static void processDiagonalTopLeftToBottomRight(bool **matrix, int n, int k, int row, int column,
                                        MatrixIndex *longestZeroSubsequenceBegin,
                                        MatrixIndex *longestZeroSubsequenceEnd, int *maxSubsequenceSize,
↪  bool *buf) {
    int bufSize = 0;
    getDiagonalTopLeftToBottomRight(matrix, n, k, row, column, buf, &bufSize);

    int indexBegin, indexEnd;
    findLongestZeroSubsequence(buf, bufSize, &indexBegin, &indexEnd);

    int subSequenceSize = indexEnd - indexBegin;
    if (subSequenceSize > *maxSubsequenceSize) {
        *maxSubsequenceSize = subSequenceSize;
```

```
        *longestZeroSubsequenceBegin = (MatrixIndex) {row + indexBegin, column + indexBegin};
        *longestZeroSubsequenceEnd = (MatrixIndex) {row + indexBegin + subSequenceSize - 1, column +
↪  indexBegin + subSequenceSize - 1};
    }
}

static void processDiagonalTopRightToBottomLeft(bool **matrix, int n, int k, int row, int column,
                                        MatrixIndex *longestZeroSubsequenceBegin,
                                        MatrixIndex *longestZeroSubsequenceEnd, int *maxSubsequenceSize,
↪  bool *buf) {
    int bufSize = 0;
    getDiagonalTopRightToBottomLeft(matrix, n, k, row, column, buf, &bufSize);

    int indexBegin, indexEnd;
    findLongestZeroSubsequence(buf, bufSize, &indexBegin, &indexEnd);

    int subSequenceSize = indexEnd - indexBegin;
    if (subSequenceSize > *maxSubsequenceSize) {
        *maxSubsequenceSize = subSequenceSize;
        *longestZeroSubsequenceBegin = (MatrixIndex) {row + indexBegin, column - indexBegin};
        *longestZeroSubsequenceEnd = (MatrixIndex) {row + indexBegin + subSequenceSize - 1, column -
↪  indexBegin - subSequenceSize + 1};
    }
}

int max(int a, int b) {
    return a > b ? a : b;
}

void findLongestZeroLineInMatrix(bool **matrix, int n, int k, MatrixIndex *longestZeroSubsequenceBegin,
                                 MatrixIndex *longestZeroSubsequenceEnd, int *maxSubsequenceSize) {
    bool *buf = malloc(sizeof(bool) * max(n, k));
    *maxSubsequenceSize = 0;
    *longestZeroSubsequenceBegin = (MatrixIndex) {0, 0};
    *longestZeroSubsequenceEnd = (MatrixIndex) {0, 0};

    // Сначала пройдёмся по строчкам
    for (int i = 0; i < n; i++) {
        processLine(matrix, k, i, longestZeroSubsequenceBegin, longestZeroSubsequenceEnd,
↪  maxSubsequenceSize, buf);
        // Для каждой строчки здесь же будем находить диагональ слева сверху до нижней правой части с
↪  левого края матрицы
        processDiagonalTopLeftToBottomRight(matrix, n, k, i, 0, longestZeroSubsequenceBegin,
                                            longestZeroSubsequenceEnd, maxSubsequenceSize, buf);
        // И диагональ сверху справа до нижней левой части с правого края матрицы
        processDiagonalTopRightToBottomLeft(matrix, n, k, i, k - 1, longestZeroSubsequenceBegin,
                                            longestZeroSubsequenceEnd, maxSubsequenceSize, buf);
    }

    // Потом пройдёмся по столбцам
```

```
     for (int i = 0; i < k; i++) {
        processColumn(matrix, n, i, longestZeroSubsequenceBegin, longestZeroSubsequenceEnd,
↪   maxSubsequenceSize, buf);
        // Для каждого столбца будем находить оба вида диагоналей.
        processDiagonalTopLeftToBottomRight(matrix, n, k, 0, i, longestZeroSubsequenceBegin,
                                            longestZeroSubsequenceEnd, maxSubsequenceSize, buf);
        processDiagonalTopRightToBottomLeft(matrix, n, k, 0, i, longestZeroSubsequenceBegin,
                                            longestZeroSubsequenceEnd, maxSubsequenceSize, buf);
    }

    free(buf);
}
```

6.6 Тестовые данные

Таблица 8

Тестовые данные findLongestZeroLineInMatrix

Тестовые данные	Ожидаемый результат
()	Пустая последовательность
$\begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$	Пустая последовательность
$\begin{pmatrix} 1 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \end{pmatrix}$	$\begin{pmatrix} 1 & 1 & 1 \\ 1 & \mathbf{0} & 1 \\ 1 & 1 & 1 \end{pmatrix}$
$\begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 \end{pmatrix}$	$\begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ 1 & 1 & 1 & 1 & 1 \end{pmatrix}$
$\begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 \end{pmatrix}$	$\begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & \mathbf{0} & \mathbf{0} & \mathbf{0} \end{pmatrix}$
$\begin{pmatrix} 1 & 1 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 0 \end{pmatrix}$	$\begin{pmatrix} 1 & 1 & \mathbf{0} \\ 1 & 1 & \mathbf{0} \\ 1 & 1 & \mathbf{0} \\ 1 & 1 & \mathbf{0} \\ 1 & 1 & \mathbf{0} \end{pmatrix}$
$\begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 0 & 1 \end{pmatrix}$	$\begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & \mathbf{0} & 1 \\ 1 & \mathbf{0} & 1 \end{pmatrix}$
$\begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \end{pmatrix}$	$\begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & \mathbf{0} \\ 1 & 1 & \mathbf{0} & 1 \\ 1 & \mathbf{0} & 1 & 1 \end{pmatrix}$

$\begin{pmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix}$	$\begin{pmatrix} 1 & 1 & 1 & 1 \\ \mathbf{0} & 1 & 1 & 1 \\ 1 & \mathbf{0} & 1 & 1 \\ 1 & 1 & \mathbf{0} & 1 \\ 1 & 1 & 1 & \mathbf{0} \end{pmatrix}$
$\begin{pmatrix} 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{pmatrix}$	$\begin{pmatrix} 1 & \mathbf{0} & 1 & 1 \\ 1 & 1 & \mathbf{0} & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{pmatrix}$
$\begin{pmatrix} 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 & 1 \end{pmatrix}$	$\begin{pmatrix} 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & \mathbf{0} & 0 \\ 1 & 1 & \mathbf{0} & 1 & 1 \\ 1 & \mathbf{0} & 0 & 0 & 1 \\ \mathbf{0} & 1 & 1 & 1 & 1 \end{pmatrix}$

6.7 Результат выполнения тестов

