



II-T (Segundo Trimestre)

Prof: Raffael Bottoli Schemmer
Disciplina: Programação I
Ano: 2019.
Módulo: II - Trimestre



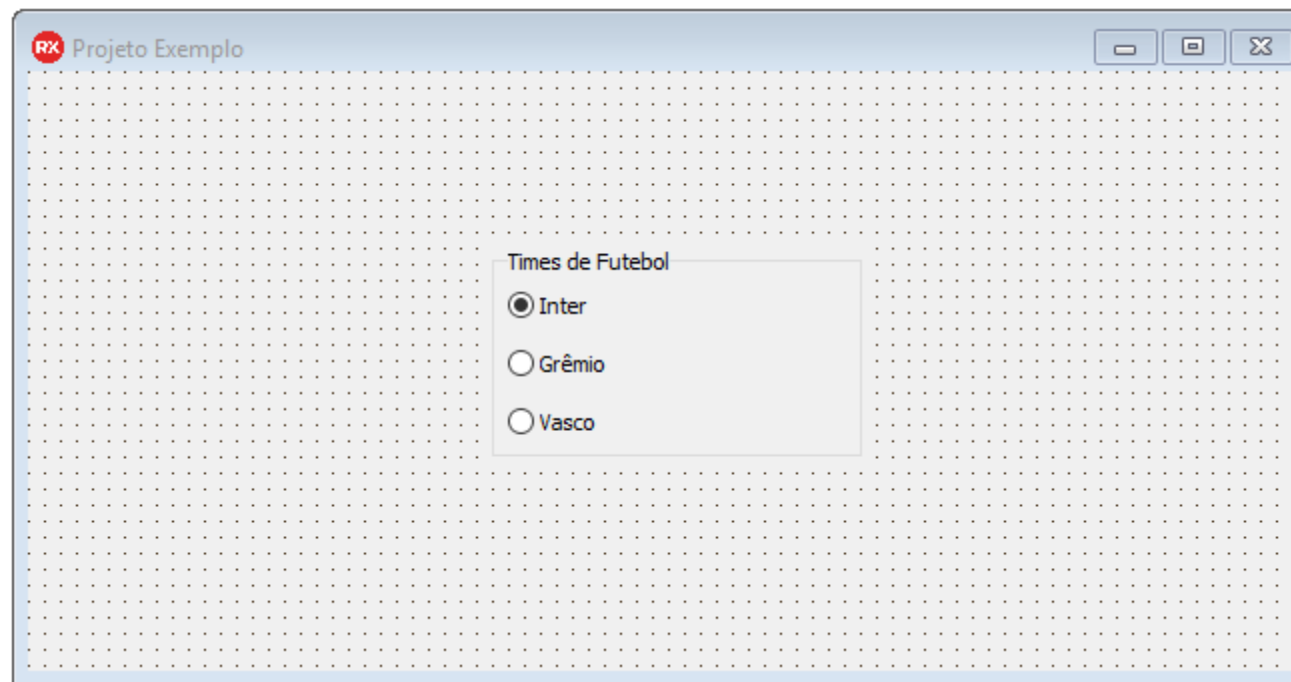
Atividades Trimestrais (II - Trimestre)



M.II-T = TI (16%) até TV (16%) + 3 Kahoot (20%)



Botão de Seleção (TRadioGroup)



Componente **TForm** com um **TRadioGroup**

Botão de Seleção (TRadioGroup)

- Todos programas possuem **botões** de **seleção**:
 - Que permitem "**dirigir**" a **entrada/execução** das **ações** (eventos) do **programa**.
 - São muito **úteis** quando **queremos construir programas "avançados"**.
- Estes componentes de interface gráfica:
 - Utilizam **estruturas** de **programação** de **seleção** de **caminhos**.
 - Estas **estruturas lógicas** utilizam **operadores** que **relacionam dados**.
- A seguir, nós aprenderemos estas **estruturas** e estes **operadores**:
 - Só então estudaremos em maior profundidade o **TRadioGroup**.



Operadores Condicionais (Relacionais)



- Operadores relacionais permitem realizar comparações simples.
- As comparações podem ser feitas com constantes e variáveis.
- Toda comparação retorna um valor lógico (TRUE ou FALSE).
- Toda comparação relacional precisa SEMPRE de dois operandos.



Operadores Condicionais (Relacionais)

Operador	Utilidade	Exemplo	Resultado
=	Compara se são iguais	2 = 2	TRUE
<>	Compara se são diferentes	2 <> 2	FALSE
>	Compara se um é menor que outro	3 > 2	TRUE
<	Compara se um é maior que outro	2 < 3	TRUE
>=	Compara se um é menor ou igual que outro	2 >= 2	TRUE
<=	Compara se um é maior ou igual que outro	5 <= 3	FALSE

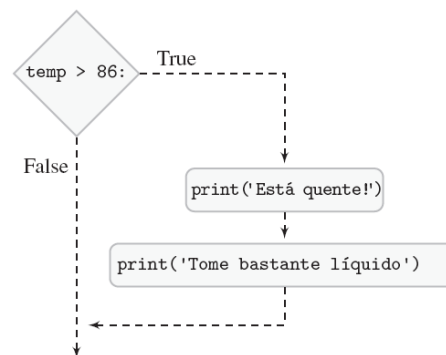
Tabela com os 6 operadores relacionais e sua funcionalidade



Operadores Condicionais (Relacionais)

Onde estes operadores devem ser utilizados?

Nas estruturas de seleção que controlam o fluxo da execução!

