Замечания:

1. В описании математич. модели и постановке задачи нужны формульные выражения, которые используются в программном коде, в описании алгоритма поиска кратчайшего пути - только выбранного метода - Форда-Беллмана, далее требуется тестирование блок-схемы алгоритма поиска кратчайшего пути.
2. По блок-схемам - не используете блоки работы с файлом (они нового типа, см. в Moodle, курс ПиОА), местами потеряны связи между блоками.
3. Пришлите, пожалуйста, отлаженный код программы со вспомогательными функциями и входной тестовый файл координат точек для проверки корректности работы программы.
4. Нет в результатах NMEA-сообщений.

МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра корабельных систем управления

КУРСОВАЯ РАБОТА

по дисциплине «Программирование и основы алгоритмизации**»**

Тема: Алгоритмы поиска минимального пути (алгоритм Форда-Беллмана)

Студент гр. 0493 _____ Воронцов А.А.

Преподаватель _____ Лукомская О.Ю.

Санкт-Петербург

Цель курсовой работы – выработка у студентов умения и практических навыков разработки блок-схем алгоритмов решения задач, написания кодов в программной среде MATLAB, а также навыков по описанию, оформлению и представлению результатов проделанной работы.

Курсовая работа включает в себя следующие основные этапы:

1. Постановка задачи и формирование математической модели.
2. Выбор алгоритма поиска кратчайшего пути и его тестирование.
3. Разработка блок-схемы алгоритма решения задачи.
4. Написание и отладка программы на языке MATLAB.
5. Представление и анализ результатов работы программы.

1. ПОСТАНОВКА ЗАДАЧИ И ФОРМИРОВАНИЕ МАТЕМАТИЧЕСКОЙ МОДЕЛИ

Одной из стадий решения вычислительной задачи является построение математической модели. Под математической моделью понимается совокупность математических объектов и отношений, отображающих объекты и отношения, существующие в некоторой области реального мира [1]. Примерами математических моделей могут быть системы дифференциальных и алгебраических уравнений, ориентированные и неориентированные графы и пр.

Вид математической модели в значительной степени определяется постановкой задачи, но не только ею. Объекты и отношения реального мира, как правило, настолько сложны, что найти для них полное математическое описание не представляется возможным. Поэтому для одного и того же объекта (или системы объектов) можно построить множество различных математических моделей, описывающих поведение реального объекта с разной степенью точности. Поскольку использование математической модели связано с применением компьютерной техники и вычислительными затратами, необходимо искать баланс между точностью описания и простотой модели (чем проще модель, тем меньше времени занимает ее вычисление). Важную роль при построении математической модели играют допущения, которые можно принять для упрощения описываемых объектов и отношений, а также ограничения, которые необходимо учесть.

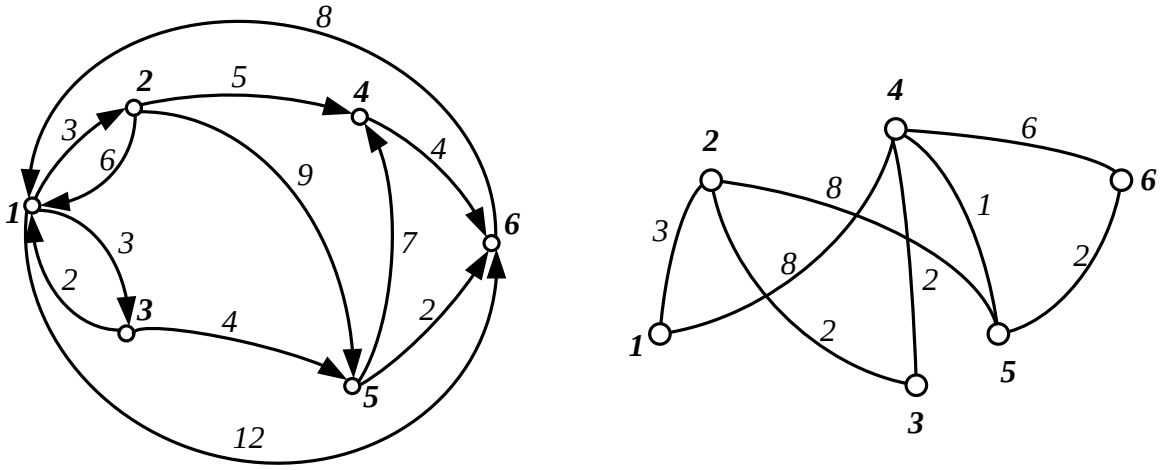
Рассмотрим процесс формирования математической модели на основе постановки задачи и ряда допущений. Задача состоит в том, чтобы перевести подвижный объект из одной точки пространства в другую за минимальное время. Это может быть автомобиль, движущийся по пересеченной местности,

надводный корабль или подводная лодка, двигающиеся на поверхности воды или на глубине, подводный робот, ползающий по морскому дну, космический аппарат в космосе. Полагаем, что подвижный объект оборудован движителем, обеспечивающим в любой момент движение в любом направлении. Движение объектов обычно можно описать при помощи второго закона Ньютона, оперируя понятиями массы и силы тяги, которая уравновешивается силой сопротивления среды. Однако в данной работе основная цель – выбор траектории движения, поэтому динамикой объекта также можно пренебречь. Считаем, что объект движется в однородной среде с постоянной скоростью, при этом его масса не меняется, что, строго говоря, для ряда реальных объектов также несправедливо, поскольку объекты (автомобили, корабли, ракеты) расходуют запас топлива. Поэтому введем допущение, что энергию для движения объект получает от аккумуляторной батареи.

Наконец, задача выбора траектории не будет иметь смысла без ограничений. С учетом уже принятых допущений ограничение очевидно: время работы аккумуляторной батареи конечно. Для реализуемости поставленной задачи в пространстве движения следует задать ряд пунктов подзарядки аккумулятора или дозаправки. Координаты расположения этих точек в пространстве можно считать постоянными. Таким образом, задача становится более конкретной: найти траекторию движения от одной точки пространства к другой за минимальное время, притом траектория должна проходить через заданные точки, а отрезки траектории ограничены с учетом времени работы аккумуляторной батареи.

После уточнения и конкретизации задачи можно выбрать вид математической модели для ее решения. В данном случае задачу удобнее всего решать с применением теории графов. Под графом понимается совокупность множества вершин и множества ребер, соединяющих вершины (рис. 1.1, *а*, *б*).

Граф называется *ориентированным*, если его ребра имеют направление (рис. 1.1, *а*) и *неориентированным*, если направление не задано. Граф называется *связным*, если последовательность ребер соединяет его две любые вершины, такая последовательность представляет собой путь из одной вершины в другую [2]. Если хотя бы для одной пары вершин не существует пути, граф называется *несвязным*.



а *б*

Рис. 1.1

Существует несколько способов математического описания графов. К примеру, графы можно описывать при помощи матрицы смежности – квадратной матрицы, размерность которой равна количеству вершин, а внедиагональные элементы равны единице, если ребро между соответствующими вершинами существует, или нулю, если не существует. Для графов, приведенных на рис. 1.1, матрицы смежности имеют вид:

$$\begin{bmatrix} 0 & 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$ – для рис. 1.1, *а*; $$\begin{bmatrix} 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 \end{bmatrix}$$ – для рис. 1.1, *б*.

Матрица будет симметричной для неориентированного графа и несимметричной для орграфа. Каждому ребру графа можно поставить в соответствие некоторое число, называемое весом ребра (рис. 1.1, *а, б*). Тогда исходная задача сводится к поиску такого пути в графе от одной вершины к другой, чтобы сумма длин ребер была минимальной.

Вес графа в общем случае зависит от характера оптимизационной задачи, т. е. от вида критерия. В нашем случае критерием служит время прохождения графа, так что веса должны быть пропорциональны времени прохождения ребер. При допущении, что скорость объекта постоянная, можно перейти от временных

характеристик пути к метрическим. Таким образом, в качестве веса ребер графа в данной задаче можно принять их длину.

Порядок расчета длины зависит от математического описания среды, которая до сих пор принималась несущественной. Возможны следующие варианты описания среды:

– плоскость;

– трехмерная поверхность;

– трехмерное пространство.

Плоскость можно использовать при моделировании движения автомобиля по дорогам, корабля по водной поверхности или подводного робота по ровному участку морского дна. Трехмерная поверхность – более универсальная модель среды (плоскость – частный случай поверхности), позволяющая учесть неровности дороги или дна, гористую местность и т. п. Трехмерное пространство позволяет описать движение в космосе, в воздухе или на глубине. Примем в качестве математического описания среды поверхность, описываемую функцией

$$z = f(x, y),$$

где x, y – координаты точки пространства в декартовой плоскости; z – высота (глубина), определяемая координатами x, y.

Таким образом, математическая модель для поставленной задачи включает:

– описание неориентированного графа;

– описание среды движения;

– порядок определения весов (длин) ребер графа;

– описание графа с учетом длин.

Данная задача даже с учетом введенных упрощений может найти применение для прикладных задач разного характера. Вычисление длин ребер может быть сведено к расчету метрических расстояний только в том случае, когда среда задана однородной плоскостью без ограничений. Если, например, представить, что вершины графа соответствуют населенным пунктам на географической карте, то расчет длин сведется к вычислению криволинейных интегралов, соответствующих соединяющим пункты дорогам. Однако общая постановка задачи и методы решения от этого не изменятся.