```
      Start 44: testTask6Empty
44/72 Test #44: testTask6Empty .................................   Passed    0.19 sec
      Start 45: testTask6OnlyOnes
45/72 Test #45: testTask6OnlyOnes ..............................   Passed    0.20 sec
      Start 46: testTask6SingleZero
46/72 Test #46: testTask6SingleZero ............................   Passed    0.19 sec
      Start 47: testTask6FullLine
47/72 Test #47: testTask6FullLine ..............................   Passed    0.19 sec
      Start 48: testTask6PartLine
48/72 Test #48: testTask6PartLine ..............................   Passed    0.19 sec
      Start 49: testTask6FullColumn
49/72 Test #49: testTask6FullColumn ............................   Passed    0.19 sec
      Start 50: testTask6PartColumn
50/72 Test #50: testTask6PartColumn ............................   Passed    0.19 sec
      Start 51: testTask6FullBottomLeftTopRightDiagonal
51/72 Test #51: testTask6FullBottomLeftTopRightDiagonal ........   Passed    0.19 sec
      Start 52: testTask6PartBottomLeftTopRightDiagonal
52/72 Test #52: testTask6PartBottomLeftTopRightDiagonal ........   Passed    0.19 sec
      Start 53: testTask6FullBottomRightTopLeftDiagonal
53/72 Test #53: testTask6FullBottomRightTopLeftDiagonal ........   Passed    0.19 sec
      Start 54: testTask6PartBottomRightTopLeftDiagonal
54/72 Test #54: testTask6PartBottomRightTopLeftDiagonal ........   Passed    0.19 sec
      Start 55: testTask6Regular
55/72 Test #55: testTask6Regular ...............................   Passed    0.19 sec
```

Рисунок 27: Результат выполнения тестов функции findLongestZeroLineInMatrix

7 Арифметика

7.1 Задание варианта

Пусть m натуральных чисел заданы своими цифрами в q-ичной системе счисления, хранящимися в строках матрицы $K(m, n)$. Найти сумму этих чисел в той же системе, не вычисляя самих чисел $(q \leq 10)$.

7.2 Обоснование

Будем перебирать цифры числа с конца. Для всех чисел будем находить сумму всех цифр и перенос из предыдущего разряда, остаток от деления этой суммы на основание сформирует в новом числе цифру, целочисленное деление - перенос в следующий разряд. Будем выполнять это жействие до тех пор пока не закончатся цифры.

7.3 Спецификация

1. Заголовок: `void sum(unsigned int **numbers, int m, int n, int q, unsigned int *ans)`
2. Назначение: вычисляет сумму чисел `numbers` количеством `m` с количеством цифр `n` в `q`-ичной системе счисления и записывает сумму в той же системе счисления в `ans`.

7.4 Блок-схема

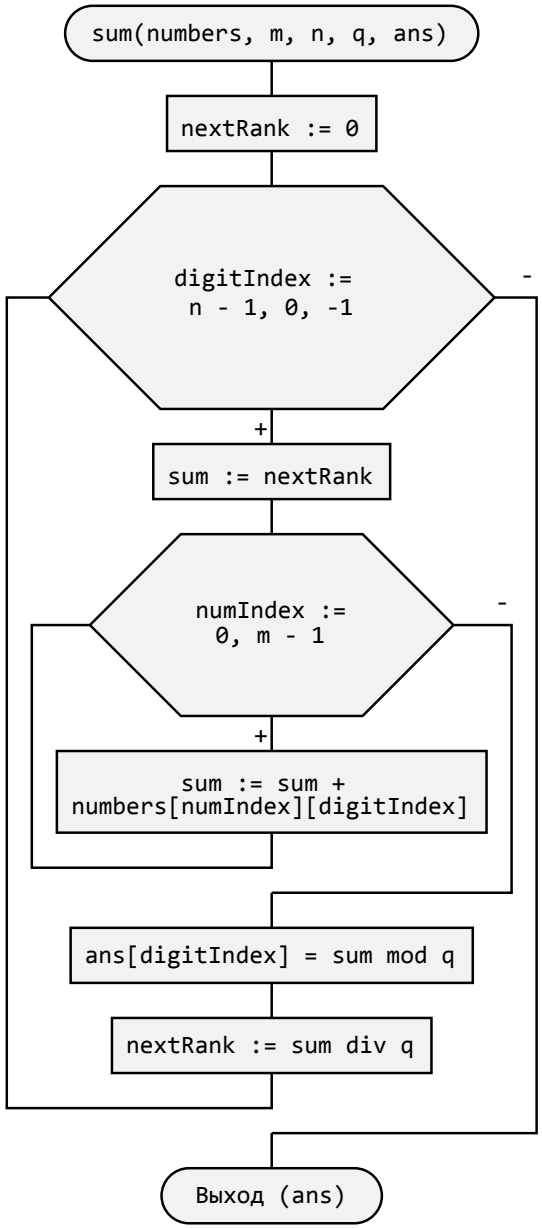

Рисунок 28: Функция sum

7.5 Код программы

```
void sum(unsigned int **numbers, int m, int n, int q, unsigned int *ans) {
    unsigned int nextRank = 0;

    // Будем идти с конца чисел
    for (int digitIndex = n - 1; digitIndex >= 0; digitIndex--) {
        // Посчитаем сумму цифр. Сумме цифр сразу присваиваем перенос из предыдущего разряда.
        unsigned int sum = nextRank;
```

```
        // Из каждого числа добавляем цифру
        for (int numIndex = 0; numIndex < m; numIndex++)
            sum += numbers[numIndex][digitIndex];

        // В текущий разряд записываем остаток от суммы
        ans[digitIndex] = sum % q;
        // В следующий разряд переносим целочисленное деление
        nextRank = sum / q;
    }
}
```

7.6 Тестовые данные

Таблица 9

Тестовые данные sum

Тестовые данные	Ожидаемый результат
numbers = (), m = 0, n = 0, q = 3	ans = 000_3
numbers = $\begin{pmatrix} 0 & 1 & 0 \\ 1 & 1 & 1 \end{pmatrix}$, m = 2, n = 3, q = 2	ans = 101_2
numbers = $\begin{pmatrix} 0 & 3 & 2 \\ 1 & 1 & 1 \end{pmatrix}$, m = 2, n = 3, q = 4	ans = 103_4
numbers = $\begin{pmatrix} 0 & 3 & 2 \\ 1 & 1 & 1 \\ 1 & 2 & 1 \\ 0 & 0 & 3 \end{pmatrix}$, m = 4, n = 3, q = 4	ans = 232_4
numbers = $\begin{pmatrix} 0 & 3 & 2 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \end{pmatrix}$, m = 4, n = 4, q = 2	ans = 1111_2
numbers = $\begin{pmatrix} 0 & 3 & 2 \\ 2 & 1 & 1 \\ 1 & 2 & 0 \\ 0 & 3 & 0 \end{pmatrix}$, m = 4, n = 3, q = 4	ans = 113_4
numbers = $\begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$, m = 3, n = 3, q = 3	ans = 000_3

7.7 Результат выполнения тестов

```
      Start 56: testTask7Zeros
56/72 Test #56: testTask7Zeros .................................   Passed    0.19 sec
      Start 57: testTask7TwoNumbersBin
57/72 Test #57: testTask7TwoNumbersBin .........................   Passed    0.19 sec
      Start 58: testTask7TwoNumbers
58/72 Test #58: testTask7TwoNumbers ............................   Passed    0.19 sec
      Start 59: testTask7MultipleNumbers
59/72 Test #59: testTask7MultipleNumbers .......................   Passed    0.19 sec
      Start 60: testTask7MultipleNumbersBin
60/72 Test #60: testTask7MultipleNumbersBin .....................   Passed    0.19 sec
      Start 61: testTask7Overflow
61/72 Test #61: testTask7Overflow ..............................   Passed    0.19 sec
      Start 62: testTask7MT
62/72 Test #62: testTask7MT ....................................   Passed    0.19 sec
```

Рисунок 29: Результат выполнения тестов функции sum

8 Геометрия и теория множеств

8.1 Задание варианта

Найти площадь многоугольника (не обязательно выпуклого), заданного координатами своих вершин на плоскости в порядке обхода по или против часовой стрелки.

8.2 Обоснование

Площадь многоугольника можно найти методом трапеций. Опустим из каждой точки многоугольника перпендикуляр к Ox, две соседние точки и их перпендикуляры будут образовывать трапецию. Если вторая точка находится правеке текущей, прибавляем сумму к итоговой, иначе - отнимаем. Итоговая сумма будет равна сумме по модулю, так как знак суммарной площади зависит от порядка обхода. Формула для вычисления площади:
$S = |\sum_{i=0}^{n} \frac{(points_i.x - points(i+1)\%n.x) \cdot (points_i.y + points_{(i+1)\%n}.y)}{2}|$, где points - массив точек.

8.3 Спецификация

1. Заголовок: `double getArea(Point *points, int pointsAmount)`
2. Назначение: воозвращает сумму многоугольника, заданного массивом точек `points` размером `pointsAmount`.

8.4 Блок-схема

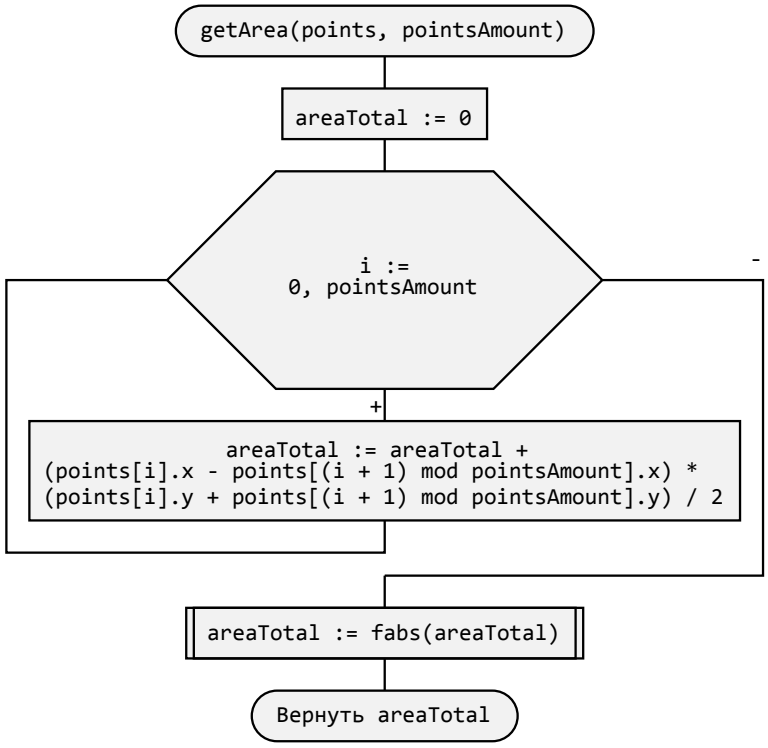

Рисунок 30: Функция getArea

8.5 Код программы

