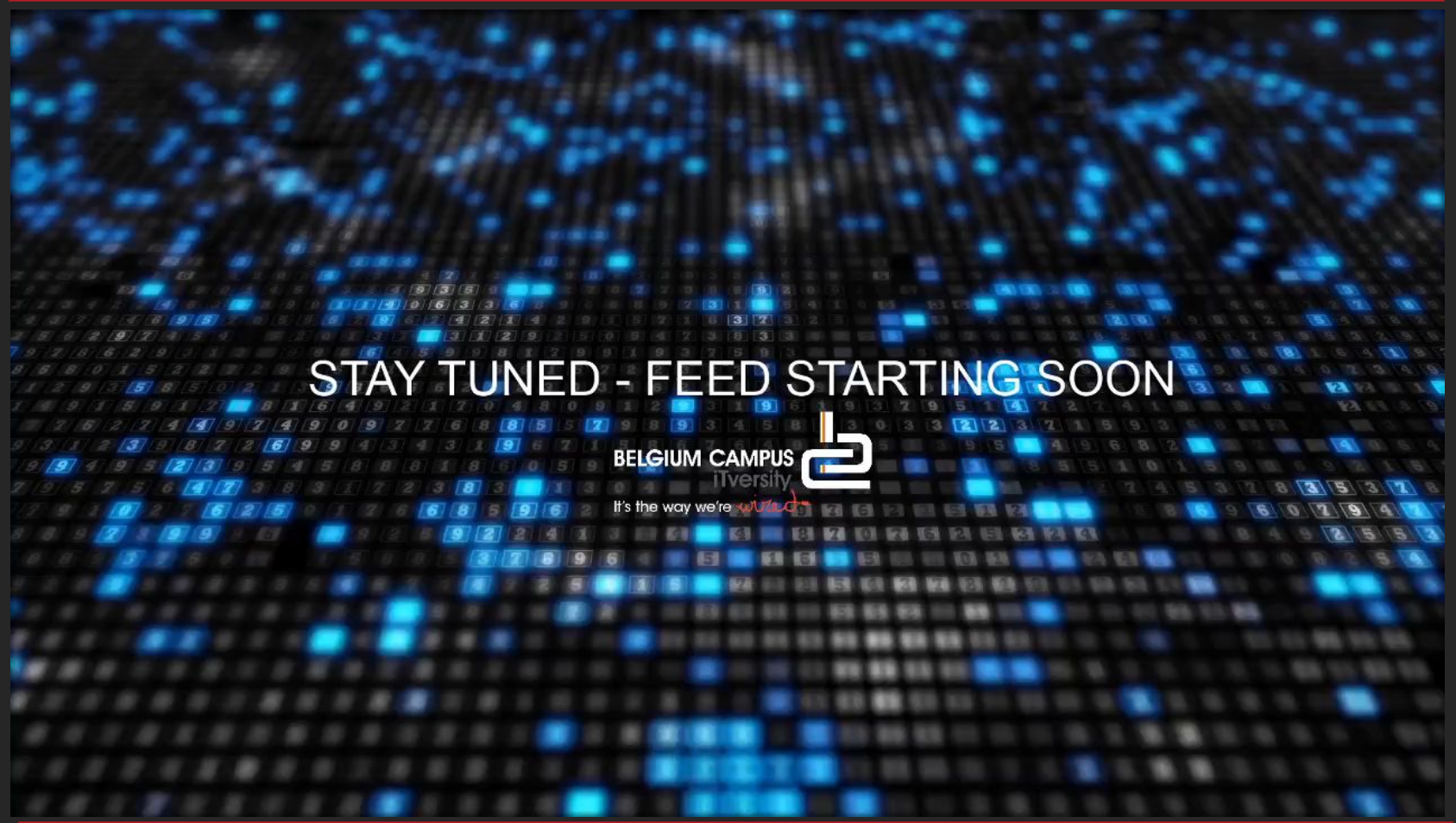

STAY TUNED - FEED STARTING SOON
BELGIUM CAMPUS
iTversity
It's the way we're wired