Средством математического описания графов с взвешенными ребрами являются матрицы весов (длин), похожие по структуре на матрицы смежности, но имеющие веса в качестве элементов. Элементы матрицы длин вычис-

ляются следующим образом:

$$L_{ij} = \begin{cases} 0, i = j, \\ l_{ij}, a_{ij} = 1, \\ \infty, a_{ij} = 0, \end{cases} \quad (1.1)$$

где a_{ij} – элементы матрицы смежности. Для графов на рис. 1.1 матрицы имеют вид

$$L_1 = \begin{bmatrix} 0 & 3 & 3 & \infty & \infty & 12 \\ 6 & 0 & \infty & 5 & 9 & \infty \\ 2 & \infty & 0 & \infty & 4 & \infty \\ \infty & \infty & \infty & 0 & \infty & 4 \\ \infty & \infty & \infty & 7 & 0 & 2 \\ 8 & \infty & \infty & \infty & \infty & 0 \end{bmatrix}; \qquad L_2 = \begin{bmatrix} 0 & 3 & \infty & 8 & \infty & \infty \\ 3 & 0 & 2 & \infty & 8 & \infty \\ \infty & 2 & 0 & 2 & \infty & \infty \\ 8 & \infty & 2 & 0 & 1 & 6 \\ \infty & 8 & \infty & 1 & 0 & 2 \\ \infty & \infty & \infty & 6 & 2 & 0 \end{bmatrix},$$

где L_1 и L_2 – матрицы длин для графов соответственно на рис. 1.1, *а* и *б*.

Уточнение математической модели. Простые математические модели описывают идеализированное поведение процессов и систем. Для учета реальных факторов, влияющих на поведение объекта, модель уточняют, добавляя различные физические эффекты.

Допустим, предложенную ранее модель для решения траекторной задачи нужно использовать для оценки общего времени пути из начальной точки в конечную. Расчет не представляет сложности, однако модель не учитывает времени, затраченного на подзарядку аккумуляторов (или дозаправку) подвижного объекта. Если считать пункты подзарядки идентичными по времени подзарядки, то учет этого времени сделает модель более реалистичной без существенных усложнений.

В данной задаче полагаем, что время подзарядки аккумуляторов одинаковое для всех пунктов и не зависит от текущего состояния аккумулятора. Можно далее уточнять модель: ввести подзарядку с учетом разрядки или задавать разное время подзарядки на разных пунктах. Правда, в последнем случае придется отказаться от метрических весов графа и рассчитывать веса ребер пропорционально затраченному времени на путь и подзарядку.

Итак, для формализации задачи получения математической модели и численного решения требуются следующие исходные данные:

– скорость подвижного объекта;

– время работы аккумулятора;

– время подзарядки аккумулятора;

– координаты начальной и конечной точек объекта;

– координаты промежуточных точек – пунктов подзарядки.

На основе этих данных можно построить неориентированный граф и сформировать матрицу длин для его описания в вычислительной машине. Теперь задача поиска минимального пути на графе справедлива.

2. АЛГОРИТМЫ ПОИСКА ОПТИМАЛЬНОЙ ТРАЕКТОРИИ

Задача поиска оптимальной траектории, соответствующей минимальной длине на взвешенном графе широко известна. Наиболее популярны следующие алгоритмы поиска:

– алгоритм Форда–Беллмана.

– алгоритм Дейкстры;

– алгоритм Флойда;

Существуют и другие алгоритмы, но они, как правило, представляют собой модифицированные варианты алгоритмов, приведенных ранее.

2.1. Алгоритм Форда–Беллмана

Алгоритм Форда–Беллмана основан на теореме, согласно которой любой участок минимального пути, соединяющий некоторые вершины графа, представляет минимальный путь между этими вершинами. Алгоритм предполагает вычисление всех минимальных путей из начальной вершины.

Исходные данные: матрица длин L, начальная вершина имеет индекс 1.

Требуется выполнить следующие действия:

1. Определить для каждой вершины набор величин $\lambda_i^{(k)}$, означающих длину пути от 1-й до i-й вершины, состоящего не более чем из k ребер. Набор формируется по правилу

$$\lambda_i^{(k+1)} = \min_j \left\{ \lambda_i^{(k)} + l_{ij} \right\}, k = 0 \ldots n-1, \quad \lambda_i^{(0)} = \begin{cases} 0, i = 1, \\ \infty, i \neq 1, \end{cases} \tag{1.2}$$

где n – число вершин; k - число ребер.

Величины $\lambda_i^{(k)}$ можно представить в виде квадратной матрицы $\mathbf{\Lambda}$. Если значение $\lambda_i^{(n-1)}$ отлично от ∞, значит, оно равно длине минимального пути в вершину i, в противном случае данная вершина недостижима (граф несвязный).

2. Определить последовательность ребер, приводящих из 1-й в i-ю вершину за путь $\lambda_i^{(k)}$. Для этого нужно определить номера $i_1, i_2, \ldots, i_k$, для которых выполняются соотношения:

$$\begin{aligned}
&\lambda_i^{(k)} = \lambda_{i_1}^{(k-1)} + l_{i_1, i}\ ;\\
&\lambda_{i_1}^{(k-1)} = \lambda_{i_2}^{(k-2)} + l_{i_2, i_1};\\
&\ldots\\
&\lambda_{i_{k-1}}^{(1)} = \lambda_{i_k}^{(0)} + l_{i_k, i_{k-1}}.
\end{aligned} \tag{1.3}$$

Для графов, приведенных на рис. 1.1, по алгоритму Форда–Беллмана матрицы минимальных путей Λ_1 и Λ_2 имеют вид

$$\Lambda_1 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ \infty & 3 & 3 & 3 & 3 & 3 \\ \infty & 3 & 3 & 3 & 3 & 3 \\ \infty & \infty & 8 & 8 & 8 & 8 \\ \infty & \infty & 7 & 7 & 7 & 7 \\ \infty & 12 & 12 & 9 & 9 & 9 \end{bmatrix}; \qquad \Lambda_2 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ \infty & 3 & 3 & 3 & 3 & 3 \\ \infty & \infty & 5 & 5 & 5 & 5 \\ \infty & 8 & 8 & 7 & 7 & 7 \\ \infty & \infty & 9 & 9 & 8 & 8 \\ \infty & \infty & 14 & 11 & 11 & 10 \end{bmatrix}.$$

Для графа, приведенного на рис. 1.1, *а*, минимальный путь до любой вершины достигается максимум за три ребра. По формулам (1.3) легко восстановить, что путь до 6-й вершины с конца проходит через 5-ю $\left(\lambda_6^{(3)} = \lambda_5^{(2)} + l_{56}\right)$, а далее через 3-ю $\left(\lambda_5^{(2)} = \lambda_3^{(1)} + l_{35}\right)$ вершины. Таким образом, искомый минимальный путь имеет вид 1–3–5–6 . Длина минимального пути равна 9. Минимальный путь для графа на рис. 1.1, *б* содержит 5 ребер. Аналогично (1.3) можно восстановить в обратном порядке искомую последовательность вершин 1–2–3–4–5–6, длина минимального пути равна 10.

2.2. Алгоритм Дейкстры

Этот алгоритм, предложенный в 1959 г. Дейкстрой, считается одним из наиболее эффективных алгоритмов решения задачи нахождения в графе минимальных путей от начальной вершины до всех остальных при положительных длинах дуг.

main.m

```
clear;clc;
matrica=matrica();
[matrix_nul,matrix_inf]=vozmoshno(matrica);
[pyt, sps]=algoritm(matrix_inf);
PP=NMEA(matrica,matrix_inf,sps);
File=fopen('Results.txt','w');
clc;
menu=input('Для работы с меню программы введите цифру:\n1 -
начальные условия\n2 - Матрица длин ребер\n3 - Матрица
возможных перемещений\n4 - Последовательность заправок\n5 -
NMEA\n6 - Матрица минимальных перемещений\n7 - График\n8 -
Конец\n'); %управление меню
while menu
clc;
    switch menu
        case 1
           fprintf(File,'%3s\n','точки:');

            sohranim(matrica,File);
        case 2

fprintf(File,'%3s\n','матрица длин ребёр');

            sohranim(matrix_inf,File);
        case 3
                          fprintf(File,'%3s\n','матрица
возможных перемещений:');

            sohranim(matrix_nul,File);
        case 4

fprintf(File,'%3s\n','последовательность заправок:');

            sohranim(sps,File);
        case 5

fprintf(File,'%3s\n','NMEA:');

            for i=1:(length(sps)-1)

fprintf(File,'$UTHDG,%0.1f,%1.1f,%.2f,%c,%.3f,%c\n',PP(i,1),PP
(i,2),PP(i,3),PP(i,4),PP(i,5),PP(i,6));%1целое число 2 точка с
запятой с-отдельный символ
```

```
fprintf('$UTHDG,%0.1f,%1.1f,%.2f,%c,%.3f,%c\n',PP(i,1),PP(i,2)
,PP(i,3),PP(i,4),PP(i,5),PP(i,6));
            end

        case 6
                            fprintf(File,'%3s\n','Матрица
минимальных перемещений:');

            sohranim(pyt,File);
        case 7
            ploskost( matrica,matrix_nul,sps );
            case 8

            return;
    end
    menu=input('Для работы с меню программы введите цифру:\n1
- Начальные условия\n2 - Матрица длин ребер\n3 - Матрица
возможных перемещений\n4 - Последовательность станций\n5 -
NMEA\n6 - Матрица минимальных перемещений\n7 - График\n8 -
Конец\n'); %управление меню


end
fclose('all');
```