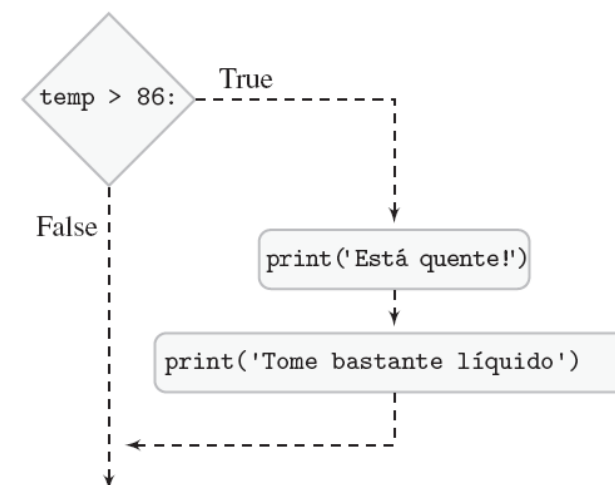


Estruturas Condicionais (IF)

- O que é o IF?
 - IF é uma **estrutura lógica** que **condiciona** a **execução** de certas **instruções**.
- Como funciona?
 - Se a **condição** possuir valor **TRUE** o **bloco IF** será **executado**.

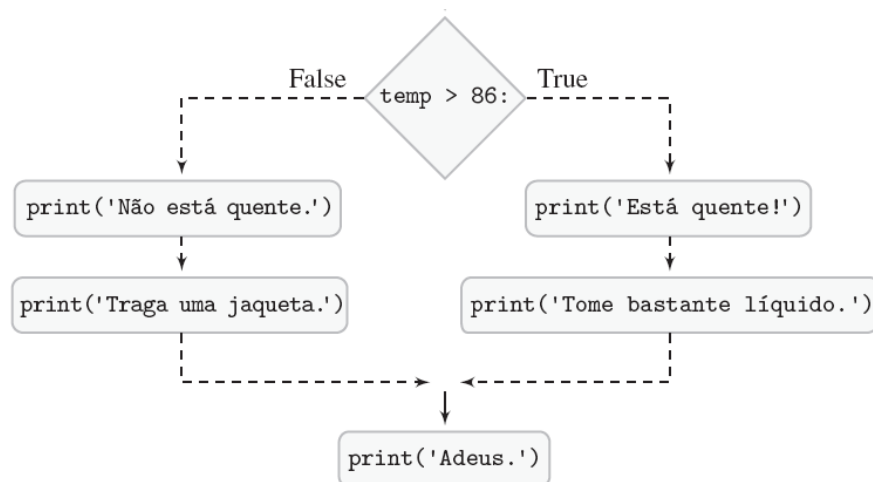
```
IF (mediaAluno >= 7.0) THEN  
BEGIN  
    SHOWMESSAGE ('Aluno Aprovado!');  
END;
```

Estrutura **condicional simples IF** com exemplo



Estruturas Condicionais (IF/ELSE)

- O que é o **IF/ELSE**?
 - Todo **IF** pode ter um **ELSE** associado ao mesmo.
 - O **ELSE** é uma **estrutura opcional** que pode ser **utilizada** caso seja **necessário**.
- Como funciona?
 - O bloco do **ELSE** sempre irá executar se a condição do **IF** resultar em **FALSE**.



```
IF (mediaAluno >= 7.0) THEN
BEGIN

    SHOWMESSAGE ('Aluno Aprovado!');

END
ELSE
BEGIN

    SHOWMESSAGE ('Aluno Não Aprovado!');

END;
```

Estrutura **condicional** composta **IF/ELSE**

Operadores Condicionais (Lógicos)

- Operadores **lógicos** permitem realizar **comparações compostas**.
- As **comparações** são feitas **obrigatoriamente** com **variáveis** e **constantes boolean**.
- Toda **comparação** retorna um valor **lógico** (TRUE ou FALSE).
- Toda **comparação lógica** precisa **SEMPRE** de **dois operandos lógicos** (boolean).



Operador Lógico (AND)

Lado Esquerdo	Lado Direito	Expressão	Resultado
FALSE	FALSE	FALSE AND FALSE	FALSE
FALSE	TRUE	FALSE AND TRUE	FALSE
TRUE	FALSE	TRUE AND FALSE	FALSE
TRUE	TRUE	TRUE AND TRUE	TRUE

Tabela verdade do **operador lógico AND** (Apenas quando **ambos** os **operandos** forem **TRUE** o **resultado** é **TRUE**)

Operadores Lógicos (AND)

- 1. AND (Operador da Conjunção)
 - Pode ser **utilizado** em **quantas variáveis** (entradas) for **necessário**.
 - "E" Se os **dois operandos** são **verdadeiros**, o **resultado** será **verdadeiro**.
 - Se **um** dos **dois operandos** é **falso**, o **resultado** será **falso**.

Operador AND



```
var A,B : Integer;  
  
BEGIN  
  
    // Só irá executar se A e B forem > 0  
    IF ( (A > 0) AND (B > 0) ) THEN  
        BEGIN  
            // Lógica do programa!  
        END;  
    END;  
END;
```

A	AND	B	S
F		F	F
F		V	F
V		F	F
V		V	V

Operador Lógico (OR)

Lado Esquerdo	Lado Direito	Expressão	Resultado
FALSE	FALSE	FALSE OR FALSE	FALSE
FALSE	TRUE	FALSE OR TRUE	TRUE
TRUE	FALSE	TRUE OR FALSE	TRUE
TRUE	TRUE	TRUE OR TRUE	TRUE

Tabela **verdade** do operador **lógico OR** (Quando **um** dos **operandos** for **TRUE** o **resultado** é **TRUE**)

Operadores Lógicos (OR)

- 2. OR (Operador da Disjunção)
 - Pode ser **utilizado** em **quantas variáveis** (entradas) for **necessário**.
 - "OU" Se **um** dos **operandos** são **verdadeiros**, o **resultado** será **verdadeiro**.
 - Se os **dois operandos** são **falsos**, o **resultado** será **falso**.