BELGIUM CAMPUS
iTversity
It's the way we're wired

DELPHI - ARRAYS

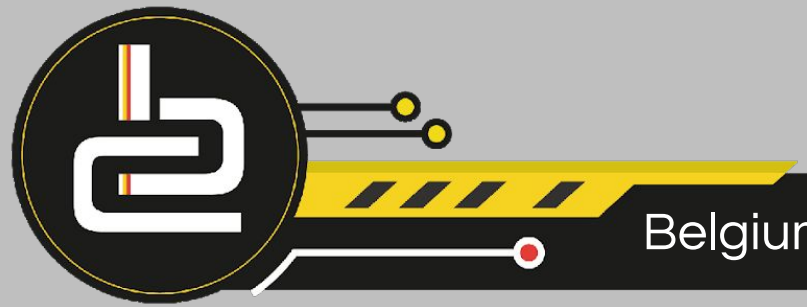
V. Pretorius

LESSON OBJECTIVES

- One dimensional arrays
- Declaration of arrays
- Adding values to arrays
- Displaying elements of an array
- Sorting Arrays (bubble sorting)

PRIOR KNOWLEDGE

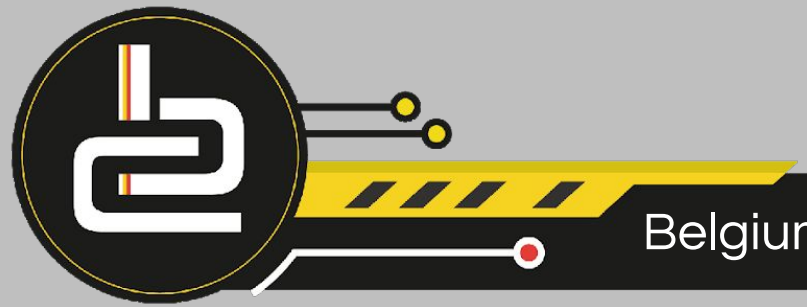
- In order to complete the content on arrays, the following prior knowledge from grades 10 and 11 Information Technology is required:
  - Character handling
  - Text files
  - Control structures
  - Error trapping



AIM

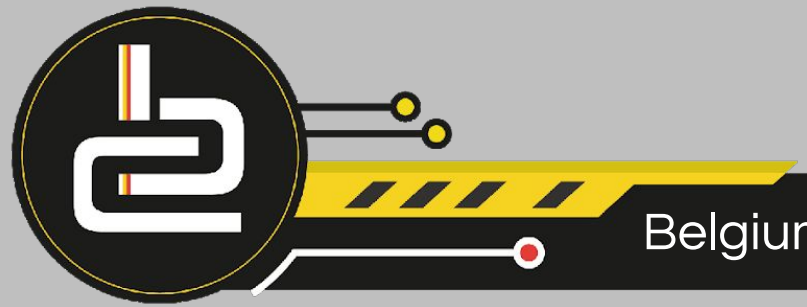
After you have completed this chapter, you should be able to:

- explain what an array is,
- use arrays as data structures,
- sort arrays.



WHAT IS AN ARRAY?

- An array is
  - a temporary data structure that only exists in memory;
  - a single data structure
    - that contains a collection of related data items of the same data type;
    - that are stored under a single variable name;
    - where every data item has a unique index;



DECLARATION OF A 1D-ARRAY

Name of the array

Datatype
(*Can only be a simple data type*)

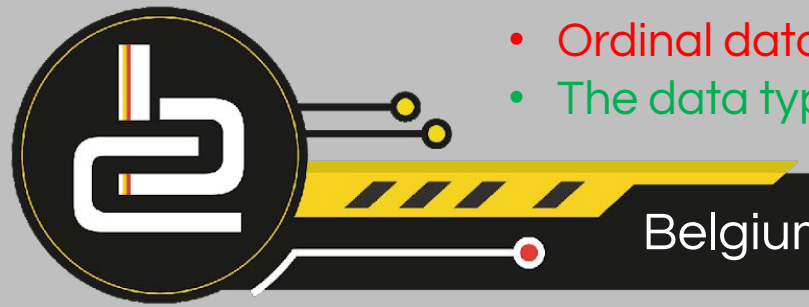
```
VAR
  arrABC : Array[1..100] of string;
```