matrica.m

```
function [ matrica ] = matrica()
    izfaila=load('CORDS.txt');
    disp('Введите координаты в виде [x,y]');
    start=input('Начальная точка: ');
    last=input('Конечная точка:  ');
    matrica(1, 1:2)=start;
    matrica(2:11,1:2)=izfaila;
    matrica(12,1:2)=last;
end
```

vozmoshno.m

```
function [matrix_nul,matrix_inf]=vozmoshno( matrica )
```

```
    matrix_nul(1:12,1:12)=0;
    matrix_inf(1:12,1:12)=inf;
    for i=1:12
        for j=1:12

proverka=(pifagor(matrica(i,1),matrica(i,2),matrica(j,1),matri
ca(j,2)));
            if proverka<=22*4.5;
                if i~=j
                    matrix_nul(i,j)=1;
                    matrix_inf(i,j)=round(proverka);
                else
                    matrix_nul(i,j)=0;
                    matrix_inf(i,j)=round(proverka);
                end
            else

            end
        end
    end
end
```

sohranim.m

```
function [] = sohranim(matrica,File)
siz=size(matrica);
    for i=1:siz(1,1)
        for j=1:siz(1,2)
            fprintf(File,'%4d',matrica(i,j));
        end
        fprintf(File ,'\r\n');
    end
    fprintf(File ,'\r\n');
    disp(matrica);
end
```

algoritm.m

```
function [pyt,sps]=algoritm( matrix_inf )
    pyt=zeros(12);
    pyt(2:12,:) = inf;      for i = 2 : 12
        for j = 2 : 12
            for k = 1 : 12
                if matrix_inf(k, j)~=0                      if
pyt(j, i-1) > pyt(k, i - 1) + matrix_inf(j, k) && pyt(j,i) >
pyt(k,i-1)+matrix_inf(j,k)
                        pyt(j, i) = pyt(k, i - 1) +
matrix_inf(j, k);
                    end
                    if pyt(j,i)==inf
                        pyt(j,i)=pyt(j,i-1);
                    end
                end
            end
        end
    end
l = pyt(12,12);
    sps = 12;
    w = 12;
    while(l)
        for i = 1:12
            if (matrix_inf(i, w)~=0) && (l - matrix_inf(i, w)
== pyt(i, 12))
                l = l - matrix_inf(i, w);
                w = i;
                sps = [w, sps];

            end
        end
    end

end
```

NMEA.m

```
function [PP]=NMEA( matrica,matrix_inf,sps )

    h=input('Время работы аккумулятора: ');
    m=input('Время зарядки аккумулятора: ');


  for i=2:(length(sps))
    DD(i-1)=matrix_inf(sps(i),sps(i-1));
    XX(i-1)=fix(DD(i-1)/22);
    YY(i-1)=(DD(i-1)/22-XX(i-1))*60;
    ZZ(i-1)=azimut(matrica(sps(i-1),1), matrica(sps(i-1),2),
matrica(sps(i),1), matrica(sps(i),2));
    if i==length(sps)
      S2(i-1)='E';
      S1(i-1)='T';
    else
      S2(i-1)='N';
      S1(i-1)='T';
     end
      if (i>2)
          if (ZZ(i-1)==ZZ(i-2))
          S1(i-1)='N';
          end
      end
  end

    PP=zeros(length(sps)-1);
    PP(:,1)=XX;
    PP(:,2)=YY;
    PP(:,3)=DD;
    PP(:,4)=S1;
    PP(:,5)=ZZ;
    PP(:,6)=S2;
    PP(1,1)=h;
    PP(1,2)=m;

end
```

pifagor.m

```
function [proverka] = pifagor(x_1, y_1, x_2, y_2)
proverka=30*sqrt((x_2-x_1)^2+(y_2-y_1)^2);
end
```

azimut.m

```
function [azi] = azimut(x_1, y_1, x_2, y_2)
azi=atan((y_2-y_1)/(x_2-x_1))*180/pi;
if and(x_2<x_1,y_2>y_1)||and(x_2<x_1,y_2<y_1)
azi=180+azi;
end
if azi<0
azi=azi+360;
end
```

ploskost.m

```
function [] = ploskost( matrica,matrix_nul,sps )
hold on
grid on
for i=1:1:12
    for j=i+1:1:12
        if matrix_nul(i,j)==1
            point(1,1)=matrica(i,1);
            point(1,2)=matrica(j,1);
            point(2,1)=matrica(i,2);
            point(2,2)=matrica(j,2);
            plot(point(1,:), point(2,:), 'r')
            plot(point(1,:), point(2,:), 'bo')
        end
    end
    text(matrica(i,1),matrica(i,2), num2str(i));
end
    for i=1:1:length(sps)-1
        point(1,1)=matrica(sps(i),1);
        point(1,2)=matrica(sps(i+1),1);
        point(2,1)=matrica(sps(i),2);
        point(2,2)=matrica(sps(i+1),2);
        plot(point(1,:), point(2,:), 'k')
    end
```

```
xlabel('Масштаб 1:30 км');
end
```

CORDS.txt

8 8;

6 6;

4 4;

2 2;

2 3;

0 0;

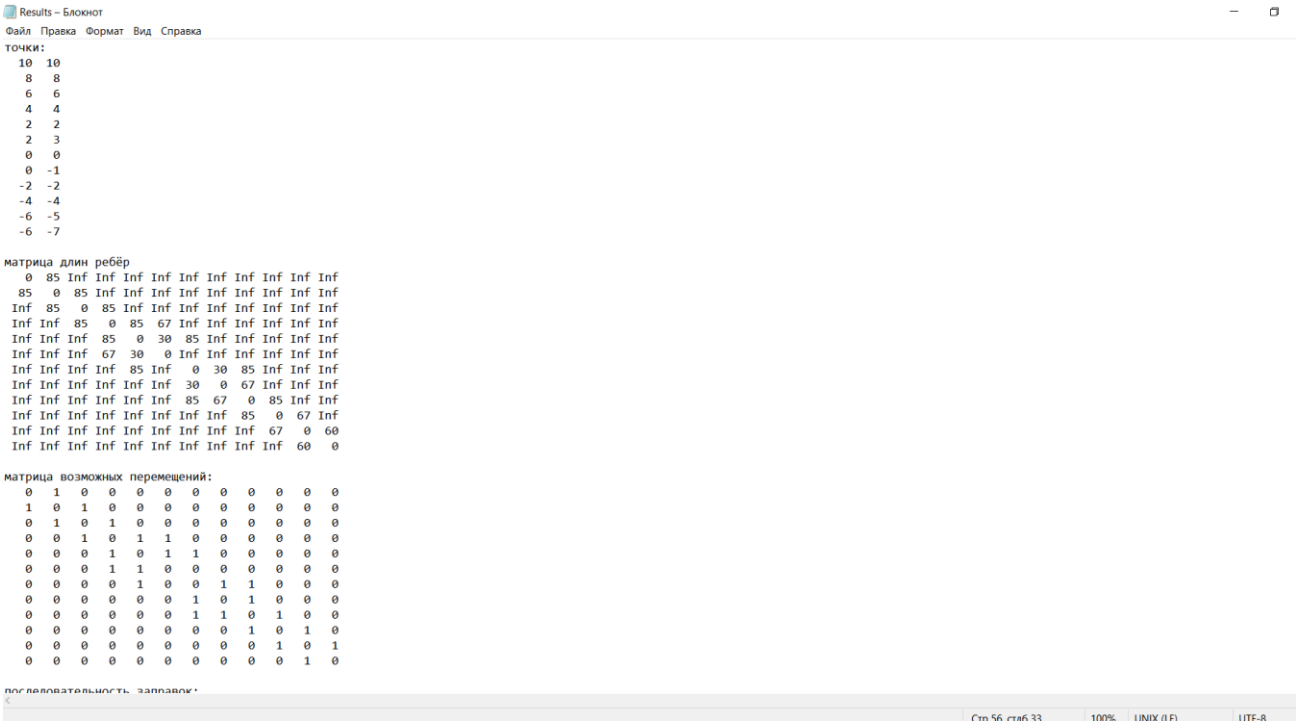
0 -1;

-2 -2;

-4 -4;

-6 -5

Результаты работы программы:



Results – Блокнот

Файл Правка Формат Вид Справка

```
точки:
  10  10
   8   8
   6   6
   4   4
   2   2
   2   3
   0   0
   0  -1
  -2  -2
  -4  -4
  -6  -5
  -6  -7

матрица длин ребёр
   0  85 Inf Inf Inf Inf Inf Inf Inf Inf Inf
  85   0  85 Inf Inf Inf Inf Inf Inf Inf Inf
 Inf  85   0  85 Inf Inf Inf Inf Inf Inf Inf
 Inf Inf  85   0  85  67 Inf Inf Inf Inf Inf
 Inf Inf Inf  85   0  30  85 Inf Inf Inf Inf
 Inf Inf Inf  67  30   0 Inf Inf Inf Inf Inf
 Inf Inf Inf Inf  85 Inf   0  30  85 Inf Inf
 Inf Inf Inf Inf Inf Inf  30   0  67 Inf Inf
 Inf Inf Inf Inf Inf Inf  85  67   0  85 Inf
 Inf Inf Inf Inf Inf Inf Inf  85   0  67 Inf
 Inf Inf Inf Inf Inf Inf Inf Inf  67   0  60
 Inf Inf Inf Inf Inf Inf Inf Inf Inf  60   0

матрица возможных перемещений:
   0   1   0   0   0   0   0   0   0   0   0
   1   0   1   0   0   0   0   0   0   0   0
   0   1   0   1   0   0   0   0   0   0   0
   0   0   1   0   1   1   0   0   0   0   0
   0   0   0   1   0   1   1   0   0   0   0
   0   0   0   1   1   0   0   0   0   0   0
   0   0   0   0   1   0   0   1   1   0   0
   0   0   0   0   0   0   1   0   1   0   0
   0   0   0   0   0   0   1   1   0   1   0
   0   0   0   0   0   0   0   0   1   0   1
   0   0   0   0   0   0   0   0   0   1   0

последовательность заправок:
```

Стр 56, стлб 33 | 100% | UNIX (LF) | UTF-8

```
последовательность заправок:
   1   2   3   4   5   7   9  10  11  12

NMEA:
$UTHDG,4.5,20.0,85.00,T,225.000,N
$UTHDG,3.0,51.8,85.00,N,225.000,N
$UTHDG,3.0,51.8,85.00,N,225.000,N
$UTHDG,3.0,51.8,85.00,N,225.000,N
$UTHDG,3.0,51.8,85.00,N,225.000,N
$UTHDG,3.0,51.8,85.00,N,225.000,N
$UTHDG,3.0,51.8,85.00,N,225.000,N
$UTHDG,3.0,2.7,67.00,T,206.565,N
$UTHDG,2.0,43.6,60.00,T,270.000,E
Матрица минимальных перемещений:
   0   0   0   0   0   0   0   0   0   0   0   0
 Inf  85  85  85  85  85  85  85  85  85  85  85
 Inf Inf 170 170 170 170 170 170 170 170 170 170
 Inf Inf Inf 255 255 255 255 255 255 255 255 255
 Inf Inf Inf Inf 340 340 340 340 340 340 340 340
 Inf Inf Inf Inf 322 322 322 322 322 322 322 322
 Inf Inf Inf Inf Inf 425 425 425 425 425 425 425
 Inf Inf Inf Inf Inf Inf 455 455 455 455 455 455
 Inf Inf Inf Inf Inf Inf 510 510 510 510 510 510
 Inf Inf Inf Inf Inf Inf Inf 595 595 595 595 595
 Inf Inf Inf Inf Inf Inf Inf Inf 662 662 662 662
 Inf Inf Inf Inf Inf Inf Inf Inf Inf 722 722 722
```

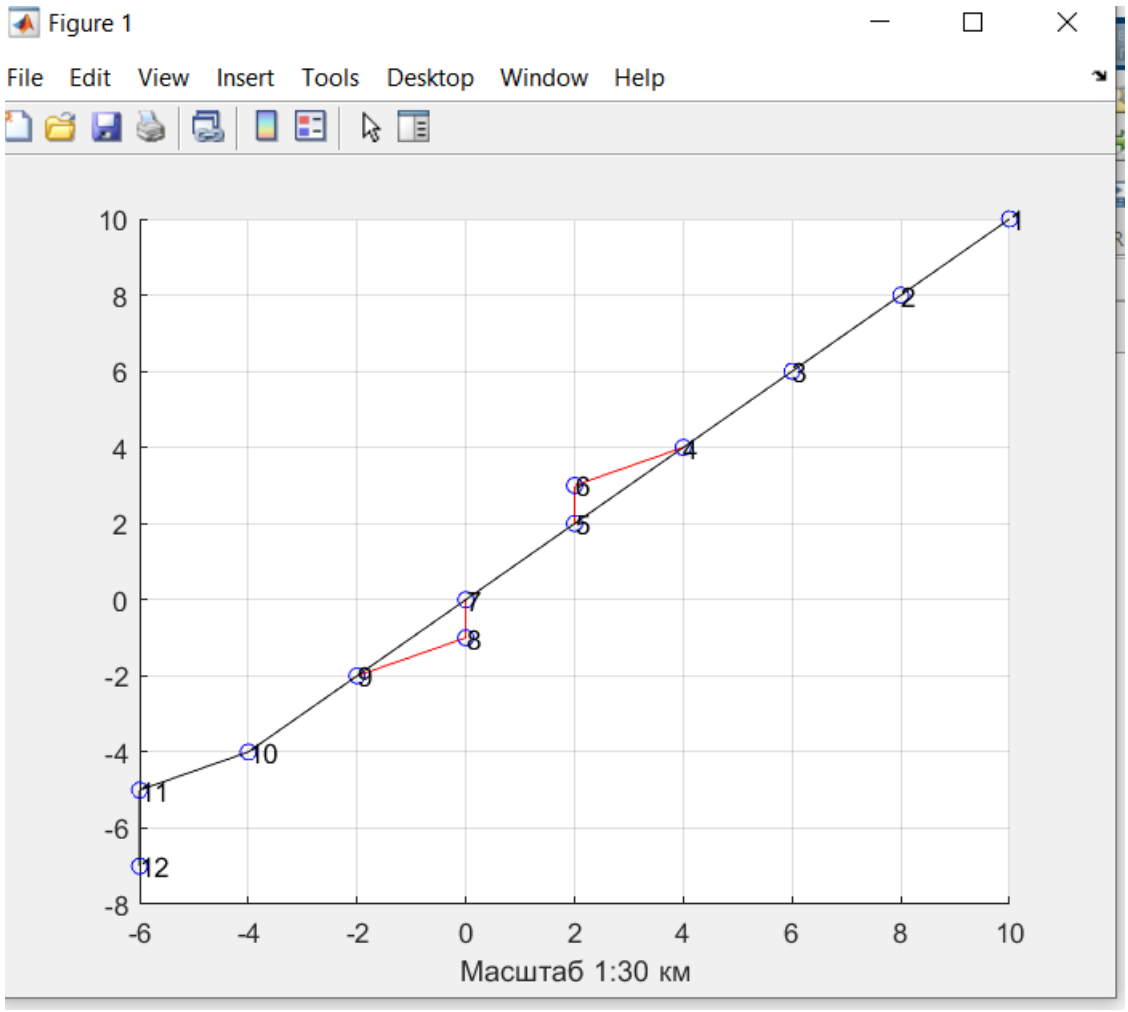
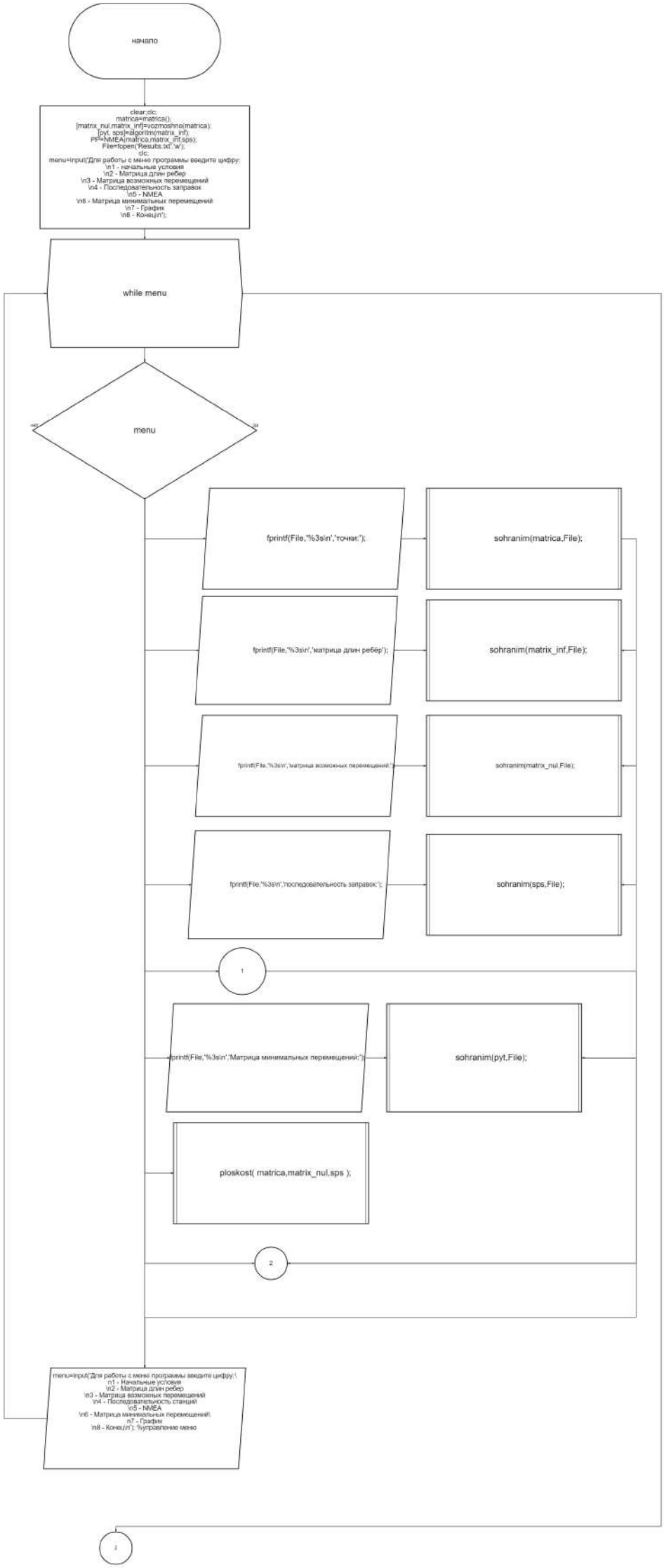
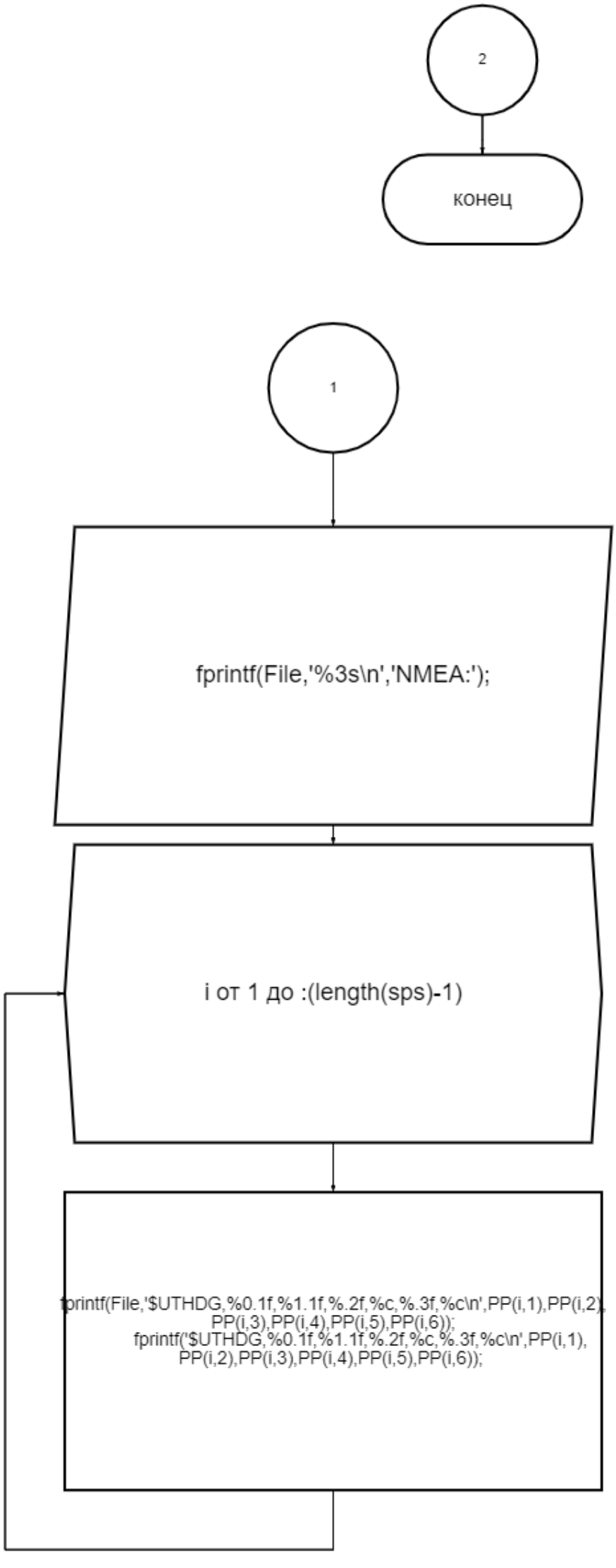
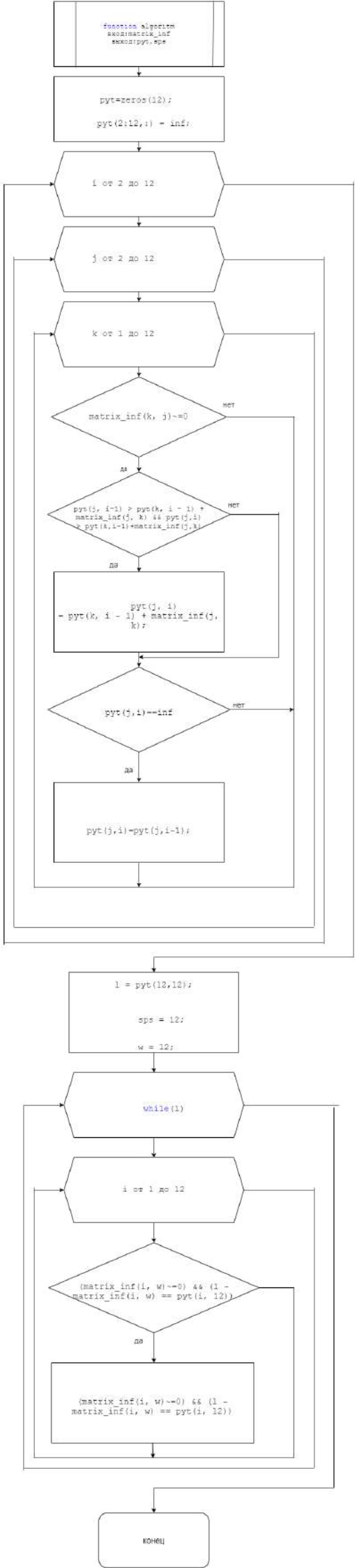


