```
{ Tab_DAR.pas

* Разработка по заданию
* https://znanija.com/task/27585062

  составить функцию в паскале которая вазвращаяет:
1 кол-во четных чисел таблицы
2 сумму нечетных чисел таблицы
3 кол-во четных чисел таблицы которые стоят на четных местах
* А что считать чётным местом для 2-мерного?
* Будем считать таковым такое, у которого сумма индексов чётная.

* Ну одну функцию, возвращающую в основнную программу 3 значения
* можно "соорудить", но принято возвращать одно.
* Тогда лучше или процедуру писать или 3 функции. Сделаем 3 функции.


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

   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.

    Е.А.Попов Экспресс курс программирования в Lazarus
    * (Краткий справочник, страниц 80. Есть различные версии разных лет
    *  v42 за 2016 год)

    В общем, хватает литературы по данному вопросу

   Общие задачи:
   * Пока особо с проверками вводимых данных не заморачиваемся.
   * Поупражняемся с 2-мерным динамическим массивом с случайным
   заполнением

 }
```

```
program Tab_DAR;
{Создаём 2-мерный динамический массив-таблицу.
* Заполняем её случайными целыми числами. И обрабатываем}

uses sysutils, math, crt;

const
    MaxABS=10000; {Ограничение по модулю чисел в таблице}
    L1=6;         {Длина поля вывода элемента массива}
var
    DATab:  Array of Array of LongInt; {Объявление динамического
    массива.}

    i, j, k     :Integer;   {Переменные счётчики}
    iMax, jMax  :Integer;   {Задаваемые 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}

{Функции обработки 2-мерного массива.}
{ Массив только глобальный. Пока не знаю,
*  можно ли динамический массив протолкнуть в качестве аргумента.
*  А главное нужен ли такой выверт?}

{Функция, считающая количество чётных чисел в массиве или его части}
function NumEven(kMAx, nMax: integer) : LongInt ;
var
    i, j            :Integer;   {Переменные счётчики}
    NTemp           :LongInt;   {Текущее число сосчитанных элементов}

begin
  NTemp:=0; {Обнуляем число сосчитанных элементов}

  {Проверяем элементы массива}
  For i:=0 to kMax DO
        begin
          For j:=0 to nMax DO
                begin
                  { Если элемент чётный, наращиваем сумму}
```

```
                if ((DATab[i,j] mod 2)=0) then NTemp:=NTemp+1;
              end;
        end;

{Возвращаем значение суммы}
 NumEven:=NTemp;
end ;

{Функция, считающая сумму нечётных чисел}
function SumOdd(kMAx, nMax: integer) : LongInt ;
var
    i,j             :Integer;   {Переменные счётчики}
    STemp           :LongInt;   {Времнное хранилище суммы}

begin
  STemp:=0; {Обнуляем сумму}

  {Проверяем элементы массива}
  For i:=0 to kMax DO
        begin
          For j:=0 to nMax DO
                begin
                  { Если элемент нечётный, наращиваем сумму}
                  if ((DATab[i,j] mod 2)<>0) then STemp:=STemp+DATab[
                  i,j];
                end;
        end;

{Возвращаем значение суммы}
 SumOdd:=STemp;
end ;

{Функция, считающая число чётных чисел на чётных местах}
function NEven_PlaceEven(kMAx, nMax: integer) : LongInt ;
var
    i,j             :Integer;   {Переменные счётчики}
    NTemp           :LongInt;   {Текущее число сосчитанных элементов }

begin
  NTemp:=0; {Обнуляем число сосчитанных элементов}

  {Проверяем элементы массива}
  For i:=0 to kMax DO
        begin
```

```
          For j:=0 to nMax DO
                begin
                  { Если место чётное, проверяем элемент}
                  if (((i+j) mod 2)=0)
                   then
                   { Если элемент чётный, наращиваем сумму}
                   if ((DATab[i,j] mod 2)<>0) then NTemp:=NTemp+1;
                end;
        end;

{Возвращаем значение суммы}
 NEven_PlaceEven:=NTemp;
end ;


{================ Основная программа ================================}
BEGIN
  {-v-  Запрос данных от пользователя -v---------------v--}

  ClrScr(); {Чистим "табло"}
  {Запрашиваем размеры массива.
  * Вводимые числа сравниваем с минимально допустимым значением.
  * Считаем это 2, ибо 1 неинтересно, ≤0, вовсе массив не получится}
  Repeat
    Writeln('Задайте число строк массива iMax больше или равно 2');
    Readln(iMax);
    if iMax<2
    then Writeln('Число строк должно быть больше или равно 2!');
  Until (iMax>2);

   Repeat
    Writeln('Задайте число столбцов массива jMax больше или равно 2');
    Readln(jMax);
    if jMax<2
    then Writeln('Число столбцов должно быть больше или равно 2!');
  Until (jMax>2);

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


  {-v-  Подготовка основного цикла---------------------- -v-}
  { создаем динамический 2 мерный массив и заполняем его}
  { При желании, можно совместить заполнение и обработку}
```

```
 SetLength(DATab, iMax, jMax); {Отводим память под массив}
 {NB нумерация элементов динамических массиввов начинается с 0!}

 {Заполняем массив}
 Randomize;
 For i:=0 to iMax-1 DO
       begin
         For j:=0 to jMax-1 DO
               begin
                 DATab[i,j]:=Random(MaxAbs);
                 { Извернёмся так, чтоб числа в таблице могли
                 * быть отрицательными}
                 if Random<0.5 then DATab[i,j]:=(-1)*DATab[i,j];
               end;
       end;

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


 {-v--  Обработка и вывод результатов                          ↲
 ------------------------------v-}
   {Вывод полученного массива для контроля результатов}

 Writeln('Таблица ');
 For i:=0 to iMax-1 DO
       begin
         StrTmp:='';
         For j:=0 to jMax-1 DO
               begin
               {Пробуем выровнять по правому краю (младшему          ↲
               разряду). }
                 StNum:=IntToStr(DATab[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}
```

```
    {Вычисляем. Просто вызываем функции. И печатаем результаты.}
  Tmp:=NumEven(iMax-1, jMax-1);
  Writeln('Количество чётных элементов в таблице ');
  Writeln('Ne=', Tmp:10);

  Tmp:=SumOdd(iMax-1, jMax-1);
  Writeln('Сумма нечётных элементов в таблице ');
  Writeln('Sodd=', Tmp:10);

  Tmp:=NEven_PlaceEven(iMax-1, jMax-1);
  Writeln('Количество чётных элементов в таблице на четных местах');
  Writeln('Nepe=', Tmp:10);

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


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