Lower index

Upper index

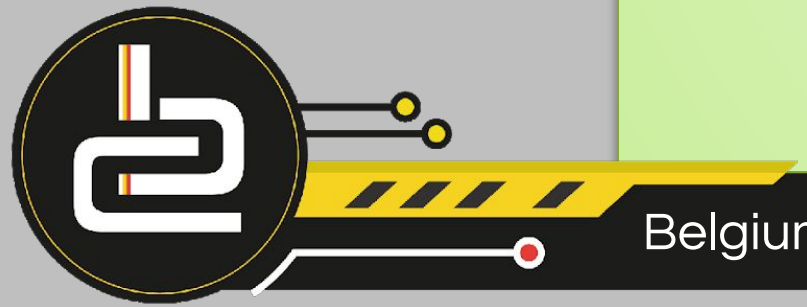
NOTE:

- The lower index and upper index MUST be of ordinal data type.
- Ordinal data types are ***Char***, ***Integer*** or ***Boolean***.
- The data type of all the elements of the array are the same.



ADDING VALUES TO THE ELEMENTS OF AN ARRAY.

- **The values of an array may be added in various ways:**
  - Assigned as part of the array declaration using a constant array,
  - Assigned on the creation of the Delphi form during the OnCreate or OnActivate event,
  - User typing the information on the keyboard using edit controls or the InputBox,
  - Read data from a text file.



VALUES ASSIGNED AS PART OF THE ARRAY DECLARATION.

Used when:

- the content of the array won't change during the execution of the program

```
CONST
    arrWeekday : array[1..7] of string =
        ('Sunday', 'Monday', 'Tuesday', 'Wednesday',
        'Thursday', 'Friday', 'Saturday');
```

NOTE:
The values of a constant array cannot change for the whole duration of the Delphi project.

The values of the array:

Index	1	2	3	4	5	6	7
Value	Sunday	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday

VALUES ASSIGNED ON CREATING OF THE DELPHI-FORM.

Used when:

- the programmer knows in advance the values that are going to be placed in the array,
- there are only a few elements...

GLOBAL Declaration:

```
{private declarations of the form}
    arrPrize : Array[1..3] of String;
```

```
Procedure TForm1.FormCreate(Sender : Tobject);
Begin
  arrPrize[1] := 'Nokia 6600';
  arrPrize[2] := 'Siemens MC60';
  arrPrize[3] := 'No Prize';
End;
```

The values of the array:

Index	1	2	3
Value:	Nokia 6600	Siemens MC60	No Prize

VALUES ASSIGNED BY TYPING FROM THE KEYBOARD.

Used when:

- The user needs to add the values to the array during execution of the project.

GLOBAL Declaration:

```
{private declarations of the form}
   arrNumbers : Array[1..100] of Real;
   Counter    : Integer;
```

```
Procedure TForm1.btnInputClick(Sender : Tobject);
Begin
  Inc(Counter, 1);
  arrNumbers[counter] := StrToFloat(Inputbox('Number','Enter a
                                                  number','0'));
End;
```

The elements in this array depend on the input from the user!

EXERCISE 1

WRITE A DELPHI PROGRAM FOR EACH OF THE FOLLOWING SCENARIOS:

A) Read 10 **names** from the keyboard into an array and display the names in a rich edit.