Operador OR



```
var SENHA : Integer;  
  
BEGIN  
  
  // Só irá executar se SENHA for 123 ou 321  
  IF ( (SENHA = 123) OR (SENHA = 321) ) THEN  
  BEGIN  
  
    // Lógica do programa!  
  
  END;  
  
END;
```

A	OR	B	S
F		F	F
F		V	V
V		F	V
V		V	V

Operador Lógico (NOT)

Lado Esquerdo	Expressão	Resultado
FALSE	NOT FALSE	TRUE
TRUE	NOT TRUE	FALSE

Tabela **verdade** do operador **lógico NOT** (Quando operador for **TRUE** o **resultado** é **FALSE**)

Operadores Lógicos (NOT)

- 3. NOT (Operador da Negação)
 - Nega o estado (IF for FALSE retorna TRUE e/ou o contrário).

```
var cont : Integer;  
  
begin  
    cont := 1;  
    REPEAT  
        cont := cont + 1;  
    UNTIL NOT cont <= 5;  
end;
```

Operador NOT



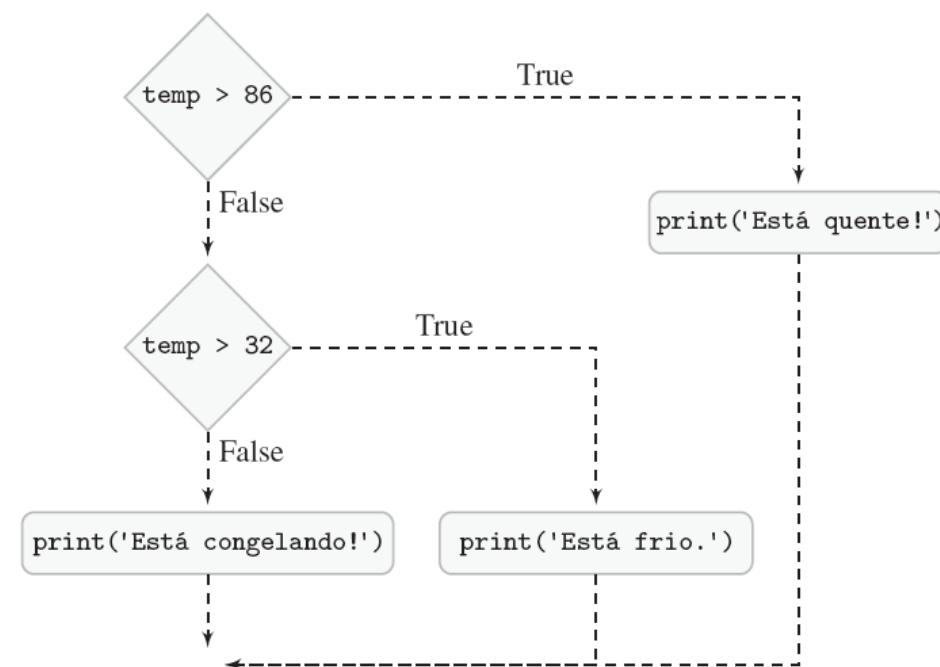
NOT	A		S
	F		V
	V		F

Estruturas Condicionais (IF/ELSE IF)

- Permite realizar uma **verificação lógica** e executa o bloco de código (IF) se a condição for **TRUE**
 - Caso contrário, uma nova pergunta (ELSE IF) é **realizada**.
 - Estrutura **ELSE** é **opcional**.

```
IF (mediaAluno >= 7.0) THEN
BEGIN
    SHOWMESSAGE('Aluno Aprovado!');
END
ELSE IF ((mediaAluno >= 5.0) AND (mediaAluno < 7.0)) THEN
BEGIN
    SHOWMESSAGE('Aluno Em Exame!');
END
ELSE
BEGIN
    SHOWMESSAGE('Aluno Reprovado!');
END;
END;
```

Estrutura condicional **encadeada IF/ELSE IF/ELSE**

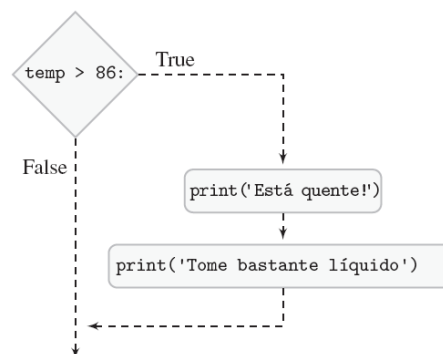


Operadores Lógicos (Relacionais)

Onde estes operadores devem ser utilizados?

Em **perguntas compostas** (múltiplas)

Nas estruturas de seleção que controlam o fluxo da execução!



Botão de Seleção (TRadioGroup): Eventos

- Os testes condicionais (IF):
 - Podem testar os valores selecionados em um TRadioGroup.
 - Vamos entender o componente VCL (propriedades/eventos).
- Principais eventos:
 - **onClick**: Executa toda vez que um click for feito na TRadioGroup.
 - **onEnter**: Executa toda vez que o TAB entrar dentro do TRadioGroup
 - **onExit**: Executa toda vez que o TAB sair dentro do TRadioGroup.



Botão de Seleção (TRadioGroup): Propriedades

- **Name:** Define o nome do componente na **Unit**.
 - O conteúdo desta **propriedade** é **sempre** uma **frase** (**String**).
- **Caption:** Define o **rótulo** do TRadioGroup.
 - O conteúdo desta **propriedade** é **sempre** uma **frase** (**String**).
- **Color:** Define a **cor** do TRadioGroup.
 - O conteúdo desta **propriedade** é sempre uma **frase** (**String**) definida pelo **DELPHI**.
- **Font:** Define a **fonte** e o **tamanho** do texto do TRadioGroup
 - Para **estilizar** esta **propriedade** você deve **pressionar** (..).
 - Nesta **propriedade** você pode **estilizar** a **fonte**, o **tamanho** da **fonte** e a **cor** da **fonte**.