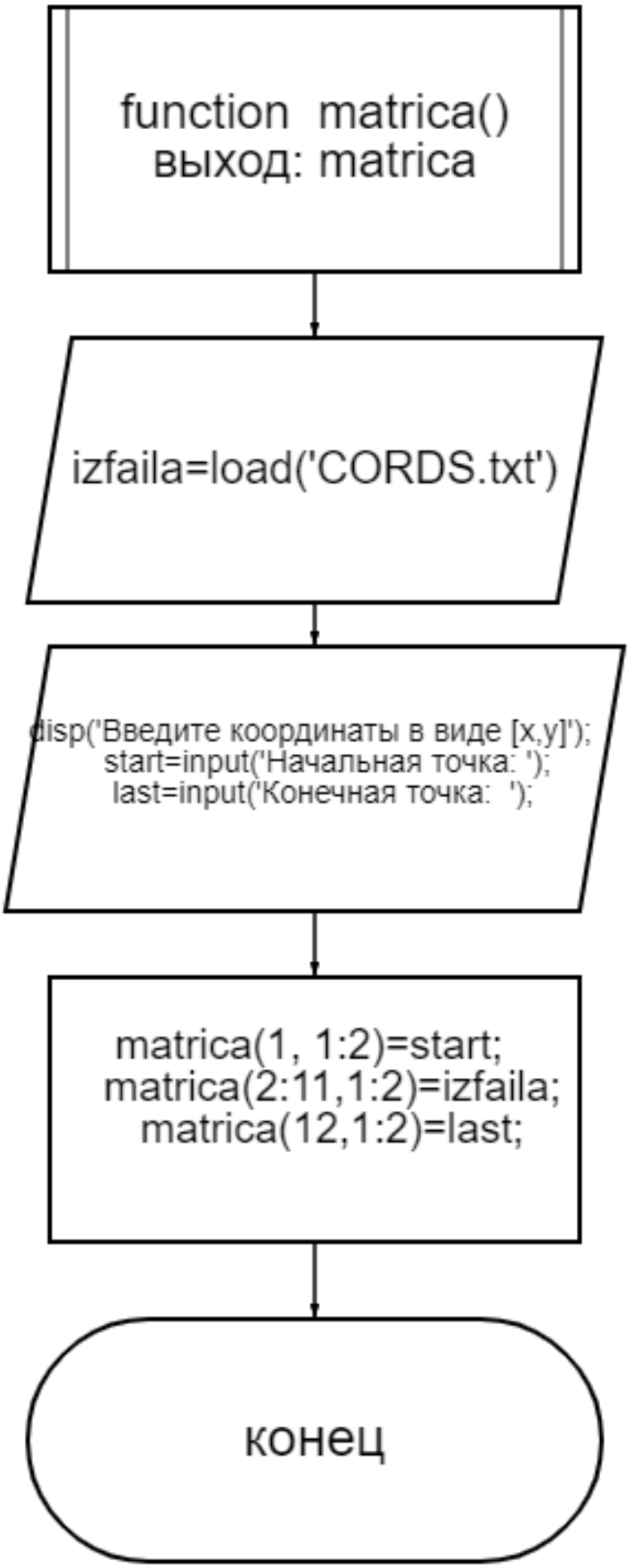
Figure 1
File Edit View Insert Tools Desktop Window Help
10
8
6
4
2
0
-2
-4
-6
-8
-6 -4 -2 0 2 4 6 8 10
1
2
3
4
5
6
7
8
9
10
11
12
Масштаб 1:30 км