```
#include <math.h>

#include "Tasks.h"

double getArea(Point *points, int pointsAmount) {
    double areaTotal = 0;

    for (int i = 0; i < pointsAmount; i++)
        areaTotal += (points[i].x - points[(i + 1) % pointsAmount].x) *
                (points[i].y + points[(i + 1) % pointsAmount].y) / 2;

    areaTotal = fabs(areaTotal);

    return areaTotal;
}
```

8.6 Тестовые данные

Таблица 10

Тестовые данные getArea

Тестовые данные	Ожидаемый результат
pointsAmount = 0	0
points = (1, 1) (5, 5) pointsAmount = 2	0
points = (0, 0) (5, 5) (5, 0) pointsAmount = 3	12.5
points = (2, 1) (5, 3) (6, 6) (5, 7) (3, 6) (2, 8) (1, 5) pointsAmount = 7	19.5
points = (2, 1) (1, 5) (2, 8) (3, 6) (5, 7) (6, 6) (5, 3) pointsAmount = 7	19.5

8.7 Результат выполнения тестов

```
      Start 63: testTask8NoPoints
63/72 Test #63: testTask8NoPoints ..............................   Passed    0.19 sec
      Start 64: testTask8TwoPoints
64/72 Test #64: testTask8TwoPoints ..............................   Passed    0.22 sec
      Start 65: testTask8Triangle
65/72 Test #65: testTask8Triangle ..............................   Passed    0.19 sec
      Start 66: testTask8Polygon
66/72 Test #66: testTask8Polygon ...............................   Passed    0.19 sec
      Start 67: testTask8RevertedPolygon
67/72 Test #67: testTask8RevertedPolygon .......................   Passed    0.18 sec
```

Рисунок 31: Результат выполнения тестов функции getArea

9 Линейная алгебра и сжатие информации

9.1 Задание варианта

Матрица $A(n, n)$ системы линейных уравнений $AX = B$ приведена к верхнетреугольному виду и упакована в одномерный массив. Найти вектор решения X последовательной подстановкой, не распаковывая A.

9.2 Обоснование

Рассмотрим для примера матрицу 3 на 3

$$\begin{pmatrix} x_1 & x_2 & x_3 \\ 0 & x_4 & x_5 \\ 0 & 0 & x_6 \end{pmatrix} \times \begin{pmatrix} t_1 & t_2 & t_3 \\ t_4 & t_5 & t_6 \\ t_7 & t_8 & t_9 \end{pmatrix} = \begin{pmatrix} b_1 & b_2 & b_3 \\ b_4 & b_5 & b_6 \\ b_7 & b_8 & b_9 \end{pmatrix}$$

$$\begin{pmatrix} x_1 \cdot t_1 + x_2 \cdot t_4 + x_3 \cdot t_7 & x_1 \cdot t_2 + x_2 \cdot t_5 + x_3 \cdot t_8 & x_1 \cdot t_3 + x_2 \cdot t_6 + x_3 \cdot t_9 \\ x_4 \cdot t_4 + x_5 \cdot t_7 & x_4 \cdot t_2 + x_5 \cdot t_5 & x_4 \cdot t_6 + x_5 \cdot t_9 \\ x_6 \cdot t_7 & x_6 \cdot t_8 & x_6 \cdot t_9 \end{pmatrix} = \begin{pmatrix} b_1 & b_2 & b_3 \\ b_4 & b_5 & b_6 \\ b_7 & b_8 & b_9 \end{pmatrix}$$

Можно заметить, что легко можно вычислить t_7, t_8, t_9. Далее на основе вычисленных t_7, t_8, t_9 можно вычислить t_4, t_5, t_6, и так далее до тех пор пока не получим полную матрицу X.

9.3 Спецификация

1. Заголовок: `void findX(double *matrixA, int n, double **matrixB, double **matrixX)`
2. Назначение: находит в уравнении вида $AX = B$ вектор X, где A - `matrixA`, верхнетреугольная упакованная матрица, `n` - размер стороны матриц, B - `matrixB`. Сохраняет ответ X в `matrixX`.

9.4 Блок-схема

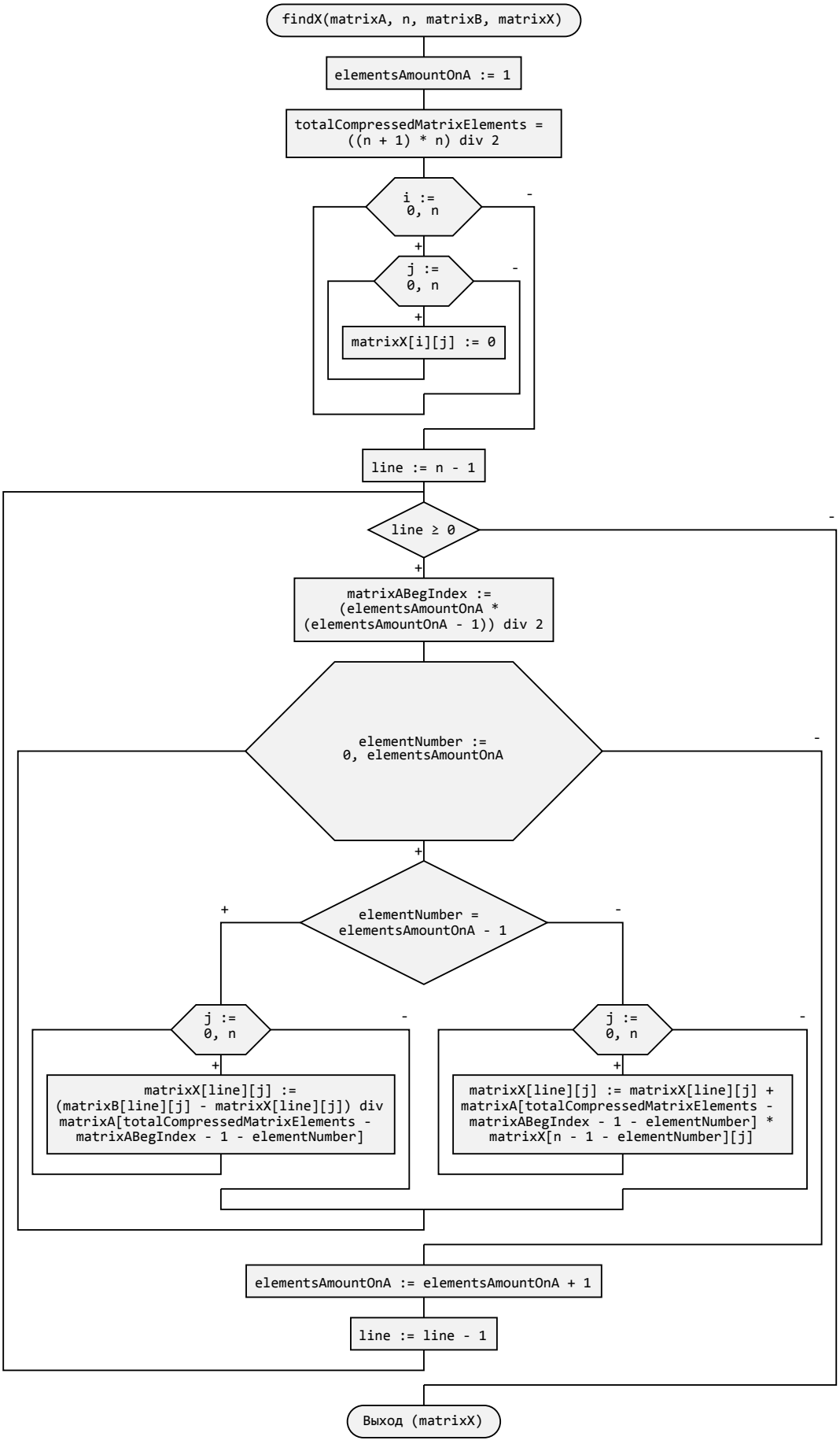

Рисунок 32: Функция findX

9.5 Код программы

```
void findX(double *matrixA, int n, double **matrixB, double **matrixX) {
    /*
    Здесь верхнетреугольная матрица matrixA

              {1, -2, 0}
     matrixA = {0, 4, 17} -> {1, -2, 0, 4, 17, 1}
              {0, 0, 1}

     */
    // Кол-во элементов в одной линии верхнетреугольной матрицы. Будем идти снизу-вверх и справа-налево
↪   по этой матрице
    int elementsAmountOnA = 1;
    // Общее количество элементов в упакованной верхнетреугольной матрице
    int totalCompressedMatrixElements = ((n + 1) * n) / 2;

    // Заполним матрицу нулями
    for (int i = 0; i < n; i++)
        for (int j = 0; j < n; j++)
            matrixX[i][j] = 0;

    // Счётчик для линии
    for (int line = n - 1; line >= 0; line--) {

        // Индекс с которого будет начинаться упакованная верхнетреугольная матрица в line
        int matrixABegIndex = (elementsAmountOnA * (elementsAmountOnA - 1)) / 2;

        // Далее будем перебирать все элементы верхнетреугольной матрицы в этой линии
        for (int elementNumber = 0; elementNumber < elementsAmountOnA; elementNumber++)
            // Если мы достигли конца линии (самый левый элемент), значит мы может легко вычислить
↪   элемент из x
            if (elementNumber == elementsAmountOnA - 1)
                for (int j = 0; j < n; j++)
                    matrixX[line][j] =
                            (matrixB[line][j] - matrixX[line][j]) /
                            matrixA[totalCompressedMatrixElements - matrixABegIndex - 1 - elementNumber];
        // Иначе будем вычислять части массива X суммируя произведение текущего элемента упакованной
↪   матрицы и
        // элемента ниже.
        else
                for (int j = 0; j < n; j++)
                    matrixX[line][j] += matrixA[totalCompressedMatrixElements - matrixABegIndex - 1 -
↪   elementNumber] *
                            matrixX[n - 1 - elementNumber][j];
```

