

Universidade Federal de Viçosa
Departamento de Engenharia Agrícola

ENG 390

Programação Aplicada à Agricultura

Aulas Práticas

Prof. Evandro de Castro Melo

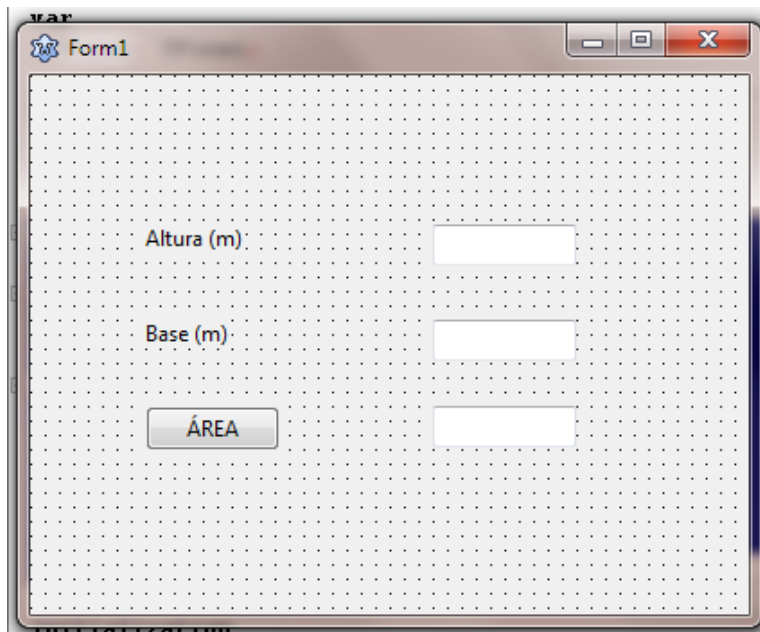
Viçosa, Junho 2010

ENG 390

Aula Prática 01

1. Faça um projeto em Lazarus para calcular a área de um triângulo.

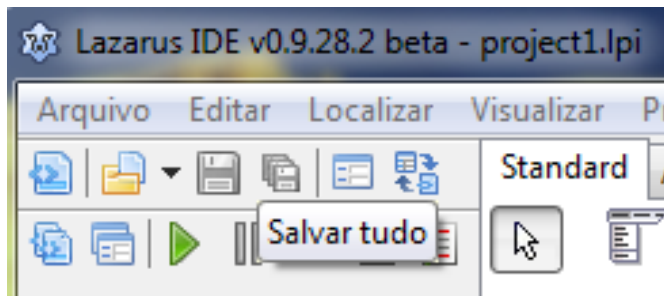
- Aplicação dos componentes: **Label, Edit, Button**
- Sugestão para o formulário **Form1**:



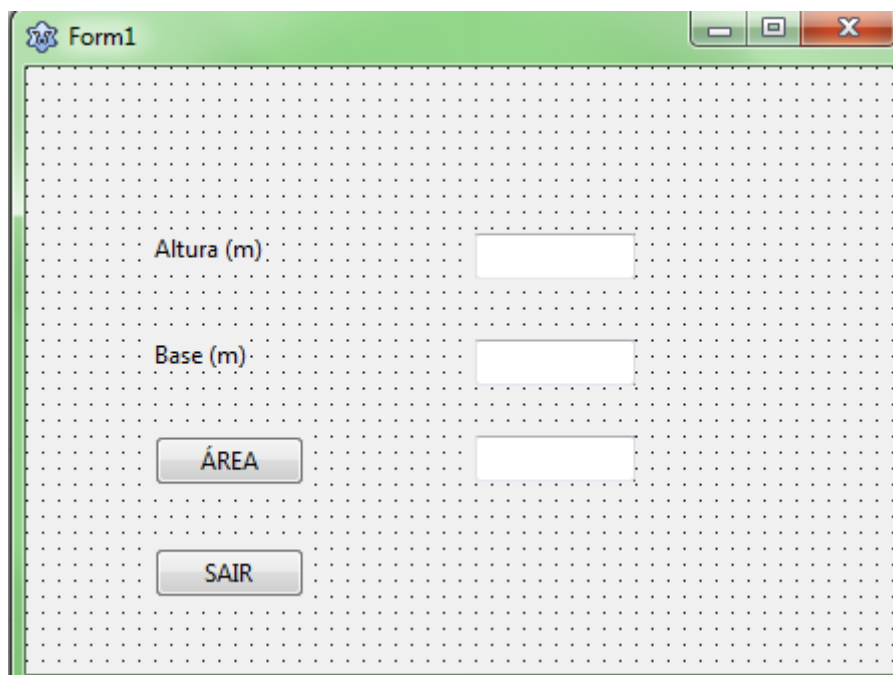
- Escrevendo o código vinculado ao botão **ÁREA**:

```
35 . procedure TForm1.Button1Click(Sender: TObject);  
36 .  
37 .  
38 . var  
39 .     altura, base, area: double;  
40 .  
41 . begin  
42 .     altura:=StrToFloat(Edit1.Text);  
43 .     base:=StrToFloat(Edit2.text);  
44 .     area:=base*altura/2;  
45 .  
46 .     Edit3.Text:=FloatToStr(area);  
47 . end;
```

- d. Salvando o projeto: vá a Barra de Botões e escolha o ícone *Salvar Tudo*. Cada projeto deverá ser guardado em uma pasta independente no disco do seu computador. Serão gravados a *Unit1.pas* e o *Project1.lpi*, cujos nomes podem ser mudados, porém devem ser diferentes.



