Arrays

One Dimensional Arrays

An Array is a data structure, that stores data of the same data type.

1. **Data structure**: a collection of data, ie: a collection of names of people, or a collection of marks in the grade
2. **Data type**: a array can only hold data of the same type ie: only integers, or only strings
3. You will always need two things when working with arrays:
    1. A `for` loop to go through all the elements inside of the array
    2. A **position** variable which, keeps track of which item in the array

Example

Let's look at an example which shows why we would want to use an array:

```
var sName1 : string; sName2 : string; sName3 : string; sName4 : string;
sName5 : string;
begin
sName1 := 'Johnny';
sName2 := 'Tommy';
sName3 := 'Sunny';
sName4 := 'Molly';
sName5 := 'Mary';
end;
```

Let's do the same thing using an array:

```
procedure TForm1.Button2Click(Sender: TObject);
var Names : array [0..4] of string;
begin
Names[0] := 'Johnny';
Names[1] := 'Tommy';
Names[2] := 'Sunny';
Names[3] := 'Molly';
Names[4] := 'Mary';
end;
```

The first element in an array is indexed at position 0, the second element at position 1 and so forth. In our example, our array contains 5 elements (0..4).

Why then do we need an array, if we can do the same thing with variables? Think about this, if we had 100 names to store, instead of 5 are we going to write 100 string variables? No we are not, instead we will we use a " data structure " like an array to easily manage the large amount of names.

```
procedure TForm1.Button2Click(Sender: TObject);
var Names : array [0..4] of string;
    Numbers : array [1..10] of integer;
begin
end;
```

Declaring an array

An array can be declared like this:

```
var Names : array [0..4] of string;
```

This declaration can be in two places.

1. Global array (accessible through all procedures)

```
unit Unit1;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls,
Forms,
  Dialogs;

type
  TForm1 = class(TForm)
  private
```

```
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;
  var Names : array [0..4] of string; // global array declaration

implementation

{$R *.dfm}

end.
```

2. Locally (accessible only inside the procedure it was declared)

```
unit Unit1;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls,
Forms,
  Dialogs, StdCtrls;

type
  TForm1 = class(TForm)
    Button1: TButton;
    procedure Button1Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;


implementation

{$R *.dfm}

procedure TForm1.Button1Click(Sender: TObject);
var Names : array [0..4] of string; // local array
begin

end;
```

```
end.
```

Initialize an array

If we wanted to put a value inside of the first position of the array *Names*:

```
procedure TForm1.Button2Click(Sender: TObject);
var Names : array [0..4] of string;

begin
    Names[0] := 'Johnny';
end;
```

> To initialize and declare an array(only **global arrays**) cant do this with local arrays

```
 var Names : array [0..4] of string =
('Johnny','Sonny','Mikey','Billy','Bobby');
```

Display/Retrieve a Value from an Array

To get the values out from the array you need:

1. A **for loop**
2. The **position** variable

```
procedure TForm1.Button1Click(Sender: TObject);
var i : integer; // position variable
begin
for i := low(Names) to high(Names) do
begin
    redOutput.Lines.Add(Names[i]);
end;
```

Alternately you could have written this *for loop* like this, since there are five elements inside the array:

```
var i : integer; // position variable
begin
for i := 0 to 4 do
begin
    redOutput.Lines.Add(Names[i]);
end;
```

Sorting an array

To sort the elements inside of the array, you have to use a **sorting algorithm** called **selection sort**. This will order the elements inside the array from lowest to highest.

> Now there are other sorting algorithms like bubble sort, but in Grade 11-12 selection sort is the one you will use

Here we have a global array with some numbers inside of it:

```
var arrNum : array[1..7] of integer = (2,4,6,1,0,9,6);
```

To sort the array using selection sort:

```
procedure TForm1.Button1Click(Sender: TObject);
var
  i, j, temp: integer;
begin
  for i := 1 to length(arrNum) - 1 do
    for j := i + 1 to length(arrNum) do
      if (arrNum[i] > arrNum[j]) then
      begin
        temp := arrNum[j];
        arrNum[j] := arrNum[i];
        arrNum[i] := temp;
      end;
end;
```