```
        // На следующей линии будет на 1 элемент больше
        elementsAmountOnA++;
    }
}
```

9.6 Тестовые данные

Таблица 11

Тестовые данные findX

Тестовые данные	Ожидаемый результат
$A = (), B = ()$	$X = ()$
$A = (4), B = (-8)$	$X = (-2)$
$A = \begin{pmatrix} 4 & 1 & 3 \\ 0 & 2 & 0 \\ 0 & 0 & -9 \end{pmatrix}$ или $A = \{4, 1, 3, 2, 0, -9\}$ в упакованном виде, $B = \begin{pmatrix} -31 & 99 & 18 \\ 8 & 0 & 2 \\ 81 & -45 & -63 \end{pmatrix}$	$X = \begin{pmatrix} -2 & 21 & -1 \\ 4 & 0 & 1 \\ -9 & 5 & 7 \end{pmatrix}$
$A = \begin{pmatrix} 4 & 1 & 3 & 3 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & -9 & 5 \\ 0 & 0 & 0 & 6 \end{pmatrix}$ или $A = \{4, 1, 3, 3, 2, 0, 0, -9, 5, 6\}$ в упакованном виде, $B = \begin{pmatrix} -19 & 72 & 54 & 87 \\ 8 & 0 & 2 & 12 \\ 101 & -90 & -3 & -37 \\ 24 & -54 & 72 & 42 \end{pmatrix}$	$X = \begin{pmatrix} -2 & 21 & -1 & 9 \\ 4 & 0 & 1 & 6 \\ -9 & 5 & 7 & 8 \\ 4 & -9 & 12 & 7 \end{pmatrix}$

9.7 Результат выполнения тестов