- 2. No formulário do item 1b, coloque um botão SAIR para encerrar o programa.**

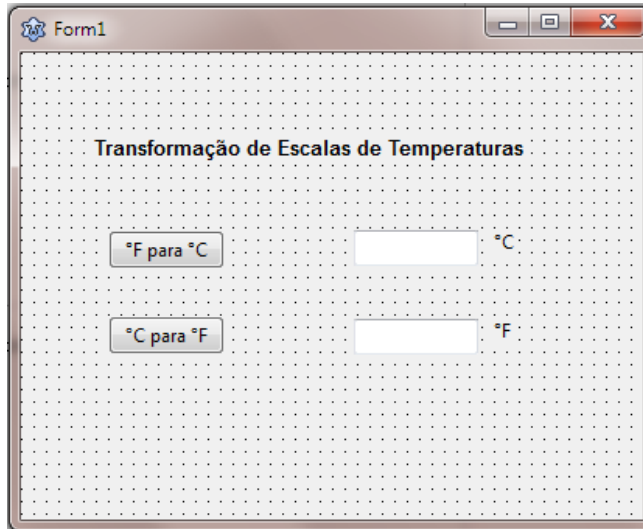


ENG 390

Aula Prática 02

1. Faça um projeto em Lazarus para transformar escalas de temperaturas em °C e em °F, de acordo com o formulário sugerido.

a. Sugestão para o formulário Form1:



b. Escrevendo o código vinculado aos dois botões:

```
. procedure TForm1.BCparaFClick(Sender: TObject);  
. var  
40     F, C: double;  
. begin  
.     C:=StrtoFloat (EgrausC.Text);  
.     F:=1.8*C+32;  
.     EgrausF.Text:=FloatToStr (F);  
45 end;  
.   
.   
. procedure TForm1.BFparaCClick(Sender: TObject);  
. var  
50     F, C: double;  
51 begin  
.     F:=StrtoFloat (EgrausF.Text);  
.     C:=5/9*(F-32);  
.     EgrausC.Text:=FloatToStr (C);  
55 end;
```

c. Colocar um botão SAIR e outro botão LIMPAR no formulário do item a.

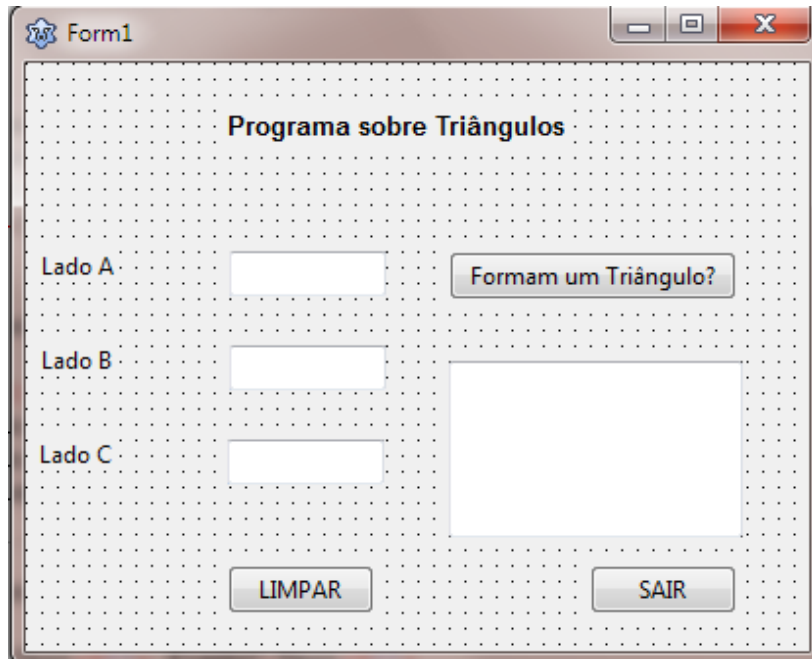
2. Desenvolver um projeto que calcule o volume de um cilindro. Valor de $\pi=3,14159$.

ENG 390

Aula Prática 03

Faça um projeto em Lazarus que leia os três lados de um triângulo e verifique se realmente eles formam um triângulo. Caso afirmativo, classificar o tipo de triângulo.

- Aplicação dos componentes: **Label, Edit, Button, Memo**
- Sugestão para o formulário **Form1**:



TMemo - este componente é semelhante ao **Edit**, com a diferença de permitir a editoração de várias linhas. Ele pode ser usado, por exemplo, para mostrar os resultados. Depois de inserir o componente **Memo** no formulário Lazarus, pode-se alterar a sua propriedade **Name** para *Mresultado*. A propriedade que controla o que será escrito no Memo é a **Lines**. Para iniciar este componente vazio, ou seja, sem nada escrito ao iniciar um programa, deve-se clicar em (...) ao lado de *Lines (TStrings)* e apagar o texto contido no **Memo**. Além disso, é importante que o usuário não consiga alterar o texto apresentado. Então, deve-se alterar a propriedade **ReadOnly** do **Memo** para *True*. Finalmente, é interessante adicionar barras de rolagem ao Memo, alterando a propriedade **ScrollBars** para *ssBoth*.

