FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Exercises on Lazarus

Andry Maykol Pinto, Paulo Portugal and José Faria

Informática Industrial

February 27, 2023

Exercises on Lazarus

Andry Maykol Pinto, Paulo Portugal and José Faria

Informática Industrial

February 27, 2023

Contents

Chapter 1

Exercises on Free Pascal: Part 1

A set of exercises are recommended to students. These exercises will cover all fundamentals of the free pascal programming language including:

- data types;
- data manipulation;
- statements.

1.1 New application on Lazarus

For each exercise you will be invited to create a new application. For doing this, please select ”File” then ”New”. After this, a new window will appear in which you should select: ”Project” and ”Application”, see figure 1.1

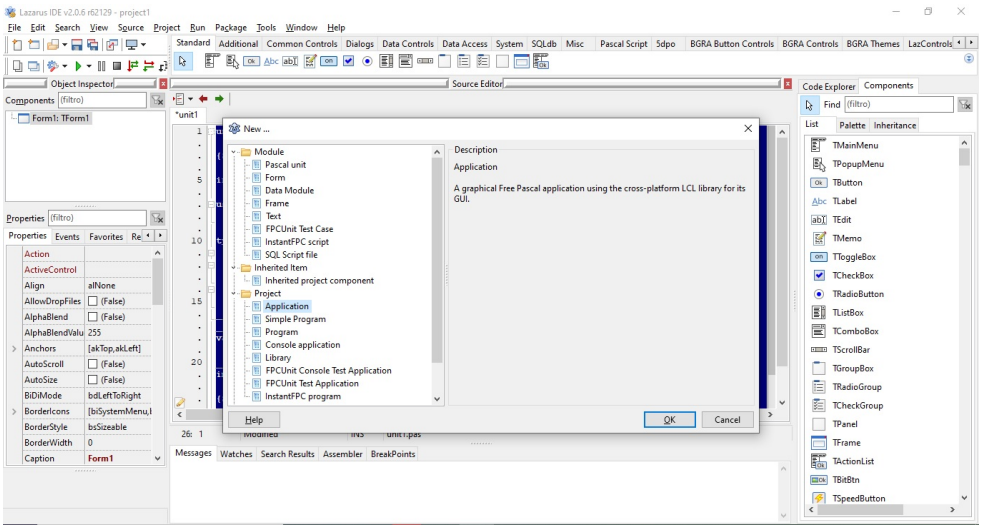

Figure 1.1: Lazarus: new application

After saving the project ("Save All"), you can notice that there is a Project1(.lpi) and Unit1(.pas), and you are invited to change the filenames of both.

In the Unit1.pas you will encounter:

```
1  unit Unit1;
2
3  {$mode objfpc}{$H+}
4
5  interface
6
7  uses
8    Classes, SysUtils, Forms, Controls, Graphics, Dialogs;
9
10 type
11   TForm1 = class(TForm)
12   private
13
14   public
15
16   end;
17
18 var
19   Form1: TForm1;
20
21 implementation
22
23 {$R *.lfm}
24
25 end.
```

code/exampleApplication.txt

1.2 Data Types

The first module will be related to manipulation of variables of different data types.

- integer;
- real;

- boolean;
- strings.

1.2.1 Adding a series of numbers

In this exercise you will add two variables to understand the difference between data types and how you can manipulate them.

Follow the figure 1.2 by adding to the Form1:

- BT_Calculate: TButton;
- Ed_Numb1: TEdit;
- Ed_Numb2: TEdit;
- Label_Numb1: TLabel;
- Label_Numb2: TLabel;
- Label_Result: TLabel;
- Memo_ShowResult: TMemo;

By double-clicking on the Button, you will create automatically the event: "**procedure BT_CalculateClick(Se TObject);**". This event will happen every time that the button is pressed.