Solution

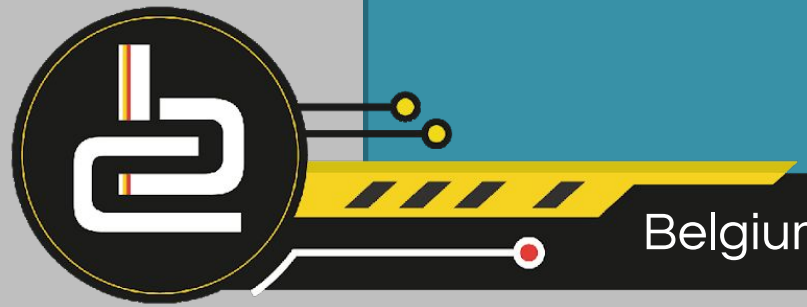
B) Read 10 **numbers** from the keyboard into an array and display the numbers in a rich edit.

Solution

C) Generate 100 **random numbers** between 100 and 1000 and store the numbers in an array. Display these numbers in 3 columns in a rich edit.

Solution

Challenge: Try similar examples and change the number of columns.



SOLUTION: EXERCISE 1

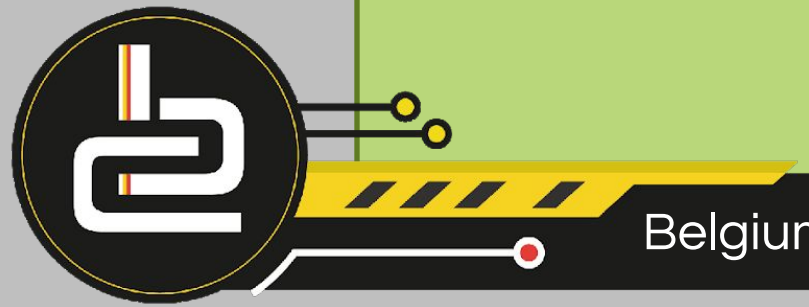
SECTIONS:

1. Declaration of arrays
2. Procedure for 1A
3. Procedure for 1B
4. Procedure for 1C

Declaration of Arrays

```
type
  myarray = array[1..10] of string;
  numarray = array[1..100] of integer;

var
  namearray: myarray;
  numberarray: myarray;
  rannumarray: numarray;
           {See Delphi unit for explanations}
```

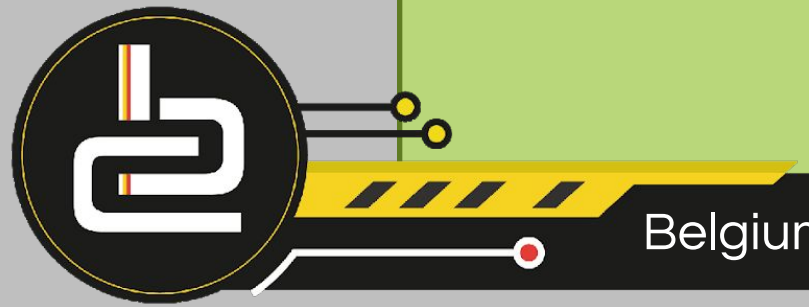


SOLUTION: EXERCISE 1

(CONTINUE)

Question 1.A.

```
procedure TForm1.Ques1AClick(Sender: TObject);
var i: integer;  // declare a loop counter
begin
   for i := 1 to 10 do
    begin
     namearray[i]:= inputbox('Names','Type in a name','');
    end;
   for i := 1 to 10 do
    begin
      redout.lines.add(namearray[i]);
    end;
end;
```

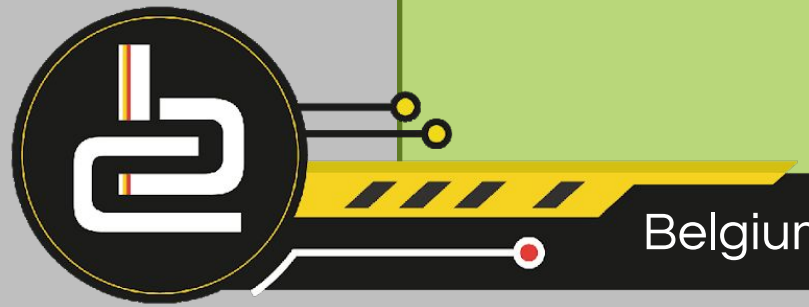


SOLUTION: EXERCISE 1

(CONTINUE)