Left	216
→ Lines	(TStrings) ...
MaxLength	0

ReadOnly	False
ScrollBars	False
ShowHint	True

→ ScrollBars	ssBoth
--------------	--------

c. Escrevendo o código vinculado aos botões do **Form1** :

```
. procedure TForm1.BtrianguloClick(Sender: TObject);  
. var  
45   a, b, c: double;  
. begin  
.   a:=StrToFloat(EladoA.Text);  
.   b:=StrToFloat(EladoB.Text);  
.   c:=StrToFloat(EladoC.Text);  
50   if ((A<(B+C)) and (B<(A+C)) and (C<(A+B)))  
.     then  
.       if ((A=B) and (B=C))  
.         then MResultado.Lines.Add('Triângulo Equilátero')  
.         else  
55           if ((A=B) or (B=C) or (C=A))  
.             then MResultado.Lines.Add('Triângulo Isósceles')  
.             else MResultado.Lines.Add('Triângulo Escaleno')  
.         else MResultado.Lines.Add('As medidas não formam Triângulo');  
.     end;  
60  
. procedure TForm1.BlimparClick(Sender: TObject);  
. begin  
.   EladoA.Text:=''; EladoB.Text:=''; EladoC.Text:='';  
.   MResultado.Lines.Clear;  
65 end;  
.   
. procedure TForm1.BsairClick(Sender: TObject);  
. begin  
69   close;  
70 end;
```

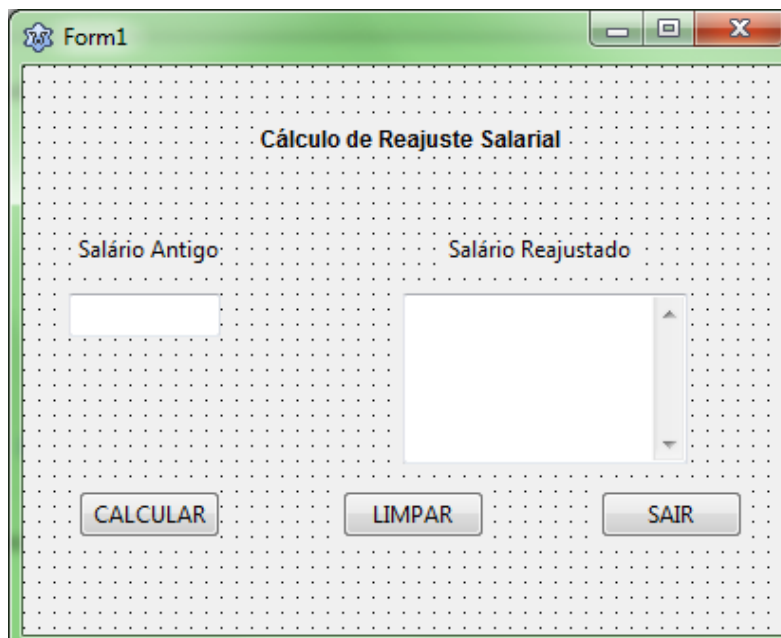
ENG 390

Aula Prática 04

Faça um projeto em Lazarus para reajuste salarial:

Para um exemplo da utilização desta estrutura considere o seguinte problema: "Elaborar um programa que efetue o cálculo do reajuste de salário de um funcionário. Considere que o funcionário deverá receber um reajuste de 15% caso seu salário seja menor que 500. Se o salário for maior ou igual a 500, mas menor ou igual a 1000, seu reajuste será de 10%; caso seja ainda maior que 1000, o reajuste deverá ser de 5%". Veja o algoritmo, diagrama de blocos e a codificação em português estruturado.

- Aplicação dos componentes: **Label, Edit, Button, Memo**
- Sugestão para o formulário **Form1**:



- Escrevendo o código vinculado aos botões do **Form1** :

```
.
40 procedure TForm1.BcalcularClick(Sender: TObject);
.
.   var
.     nsal, sal: double;
.
.   begin
.     sal:=StrToFloat(Edit1.Text);
45     if sal<500
.       then nsal:=sal*1.15
.       else
.         if sal<=1000
.           then nsal:=sal*1.10
50          else nsal:=sal*1.05;
.     Memo1.Lines.Add ('Novo Salário (R$): ' + FloatToStr(nsal));
.   end;
```

ENG 390

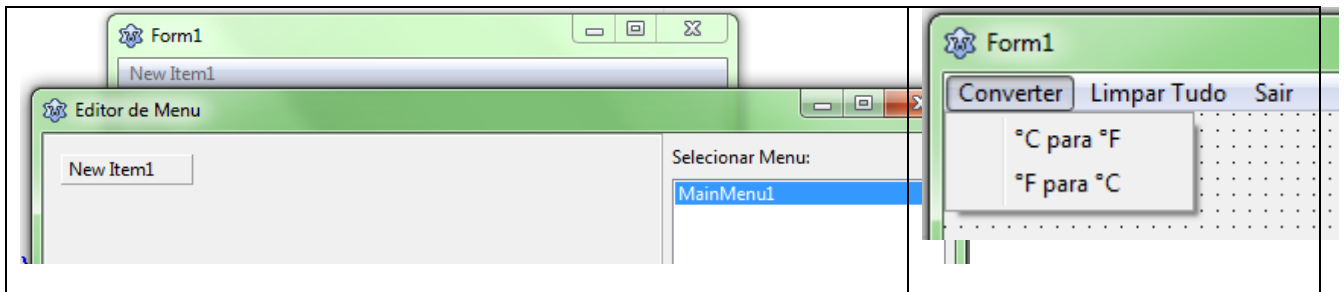
Aula Prática 05

Aplicar o componente **MainMenu** ao projeto de transformação de temperaturas ($^{\circ}\text{C} \rightarrow ^{\circ}\text{F}$)