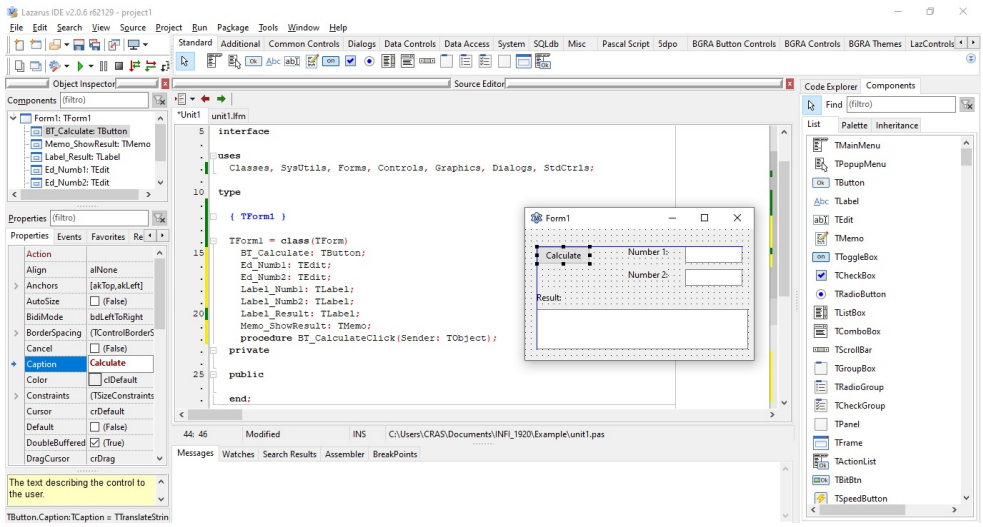

Figure 1.2: Lazarus: new application

Now, you want to code your program that will add two **integer** numbers using Editboxes as inputs and the Memobox as output. Several questions will arrive:

- The structure of a typical ”procedure”.
- Declaration of local variables.
- Assignment operator.
- Adding operator.
- String versus integer.
- ”StrToInt” does what? What about the ”StrToIntDef”.
- Printing the result in the MemoBox.

The code for the procedure is:

```
1  procedure TForm1.BT_CalculateClick(Sender: TObject);
2  var
3    numb1: integer;
4    numb2: integer;
5    res_cal: integer = 0;
6
7  begin
8    numb1:= StrToIntDef(Ed_Numb1.Text, 0);
9    numb2:= StrToIntDef(Ed_Numb2.Text, 0);
10   res_cal := numb1 + numb2;
11
12   Memo_ShowResult.Clear;
13   Memo_ShowResult.Append(IntToStr(res_cal));
14 end;
```

code/ex1Adding.txt

1.2.2 Real and string manipulations

Consider the same example but with :

1. **real** data-type. The ”res_cal” should be divided by 2.5.
2. **string** data-type.
    - by ”adding” two strings: ”INFI” and ” 2020”.

- print the size of ”res_cal” which should be a string.
- verify if ”res_cal” contains the text ”FI”.
- locate the index of ”20” (the location of the first appearance in the string).

1.2.3 String

Read a string and write it out with a space between each adjacent pair of characters. Example:

- Enter: waffle
- Print: w a f f l e

1.2.4 String compare

Read a password P using an EditBox, then clear the MemoBox and repeatedly read password guesses in the other EditBox until the password P is entered. Example:

- Enter password: pineapple
- Password: apple
- Print: wrong!

1.2.5 Counting letters

Write a program that reads a single line of input text, and determines which letter from A to Z appears most frequently in the input text. You should ignore case, considering ”a” and ”A” to be the same letter.

1.2.6 String manipulations

Write a function capitalize(s: string): string that capitalizes the first letter of every word in a string, where words are separated by one or more spaces.

1.2.7 Concatenate and print different data types

Declare two string variables: Name and Surname. Declare an integer variable: Age. Use three EditBoxes: to assign to Name and Surname your name and surname and, your age to Age. Use a ”ShowMessage” to display the text ”My name is $name $surname. I am $age years old.” after the button is pressed.

1.2.8 Relational expressions and IF statement

Read two integers X and Y and report which is greater (or that they are equal).
Example:

- Enter X: 4
- Enter Y: 6
- Print: Y is greater

1.2.9 Quotient

Read two integers X and Y, and print their quotient if X is exactly divisible by Y, or ”indivisible” otherwise. Example:

- Enter X: 10
- Enter Y: 3
- indivisible

1.2.10 Arithmetic expression

Read two reals X and Y and print the result of:

- (X / Y)
- (X * Y)
- (X div Y)
- (X mod Y)

1.2.11 Arithmetic expression

Write a program that prompts the user to enter the radius of a circle (integer) and outputs the circumference and the area. Both amounts should be displayed to two decimal places

1.2.12 Boolean expressions

Read two booleans X and Y and print the result of:

- (X AND Y)
- (X OR Y)
- (NOT X)
- (X XOR Y)