Botão de Seleção (TRadioGroup): Propriedades

- **Enabled**: Define se o botão será habilitado
 - O conteúdo desta propriedade é sempre um boolean (true/false).
- **TabOrder**: Define a ordem de chamada do TAB no TRadioGroup.
 - O conteúdo desta propriedade é sempre um número (Integer).
- **ItemIndex**: Define qual dos botões será marcado (zero é o primeiro).
 - O conteúdo desta propriedade é sempre um número (Integer).
- **Items**: Define os botões no TRadioGroup.
 - O conteúdo desta propriedade é sempre um conjunto (lista) de strings.



Botão de Seleção (TRadioGroup): Propriedades

- **Columns**: Define quantas colunas o deve ter TRadioGroup.
 - O conteúdo desta **propriedade** é sempre um **número** (**Integer**).
- **Hint**: Define uma dica para o TRadioGroup.
 - O conteúdo desta **propriedade** é sempre uma **frase** (**String**).
- **ShowHint**: Define se a dica será mostrada ou não.
 - O conteúdo desta **propriedade** é **sempre um boolean** (**true/false**).
- **Visible**: Define se o botão será visível ou não.
 - O conteúdo desta **propriedade** é **sempre um boolean** (**true/false**).



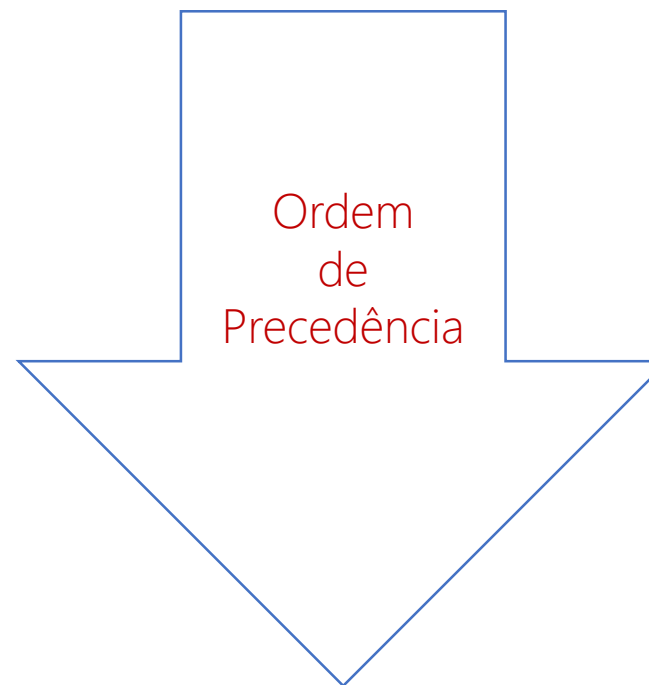
Botão de Seleção (TRadioGroup): Exemplo

```
procedure TForm2.RdGrpClick(Sender: TObject);  
begin  
    RdGrp.Caption := 'Times de Futebol!';  
  
    if RdGrp.ItemIndex = 0 then  
    begin  
        showmessage('Time Esolhido: ', RdGrp.Items[RdGrp.ItemIndex]);  
    end  
    else if RdGrp.ItemIndex = 1 then  
    begin  
        showmessage('Time Esolhido: ', RdGrp.Items[RdGrp.ItemIndex]);  
    end;  
end;
```

Tratamento do **evento onClick** do **TRadioGroup**
Seleciona uma das opções do **RadioGroup** utilizando **ItemIndex**
Captura o **conteúdo** da **posição ItemIndex** utilizando **Items**

Precedência dos Operadores

- 1. Parênteses mais **internos**
 - ()
- 2. Operadores **aritméticos**
 - * / DIV MOD + -
- 3. Operadores **relacionais**
 - = <> > >= < <=
- 4. Operadores **lógicos**
 - AND OR NOT



Decimo Primeiro Trabalho da Disciplina (2TXI)

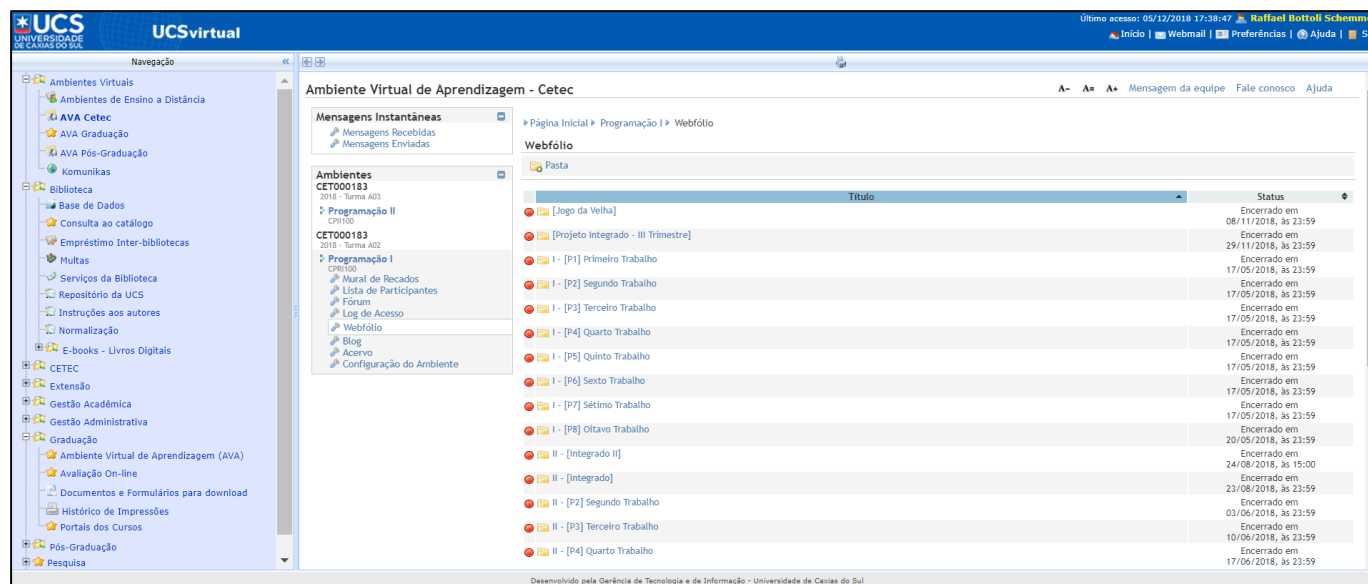
- Calculadora de moedas:



Utilize o exemplo disponível do GitHub para desenvolver o trabalho



Publicação do TXI no AVA



Último acesso: 05/12/2018 17:38:47 **Rafael Bottoli Schiemmer**
Início | Webmail | Preferências | Ajuda | Sair

UCSvirtual

Ambiente Virtual de Aprendizagem - Cetec

Mensagens Instantâneas
Mensagens Recebidas
Mensagens Enviadas

Ambientes
CET000183
2018 - Turno A02
Programação II
CET000183
2018 - Turno A02
Programação I

Webfólio

Título	Status
[Jogo da Velha]	Encerrado em 08/11/2018, às 23:59
[Projeto Integrado - III Trimestre]	Encerrado em 29/11/2018, às 23:59
I - [P1] Primeiro Trabalho	Encerrado em 17/05/2018, às 23:59
I - [P2] Segundo Trabalho	Encerrado em 17/05/2018, às 23:59
I - [P3] Terceiro Trabalho	Encerrado em 17/05/2018, às 23:59
I - [P4] Quarto Trabalho	Encerrado em 17/05/2018, às 23:59
I - [P5] Quinto Trabalho	Encerrado em 17/05/2018, às 23:59
I - [P6] Sexto Trabalho	Encerrado em 17/05/2018, às 23:59
I - [P7] Sétimo Trabalho	Encerrado em 17/05/2018, às 23:59
I - [P8] Oitavo Trabalho	Encerrado em 20/05/2018, às 23:59
II - [Integrado II]	Encerrado em 24/08/2018, às 15:00
II - [Integrado]	Encerrado em 23/08/2018, às 23:59
II - [P2] Segundo Trabalho	Encerrado em 03/06/2018, às 23:59
II - [P3] Terceiro Trabalho	Encerrado em 10/06/2018, às 23:59
II - [P4] Quarto Trabalho	Encerrado em 17/06/2018, às 23:59

Webfólio do Ambiente Virtual de Aprendizagem (CETEC)

Avaliação Kahoot! (Maio)



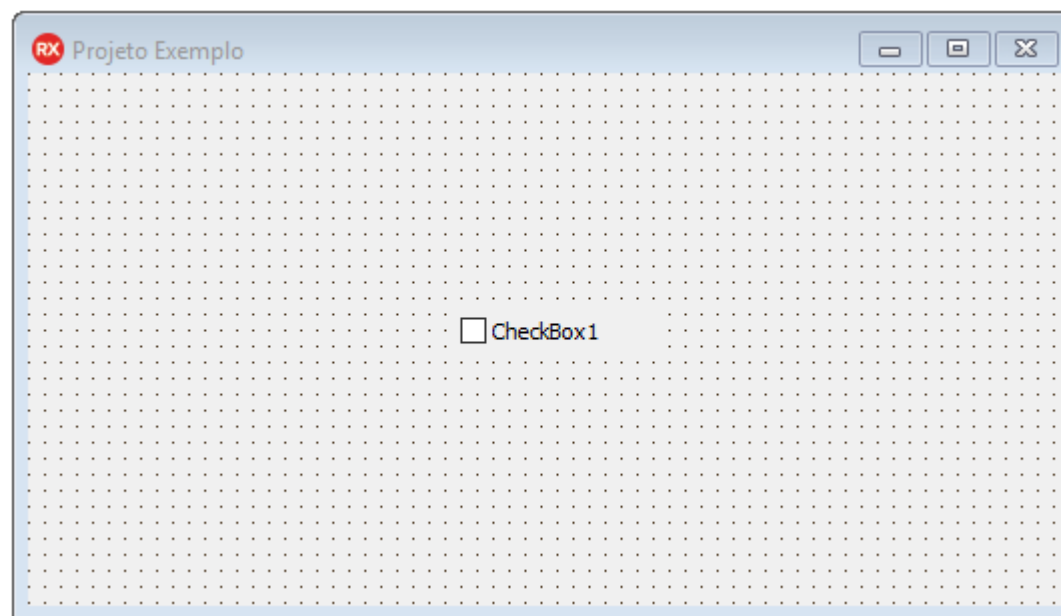
Prova Mensal de Conceitos!



(10 Questões sobre as 4 Semanas de Maio)



Caixa de Marcação (TCheckBox)



Componente `TForm` com um `TCheckBox`

Caixa de Marcação (TCheckBox) : Definição

- A **caixa** de **marcação**:
 - Permite o **usuário** **marcar** quais **opções** **deseja**.
- Diferente dos **botões** de **seleção**:
 - As **caixas** devem ser **inseridas** **manualmente** uma a uma.
 - Cada **caixa** deve ser **verificada** **individualmente**.
 - O **usuário** poderá **marcar** quantas **caixas** quiser.
- Observe como nos **botões** a **seleção** é **única**.
- Com a **caixa** a **marcação** é **múltipla**.