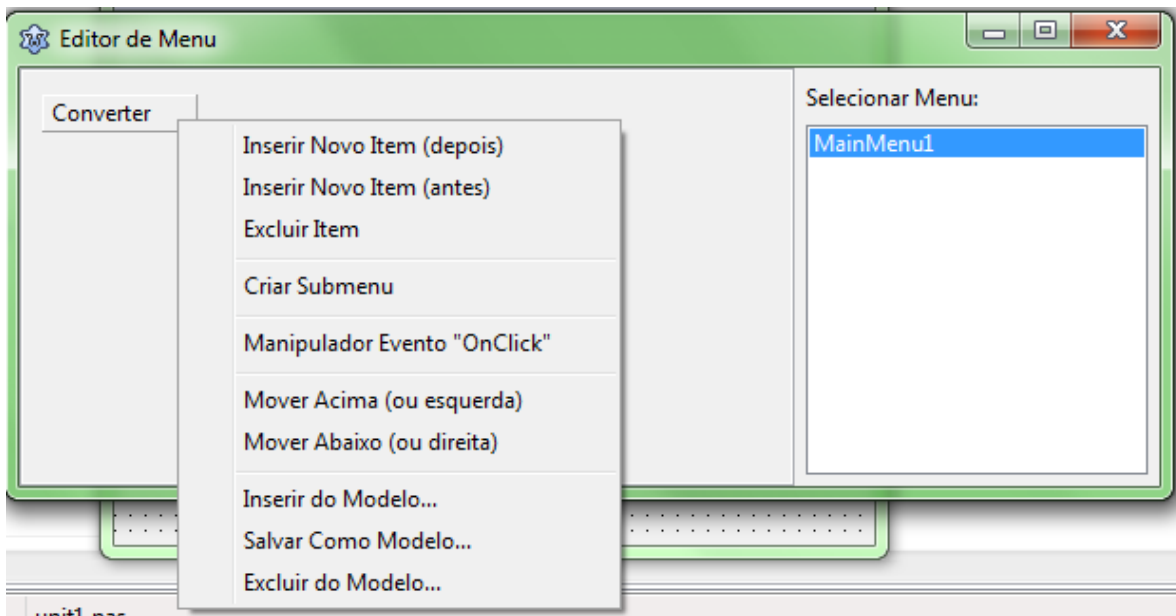
a. Aplicação do componente **MainMenu**



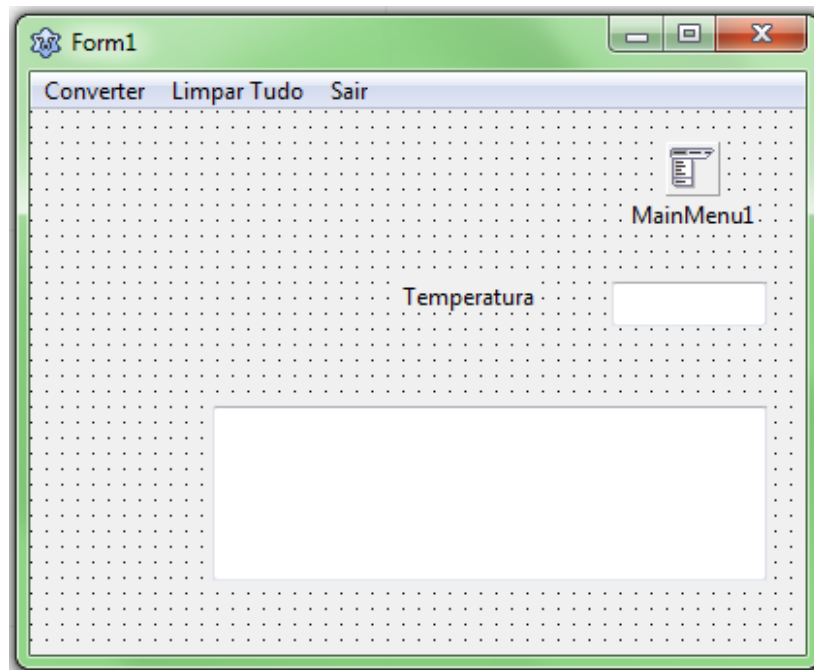
TMainMenu - Este componente permite criar barras de menus contendo vários níveis (como o menu principal dos aplicativos Windows). Para usá-lo, basta inserir a figura que o representa no Formulário do seu Projeto, localizando-o, por exemplo, no canto superior direito. Para inserir as opções necessárias ao seu projeto, clique duas vezes no símbolo do *MainMenu*. Aparecerá o *Editor de Menu*. Para alterar seu nome, de acordo com o menu proposto, deve-se alterar a sua propriedade *Caption* no *Inspetor de Objetos*.



Para criar outros tópicos, pressiona-se o botão direito do mouse em cima do item e aparecerá a seguinte janela, com as opções para edição de itens no seu menu:



b. Sugestão para o formulário **Form1**:



c. Escrevendo o código vinculado às opções do **MainMenu** no **Form1**:

```
50 procedure TForm1.CparaFClick(Sender: TObject);
.   var
.     C,F: double;
.
.   begin
55     C:=StrToFloat (ETemp.Text);
.     F:=1.8*C + 32;
.     MResultado.Lines.Add(' Temperatura °C = '+FloatToStr (C));
.     MResultado.Lines.Add(' Temperatura °F = '+FloatToStr (F));
.     MResultado.Lines.Add(' ');
60     end;
.
.   procedure TForm1.FparaCClick(Sender: TObject);
.   var
.     C,F: double;
65
.   begin
.     F:=StrToFloat (ETemp.Text);
.     C:=5/9*(F-32);
.     MResultado.Lines.Add(' Temperatura °F = '+FloatToStr (F));
70     MResultado.Lines.Add(' Temperatura °C = '+FloatToStr (C));
.     MResultado.Lines.Add(' ');
.     end;
.
.   procedure TForm1.LimparClick(Sender: TObject);
75   begin
.     ETemp.text:='';
.     MResultado.Lines.Clear;
.     end;
.
.   procedure TForm1.SairClick(Sender: TObject);
80   begin
.     close;
.     end;
```