1.3 Statements

1.3.1 IF and ELSE

Read three numbers and write whether or not they can form a triangle. Note - to form the sides of a triangle, each value must be less than the sum of the other two. You should categorize the triangle into: ”EQUILÁTERO”, ”ISÓSCELES” or ”ESCALENO”.

```
1  if (A < B AND (B < A + C) AND (C < A + B) then
2          begin
3             { ('Os numeros formam os lados de um triangulo
                 '); }
4             if (A = B) and (B = C) then
5                { ('equilatero ') }
6             else
7               if (A <> B) and (A <> C) and (B <> C) then
8                  { ('escaleno') }
9               else
10                { ('isosceles'); }
11         end
12 else
13 {  nao triangulo }
```

code/ex3IF.txt

1.3.2 CASE and Sub-range

Read a year from 1980 to 2020 using the EditBox. Print the decade that it was in. Example:

- Year: 1995
- Print: 1995 was in the 1990s

1.3.3 FOR

Ask the user for a number N, then write the word ”strawberries” N times. Example:

- How many strawberries: 3
- strawberries
- strawberries
- strawberries

1.3.4 Twin FOR

Read a number N, then print an N x N triangle of asterisks.

- Enter N: 6
- *
- **
- ***
- ...
- ******

1.3.5 FOR and Random

Simulate coin flips, printing an ”H” each time the coin falls on heads and ”T” each time it falls on tails. Once you reach the third H, stop and write how many coins were flipped. Example:

- HTTTHTH
- 7 flips
- ===

1.3.6 While

Write the same loop using while instead of for.

```
1  procedure TForm1.BT_CalculateClick(Sender: TObject);
2  var
3    numb1: integer;
4    numb2: integer;
5    res_cal: integer = 0;
6    i: integer;
7
8  begin
9    Memo_ShowResult.Clear;
10
11   for i := 1 to 5 do
12       Memo_ShowResult.Append(IntToStr(i));
13 end;
```

code/ex2While.txt

1.3.7 Nasty while

Update the code of the previous exercise. The ”res_cal” should be printed in the MemoBox while ”numb2 < 10 ’. Consider the value of ”res_cal” as ”numb1 * i”. What happened?

Chapter 2

Exercises on Free Pascal: Part 2

These exercises will cover the following topics: record data type; procedures; functions and, passing arguments by value or reference.

2.0.1 Creating a structure

Follow the figure 2.1 to create a new application in Lazarus:

- BT_AddOrder: TButton;
- CB_PartType: TComboBox;
- CB_Destination: TComboBox;
- Ed_Numb1: TEdit;
- Label_Numb1: TLabel;

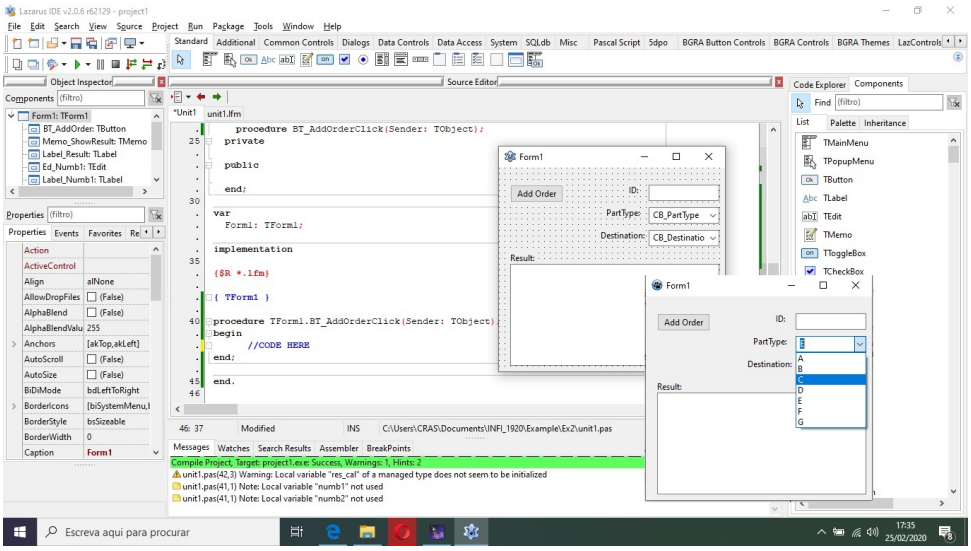

Figure 2.1: Lazarus: example for record data types

- Label_PartType: TLabel;
- Label_Destination: TLabel;
- Label_Result: TLabel;
- Memo_ShowResult: TMemo;