Botão de Seleção

☐ 1

☐ 2

☒ 3

☐ 4

Caixa de Marcação

☒ 1

☐ 2

☒ 3

Caixa de Marcação (TCheckBox) : Propriedades

- **Name:** Define o nome do componente na Unit.
 - O conteúdo desta **propriedade** é **sempre** uma **frase** (String).
- **Caption:** Define o rótulo do TCheckBox.
 - O conteúdo desta **propriedade** é **sempre** uma **frase** (String).
- **Font:** Define a fonte e o tamanho do texto do TCheckBox
 - Para **estilizar** esta **propriedade** você deve **pressionar** (..).
 - Nesta **propriedade** você pode **estilizar** a **fonte**, o **tamanho** da **fonte** e a **cor** da **fonte**.
- **Enabled:** Define se o TCheckBox será habilitado
 - O **conteúdo** desta **propriedade** é **sempre** um **boolean** (true/false).



Caixa de Marcação (TCheckBox) : Propriedades



- **TabOrder**: Define a ordem de chamada do TAB no TCheckBox.
 - O conteúdo desta **propriedade** é sempre um **número** (Integer).
- **Hint**: Define uma dica para o TCheckBox.
 - O conteúdo desta **propriedade** é **sempre** uma **frase** (String).
- **ShowHint**: Define se a dica será mostrada ou não.
 - O **conteúdo** desta **propriedade** é **sempre** um **boolean** (true/false).



Caixa de Seleção (TCheckBox) : Propriedades

- **Visible**: Define se o TCheckBox será visível ou não.
 - O conteúdo desta propriedade é sempre um boolean (true/false).
- **State**: Permite modificar o tipo de checagem.
 - O conteúdo desta propriedade é uma constante (cbChecked/cbGrayed).
- **Checked**: Permite marcar o TCheckBox.
 - O conteúdo desta propriedade é sempre um boolean (true/false).



Caixa de Seleção (TCheckBox) : Eventos

- **onEnter**: Executa toda vez que o TAB entrar dentro do TCheckBox.
- **onExit**: Executa toda vez que o TAB sair dentro do TCheckBox.
- **onClick**: Executa toda vez que um clique for feito no TCheckBox.
- **onMouseEnter**: Executa toda vez que o mouse entrar no TCheckBox.
- **onMouseLeave**: Executa toda vez que o mouse sair no TCheckBox.

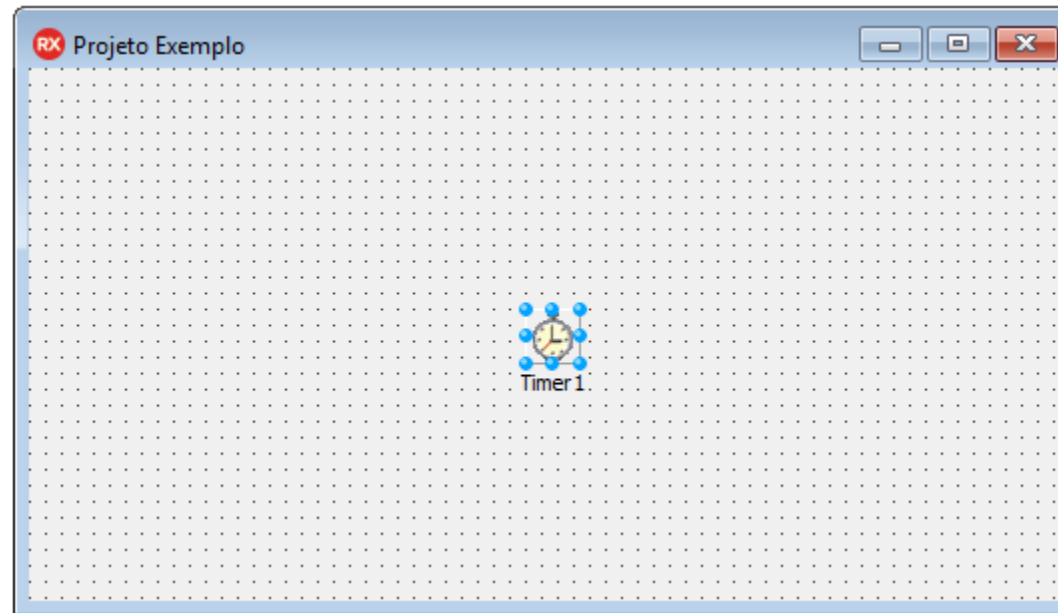


TCheckBox (Seleção): Exemplo

```
procedure TForm2.RdGrpClick(Sender: TObject);  
begin  
  
    ChkBox.Caption := 'Devo Comparar?';  
  
    if ChkBox.Checked = TRUE then  
    begin  
        // Bloco que implementa o cálculo  
    end;  
  
end;
```

Tratamento do evento `onClick` do botão
Define o conteúdo do rótulo do `TCheckBox` e verifica se o mesmo foi marcado

TTimer (Temporizador): Definição



Componente **TForm** com um **TTimer**

TTimer (Temporizador): Definição

- Permite disparar eventos sincronizados por um relógio.
- São úteis quando você quer fazer o código disparar algo.
- Utilizaremos o relógio em nossos próximos projetos.
- Após o usuário fazer algo:
 - O relógio irá a cada X tempo atualizar a tela (fazer o programa responder).
 - O relógio funciona como um contador regressivo de X até 0.



TTimer (Temporizador): Propriedades

- **Name:** Define o nome do componente na Unit.
 - O conteúdo desta **propriedade** é **sempre** uma **frase** (String).
- **Interval:** Define o intervalo de tempo do Timer.
 - O conteúdo desta **propriedade** é sempre um **número** (Integer).
- **Enabled:** Define se o Timer será habilitado.
 - O **conteúdo** desta **propriedade** é **sempre** um **boolean** (true/false).



TTimer (Temporizador): Eventos

- **onTimer**: Executa toda vez que o interval chegar a zero.

