```
{ Matrix_NN.pas

* Разработка по заданию
* https://znanija.com/task/27703988

  На плоскости заданы своими координатами n точек. Создать матрицу,
  * элементами которой являются расстояние между каждой парой точек.
  * Размер матрицы n*n. Т.е. расстояние точки от самой себя  тоже
  считаем.


   Программа Испытывалась на
   * Free Pascal Compiler version 3.0.0 [2015/12/05] for x86_64

ВНИМАНИЕ! НАБЛЮДАЮТСЯ ГЛЮКИ ПРИ N=6 и при N=14! Возможно ещё!
* Причина пока не ясна. Но похоже проявляется при чистке памяти.

   Теорию можно глянуть тут:
   *    Алексеев Е. Р., Чеснокова О. В., Кучер Т. В.
   *    "Free Pascal и Lazarus: Учебник по программированию "

    Или тут:
    * Кетков, Ю. Л.
    * Свободное программное обеспечение. FREE PASCAL для студентов и
    школьников

   Или тут:
    * Мансуров К.Т. Основы программирования в среде Lazarus, 2010.


    В общем, хватает литературы по данному вопросу

   Общие задачи:
   * Пока особо с проверками вводимых данных не заморачиваемся.
   * Упражняемся с 2-мерными динамическим массивами.
   * Заполненяем случайными вещественными числами с точностью до 0,1.
   * Кто хочет, может организовать Задание координат с клавиатуры.
   * Несколько точек можно и так.

 }


program Matrix_NN;
{Создаём 2-мерный динамический массив-таблицу.
```

```
* Заполненяем случайными вещественными числами
* от -100 до 100 с точностью до 0,1.
* И обрабатываем}

uses sysutils, math, crt;

const
    MaxABS10=1000; {Ограничение по модулю чисел в таблице.
    Координата *10}
    L1=5;          {Длина поля вывода элемента массива}
    DC=10;      {1/(точность дроби)}
var
    {Объявление динамического массива, содержащего координаты точек.}
    TabCoord:  Array of Array of real;

    {Объявление динамического массива, содержащего расстояния между
    точками}
    Matrix_Dist:  Array of Array of real;

    i, j, k     :Integer;   {Переменные счётчики}
    N           :Integer;   {Задаваемое количество точек (размеры
    массива) }

    Tmp         :LongInt;   {Переменная для хранения результатов чисел}
    StrTmp      :String;    {Переменная для вывода результатов строк}
    StNum       :String[7];


    {Дополнительные параметры, в зависимости от задачи}

{Функция расчета расстояния между точками из 2-мерного массива.}
{ Массивы только глобальные. Пока не знаю,
*  можно ли динамический массив протолкнуть в качестве аргумента.
*  А главное нужен ли такой выверт?}

function Distance(k, n: integer) : real ;
begin
{Возвращаем значение расстояния между точками с номерами k и n}
 Distance:=SQRT(SQR(TabCoord[0,k]-TabCoord[0,n])+SQR(TabCoord[1,k]-
 TabCoord[1,n]));
end ;


{================ Основная программа ================================}
```

```
BEGIN
  {-v-  Запрос данных от пользователя -v---------------v--}

  ClrScr(); {Чистим "табло"}
  {Запрашиваем число точек массива.
  * Вводимые числа сравниваем с минимально допустимым значением.
  * Считаем это 2, ибо 1 неинтересно, ≤0, вовсе массив не получится}

  Repeat
    Writeln('Задайте число точек целое N больше или равно 2');
    Readln(N);
    if N<2
    then Writeln('Число точек должно быть больше или равно 2!');
  Until (N≥2);

  {^    конец запроса данных от пользователя -----------}
  Writeln(); {пустая строка для отделения результатов}


  {-v-  Подготовка основного цикла---------------------- -v-}
  { создаем динамический 2 мерный массив координат точек.
  NB нумерация элементов динамических массиввов начинается с 0!
  * Номерами точек будут номера столбцов.
  * X координаты хранятся в 0-й строке.