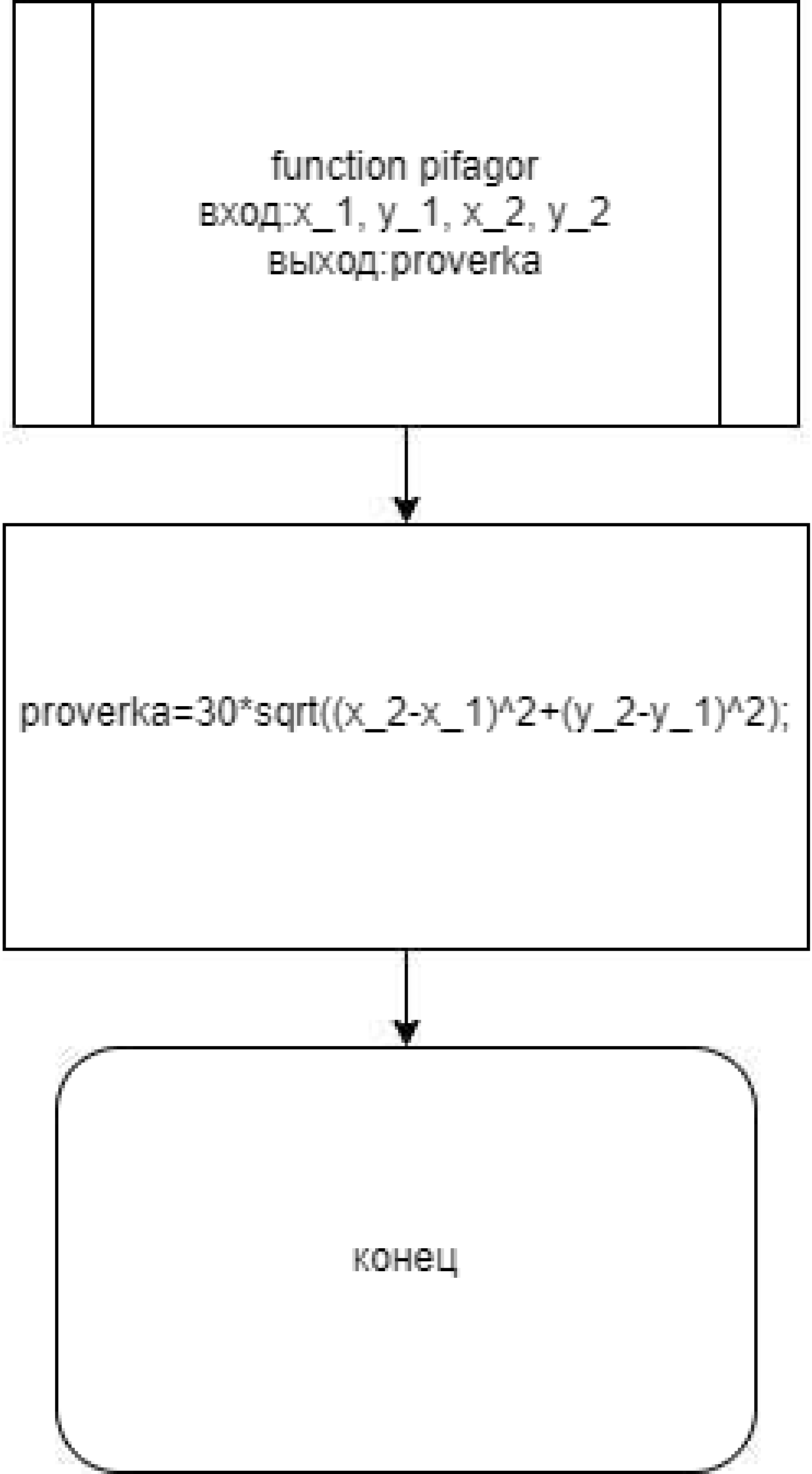
Блок-схемы:

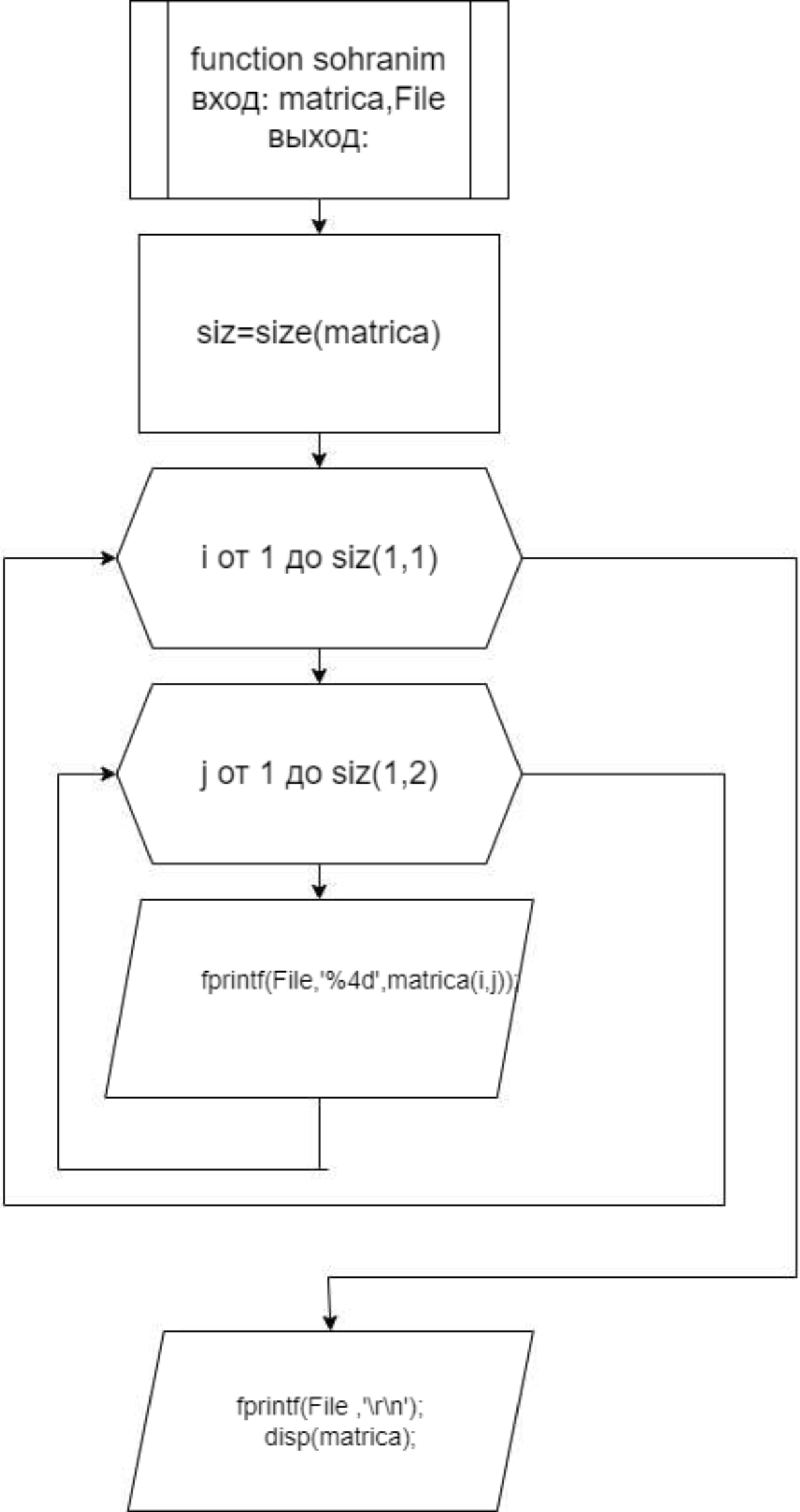

начало
clear;clc;
matrica=matrica();
[matrix_nul,matrix_inf]=vozmoshno(matrica);
[pyt, sps]=algoritm(matrix_inf);
PP=NMEA(matrica,matrix_inf,sps);
File=fopen('Results.txt','w');
clc;
menu=input('Для работы с меню программы введите цифру:
\n1 - начальные условия
\n2 - Матрица длин ребер
\n3 - Матрица возможных перемещений
\n4 - Последовательность заправок
\n5 - NMEA
\n6 - Матрица минимальных перемещений
\n7 - График
\n8 - Конец\n');
while menu
menu
fprintf(File,'%3s\n','точки:');
sohranim(matrica,File);
fprintf(File,'%3s\n','матрица длин ребёр');
sohranim(matrix_inf,File);
fprintf(File,'%3s\n','матрица возможных перемещений:');
sohranim(matrix_nul,File);
fprintf(File,'%3s\n','последовательность заправок:');
sohranim(sps,File);
1
fprintf(File,'%3s\n','Матрица минимальных перемещений:');
sohranim(pyt,File);
ploskost(matrica,matrix_nul,sps);
2
menu=input('Для работы с меню программы введите цифру:\
n1 - Начальные условия
\n2 - Матрица длин ребер
\n3 - Матрица возможных перемещений
\n4 - Последовательность станций
\n5 - NMEA
\n6 - Матрица минимальных перемещений\
n7 - График
\n8 - Конец\n'); %управлению меню
3