Find the lowest element inside of an array

To find the lowest element inside of an array, we say let the first element be the lowest. The we use the for loop to go over all the elements inside of the array, and compare it to our lowest. If one of them is lower than the lowest that we set, we make that new value to the lowest.

```
procedure TForm1.btnFindLowestClick(Sender: TObject);
var lowest,i: integer;
begin
lowest := arrNum[1]; // set the lowest to the first element
   for i:= 1 to 7 do
   begin
     if arrNum[i] < lowest then
      lowest := arrNum[i];
   end;
ShowMessage('The lowest element inside of arrNum is: ' +
inttostr(lowest));
end;
```

Find the highest element inside of an array

This is the same procedure as finding the lowest, except we use the **greater than sign**(>) inside of the `for` loop.

```
procedure TForm1.btnFindHighestClick(Sender: TObject);
var highest,i: integer;
begin
highest := arrNum[1]; // set the highest to the first element
   for i:= 1 to 7 do
   begin
     if arrNum[i] > highest then
      highest := arrNum[i];
   end;
ShowMessage('The lowest element inside of arrNum is: ' +
inttostr(highest));
end;
```

Searching an array for an element

If we wanted to check if something exists inside of an array:

1. First ask the user what they want to search for (inputbox)
2. The use a ```for`` loop to go through all the elements inside of the array, and check if they are equal to what the user wants to search for.
    1. If it is equal, set a variable called `bFound` to true
3. If `bFound` is true, use a `showmessage` to say the item was found

```
procedure TForm1.btnSearchClick(Sender: TObject);
var i:  integer;
bFound: boolean;
sItem : string;
begin
  sItem := inputBox('Search Item', 'Enter in a item to search for','');
  bFound := false; // variable to see if item is found

  for i := low(arrNum) to high(arrNum) do
  begin
      if strtoint(sItem) = arrNum[i] then
        bFound := true;
  end;

  if bFound = True then
    ShowMessage(sItem + ' exists inside of the array')
  else
    ShowMessage(sItem + ' does not exist inside of the array');
end;
```

Two Dimensional Arrays

The 2D array is a data structure that can store data in a table format.

> It can only hold data of the same data type

Here we can see a table, that shows the soccer stats for Christiano Ronaldo in the year 2014/2015. The data is arranged in a table format.

Declare a 2D Array

To declare a 2D array:

```
arrStats : array[1..6,1..7] of Integer;
```

1. `1..6` represents the number of rows inside the 2D array. In this case 6.
2. `1..7` represents the number of columns inside the 2D array. In this case 7.

> Note: Since a 2D array can only hold data of the same type, the 2D array will only contain the numeric stats and not the headings such as the Goals, Assists, La Liga etc.

To declare and initialize a 2D array with values, you do this **globally**:

```
arrStats : array [1..6,1..7] of Integer = ((35,48,16,6,1,10,3099),
(1,2,0,0,0,
  0,90),(2,0,0,1,0,0,90),(2,1,0,0,0,0,117),(12,10,3,1,0,3,1065)
  ,(1,0,0,0,0,0,67));
```

Printing the contents of a Two Dimensional Array

To print the contents of a 2D array, we make use of two for loops, and a output variable.

1. The first `for` loop is `for` the **number of rows.**
2. The second `for` loop is for the **number of columns**.
3. A string variable called `sOuput`, which will hold each rows data so that we can add it to a richedit for display.

```
procedure Tfrm2DArrays.btnDisplayClick(Sender: TObject);
var row,col : integer;
sOutput : string;
begin
  for row := 1 to 6 do
  begin
    sOutput := '';
    for col := 1 to 7 do
    begin
      sOutput := sOutput + inttostr(arrStats[row,col]) + #9;
    end;
    redOutput.Lines.Add(sOutput);
```

```
    end;
  end;
```

Recall how data is printed inside of a **richedit**:

> It is printed line by line.