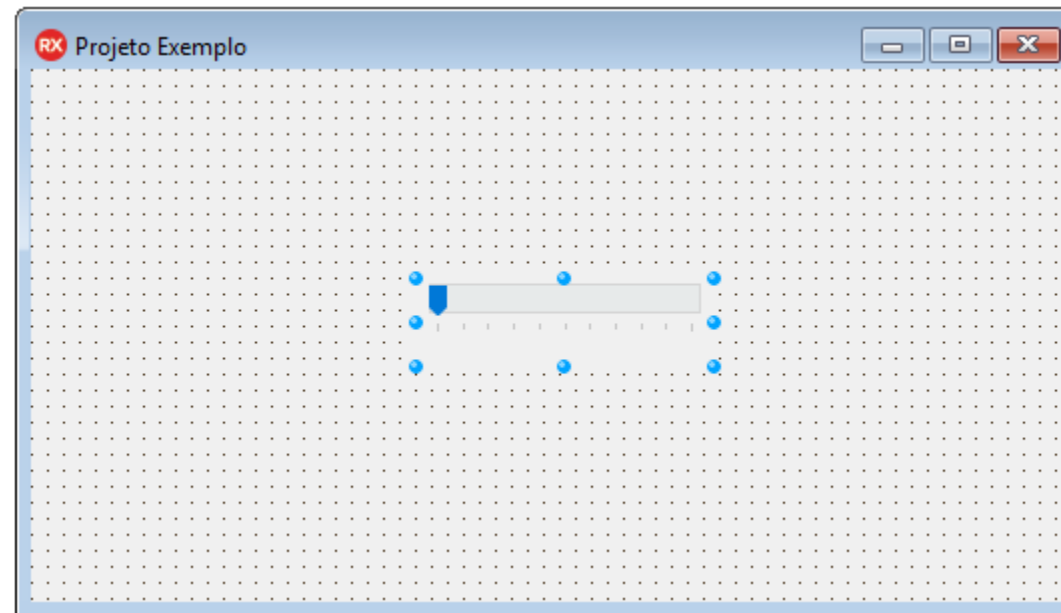


TTimer (Temporizador): Exemplo

```
procedure TForm1.RelogioTimer(Sender: TObject);  
begin  
    FrmMenu.Color := clBlue;  
end;
```

Tratamento do evento `onTimer` do Relógio
Atualiza a `cor` do formulário.

TTrackBar (Rolagem): Definição



Componente `TForm` com um `TTrackBar`

TTrackBar (Barra Rolagem): Conceito

- Implementa uma barra de rolagem:
 - Que permite que números inteiros sejam informados.
- Sempre que você precisar capturar um número em um intervalo:
 - Utilize a barra de rolagem para facilitar a captura.
- Barras de rolagem como o TTrackBar:
 - Protegem a entrada de números fora de um intervalo padrão.
- Fora que tornam o programa mais iterativo (dinâmico).



TTrackBar (Barra Rolagem): Propriedades

- **Name**: Define o nome do componente na Unit.
 - O conteúdo desta **propriedade** é **sempre** uma **frase** (**String**).
- **Enabled**: Define se o botão será habilitado.
 - O **conteúdo** desta **propriedade** é **sempre** um **boolean** (**true/false**).
- **Frequency**: Define a frequência da barra de rolagem.
 - O conteúdo desta **propriedade** é sempre um **número** (**Integer**).
- **Width**: Define a largura do componente.
 - O conteúdo desta **propriedade** é sempre um **número** (**Integer**).



TTrackBar (Barra Rolagem): Propriedades

- **Height:** Define a altura do componente.
 - O conteúdo desta **propriedade** é sempre um **número (Integer)**.
- **Min:** Intervalo mínimo do TTrackBar.
 - O conteúdo desta **propriedade** é sempre um **número (Integer)**.
- **Max:** Intervalo máximo do TTrackBar.
 - O conteúdo desta **propriedade** é sempre um **número (Integer)**.
- **Position:** Posição atual do TTrackBar.
 - O conteúdo desta **propriedade** é sempre um **número (Integer)**.



TTrackBar (Barra Rolagem): Propriedades

- **BorderWidth**: Define uma borda ao redor do componente.
 - O conteúdo desta **propriedade** é sempre um **número** (**Integer**).
- **SelStart**: Define a posição inicial do seletor.
 - O conteúdo desta **propriedade** é sempre um **número** (**Integer**).
- **ShowSelRange**: Muda o tipo de seletor.
 - O conteúdo desta **propriedade** é sempre um **boolean** (**true/false**).
- **SliderVisible**: Permite ocultar a chave de seleção.
 - O conteúdo desta **propriedade** é sempre um **boolean** (**true/false**).
- **TickMarks**: Permite inverter a barra de rolagem.
 - O conteúdo desta **propriedade** é uma **constante** (**tmBoth/tmTopLeft/tmBottomRight**).

TTrackBar (Barra Rolagem): Propriedades

- **TabOrder**: Define a ordem de chamada do TAB no botão.
 - O conteúdo desta **propriedade** é sempre um **número** (**Integer**).
- **Hint**: Define uma dica para o botão.
 - O conteúdo desta **propriedade** é **sempre** uma **frase** (**String**).
- **ShowHint**: Define se a dica será mostrada ou não.
 - O **conteúdo** desta **propriedade** é **sempre um boolean** (**true/false**).
- **Visible**: Define se o botão será visível ou não.
 - O **conteúdo** desta **propriedade** é **sempre um boolean** (**true/false**).
- **Orientation**: Define a orientação do TTrackBar (Barra de rolagem).
 - O **conteúdo** desta **propriedade** é **uma constante** (**trHorizontal/trVertical**).

TTrackBar (Barra Rolagem): Eventos

- **onChange**: Executa toda vez que o TTrackBar for modificado.
- **onEnter**: Executa toda vez que o TAB entrar dentro do TTrackBar.
- **onExit**: Executa toda vez que o TAB sair dentro do TTrackBar.