Question 1.B.

```
procedure TForm1.Ques1BClick(Sender: TObject);
var i: integer;  // declare a loop counter
begin
   for i := 1 to 10 do
    begin
     numberarray[i]:= inputbox('Numbers','Type in a number','0');
     //always try to give a default value so that the program will not crash
    end;
   for i := 1 to 10 do
    begin
      redout.lines.add(numberarray[i]);
    end;
end;
```



SOLUTION: EXERCISE 1

(CONTINUE)

Question 1.C.

```
procedure TForm1.Ques1CClick(Sender: TObject);
var i: integer;  // declare a loop counter
begin
   for i := 1 to 100 do  //using a for loop as we know the number of repitions
    begin
     rannumarray[i]:= random(900)+101;
    end;
    i:= 0;
    redout.Clear;
    redout.Lines.Add('Col1'+#9+'Col2'+#9+'Col3');
   while i <= 96 do
    begin
      inc(i,3);  // increase i by 3 each time you enter the loop
      // will need to convert all integers to string before display
      redout.lines.add(IntToStr(rannumarray[i-2])+#9+IntToStr(rannumarray[i-1])
      +#9+IntToStr(rannumarray[i]));
    end;
      redout.lines.add(IntToStr(rannumarray[100]));
end;
```

SORTING AN ARRAY

Sorting method A:

GLOBAL Declaration:

```
{private declarations of the form}
   arrSurname : Array[1..100] of String;
   Count      : Integer;
```

NOTE

You only need to know ONE sorting method.

The method you use depends on your preference.

```
Procedure TForm1.btnSort_A_ArrayClick(Sender : Tobject);
VAR
   a, b : Integer; Dummy : String;
Begin
  For A := Count downto 2 do
    For B := 1 to (a-1) do
      IF (arrSurname[B] > arrSurname[B+1])
       then
        begin
          Dummy          := arrSurname[B];
          arrSurname[B]  := arrSurname[B+1];
          arrSurname[B+1] := dummy;
        end;//sort
End;
```

Count is used to indicate the maximum number of elements in the array.

When sorting:
- ascending [A..Z] use >
- descending [Z..A] use <

If the array needs to be **DESCENDING** change the > to <

SORTING AN ARRAY (CONTINUE)

Sorting method B:

GLOBAL Declaration:

```
{private declarations of the form}
    arrSurname : Array[1..100] of String;
    Count      : Integer;
```

NOTE

You only need to know ONE sorting method.

The method you use depends on your preference.

```
Procedure TForm1.btnSort_B_ArrayClick(Sender : Tobject);
VAR
   a, b : Integer; Dummy : String;
Begin
  For A := 1 to (Count-1) do
    For B := (A+1) to Count do
      IF (arrSurname[A] > arrSurname[B])
       then
        begin
          Dummy           := arrSurname[A];
          arrSurname[A]   := arrSurname[B];
          arrSurname[B]  := dummy;
        end;//sort
End;
```

Count is used to indicate the maximum number of elements in the array.

EXERCISE 2

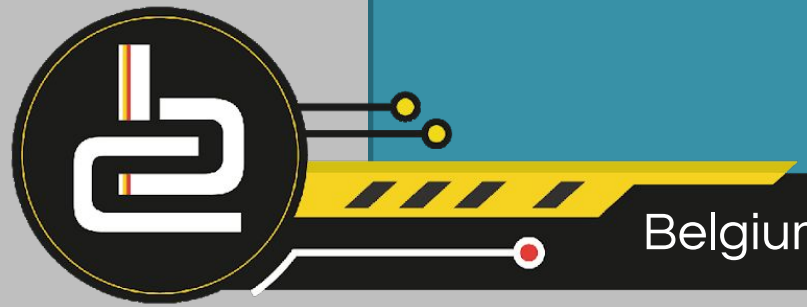
WRITE A DELPHI PROGRAM FOR EACH OF THE FOLLOWING SCENARIOS:

A)Read 10 **names** from the keyboard into an array and **display an alphabetical list**.