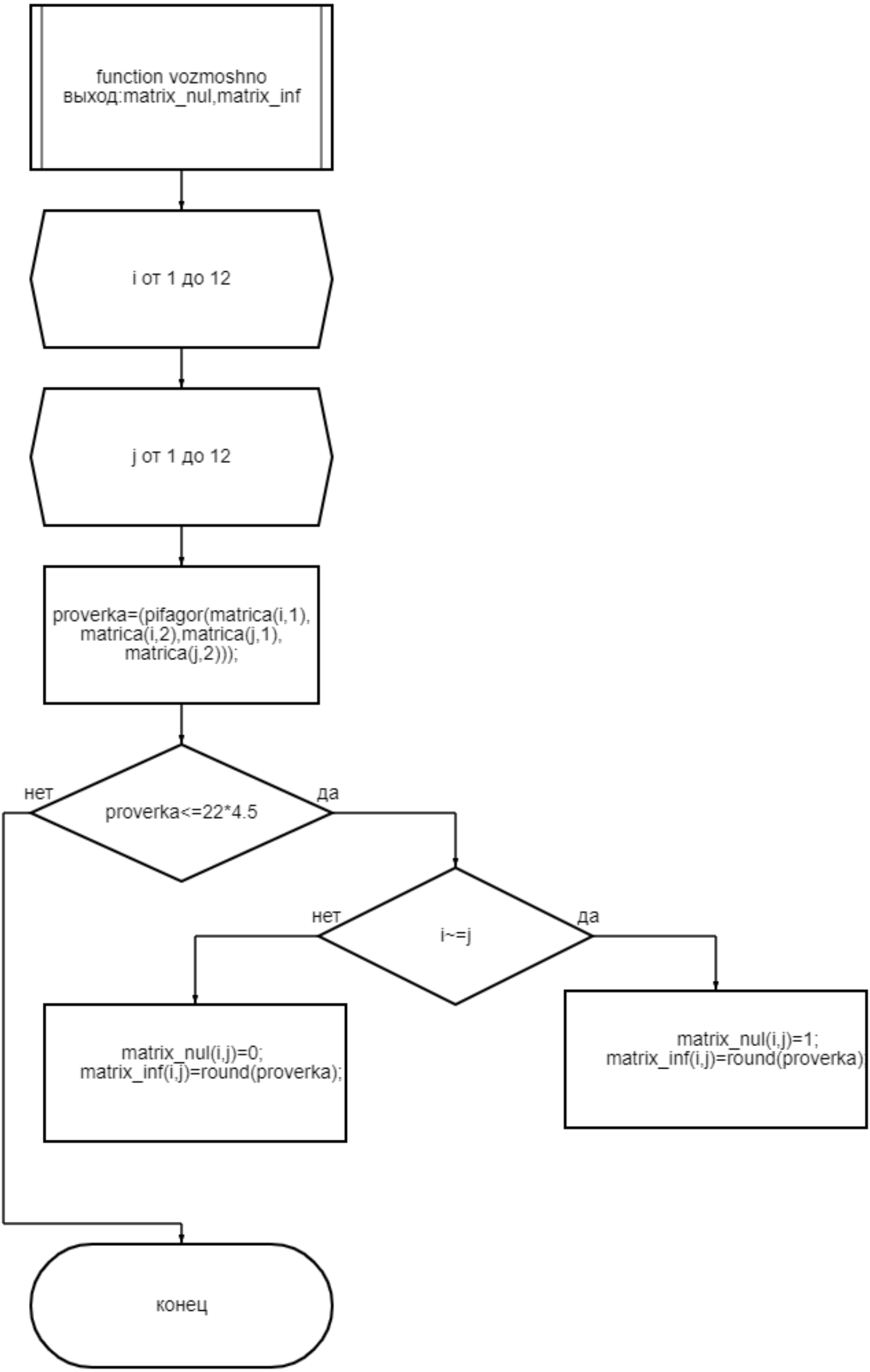

2
конец
1
fprintf(File,'%3s\n','NMEA:');
i от 1 до :(length(sps)-1)
fprintf(File,'$UTHDG,%0.1f,%1.1f,%.2f,%c,%.3f,%c\n',PP(i,1),PP(i,2),
PP(i,3),PP(i,4),PP(i,5),PP(i,6));
fprintf('$UTHDG,%0.1f,%1.1f,%.2f,%c,%.3f,%c\n',PP(i,1),
PP(i,2),PP(i,3),PP(i,4),PP(i,5),PP(i,6));

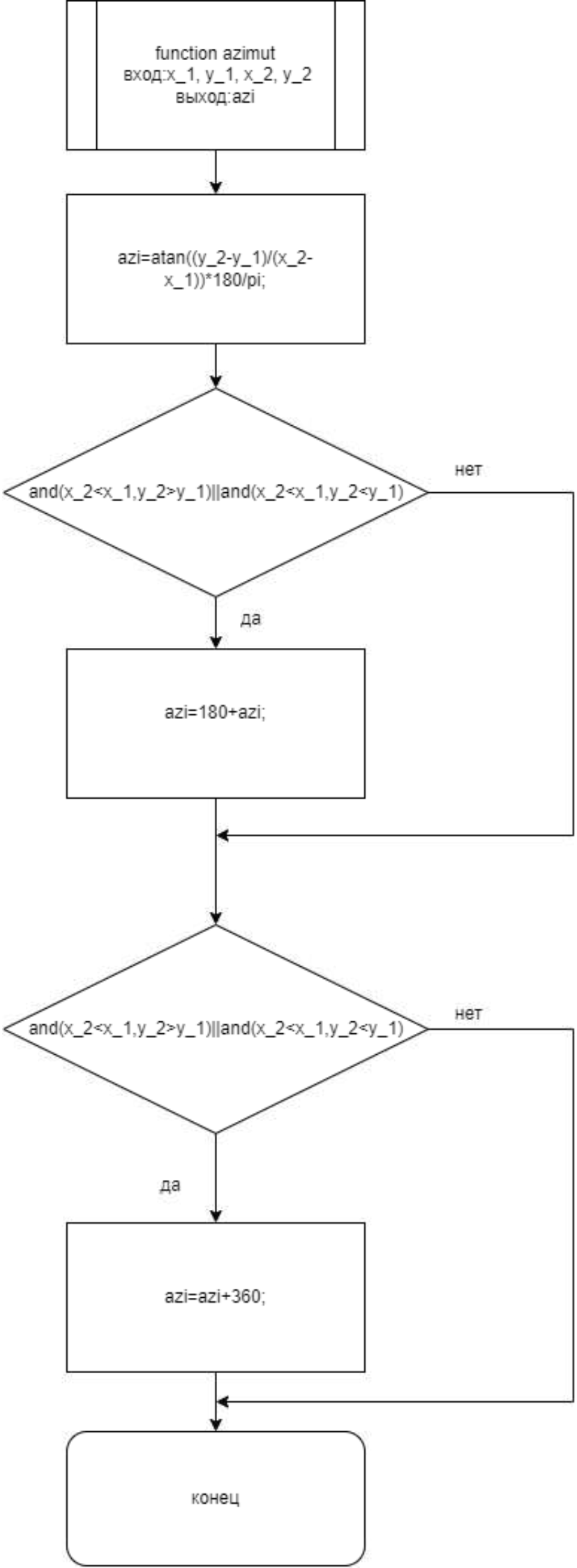

function algoritm
вход:matrix_inf
выход:pyt,sps
pyt=zeros(12);
pyt(2:12,:) = inf;
i от 2 до 12
j от 2 до 12
k от 1 до 12
matrix_inf(k, j)~=0
нет
да
pyt(j, i-1) > pyt(k, i - 1) + matrix_inf(j, k) && pyt(j,i) > pyt(k,i-1)+matrix_inf(j,k)
нет
да
pyt(j, i) = pyt(k, i - 1) + matrix_inf(j, k);
pyt(j,i)~=inf
нет
да
pyt(j,i)=pyt(j,i-1);
l = pyt(12,12);
sps = 12;
w = 12;
while(1)
i от 1 до 12
(matrix_inf(i, w)~=0) && (l - matrix_inf(i, w) == pyt(i, 12))
да
(matrix_inf(i, w)~=0) && (l - matrix_inf(i, w) == pyt(i, 12))
конец