Therefore to print the **column** headings, we have to specify this first. Modify the code to include the column headings:

```
procedure TForm1.btnDisplayClick(Sender: TObject);
var row,col : integer;
sOutput : string;
begin
  redOutput.Lines.Add('' + #9 + 'MP' + #9 + 'Goals' + #9 + 'Assists' + #9
+
  'Yellow' + #9 + 'Red' + #9 + 'Penalty' + #9 + 'Min');
  for row := 1 to 6 do
  begin
    sOutput := '';
    for col := 1 to 7 do
    begin
      sOutput := sOutput + inttostr(arrStats[row,col]) + #9;
    end;
    redOutput.Lines.Add(sOutput);
  end;
end;
```

To add in the **row** headings, we cant use the same method because a `for` is being used to print each row. Therefore we will first declare a global array to store the row headings:

```
arrRow : array[1..6] of String = ('La Liga', 'UEFA Super Cup',
  'Supercopa de Espana','Copa del Rey','UEFA Champsions Legue',
  'International Friendlies');
```

Next we modify our printing code by outputting the **row** headings first:

```
procedure TForm1.btnDisplayClick(Sender: TObject);
var row,col : integer;
sOutput : string;
begin
  redOutput.Lines.Add('' + #9 + 'MP' + #9 + 'Goals' + #9 + 'Assists' + #9
+
  'Yellow' + #9 + 'Red' + #9 + 'Penalty' + #9 + 'Min');
  for row := 1 to 6 do
  begin
    sOutput := '';
    sOutput := arrRow[row] + #9; // output the row name first then the
```

```
data
    for col := 1 to 7 do
    begin
      sOutput := sOutput + inttostr(arrStats[row,col]) + #9;
    end;
    redOutput.Lines.Add(sOutput);
  end;
end;
```

Adding the columns of a Two Dimensional Array

Sometimes it useful to have the ability to add up the columns to determine a sum/total. For example, if we wanted to determine how many total goals where scored during the 2014/2015 season we would have to add each item in the Goals columns.

```
procedure TForm1.btnAddColumnsClick(Sender: TObject);
var row, col, colSum : integer;
colOutput : string;
begin
  colOutput := 'Total' + #9;
  for col := 1 to 7 do
  begin
  colSum := 0;
    for row := 1 to 6 do
    begin
      colSum := arrStats[row,col] + colSum;
    end;
    colOutput := colOutput + inttostr(colSum) + #9;
  end;
  redOutput.Lines.Add(colOutput);
end;
```

Adding the rows of a Two Dimensional Array

In this example, it is not useful to add the values of the rows up. However, if asked this is how you would do it:

```
procedure TForm1.btnAddRowsClick(Sender: TObject);
var row, col, rowSum : integer;
colOutput : string;
begin
  colOutput := 'Row Totals' + #9;
  for row := 1 to 6 do
  begin
  rowSum := 0;
    for col := 1 to 7 do
    begin
      rowSum := arrStats[row,col] + rowSum;
```

```
    end;
    colOutput := colOutput + inttostr(rowSum) + #9;
  end;
  redOutput.Lines.Add(colOutput);
end;
```

The code is exactly the same as adding the columns, except the order of the `for` loops change: ie: in adding the columns, the col `for` loop came first and in adding the rows the row `for` loop comes first.

2D Exam Question

In this example we will be working with the **MAX** exam question.

When you open the question, and go into the code section you are presented with two arrays a global variable called `iStartWeek`.

```
var
  arrDepartments: array [1..8] of String = (
    'PCs & Notebooks',
    'Tablets & eReaders',
    'Software',
    'Printers, Toners and Ink',
    'Cellphones',
    'Gaming & Drones',
    'Network Equipment',
    'Accessories'
  );

arrSales : array[1..8, 1..6] of Real = (
      (935.89,965.99,4056.77,5023.89,3802.66,1146.98),
      (2667.78,2491.78,1989.65,2647.88,1601.56,1921.99),
      (6702.45,4271.56,3424.45,3924.55,3085.45,3359.77),
      (6662.34,6658.45,8075.43,2360.66,2635.44,7365.69),
      (16405.33,9741.37,13381.56,18969.76,8604.55,20207.56),
      (10515.29,7582.66,9856.56,7537.68,9115.67,8401.55),
      (7590.99,9212.65,9070.98,6439.99,7984.88,8767.45),
      (9220.65,8097.99,10067.44,9960.87,10109.56,6571.66));

  iStartWeek: Integer = 1;
```

When analyzing the 2D array `arrSales` you need to note:

1. DataType : float
2. Rows: 1..8 ie: 8 elements
3. Columns: 1..6 6 elements

Question 3.1

First go into Button 3.1 code and add in the columns headings, since that is the first line in the richedit.

```
procedure TfrmQuestion3.btnQ3_1Click(Sender: TObject);
begin
  //Question 3.1
  redQ3.Lines.Clear;

redQ3.Lines.Add('Department'+#9+'Week1'+#9+'Week2'+#9+'Week3'+#9+'Week4'+#
9);
end;
```

Next to we need to print the data of the of the 2D array `arrSales`. Refer to Printing the contents of a Two Dimensional Array First lets modify our code to add in the data from the 2D array.

```
procedure TfrmQuestion3.btnQ3_1Click(Sender: TObject);
var
  row, col: Integer;
  str: string;
begin
  // Question 3.1
  redQ3.Lines.Clear;
   redQ3.Lines.Add('Department' + #9 + 'Week1' + #9 + 'Week2' + #9 +
'Week3' +
      #9 + 'Week4' + #9 + 'Week5' + #9 + 'Week6');
  for row := 1 to 8 do
  begin
    str := '';
    for col := 1 to 6 do
    begin
      str := str + floattostrf(arrSales[row, col], ffcurrency, 10, 2) +
#9;
    end;
    redQ3.Lines.Add(str);
  end;
end;
```redQ3.Lines.Add('Department' + #9 + 'Week1' + #9 + 'Week2' + #9 +
'Week3' +
      #9 + 'Week4' + #9);

Recall that the row headings are contained in the one dimensional array
```arrDepartments```. Modify the code to include the row headings.

```pascal
procedure TfrmQuestion3.btnQ3_1Click(Sender: TObject);
var
  row, col: Integer;
  str: string;
begin
  // Question 3.1
  redQ3.Lines.Clear;
  redQ3.Lines.Add('Department' + #9 + 'Week1' + #9 + 'Week2' + #9 +
```
```

```
'Week3' +
      #9 + 'Week4' + #9 + 'Week5' + #9 + 'Week6');
  for row := 1 to 8 do
  begin
    str := '';
    str := arrDepartments[row] + #9;
    for col := 1 to 6 do
    begin
      str := str + floattostrf(arrSales[row, col], ffcurrency, 10, 2) +
#9;
    end;
    redQ3.Lines.Add(str);
  end;
end;
```

Question 3.2

This question wants to make some sort of report. We have to create a report based on when a department is under performing. A department is under performing when:

> Its sales figure is lower than the average sales for every department per week

This raises the question how do we get the "average sales for every department per week" If you look at question 3.1 - you will see that we need to add up the colums and divide by 8.

The first thing we need to do is work out the average per week. The week data is contained inside of each columns. Therefore lets add up our columns, and divide by 8(since) since there are 8 rows of data.

Now we have to compare each departments weekly sales figure, and if that sale value is less than the average per week, output it to the rich edit.

```
  for row := 1 to 8 do
    begin
      if (arrSales[row, col] < average) then
      begin
        redQ3.Lines.Add(arrDepartments[row] + #9 +
floattostrf(arrSales[row,
            col], ffcurrency, 10, 2));
      end;
```

The complete code now looks like this:

```
//
=============================================================================
=
```

```
// Question 3.2
//
================================================================================
=
procedure TfrmQuestion3.btnQ3_2Click(Sender: TObject);
var
  row, col: Integer;
  average: Real;
begin
  // Question 3.2
  redQ3.Clear;
  redQ3.Lines.Add('Underperforming depeartments per week');
  average := 0;
  for col := 1 to 6 do
  begin
    average := 0;
    for row := 1 to 8 do
    begin
      average := average + arrSales[row, col];
    end;
    average := average / 8;
    redQ3.Lines.Add(''); // spacing between weeks
    redQ3.Lines.Add('Week ' + inttostr(col) + ' Average sales figure ' +
        floattostrf(average, ffcurrency, 10, 2));


    for row := 1 to 8 do
    begin
      if (arrSales[row, col] < average) then
      begin
        redQ3.Lines.Add(arrDepartments[row] + #9 +
floattostrf(arrSales[row,
            col], ffcurrency, 10, 2));
      end;
    end;
  end;
end;
```