```
      Start 68: testTask9ZeroSize
68/72 Test #68: testTask9ZeroSize ..............................   Passed    0.19 sec
      Start 69: testTask9OneSize
69/72 Test #69: testTask9OneSize ...............................   Passed    0.19 sec
      Start 70: testTask9ThreeSize
70/72 Test #70: testTask9ThreeSize .............................   Passed    0.46 sec
      Start 71: testTask9FourSize
71/72 Test #71: testTask9FourSize ..............................   Passed    0.19 sec
```

Рисунок 33: Результат выполнения тестов функции findX

10 Алгоритмы обработки символьной информации

10.1 Задание варианта

Заданный список русских фамилий (вместе с именами и отчествами) упорядочить по алфавиту. Проверить (и исправить, если нужно) написание собственных имен с прописных букв.

10.2 Обоснование

Задача разделяется на две подзадачи: сделать первые буквы ФИО заглавными и отсортировать список ФИО по фамилии. Русские символы в кодировке UTF-8 состоят из двух char и имеют следующие значения

Таблица 12

Русские символы в UTF-8 кодировке

Символ	HEX
Ё	0xDO, 0x81
А	0xDO, 0x90
Б	0xDO, 0x91
В	0xDO, 0x92
Г	0xDO, 0x93
Д	0xDO, 0x94
Е	0xDO, 0x95
Ж	0xDO, 0x96
З	0xDO, 0x97
И	0xDO, 0x98
Й	0xDO, 0x99
К	0xDO, 0x9A
Л	0xDO, 0x9B
М	0xDO, 0x9C
Н	0xDO, 0x9D
О	0xDO, 0x9E
П	0xDO, 0x9F
Р	0xDO, 0xA0
С	0xDO, 0xA1
Т	0xDO, 0xA2
У	0xDO, 0xA3
Ф	0xDO, 0xA4
Х	0xDO, 0xA5
Ц	0xDO, 0xA6
Ч	0xDO, 0xA7
Ш	0xDO, 0xA8
Щ	0xDO, 0xA9
Ъ	0xDO, 0xAA
Ы	0xDO, 0xAB

Ь	0xDO, 0xAC
Э	0xDO, 0xAD
Ю	0xDO, 0xAE
Я	0xDO, 0xAF

Символ	HEX
а	0xDO, 0xB0
б	0xDO, 0xB1
в	0xDO, 0xB2
г	0xDO, 0xB3
д	0xDO, 0xB4
е	0xDO, 0xB5
ж	0xDO, 0xB6
з	0xDO, 0xB7
и	0xDO, 0xB8
й	0xDO, 0xB9
к	0xDO, 0xBA
л	0xDO, 0xBB
м	0xDO, 0xBC
н	0xDO, 0xBD
о	0xDO, 0xBE
п	0xDO, 0xBF
р	0xD1, 0x80
с	0xD1, 0x81
т	0xD1, 0x82
у	0xD1, 0x83
ф	0xD1, 0x84
х	0xD1, 0x85
ц	0xD1, 0x86
ч	0xD1, 0x87
ш	0xD1, 0x88
щ	0xD1, 0x89
ъ	0xD1, 0x8A
ы	0xD1, 0x8B
ь	0xD1, 0x8C

э	0xD1, 0x8D
ю	0xD1, 0x8E
я	0xD1, 0x8F
ё	0xD1, 0x91

Символ в UTF-8 нельзя уместить в char, однако можно поместить в short. Первая вспомогательная функция будет возвращать код русского символа в short - он будет содержать два char. Вторая вспомогательная функция - запись short в массив. Согласно *таблице 12*, прописные буквы подразделяются на три неразрывные области: ё (0xD191), а-п [0xDOB0; 0xDOBF] и р-я [0xD180; 0xD18F]. На основании этих областей будем отнимать от кода текущего символа разницу между кодом прописного и заглавного. Например для е получение кода заглавного Е будет выглядеть так: code(e) - (code(a) - code(A)). Полученный код заглавного символа запишем в массив. Перейдём к второй подзадаче. Для сортировки