This exercise introduces the ”TComboBox”. Please explore this component, in particular, the ”items” and ”itemIdex”. Note that you can access the item that was selected by the user with both properties: ”CB_PartType.items[CB_PartType.ItemIndex]”. We want to create a structure called ”TOrder” which will be formed by the following members:

- id : integer;
- fromType: tPartType;
- destination : tRouter;

2.0.1.1 Data types: record, enumerated and sub-range

By pressing the TButton, a new instance of ”TOrder” should be locally created and printed in the TMemoBox. Create a record data type (”TOrder”) using the:

- ”id” variable that can assume an integer value (no repetition should be allowed).
- ”tPartType” variable that can assume a value of ”A”, ”B”, ... or ”G”;
- ”tRouter” variable that can assume a value within the range 0 .. 7;

2.0.1.2 With statement

Use the ”With” statement for printing each order that is created when the TButton is pressed.

Try yourself and then compare with the following code. Analyze and discuss the differences.

```
1 unit Unit1;
2
3 {$mode objfpc}{$H+}
4
5 interface
6
7 uses
8   Classes, SysUtils, Forms, Controls, Graphics, Dialogs,
      StdCtrls;
```

```
9
10 type
11    tRouter = 0..7;
12    tPartType = (PartA = 0, PartB, PartC, PartD, PartE,
          PartF, PartG);
13    TOrder = record
14      id            : integer;
15      PartType      : tPartType;
16      Route         : tRouter;
17    end;
18
19   { TForm1 }
20
21   TForm1 = class(TForm)
22     BT_AddOrder: TButton;
23     CB_PartType: TComboBox;
24     CB_Destination: TComboBox;
25     Ed_Numb1: TEdit;
26     Label_Numb1: TLabel;
27     Label_PartType: TLabel;
28     Label_Destination: TLabel;
29     Label_Result: TLabel;
30     Memo_ShowResult: TMemo;
31     procedure BT_AddOrderClick(Sender: TObject);
32   private
33
34   public
35
36   end;
37
38 var
39  Form1: TForm1;
40
41 implementation
42
43 {$R *.lfm}
44
45 { TForm1 }
46
```

```
47 procedure TForm1.BT_AddOrderClick(Sender: TObject);
48 var
49  localOrder : TOrder;
50 begin
51   Randomize;
52   localOrder.id       := Random(1000);
53   localOrder.PartType:= tPartType(CB_PartType.ItemIndex);
54   localOrder.Route    :=
       StrToInt(CB_Destination.Items[CB_Destination.ItemIndex]);
55
56   With  localOrder do
57   Begin
58         Memo_ShowResult.Clear;
59         Memo_ShowResult.Append('ID: '+IntToStr(id));
60         Memo_ShowResult.Append('PartType:
             '+IntToStr(Integer(PartType)));
61         Memo_ShowResult.Append('Route: '+IntToStr(Route));
62   End;
63 end;
64
65 end.
```

code/exampleAppRecord.txt

2.0.1.3 Procedure

Declare a procedure that will be responsible for printing an instance of the type TOrder in a MemoBox. Then, this procedure will have two arguments: TOrder and a TMemo.

2.0.1.4 Arguments by Value and by Reference

Update the procedure created in the last exercise. This new procedure should be able to print all information and copy the ID of an instance TOrder to an argument. Then, this procedure will have three arguments: TOrder, a TMemo and an integer variable.

2.0.1.5 Function and dynamic array

Develop an function that adds the ID members of two instances TOrder and returns the instance TOrder with a lower Route destination. This function will have three arguments: two TOrder and one integer.
Create other TButton to call this function. Moreover, you must implement a dynamic array of TOrder that is incremented by a new order when the BT_AddOrder is pressed.