ENG 390


Aula Prática 06

Fazer um projeto no ambiente Lazarus para calcular o índice de massa corporal (IMC) e classificar a pessoa segundo a tabela da O.M.S.:

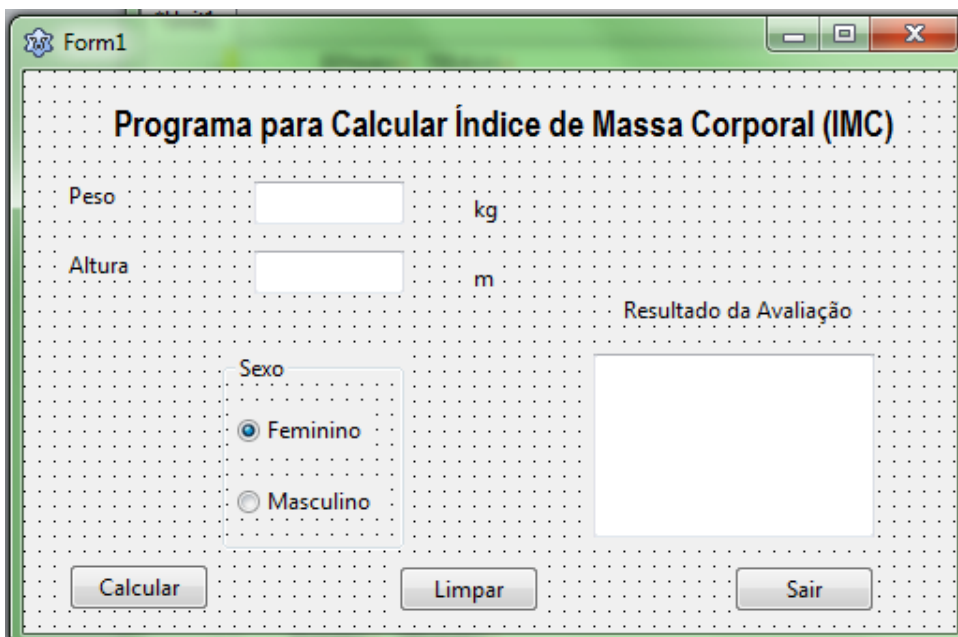
Condição	IMC em Mulheres	IMC em Homens
abaixo do peso	< 19,1	< 20,7
no peso normal	19,1 - 25,8	20,7 - 26,4
marginalmente acima do peso	25,8 - 27,3	26,4 - 27,8
acima do peso ideal	27,3 - 31,1	27,8 - 32,3
obeso	> 31,1	> 32,3

$$IMC = \text{peso} / \text{altura}^2 \text{ (peso em kg e altura em m)}$$

a. Aplicação do componente **RadioGroup**:

 **TRadioGroup** - Apresenta vários **RadioButtons** agrupados, sendo que apenas um deles poderá ser selecionado por vez. O **RadioGroup** conterá as opções do sexo da pessoa. No **Inspetor de Objetos**, sua propriedade **Name** deverá ser alterada para **RgSexo**, sua propriedade **Caption** para **Sexo** e a propriedade **Items** deverá conter as opções **Feminino** e **Masculino** (uma em cada linha do editor de itens). Cada opção conterá um índice, iniciando-se do zero. Quando o sexo for selecionado, o valor de seu índice será atribuído à propriedade **Itemindex** do **RgSexo**. O programa deve ser iniciado com uma opção já selecionada. Para tal, basta atribuir à propriedade **Itemindex** do **Rgsexo** o valor **0** (índice da primeira opção). Quando o usuário alterar a seleção a propriedade **ItemIndex** será modificada automaticamente.

b. Sugestão para o formulário **Form1**:



c. Escrevendo o código vinculado às opções no **Form1**:

```
. procedure TForm1.BCalcularClick(Sender: TObject);
. var
.   peso, alt, imc: double;
50 begin
.   peso:=StrToFloat(EPeso.Text);
.   alt :=StrToFloat(EAltura.Text);
.   imc :=peso/sqr(alt);
.   if RgSexo.ItemIndex = 0           // sexo feminino
55   then
.     if imc<19.1
.     then MResultado.Lines.Add('Abaixo do Peso')
.     else
.       if imc<25.8
60       then MResultado.Lines.Add('Peso Normal')
.       else
.         if imc<27.3
.         then MResultado.Lines.Add('Marginalmente Acima do Peso')
.         else
65         if imc<31.1
.         then MResultado.Lines.Add('Acima do Peso Ideal')
.         else MResultado.Lines.Add('OBESO!')
.     else // sexo masculino
.       if imc<20.7
70       then MResultado.Lines.Add('Abaixo do Peso')
.       else
.         if imc<26.4
.         then MResultado.Lines.Add('Peso Normal')
.         else
75         if imc<27.8
.         then MResultado.Lines.Add('Marginalmente Acima do Peso')
.         else
.           if imc<32.3
.           then MResultado.Lines.Add('Acima do Peso Ideal')
80         else MResultado.Lines.Add('OBESO!')
.   end;

. procedure TForm1.BLimparClick(Sender: TObject);
. begin
85   EPeso.Text:='';
.   EAltura.Text:='';
.   MResultado.Lines.Clear;
. end;
```