воспользуемся функцией qsort, если бы исходный текст был в кодировке ASCII, где каждый символ занимает 8 бит, в качестве компаратора мы могли бы использовать стандартный strcmp, однако мы используем русские символы в кодировке UTF-8, где используется 16 бит на русский символ. Кроме того, порядок символов согласно *таблице 12* имеет другой порядок относительно алфавита, поэтому необходимо определить ещё одну функцию, которая определяет приоритет символов в алфавите. И уже с использованием этой функции можно сранивать русские строки аналогично strcmp.

10.3 Спецификация

1. Заголовок: `static uint16_t getRuCode(unsigned char *beg)`
2. Назначение: возвращает значение двухбайтового символа `beg`.

---

1. Заголовок: `static void writeRuCodeToChar(uint16_t code, unsigned char *beg)`
2. Назначение: записывает двухбайтовое значение символа `code` в `beg`.

---

1. Заголовок: `static void firstLetterToupperRu(unsigned char *beg)`
2. Назначение: делает первый символ русской строки `beg` заглавным.

---

1. Заголовок: `void validateProperName(ProperName name)`
2. Назначение: проверяет и исправляет имя собственное `name`.

---

1. Заголовок: `void validateFullName(FullName fullName)`
2. Назначение: проверяет и исправляет фамилию, имя, отчество в `fullName`.

---

1. Заголовок: `int getRuPriority(unsigned char *a)`

2. Назначение: возвращает приоритет русского символа `a`.

---

1. Заголовок: `int getRuPriority(unsigned char *a)`
2. Назначение: возвращает приоритет русского символа `a`.

---

1. Заголовок: `static int fullNameComparator(const void *a, const void *b)`
2. Назначение: возвращает 0, если фамилия `a` = фамилии `b`, значение > 0 если фамилия `a` > фамилии `b`, иначе - значение < 0.

---

1. Заголовок: `void sortBySurname(FullName *namesList, int size)`
2. Назначение: проверяет и исправляет имена собственые а также сортирует по фамилии список ФИО `namesList` размером `size`.

10.4 Блок-схема

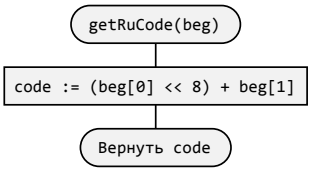

Рисунок 34: Функция getRuCode

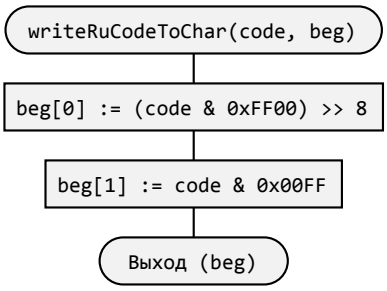

Рисунок 35: Функция writeRuCodeToChar

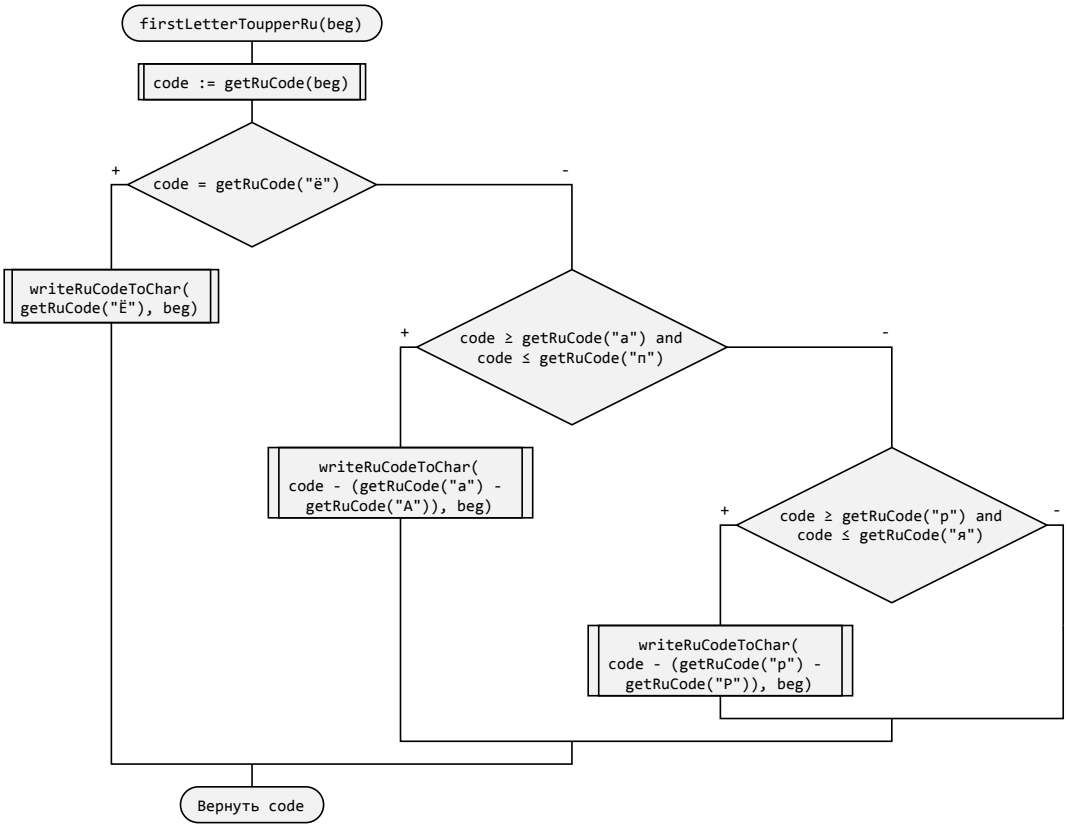

Рисунок 36: Функция firstLetterToupperRu

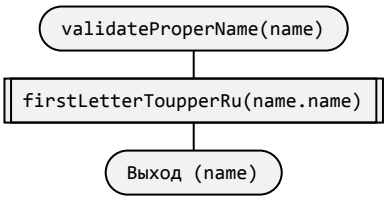

Рисунок 37: Функция validateProperName

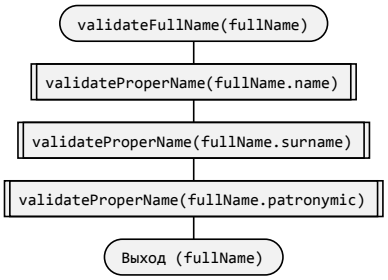

Рисунок 38: Функция validateFullName

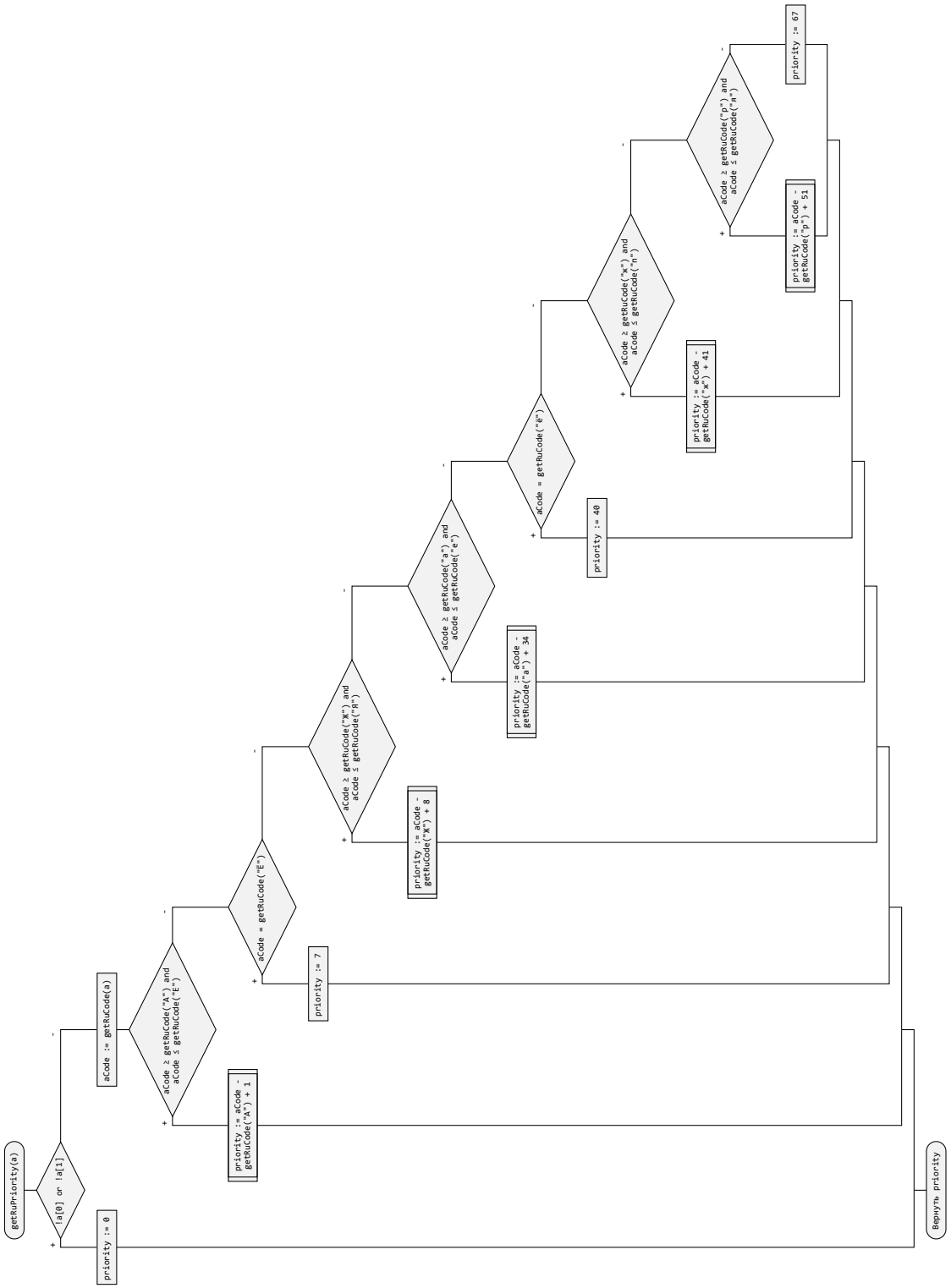

Рисунок 39: Функция getRuPriority

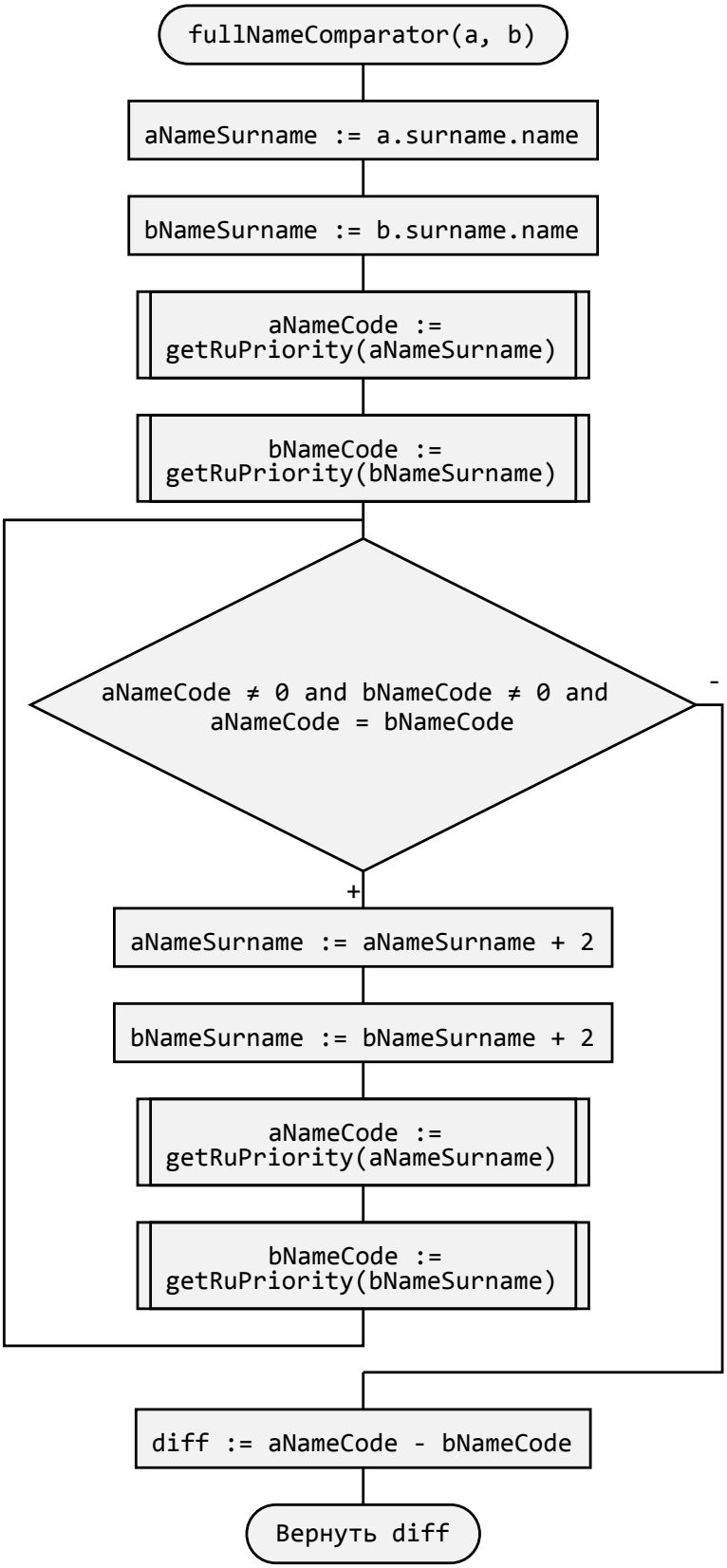

Рисунок 40: Функция fullNameComparator

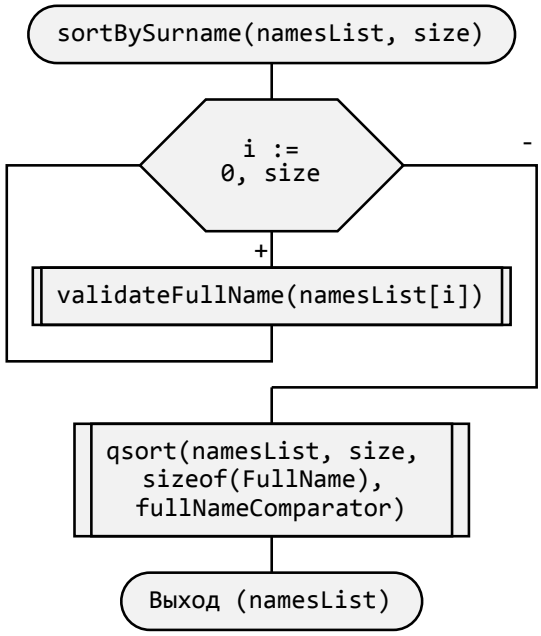

Рисунок 41: Функция sortBySurname

10.5 Код программы

Код программы:

```c
#include <stdlib.h>
#include <stdint.h>

#include "Tasks.h"

// beg содержит ссылку на русский символ в кодировке UTF-8
static uint16_t getRuCode(unsigned char *beg) {
    // Русский символ в UTF-8 состоит из двух байтов, поэтому сохраняем результат в двухбайтовый
↪  беззнаковый тип.
    return (((uint16_t) beg[0]) << 8) + ((uint16_t) beg[1]);
}

// beg содержит ссылку на русский символ в кодировке UTF-8
static void writeRuCodeToChar(uint16_t code, unsigned char *beg) {
    // Русский символ в UTF-8 состоит из двух байтов, поэтому просто копируем их
    beg[0] = (code & 0xFF00) >> 8;
    beg[1] = code & 0x00FF;
}

static void firstLetterToupperRu(unsigned char *beg) {
    uint16_t code = getRuCode(beg);

    // Отдельная проверка для ё, потому что она совсем в другом месте в таблице UTF-8
```

```
    if (code == getRuCode("ё"))
        writeRuCodeToChar(getRuCode("Ё"), beg);
    // Проверка для а-п, их коды: 0xD0B0 - 0xD0BF
    else if (code >= getRuCode("а") && code <= getRuCode("п"))
        writeRuCodeToChar(code - (getRuCode("а") - getRuCode("А")), beg);
    // Почему-то между п и р большой пробел, причём только для прописных букв.
    // Поэтому выполняем отдельную проверку для них. Границы р-я: 0xD1080 - 0xD18F
    else if (code >= getRuCode("р") && code <= getRuCode("я"))
        writeRuCodeToChar(code - (getRuCode("р") - getRuCode("Р")), beg);
}

void validateProperName(ProperName name) {
    // Делаем первый символ имени заглавным
    firstLetterToupperRu(name.name);
}

void validateFullName(FullName fullName) {
    // Проверяем имя, фамилию и отчество на валидность
    validateProperName(fullName.name);
    validateProperName(fullName.surname);
    validateProperName(fullName.patronymic);
}

// Приоритет русских символов
int getRuPriority(unsigned char *a) {
    // Если хоть один из символов ноль-символ - возвращаем 0. Для удобства работы с аналогом strcmp.
    if (!*a || !a[1]) return 0;

    // Получаем код символа
    int aCode = getRuCode(a);

    if (aCode >= getRuCode("А") && aCode <= getRuCode("Е"))
        return aCode - getRuCode("А") + 1;
    else if (aCode == getRuCode("Ё"))
        return 7;
    else if (aCode >= getRuCode("Ж") && aCode <= getRuCode("Я"))
        return aCode - getRuCode("Ж") + 8;
    else if (aCode >= getRuCode("а") && aCode <= getRuCode("е"))
        return aCode - getRuCode("а") + 34;
    else if (aCode == getRuCode("ё"))
        return 40;
    else if (aCode >= getRuCode("ж") && aCode <= getRuCode("п"))
        return aCode - getRuCode("ж") + 41;
    else if (aCode >= getRuCode("р") && aCode <= getRuCode("я"))
        return aCode - getRuCode("р") + 51;

    // Если получить приоритет не удалось - ставим символ в самую последнюю очередь.
    return 67;
}
```

```c
// Замена strcmp, но для русских символов в UTF-8
static int fullNameComparator(const void *a, const void *b) {
    unsigned char* aNameSurname = (*(FullName *) a).surname.name;
    unsigned char* bNameSurname = (*(FullName *) b).surname.name;

    int aNameCode = getRuPriority(aNameSurname);
    int bNameCode = getRuPriority(bNameSurname);
    while (aNameCode && bNameCode && aNameCode == bNameCode) {
        aNameSurname += 2;
        bNameSurname += 2;

        aNameCode = getRuPriority(aNameSurname);
        bNameCode = getRuPriority(bNameSurname);
    }

    return aNameCode - bNameCode;
}

// Так как кодировка в задании не была уточнена, реализовал алгоритм для кодировки UTF-8.
// получилось "самую капельку" сложнее, зато интереснее.
void sortBySurname(FullName *namesList, int size) {
    for (int i = 0; i < size; i++)
        validateFullName(namesList[i]);

    qsort(namesList, size, sizeof(FullName), fullNameComparator);
}
```

10.6 Тестовые данные

Таблица 13

Тестовые данные findX

Тестовые данные	Ожидаемый результат
namesList =	namesList =
фадей трифонов витальевич	Фадей Богданов Витальевич
инесса филатова фомовна	Инесса Ё Фомовна
белла пестова даниловна	Белла Ёа Даниловна
ибрагим пестоваееее романович	Ибрагим Ёааа Романович
степан пестоваб игнатович	Степан Ёаав Игнатович
фадей богданов витальевич	Белла Пестова Даниловна
инесса ё фомовна	Ибрагим Пестоваееее Романович
белла ёа даниловна	Степан Пестоваб Игнатович
ибрагим ёааа романович	Фадей Трифонов Витальевич
степан ёаав игнатович	Инесса Филатова Фомовна

10.7 Результат выполнения тестов