2.0.2 Implementation clues

The code below can assist you during the resolution of exercises presented in section 2.0.1.

```
unit Unit1;

{$mode objfpc}{$H+}

interface

uses
  Classes, SysUtils, Forms, Controls, Graphics, Dialogs,
    StdCtrls;

type
   tRouter = 0..7;
   tPartType = (PartA = 0, PartB, PartC, PartD, PartE,
     PartF, PartG);

   TOrder = record
     id            : integer;
     PartType      : tPartType;
     Route         : tRouter;
   end;

   tarray_orders = array of TOrder;

  { Ts }

  Ts = class(TForm)
    BT_AddOrder: TButton;
    Bt_MinOrder: TButton;
    CB_PartType: TComboBox;
    CB_Destination: TComboBox;
    Ed_Numb1: TEdit;
    Label_Numb1: TLabel;
    Label_PartType: TLabel;
    Label_Destination: TLabel;
    Label_Result: TLabel;
    Memo_ShowResult: TMemo;
```

```
35      procedure BT_AddOrderClick(Sender: TObject);
36      procedure Bt_MinOrderClick(Sender: TObject);
37      procedure FormCreate(Sender: TObject);
38    private
39
40    public
41       array_orders : tarray_orders;
42       numb_orders  : integer;
43    end;
44
45 var
46  s: Ts;
47
48
49
50 implementation
51
52 {$R *.lfm}
53
54 { Ts }
55
56 procedure Ts.FormCreate(Sender: TObject);
57 begin
58    numb_orders := 0;
59 end;
60
61
62 procedure printOrder(order: TOrder; memo:TMemo);
63 begin
64    With  order do
65    Begin
66          memo.Clear;
67          memo.Append('ID: '+IntToStr(id));
68          memo.Append('PartType:
               '+IntToStr(Integer(PartType)));
69          memo.Append('Route: '+IntToStr(Route));
70    End;
71 end;
72
```

```
73
74 procedure printOrder_AndCopy(order: TOrder; memo:TMemo;
     var out_id: integer);
75 begin
76   With  order do
77   Begin
78         memo.Clear;
79         memo.Append('ID: '+IntToStr(id));
80         memo.Append('PartType:
             '+IntToStr(Integer(PartType)));
81         memo.Append('Route: '+IntToStr(Route));
82         out_id := id;
83   End;
84 end;
85
86 function minRoute_Order(orderA, orderB: TOrder; var
     addValue: integer): TOrder;
87 begin
88      addValue:= 0;
89      addValue:= orderA.id + orderB.id;
90      ShowMessage(IntToStr(orderA.id)+'
          '+IntToStr(orderB.id)+' r='+IntToStr(addValue));
91      if ( orderA.Route > orderB.Route) then
92      begin
93          minRoute_Order := orderB;
94      end
95      else begin
96             minRoute_Order := orderA;
97      end
98 end;
99
100 procedure Ts.BT_AddOrderClick(Sender: TObject);
101 var
102  localOrder : TOrder;
103  c_idx      : integer = -1;
104 begin
105   Randomize;
106   localOrder.id       := Random(1000);
107   localOrder.PartType:= tPartType(CB_PartType.ItemIndex);
```

```
108   localOrder.Route    :=
          StrToInt(CB_Destination.Items[CB_Destination.ItemIndex]);
109   //printOrder_AndCopy( localOrder, Memo1, c_idx);
110   //Memo_ShowResult.Append('Variable received from
          procedure:' + IntToStr(c_idx));
111
112
113   { Resize the length of the dynamic array }
114   SetLength(array_orders, numb_orders+1);
115   { Copy local variable to global variable }
116   array_orders[numb_orders] := localOrder;
117   { Show data }
118   printOrder(array_orders[numb_orders], Memo_ShowResult);
119   { Increment number of elements in array }
120   inc(numb_orders);
121   Memo_ShowResult.Append('numbOrders:' +
          IntToStr(numb_orders));
122 end;
123
124 procedure Ts.Bt_MinOrderClick(Sender: TObject);
125 var
126   addIDS : integer;
127   auxOrder : TOrder;
128 begin
129     auxOrder :=  minRoute_Order(array_orders[0],
            array_orders[1], addIDS);
130     printOrder(auxOrder, Memo_ShowResult);
131     Memo_ShowResult.Append('Variable received from
            function:' + IntToStr(addIDS));
132 end;
133
134
135
136
137 end.
```

code/solutionsEx2.txt

2.0.3 TTimers

Lazarus works with ”events”. For a procedure or function be capable of running with a specific frequency, you must use a TTimer (that is available on the TAB ”System”). Please add one TTimer and create the event **onTimer** that will present a ShowMessage with the text ”This is an event”. Modify the property ”Interval” and analyze the change in the program behavior.