Solution

B) Read 10 **numbers** from the keyboard into an array and **display a numerical list.**

Solution



SOLUTION: EXERCISE 2

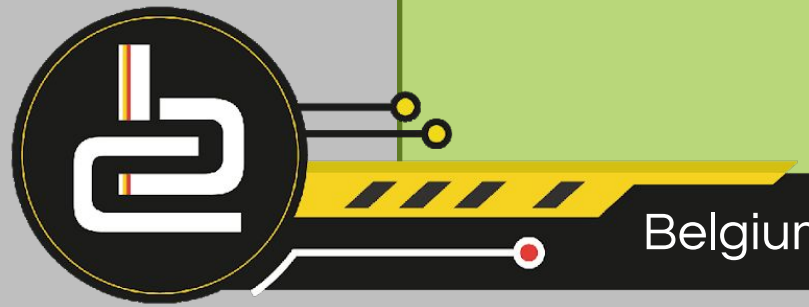
SECTIONS:

1. Declaration of arrays
2. Procedure for 2A
3. Procedure for 2B

Declaration of Arrays

```
type
  myarray = array[1..10] of string;

var
  namearray: myarray;
  numberarray: myarray;
```



SOLUTION: EXERCISE 2

(CONTINUE)

Question 2.A.

```
procedure TForm1.btnnameClick(Sender: TObject);
var i,j : integer;
 temp : string;   // must be same type as array declaration
begin
 for i := 1 to 10 do
begin
  namearray[i]:= inputbox('Names','Type in a name','');
  end;

  for i := 1 to 9 do     // 1 to total - 1
    for j:= 1+i to 10 do  // counter + 1 to total
      if  namearray[i] > namearray[j]
      then
        begin
          temp         := namearray[i];
          namearray[i] := namearray[j];
          namearray[j] := temp;
        end;  // swapping the names around

   redout.Lines.Add('Names in alphabetical order');
        // always display a heading outside the loop
  for i:= 1 to 10 do
    redout.lines.Add(namearray[i]);
  // do not need begin and end as there is only 1 instruction in the loop
End;
```

SOLUTION: EXERCISE 2

(CONTINUE)

Question 2.B.

```
procedure TForm1.btnNumClick(Sender: TObject);
var i,j : integer;
 temp : string;   // must be same type as array declaration
begin
 for i := 1 to 10 do
  // use a for loop when you know how many retitions there are
  begin
  numberarray[i]:= inputbox('Numbers','Type in a number','0');
  end;

  for i := 1 to 9 do     // 1 to total - 1
    for j:= 1+i to 10 do  // counter + 1 to total
      if  numberarray[i] > numberarray[j]
       then
        begin
          temp           := numberarray[i];
          numberarray[i] := numberarray[j];
          numberarray[j] := temp;
        end; // swapping the numbers around

   redout.Lines.Add('Numerical list of numbers');
        // always display a heading outside the loop
  for i:= 1 to 10 do
    redout.lines.Add(numberarray[i]);
        // do not need begin and end as there is only 1 instruction in the loop
end;
```

EXERCISE 3

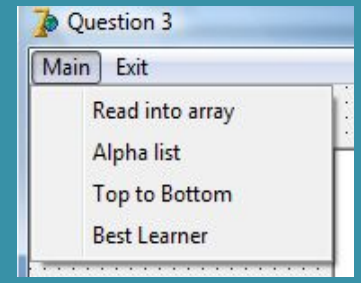
WRITE A DELPHI PROGRAM FOR EACH OF THE FOLLOWING SCENARIOS:

Read **10 names and 10 averages into two arrays**. Give the user the option to display:

A. An **alphabetical list** with corresponding numbers — Solution

B. The list of learners in order from **best to weakest** — Solution

C. The **top learner** according to average — Solution



Challenge: Check to see if there is more than 1 learner with the highest average.