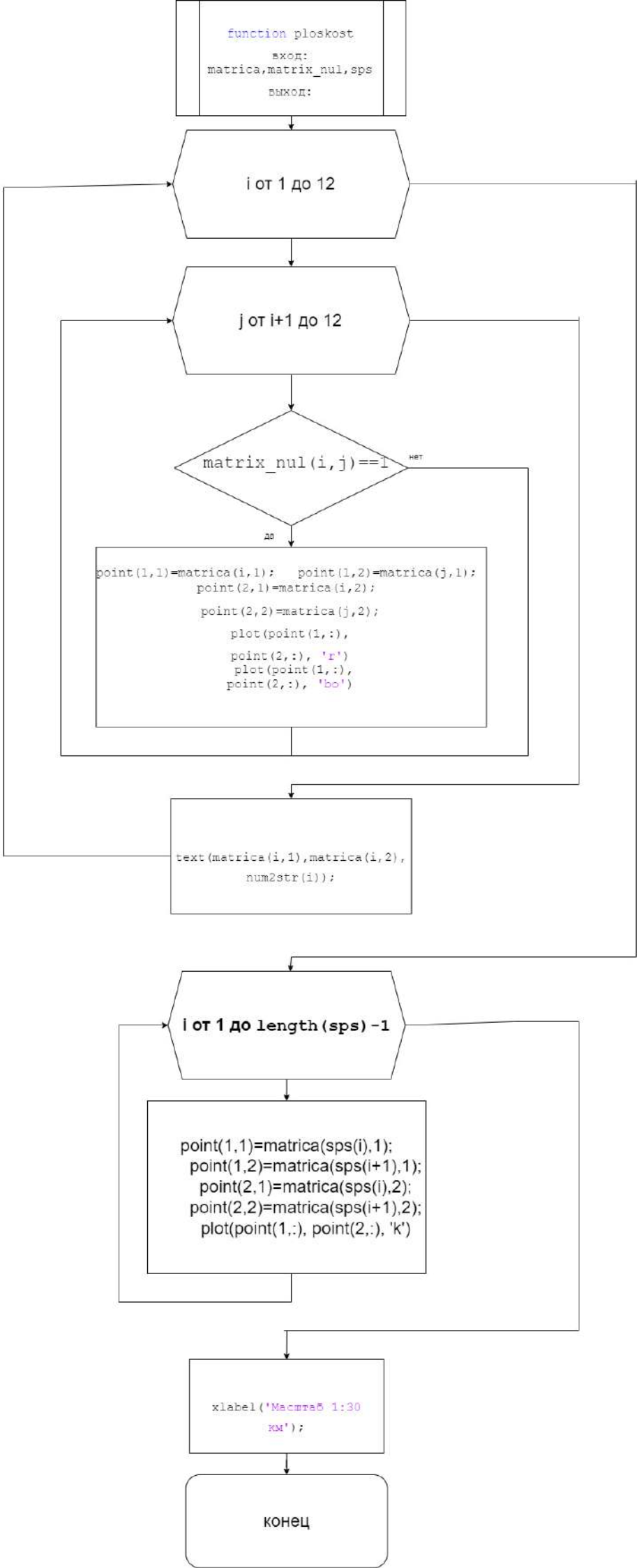

function matrica()
выход: matrica
izfaila=load('CORDS.txt')
disp('Введите координаты в виде [x,y]');
start=input('Начальная точка: ');
last=input('Конечная точка: ');
matrica(1, 1:2)=start;
matrica(2:11,1:2)=izfaila;
matrica(12,1:2)=last;
конец

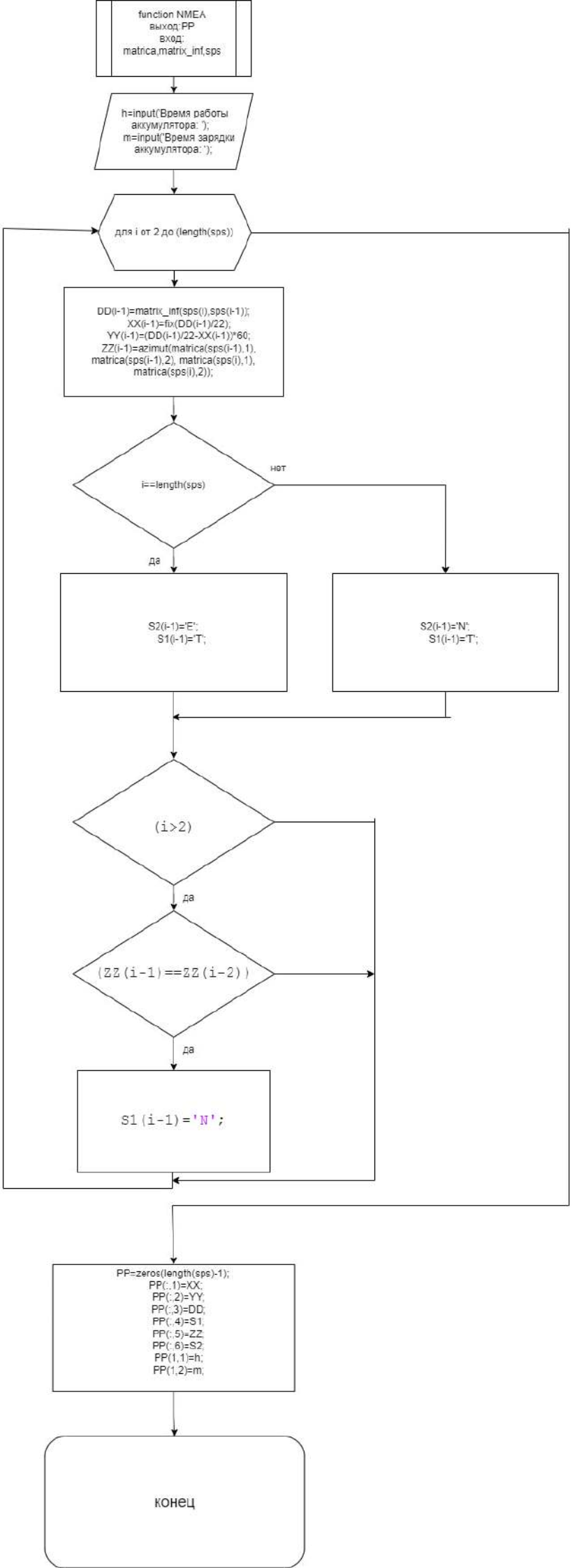

function pifagor
вход:x_1, y_1, x_2, y_2
выход:proverka
proverka=30*sqrt((x_2-x_1)^2+(y_2-y_1)^2);
конец


function sohranim
вход: matrica,File
выход:
siz=size(matrica)
i от 1 до siz(1,1)
j от 1 до siz(1,2)
fprintf(File,'%4d',matrica(i,j));
fprintf(File ,'\r\n');
disp(matrica);


function vozmoshno
выход:matrix_nul,matrix_inf
i от 1 до 12
j от 1 до 12
proverka=(pifagor(matrica(i,1),
matrica(i,2),matrica(j,1),
matrica(j,2)));
нет
proverka<=22*4.5
да
нет
i~=j
да
matrix_nul(i,j)=0;
matrix_inf(i,j)=round(proverka);
matrix_nul(i,j)=1;
matrix_inf(i,j)=round(proverka)
конец


function azimut
вход:x_1, y_1, x_2, y_2
выход:azi
azi=atan((y_2-y_1)/(x_2-x_1))*180/pi;
and(x_2<x_1,y_2>y_1)||and(x_2<x_1,y_2<y_1)
нет
да
azi=180+azi;
and(x_2<x_1,y_2>y_1)||and(x_2<x_1,y_2<y_1)
нет
да
azi=azi+360;
конец


function ploskost
вход:
matrica,matrix_nul,sps
выход:
i от 1 до 12
j от i+1 до 12
matrix_nul(i,j)==1
нет
да
point(1,1)=matrica(i,1); point(1,2)=matrica(j,1);
point(2,1)=matrica(i,2);
point(2,2)=matrica(j,2);
plot(point(1,:),
point(2,:), 'r')
plot(point(1,:),
point(2,:), 'bo')
text(matrica(i,1),matrica(i,2),
num2str(i));
i от 1 до length(sps)-1
point(1,1)=matrica(sps(i),1);
point(1,2)=matrica(sps(i+1),1);
point(2,1)=matrica(sps(i),2);
point(2,2)=matrica(sps(i+1),2);
plot(point(1,:), point(2,:), 'k')
xlabel('Масштаб 1:30
км');
конец


function NMEA
выход:PP
вход:
matrica,matrix_inf,sps
h=input('Время работы аккумулятора: ');
m=input('Время зарядки аккумулятора: ');
для i от 2 до (length(sps))
DD(i-1)=matrix_inf(sps(i),sps(i-1));
XX(i-1)=fix(DD(i-1)/22);
YY(i-1)=(DD(i-1)/22-XX(i-1))*60;
ZZ(i-1)=azimut(matrica(sps(i-1),1), matrica(sps(i-1),2), matrica(sps(i),1), matrica(sps(i),2));
i==length(sps)
нет
да
S2(i-1)='E';
S1(i-1)='T';
S2(i-1)='N';
S1(i-1)='T';
(i>2)
да
(ZZ(i-1)==ZZ(i-2))
да
S1(i-1)='N';
PP=zeros(length(sps)-1);
PP(:,1)=XX;
PP(:,2)=YY;
PP(:,3)=DD;
PP(:,4)=S1;
PP(:,5)=ZZ;
PP(:,6)=S2;
PP(1,1)=h;
PP(1,2)=m;
конец

Выводы:

В данной работе был продемонстрирован алгоритм Форда-Беллмана для поиска кратчайшего пути взвешенного графа. Были составлены блок-схемы по каждому фрагменту кода, получено изображение графа, все результаты работы, как и требовалось, выводятся в файл, а координаты вершин графа берутся так же из текстового файла.