ENG 390

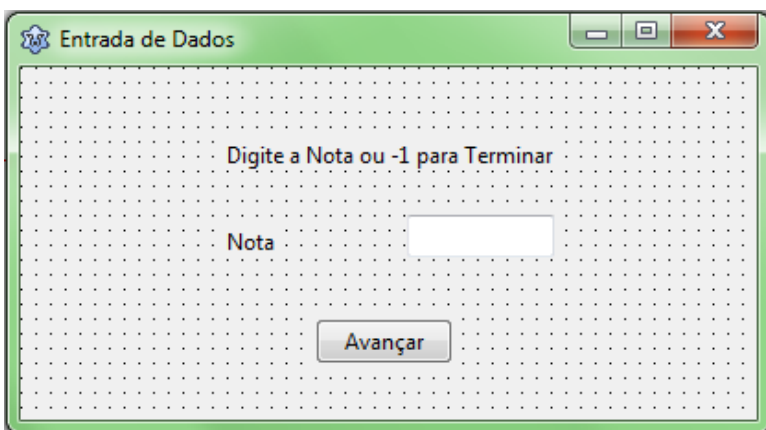
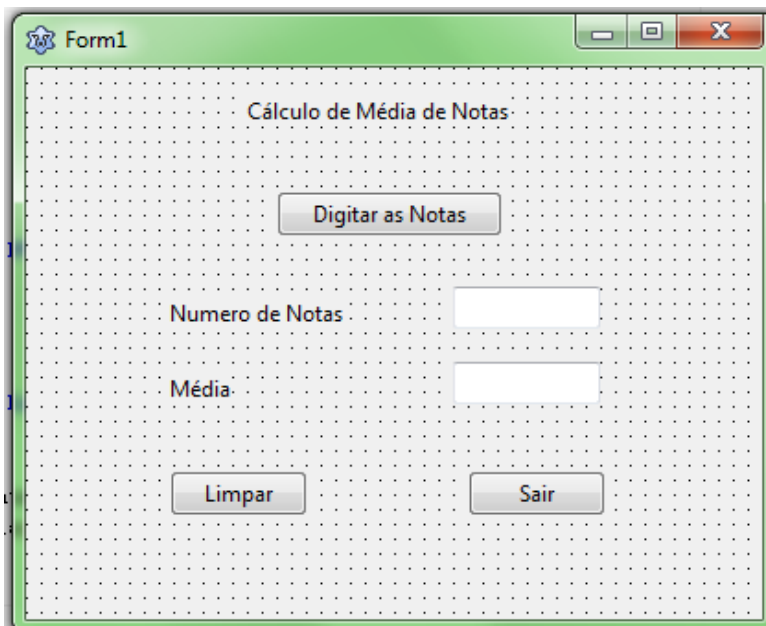
Aula Prática 07

Fazer um projeto em Lazarus que calcula a Média das Provas de Cálculo II, cujo algoritmo já foi apresentado e discutido em aula teórica.

a. Uso de duas **Unit** no mesmo Projeto:

Este projeto é constituído de duas unidades: Unit1 e Unit2 e portanto de dois formulários: **Form1** e **Form2**. A propriedade **Name** do **Form2** será modificada para **FEntrada**. Para que ambas as unidades possam se comunicar dentro do projeto, devemos empregar o comando **Uses**, fazendo, dentro de cada uma, referencia à outra unidade. A variável **Nota** será usada em ambas as Unidades, portanto, ela deve ser declarada como variável pública abaixo de **Public { Public declarations }** na Unidade **Unit1**. O formulário **FEntrada** será chamado dentro do formulário **Form1** empregando-se o comando **NomeFormulario.ShowModal**, ou seja, **FEntrada.ShowModal**.

b. Sugestão para os formulários **Form1** e **Form2**:



c. Escrevendo o código vinculado ao **Form1**:

```
. private
.   { private declarations }
. public
30   { public declarations }
.   nota: double;
. end;

. implementation
.   Uses Unit2; //aqui avisamos que será usada a Unit2
.   { TForm1 }
40
. procedure TForm1.BdigitarClick(Sender: TObject);
.   const
.     sentinela=-1;
.   var
45     numNotas,soma,media:double;
.   begin
.     numNotas:=0;
.     soma:=0;
49     FEntrada.showmodal; // Ler a primeira nota
50     while nota <> SENTINELA do
.       begin
.         numNotas:=numNotas+1;
.         soma:=soma+nota;
.         FEntrada.showmodal; // Ler as seguintes notas
55       end;
.     media:=soma/numNotas;
.     EnumNotas.Text:=(FloatToStr(numNotas));
.     Emedia.Text:=(FloatToStr(media));
.
60   end;

. procedure TForm1.BlimparClick(Sender: TObject);
.   begin
.     Emedia.Text:='';
65     ENumNotas.Text:='';
.   end;

. procedure TForm1.BSairClick(Sender: TObject);
.   begin
70     close;
.   end;
```

d. Escrevendo o código vinculado ao **Form2**:

```
. uses
.   Classes, SysUtils, FileUtil, LResources, Forms, Controls, Graphics, Dialogs,
.   StdCtrls, Unit1;
10

. procedure TForm2.Button1Click(Sender: TObject);
35 begin
.   Form1.nota:=StrToFloat(edit1.text);
.   edit1.text:='';
.   close;
. end;
```

ENG 390

Aula Prática 08

Queda de pressão em camada de grãos – Equação de Shedd – *Usando esta equação, fazer um projeto em Lazarus com opção para arroz, milho, soja e trigo. Os dados de entrada são os parâmetros Q e h_g.*

a. Aplicação do comando **CASE**:

$$\Delta P_g = (a \cdot Q^2 \cdot h_g) / \ln(1 + b \cdot Q)$$

ΔP_g = queda de pressão devido à resistência do produto, mmCA;
 Q = fluxo de ar, m³/min.m²;
 h_g = altura da massa, m; e
 a, b - constantes que dependem do produto (Tabela 1).