```
      Start 72: testTask10SortTest
72/72 Test #72: testTask10SortTest ..............................   Passed    0.00 sec
```

Рисунок 42: Результат выполнения тестов функции sortBySurname

11 Аналитическая геометрия

11.1 Задание варианта

Даны точки $A(-4;0), B(1;-3), C(4;-2)$. Построить прямую l, проходящую через A и параллельно BC в диапазоне $x \in [-1;3]$ с шагом $\triangle = 0.25$

11.2 Обоснование

Для уравнения $y = kx + b$ найдём коэффициенты. Так как y должен быть параллелен BC, то

$k_y = k_{BC}$

$k_{BC} = \frac{\triangle y}{\triangle x} = \frac{-3-(-2)}{1-4} = \frac{-1}{-3} = \frac{1}{3}$

Прямая y должна проходить через точку A, следовательно

$0 = \frac{1}{3} \cdot (-4) + b$

$b = \frac{4}{3}$

Таким образом получили уравнение: $y = \frac{1}{3}x + \frac{4}{3}$. Уравнение y описывает искомую прямую l.

11.3 Таблица значений

Таблица 14

Таблица значений функции y

x	y
-1	1
-0.75	$1\frac{1}{12}$
-0.5	$1\frac{1}{6}$
-0.25	$1\frac{1}{4}$
0	$1\frac{1}{3}$
0.25	$1\frac{5}{12}$
0.5	$1\frac{1}{2}$
0.75	$1\frac{7}{12}$
1	$1\frac{2}{3}$
1.25	$1\frac{3}{4}$
1.5	$1\frac{5}{6}$
1.75	$1\frac{11}{12}$
2	2
2.25	$2\frac{1}{12}$
2.5	$2\frac{1}{6}$
2.75	$2\frac{1}{4}$
3	$2\frac{1}{3}$

11.4 График

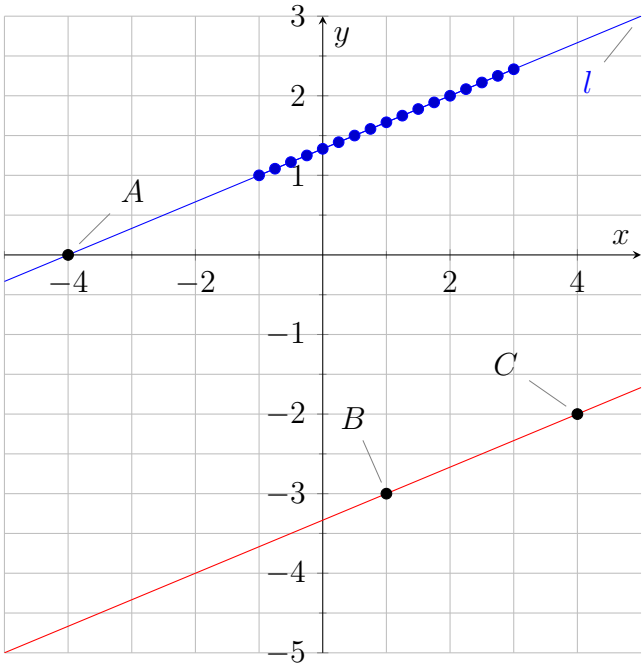

Рисунок 43: Прямая l, проходящая через A параллельная BC

12 Кривые второго порядка на плоскости

12.1 Задание варианта

Постройте гиперболы (диапазон и шаг выберите самостоятельно).

1)$16x^2 - 9y^2 = 144$

2)$\frac{y^2}{81} - \frac{x^2}{64} = 1$

12.2 Обоснование

1. $16x^2 - 9y^2 = 144$

   $y^2 = \frac{16x^2-144}{9}$

   $y = \frac{\pm\sqrt{16x^2-144}}{3}$

   $D(y) : 16x^2 - 144 \geq 0$

   $(4x - 12)(4x + 12) \geq 0$ - гипербола с ветвями направленными вверх, значит

   $D(y) \in (-\infty; -3] \cup [3; \infty)$.

2. $\frac{y^2}{81} - \frac{x^2}{64} = 1$

   $y^2 = 81(\frac{x^2+64}{64})$

   $y = \frac{9}{8}\sqrt{64 + x^2}$

   $D(y) \in (-\infty; \infty)$, так как $64 + x^2 > 0$ при любом x.

12.3 Таблица значений

1. Область перебора: $[-5; -3] \cup [3; 5]; \triangle = 0.5$.

Таблица 15

Таблица значений функции y

x	y
-5	$\pm 5\frac{1}{3}$
-4.5	$\pm 2\sqrt{5}$
-4	$\pm\frac{4\sqrt{7}}{3}$
-3.5	$\pm\frac{2\sqrt{13}}{3}$
-3	0
3	0
3.5	$\pm\frac{2\sqrt{13}}{3}$
4	$\pm\frac{4\sqrt{7}}{3}$
4.5	$\pm 2\sqrt{5}$
5	$\pm 5\frac{1}{3}$

2. Область перебора: $[-5; 5]; \triangle = 0.5$.

Таблица 16

Таблица значений функции y

x	y
-5	$\pm\frac{9\sqrt{89}}{8}$
-4.5	$\pm\frac{9\sqrt{337}}{16}$
-4	$\pm\frac{9\sqrt{5}}{2}$
-3.5	$\pm\frac{9\sqrt{305}}{16}$
-3	$\pm\frac{9\sqrt{73}}{8}$
-2.5	$\pm\frac{9\sqrt{281}}{16}$
-2	$\pm\frac{9\sqrt{17}}{4}$
-1.5	$\pm\frac{9\sqrt{265}}{16}$
-1	$\pm\frac{9\sqrt{65}}{8}$
-0.5	$\pm\frac{9\sqrt{257}}{16}$
0	± 9
0.5	$\pm\frac{9\sqrt{257}}{16}$
1	$\pm\frac{9\sqrt{65}}{8}$
1.5	$\pm\frac{9\sqrt{265}}{16}$
2	$\pm\frac{9\sqrt{17}}{4}$
2.5	$\pm\frac{9\sqrt{281}}{16}$
3	$\pm\frac{9\sqrt{73}}{8}$
3.5	$\pm\frac{9\sqrt{305}}{16}$
4	$\pm\frac{9\sqrt{5}}{2}$
4.5	$\pm\frac{9\sqrt{337}}{16}$
5	$\pm\frac{9\sqrt{89}}{8}$

12.4 График

1.

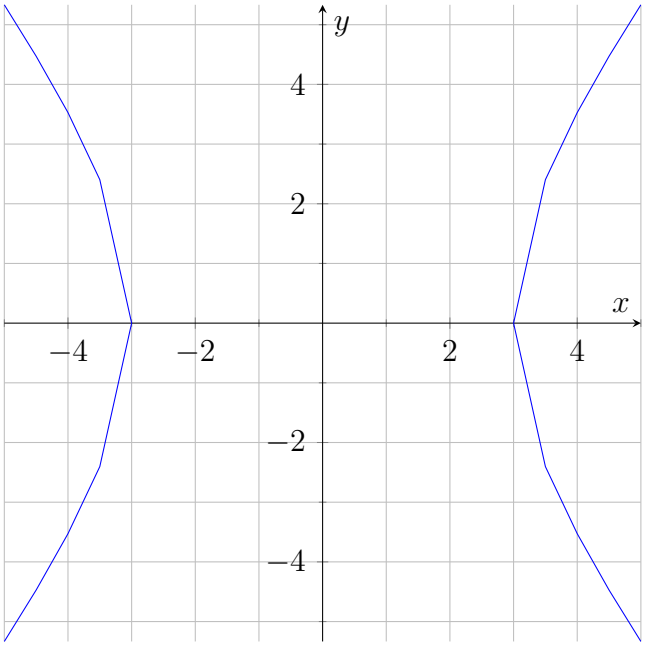

Рисунок 44: График гиперболы $16x^2 - 9y^2 = 144$

2.

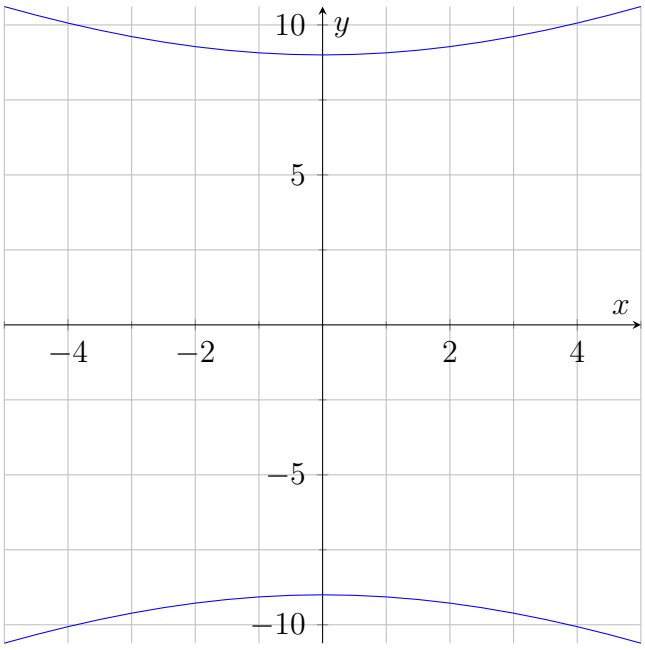

Рисунок 45: График гиперболы $\frac{y^2}{81} - \frac{x^2}{64} = 1$

13 Графическое решение систем уравнений

13.1 Задание варианта

$$\begin{cases} y^2 = x \\ y^2 = -x^2 + 9 \end{cases}$$ в диапазоне $0 \leq x \leq 4$ с шагом $\triangle = 0.2$

13.2 Обоснование

$y_1^2 = x$

$y_1 = \pm\sqrt{x}$

$y_2^2 = 9 - x^2$

$y_2 = \pm\sqrt{9 - x^2}$

13.3 Таблица значений

Таблица 17

Таблица значений функций y_1 и y_2

x	y_1	y_2
0	0	± 3
0.2	± 0.447213595	± 2.993325909
0.4	± 0.632455532	± 2.973213749
0.6	± 0.774596669	± 2.939387691
0.8	± 0.894427191	± 2.891366459
1	± 1	± 2.828427125
1.2	± 1.095445115	± 2.749545417
1.4	± 1.183215957	± 2.653299832
1.6	± 1.264911064	± 2.537715508
1.8	± 1.341640786	± 2.4
2	± 1.414213562	± 2.236067977
2.2	± 1.483239697	± 2.039607805
2.4	± 1.549193338	± 1.8
2.6	± 1.61245155	± 1.496662955
2.8	± 1.673320053	± 1.077032961
3	± 1.732050808	0
3.2	± 1.788854382	Нет решения
3.4	± 1.843908891	Нет решения
3.6	± 1.897366596	Нет решения
3.8	± 1.949358869	Нет решения
4	± 2	Нет решения

13.4 График

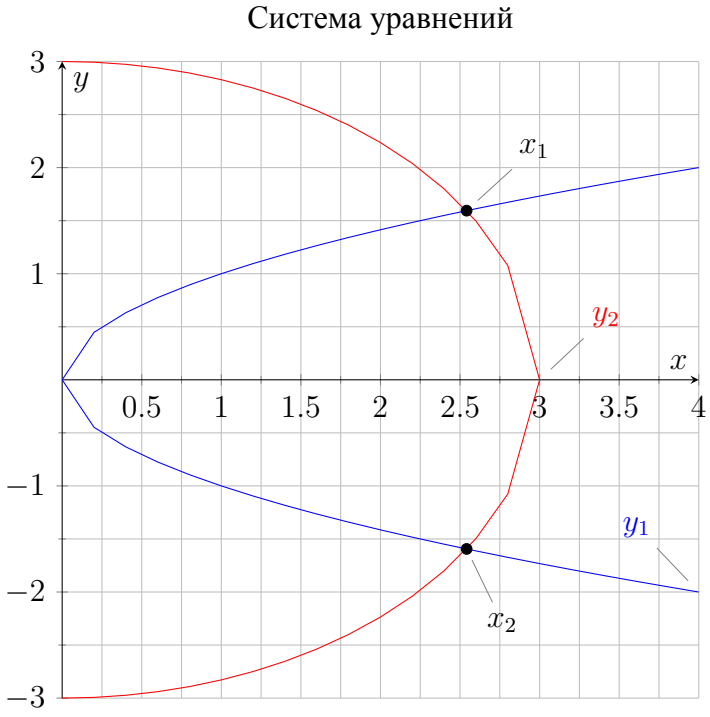

Рисунок 46: Графики функций y_1, y_2

Решение системы уравнений: $(2.541381265, 1.594171027)$,
$(2.541381265, -1.594171027)$.

14 Плоскость в трёхмерном пространстве

14.1 Задание варианта

Построить плоскость, параллельную плоскости Oxy и пересекающую ось Oz в точке $M(0, 0, 20)$; при $0 \leq x \leq 10$ с шагом $\triangle = 1$; $-5 \leq y \leq 5$ с шагом $\triangle = 1$.

14.2 Обоснование

Плоскость, параллельная Oxy имеет вид $Cz + D = 0$. Подставим в уравнение координаты точки M.

$C \cdot 20 + D = 0$

$D = -C \cdot 20$

$Cz - C \cdot 20 = 0 | : C$

$z = 20$ - искомое уравнение.

14.3 Таблица значений

Таблица 18

Таблица значений функции плоскости

x / y	0	1	2	3	4	5	6	7	8	9	10
-5	20	20	20	20	20	20	20	20	20	20	20
-4	20	20	20	20	20	20	20	20	20	20	20
-3	20	20	20	20	20	20	20	20	20	20	20
-2	20	20	20	20	20	20	20	20	20	20	20
-1	20	20	20	20	20	20	20	20	20	20	20
0	20	20	20	20	20	20	20	20	20	20	20
1	20	20	20	20	20	20	20	20	20	20	20
2	20	20	20	20	20	20	20	20	20	20	20
3	20	20	20	20	20	20	20	20	20	20	20
4	20	20	20	20	20	20	20	20	20	20	20
5	20	20	20	20	20	20	20	20	20	20	20

14.4 График

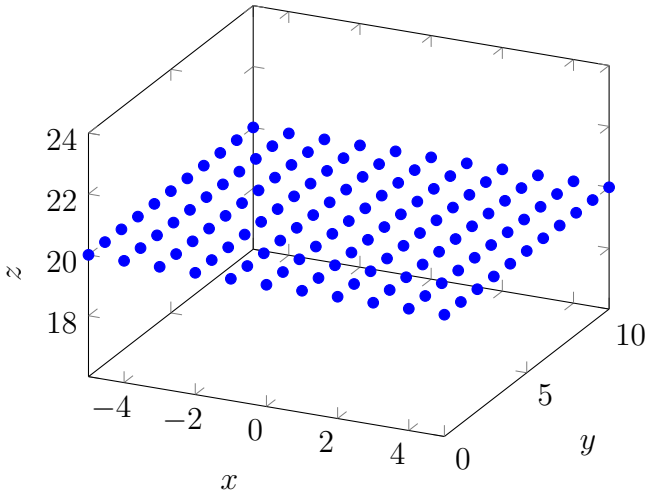

Рисунок 47: График плоскости, параллельной Oxy проходящей через $M(0, 0, 20)$

15 Поверхность второго порядка в трёхмерном пространстве

15.1 Задание варианта

Построить верхнюю часть гиперболоида, заданного уравнением $\frac{x^2}{25} + \frac{y^2}{16} - \frac{z^2}{9} = 1$, лежащую в диапазоне $-5 \leq x \leq 5$ с шагом $\triangle = 1$ и $-3 \leq y \leq 3$ с шагом $\triangle = 0.5$.

15.2 Обоснование

$\frac{x^2}{25} + \frac{y^2}{16} - \frac{z^2}{9} = 1$
$\frac{z^2}{9} = \frac{x^2}{25} + \frac{y^2}{16} - 1$
$z^2 = \frac{9x^2}{25} + \frac{9y^2}{16} - 9$
$z = \sqrt{\frac{9x^2}{25} + \frac{9y^2}{16} - 9}$, так как по условию нужно получить только верхнюю часть гиперболоида, будем брать результат с плюсом.

15.3 Таблица значений

Таблица 19

Таблица значений гиперболоида

x / y	-5	-4	-3	-2	-1	0	1	2	3	4	5
-3	2.88140591	1.8	∅	∅	∅	∅	∅	∅	∅	1.8	2.88140591
-2.5	2.70416346	1.5	∅	∅	∅	∅	∅	∅	∅	1.5	2.70416346
-2	2.55	1.2	∅	∅	∅	∅	∅	∅	∅	1.2	2.55
-1.5	2.42332416	0.9	∅	∅	∅	∅	∅	∅	∅	0.9	2.42332416
-1	2.3286262	0.6	∅	∅	∅	∅	∅	∅	∅	0.6	2.3286262
-0.5	2.26991189	0.3	∅	∅	∅	∅	∅	∅	∅	0.3	2.26991189
0	2.25	0	∅	∅	∅	∅	∅	∅	∅	0	2.25
0.5	2.26991189	0.3	∅	∅	∅	∅	∅	∅	∅	0.3	2.26991189
1	2.3286262	0.6	∅	∅	∅	∅	∅	∅	∅	0.6	2.3286262
1.5	2.42332416	0.9	∅	∅	∅	∅	∅	∅	∅	0.9	2.42332416
2	2.55	1.2	∅	∅	∅	∅	∅	∅	∅	1.2	2.55
2.5	2.70416346	1.5	∅	∅	∅	∅	∅	∅	∅	1.5	2.70416346
3	2.88140591	1.8	∅	∅	∅	∅	∅	∅	∅	1.8	2.88140591

15.4 График

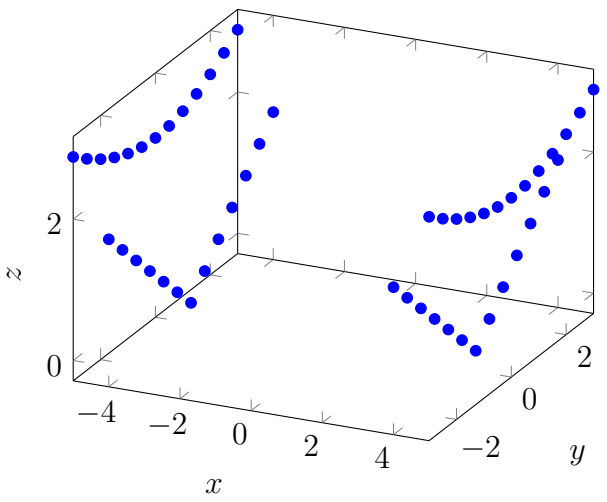

Рисунок 48: График поверхности $\frac{x^2}{25} + \frac{y^2}{16} - \frac{z^2}{9} = 1$