SOLUTION: EXERCISE 3

SECTIONS:

1. Declaration of arrays & Creating two columns
2. Procedure for 3A
3. Procedure for 3B
4. Procedure for 3C

Declaration of Arrays

```
type
  myarray1 = array[1..10] of string;
  myarray2 = array[1..10] of real;

var
  namearray: myarray1;
  avearray: myarray2;
```

Formatting for two column output:

```
procedure TForm1.FormActivate(Sender:TObject);
begin
  redout.paragraph.TabCount := 2;
  redout.Paragraph.Tab[0]   := 80;
  redout.Paragraph.Tab[1]   := 100;
end;
```

For the input of the values into the array refer back to Exercise 1.

SOLUTION: EXERCISE 3

(CONTINUE)

Question 3.A.

```
procedure TForm1.AlphaClick(Sender: TObject);
var i, j : integer;
  temp: string;
  temp2:real;
begin
redout.Clear;
for i := 1 to 9 do
  for j:= 1 +i to 10 do
       if namearray[i] > namearray[j]
         then
           begin
             temp          := namearray[i];
             namearray[i]  := namearray[j];
             namearray[j]  := temp;
             temp2         := avearray[i];
             avearray[i]   := avearray[j];
             avearray[j]   := temp2;
           end;
  redout.Lines.Add('Names'+#9+'Averages'); // headings
  for i := 1 to 10 do
  redout.Lines.Add(namearray[i]+#9+FloatToStrF(avearray[i],ffFixed,5,1));
      // displays the averages correct to 1 decimal place
end;
```

Sample of output:

Names	Averages
a	1.0
b	3.0
c	44.0
d	4.0
e	55.0
f	66.0
g	77.0
h	88.0
i	99.0
j	12.0

SOLUTION: EXERCISE 3

(CONTINUE)

Question 3.B.

```
procedure TForm1.ToptobottomClick(Sender: TObject);
var i, j : integer;
  temp: string;
  temp2:real;
begin
redout.Clear;
for i := 1 to 9 do
  for j:= 1 +i to 10 do
       if avearray[i] < avearray[j]
         then
           begin
             temp := namearray[i];
             namearray[i]:=namearray[j];
             namearray[j]:=temp;
             temp2:= avearray[i];
             avearray[i]:=avearray[j];
             avearray[j]:=temp2;
           end;
  // if you want to sort numbers from small to big,
  // change the sign from < to >
  redout.Lines.Add('Averages'+#9+'Names');  // headings
  for i := 1 to 10 do
  redout.Lines.Add(FloatToStrF(avearray[i],ffFixed,5,1)+#9+namearray[i]);
      // displays the averages correct to 1 decimal place
end;
```

Sample of output:

Averages	Names
99.0	i
88.0	h
77.0	g
66.0	f
55.0	e
44.0	c
12.0	j
4.0	d
3.0	b
1.0	a

SOLUTION: EXERCISE 3

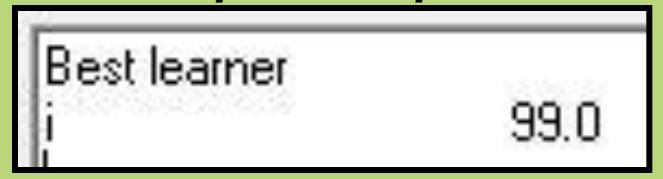
(CONTINUE)

Question 3.C.

```
procedure TForm1.BestLearner2Click(Sender: TObject);
begin
  redout.Clear;
  redout.lines.Add('Best learner');
  Tform1.MethodName(ToptoBottom);
  // calls and uses the existing sorting procedure.
  // Assumes that the averages are unique
  redout.Lines.Add(namearray[1]+#9+FloatToStrF(avearray[1],ffFixed,4,1));
end;
```

Use index 1 because when array is sorted the values in index 1 is the best.

Sample of output:



The call for the method could also be:

TopToBottom.Click;

because it is a menu event of the form.