Produto	a	b
Arroz em casca	0,722	0,197
Aveia	0,718	0,243
Café Pergaminho	Usar os valores para soja	
Café Coco	0,017	3,900
Milho	0,583	0,512
Soja	0,333	0,302
Trigo	0,825	0,164

b. Sugestão para o formulário **Form1**

The screenshot shows a Lazarus form window titled "Form1". The form is titled "Queda de Pressão em Grãos" and is set against a grid background. It features a radio button group labeled "Tipo de Grão" with options for "Arroz", "Milho", "Soja", and "Trigo". To the right, there are three input fields: "Fluxo de Ar (m3/min.m2)", "Altura da massa de grãos(m)", and "Queda de Pressão (mm CA)". At the bottom, there are two buttons: "Limpar" and "Sair".

c. Escrevendo o código vinculado ao **Form1**:

```
.
.
. procedure TForm1.Button1Click(Sender: TObject);
. var
50   tg: integer;
.   Q, hg, Pg: double;
.   a, b: double;
. begin
.   Q := StrToFloat(Edit1.Text);
.   hg:= StrToFloat(Edit2.Text);
55   tg:= RgTipo.ItemIndex;
.   CASE tg of
.     0: begin
.         a:=0.722; b:= 0.197; //arroz
60     end;
.     1: begin
.         a:= 0.583; b:=0.512; //milho
.         end;
.     2: begin
65     a:=0.333; b:=0.302; //soja
.         end;
.     3: begin
.         a:=0.825; b:=0.164; //trigo
.         end;
70   end;
.   Pg:= (a*sqr(Q)*hg)/ln(1+b*Q);
.   Edit3.Text:=FloatToStr(Pg);
.   end;
.
.
75 procedure TForm1.Button2Click(Sender: TObject);
. begin
.   Edit1.Text:='';
.   Edit2.Text:='';
.   Edit3.Text:='';
80 end;
.
.
. procedure TForm1.Button3Click(Sender: TObject);
. begin
84   close;
```

ENG 390

Aula Prática 09 - A

Parte A. Faça um projeto em *Lazarus* para resolver o seguinte problema: **têm-se duas cidades A e B com populações iniciais diferentes e taxas de crescimento também diferentes. A cidade A tem menor população inicial que B, porém uma taxa de crescimento percentual ao ano maior que B. Então, calcule o tempo para a população de A ultrapassar a população da cidade B.**

a. Aplicação do comando **WHILE ... DO**.

b. Sugestão para o formulário **Form1**

Crescimento Populacional

População Cidade A: 5000 habitantes. Taxa crescimento A: 5 %

População Cidade B: 7000 habitantes. Taxa crescimento B: 3 %

Tempo Transcorrido: 18 Anos.

População Final A: 12019 habitantes.

População Final B: 11906 habitantes.

Calcular Limpar Sair

c. Escrevendo o código em **Portugol**. Transcreva-o para **Lazarus**, de acordo com Form1 acima.

```
Algoritmo População
VAR
    popiA, poiB, popA, popB, taxaA, taxaB: real
    tempo:inteiro
INICIO
    Ler (popiA, taxaA)
    Ler (poiB, taxaB)
    taxaA := taxaA/100 + 1
    taxaB := taxaB/100 + 1
    tempo:=0
    popA:=popiA
    popB:=poiB

    ENQUANTO (popA <= popB) FAÇA
        inicio
            popA:=popA * taxaA
            popB:=popB * taxaB
            tempo := tempo + 1
        fim

    escrever (tempo, popA, popB)
FIM
```

ENG 390

Aula Prática 09 - B

Parte B. Acrescente ao projeto anterior, o gráfico que mostre a evolução do crescimento de ambas as populações.

The screenshot shows a Delphi application window titled "Comando While.. Do" with the subtitle "Crescimento Populacional". The window contains several input fields for population and growth rate, a graph area, and three buttons: "Calcular", "Limpar", and "Sair".

Input fields:

- População Cidade A: habitantes.
- População Cidade B: habitantes.
- Taxa Crescimento A: %
- Taxa Crescimento B: %
- Tempo transcorrido: anos
- População Final A: habitantes.
- População Final B: habitantes.

Buttons:

Graph area: A coordinate system with a vertical axis ranging from -1 to 1 and a horizontal axis. The graph is currently empty.

```
procedure TForm1.btnCalcularClick(Sender: TObject);
var popia, popib, popfa, popfb, taxaa, taxab : Real;
    i, tempografico, tempo : integer;
begin
    // Construcao do Grafico
    FreeAndNil(FLine);
    FreeAndNil(FLineB);
    InitLine;
    popia := StrToFloat(edPopCida.Text); // Populacao Inicial Cidade A
    popib := StrToFloat(edPopCidB.Text); // Populacao Inicial Cidade B
    taxaa := (StrToFloat(edTaxaA.Text)/100) +1; // Taxa Crescimento A
    taxab := (StrToFloat(edTaxaB.Text)/100) +1; // Taxa Crescimento B
    popfa := popia; // Populacao Final de A = Inicial de A para iniciar o laço
    popfb := popib; // Populacao Final de B = Inicial de B para iniciar o laço
    tempo := 0;

    while popfa <= popfb do
    begin
        FLine.AddXY(tempo, popfa, '|', clGreen); // Adiciona os dados da Pop. A
        FLineB.AddXY(tempo, popfb, '|', clGreen); // Adiciona os dados da Pop. B

        popfa := popfa * taxaa; // Calculo acumulativo da Pop. A
        popfb := popfb * taxab; // Calculo acumulativo da Pop. B
        Inc(tempo);
    end;

    tempografico := Round(tempo*1.4); // Tempo para adicionar os dados no grafico para tempo + 30% do mesmo.
    edAnos.Text := IntToStr(tempo); // Tempo Transcorrido |
    edPopFinalA.Text := FormatFloat('##0', popfa);
    edPopFinalB.Text := FormatFloat('##0', popfb);

    while tempo <= tempografico do
    begin
        Inc(tempo);
        popfa := popfa * taxaa;
        popfb := popfb * taxab;
        FLine.AddXY(tempo, popfa, '|', clGreen);
        FLineB.AddXY(tempo, popfb, '|', clGreen);
    end;
end;
```

```

procedure TForm1.btnSairClick(Sender: TObject);
begin
    Close;
end;

procedure TForm1.InitLine;
begin
    FLine := TLineSeries.Create(Chart1); // Cria o gráfico de séries do tipo linha
    FLine.ShowLines := true; // Exibir Linha
    FLine.ShowPoints := False; // Exibir o valor do ponto adicionado (na linha do gráfico)
    FLine.Pointer.Brush.Color := clBlue; // Cor do ponto adicionado caso a propriedade ShowPoints for True
    FLine.Title := 'Pop. A'; // Titulo da Linha de serie
    FLine.SeriesColor := clRed; // Cor da linha do grafico
    Chart1.AddSeries(FLine); // Adiciona a serie de dados para grafico do tipo linha

    // Idem FLine propriedade
    FLineB := TLineSeries.Create(Chart1);
    FLineB.ShowLines := true;
    FLineB.ShowPoints := False;
    FLineB.Title := 'Pop. B';
    FLineB.Pointer.Brush.Color := clGreen;
    FLineB.SeriesColor := clGreen;
    Chart1.AddSeries(FLineB);

    Chart1.Legend.Visible := True; // Exibir Legenda
end;

```

```

procedure TForm1.btnLimparClick(Sender: TObject);
begin
    edPopCidA.Clear;
    edPopCidB.Clear;
    edAnos.Clear;
    edPopFinalA.Clear;
    edPopFinalB.Clear;
    edTaxaA.Clear;
    edTaxaB.Clear;
    FreeAndNil(FLine);
    FreeAndNil(FLineB);
end;

```

ENG 390

Aula Prática 10

Fazer um projeto em Lazarus para calcular a potência de um ventilador em sistemas de secagem de grãos.

- a. Aplicação do conceito de **ARRANJO** unidimensional, no caso **VETOR**: Utilizar o mesmo **Form** da Aula Prática 8, modificando apenas o seu código. Os parâmetros **a** e **b**, na equação de perda de carga serão agora vetores, cujos índices se referem ao tipo de grão, de acordo com a seleção de entrada em **RgTipo.ItemIndex**.

Entrada de Dados	Cálculos
Q: Vazão (m3/min);	Área da base do silo (m2): $A = (PI * D^2) / 4 * hg$
hg: Altura de grãos no silo (m);	Perda de carga nos grãos: $Pg = (a[i]*(Q/A)^2*hg) / (1+b[i]*(Q/A))$
D: Diâmetro do silo (m);	Potência do ventilador: $Pot = Q * 1,2 * Pg / rv$
rv: Rendimento do ventilador (decimal).	

- b. Sugestão para o formulário **Form1**

Potência de Ventilador em Sistemas de Secagem de Grãos:

Tipo de Grão

Arroz

Milho

Soja

Trigo

Vazão de Ar (m3/min):

Altura de Grãos (m):

Diâmetro do Silo (m):

Rendimento do ventilador:

SAIR

Potência Ventilador (cv):

- c. Escrevendo o código em **Portugol e no ambiente Lazarus**, de acordo com Form1 sugerido

```
Portugol
```

```
Algoritmo Ventilador  
Var  
  tg: inteiro  
  Q, Qa, hg, Pg, Pgm, Ds, Area, Pot, rv: real  
  a, b: arranjo [0..3] de reais  
  
Inicio  
  a[0]:=0.772  
  a[1]:=0.583  
  a[2]:=0.333  
  a[3]:=0.825  
  b[0]:=0.197  
  b[1]:=0.512  
  b[2]:=0.302  
  b[3]:=0.164  
  Ler (Q)  
  Ler (hg)  
  Ler (Ds)  
  Ler (rv)  
  
  Area := pi * Ds^2 / 4 * hg  
  Qa := Q/Area  
  
  Pgm := (Qa^2 * a[tg]) / (ln(1+b[tg]*Qa))  
  Pg := Pgm * hg;  
  
  Pot := ( Q * 1.2*Pg ) / rv;  
  
  Escrever (Pot)  
Fim
```

```
Delphi
```

```
procedure TForm1.Button1Click(Sender: TObject);  
  
var  
  tg: integer;  
  Q, Qa, hg, Pg, Pgm, Ds, Area, Pot, rv: double;  
  a, b: array [0..3] of double;  
  
begin  
  
  a[0]:=0.772; a[1]:=0.583; a[2]:=0.333; a[3]:=0.825;  
  b[0]:=0.197; b[1]:=0.512; b[2]:=0.302; b[3]:=0.164;  
  
  Q := StrToFloat(Edit1.Text);  
  hg:= StrToFloat(Edit2.Text);  
  Ds:= StrToFloat(Edit3.Text);  
  rv:= StrToFloat(Edit4.Text);  
  tg:= RgTipo.ItemIndex;  
  
  Area := pi * sqr (Ds) / 4 * hg;  
  Qa := Q/Area ;  
  
  Pgm := (sqr(Qa) * a[tg]) / (ln(1+b[tg]*Qa));  
  Pg := Pgm * hg;  
  
  Pot := ( Q * 1.2*Pg ) / rv;  
  Edit5.text := FloatToStr (Pot);  
  
end;
```