  * Y координаты хранятся во 1-й строке.}

  SetLength(TabCoord, N, 2); {Отводим память под массив}

  {Заполняем массив}
  Randomize;
  For j:=0 to N-1 DO {Пробегаем по столбцам}
        begin
          For i:=0 to 1 DO {Пробегаем по строкам}
                begin
                  TabCoord[i,j]:=Random(MaxAbs10)/DC;

                  { Извернёмся так, чтоб числа в таблице могли
                  * быть отрицательными}
                  if Random<0.5 then TabCoord[i,j]:=(-1)*TabCoord[i,j];
                end;
        end;

 {^ Подготовка основного цикла----------------------^--------^}
```

```
{-v--  Обработка и вывод результатов                                    ↵
-----------------------------v-}

  {Вывод полученного массива координат для контроля результатов}
Writeln('Таблица координат точек');

{Формируем первую строку с номерами точек}
StrTmp:='№ ';
For j:=0 to N-1 DO {Пробегаем по столбцам}
               begin
               {Пробуем выровнять по правому краю (младшему                ↵
               разряду). }
                 StNum:=IntToStr(j+1); {Ставим номера точек от 1 до N}
                 Tmp:=L1-Length(StNum);
                {Если длина элемента <L1, дополняем элемент                 ↵
                пробелами слева}
                 IF Tmp≥0
                 then For k:=0 to Tmp DO StrTmp:=StrTmp+' ';
                 StrTmp:=StrTmp+StNum+'  ';
               end;
Writeln(StrTmp);

For i:=0 to 1 DO {Пробегаем по строкам}
      begin
        if (i=0) then StrTmp:='X ';
        if (i=1) then StrTmp:='Y ';
        For j:=0 to N-1 DO {Пробегаем по столбцам}
               begin
               {выравниваем по правому краю (младшему разряду).
               * Не очень удачно в нашем случае.
               * Лучше или по целым равнять,
               * или использовать формат с фиксированным числом            ↵
               знаков после запятой.}
                 StNum:=FloatToStr(TabCoord[i,j]);
                 Tmp:=L1-Length(StNum);
                {Если длина элемента <L1, дополняем элемент                 ↵
                пробелами слева}
                 IF Tmp≥0
                 then For k:=0 to Tmp DO StrTmp:=StrTmp+' ';
                 StrTmp:=StrTmp+StNum+'  ';
               end;
        Writeln(StrTmp);
      end;
```

```
Writeln(); {пустая строка для отделения результатов}

{-------------------------------------------------------------------
-}

{ создаем динамический 2 мерный массив матрицу расстояний размером
N*N.
NB нумерация элементов динамических массиввов начинается с 0!
* Потому индексы меняются от 0 до N-1.
}

SetLength(Matrix_Dist, N, N); {Отводим память под массив}

{Заполняем массив. Вычисляем расстояния}
For i:=0 to N-1 DO {Пробегаем по строкам}
      begin
        For j:=0 to N-1 DO {Пробегаем по столбцам}
              begin
                Matrix_Dist[i,j]:=Distance(i,j);
              end;
      end;

Writeln(); {пустая строка для отделения результатов}

{-------------------------------------------------------------------
-}

{Вывод полученного массива расстояний}
Writeln('Таблица расстояний');
Writeln();

For i:=0 to N-1 DO {Пробегаем по строкам}
      begin
        StrTmp:='';
        For j:=0 to N-1 DO {Пробегаем по столбцам}
              begin
              {выравниваем по правому краю (младшему разряду).
              * Не очень удачно в нашем случае.
              * Лучше или по целым равнять,
              * или использовать формат с фиксированным числом
              знаков после запятой.}
                StNum:=FloatToStr(Matrix_Dist[i,j]);
                Tmp:=L1-Length(StNum);
```

```
                    {Если длина элемента <L1, дополняем элемент
                    пробелами слева}
                     IF Tmp≥0
                     then For k:=0 to Tmp DO StrTmp:=StrTmp+' ';
                     StrTmp:=StrTmp+StNum+'  ';
                   end;
            Writeln(StrTmp);
          end;

  Writeln(); {пустая строка для отделения результатов}


 {---^--    Обработка и вывод результатов ----------------------^-}


 {-v-- Завершение работы программы ---v--------------------v-}
 {Чистим память массивов}
//  TabCoord:=nil;
//  Matrix_Dist:=nil;

  Writeln('');
  Writeln('Работа программы закончена.');
  Writeln('Нажмите любую клавишу для выхода');
  {--   Ожидание нажатой клавиши задержка -------------------}
  ReadKey();
  {^    Ожидание нажатой клавиши        ---------------------}
END.
```