TTrackBar (Barra Rolagem): Eventos

- **onClick**: Executa toda vez que um clique for feito no TTrackBar.
- **onDbClick**: Executa toda vez que dois clicks forem feitos no TTrackBar.
- **onMouseEnter**: Executa toda vez que o mouse entrar no TTrackBar.
- **onMouseLeave**: Executa toda vez que o mouse sair no TTrackBar.

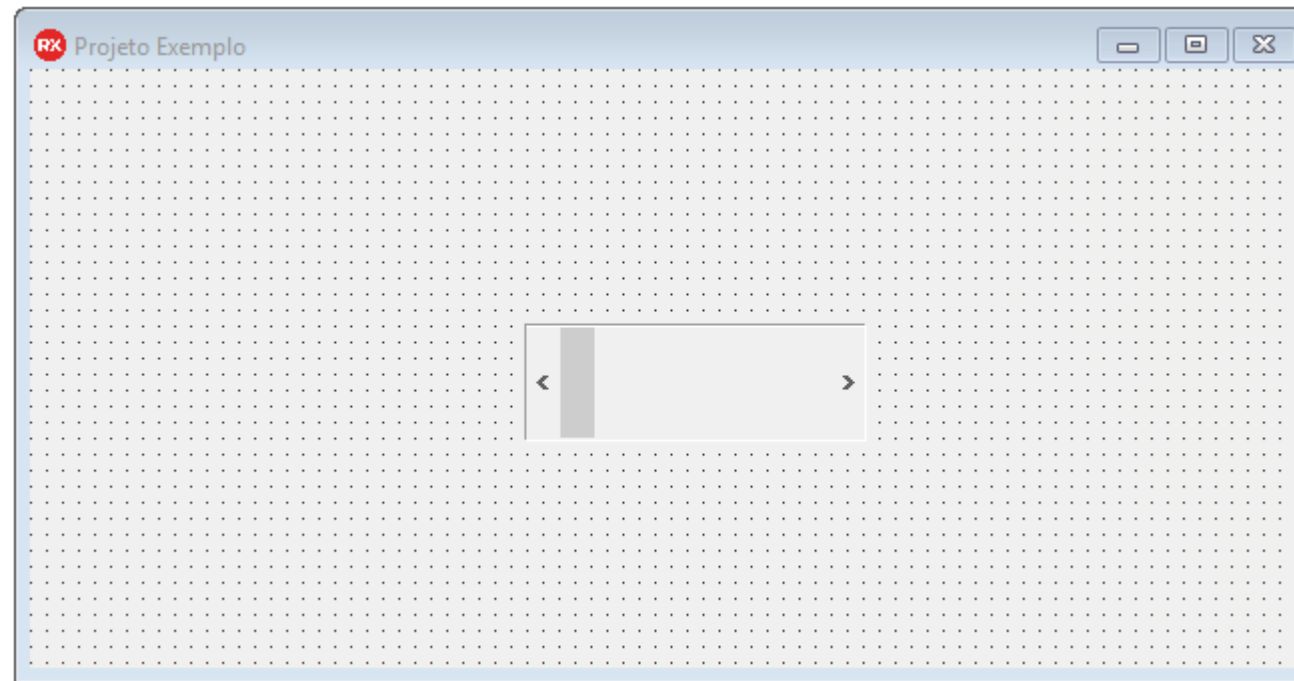


TTrackBar (Barra Rolagem): Exemplo

```
procedure TForm18.BtnClick(Sender: TObject);  
begin  
  
    TrkBar.Min := 0;  
    TrkBar.Max := 100;  
    TrkBar.Frequency := 10;  
    TrkBar.SelStart := 0;  
    TrkBar.SliderVisible := TRUE;  
  
end;
```

Tratamento do evento `onClick` da barra de rolagem
Define o conteúdo do `TTrackBar` com `min` em 0 e `max` em 100 com frequência de 10

TScrollBar (Barra Rolagem): Definição



Componente `TForm` com um `TScrollBar`

TScrollBar (Barra Rolagem): Conceito

- Implementa uma barra de rolagem:
 - Que permite que números inteiros sejam informados.
 - Muito similar a TTrackBar.
- Sempre que você precisar capturar um número em um intervalo:
 - Utilize a barra de rolagem para facilitar a captura.
- Barras de rolagem como o TScrollBar:
 - Protegem a entrada de números fora de um intervalo padrão.
- Fora que tornam o programa mais iterativo (dinâmico).



TScrollBar (Barra Rolagem): Propriedades

- **Name**: Define o nome do componente na Unit.
 - O conteúdo desta **propriedade** é **sempre** uma **frase** (**String**).
- **Width**: Define a largura do componente.
 - O conteúdo desta **propriedade** é sempre um **número** (**Integer**).
- **Height**: Define a altura do componente.
 - O conteúdo desta **propriedade** é sempre um **número** (**Integer**).
- **TabOrder**: Define a ordem que o componente será selecionado pelo TAB.
 - O conteúdo desta **propriedade** é sempre um **número** (**Integer**).

TScrollBar (Barra Rolagem): Propriedades

- **Visible:** Torna o componente ocultável
 - O conteúdo desta propriedade é sempre um boolean (true/false).
- **Enabled:** Habilita o acesso ao componente.
 - O conteúdo desta propriedade é sempre um boolean (true/false).
- **Hint:** Define uma dica para o ScrollBar.
 - O conteúdo desta propriedade é sempre uma frase (String).
- **Kind:** Define se a barra de rolagem será horizontal ou vertical.
 - O conteúdo desta propriedade é uma constante (sbHorizontal/sbVertical).

TScrollBar (Barra Rolagem): Propriedades

- **Min:** Intervalo mínimo do ScrollBar.
 - O conteúdo desta **propriedade** é sempre um **número** (**Integer**).
- **Max:** Intervalo máximo do ScrollBar.
 - O conteúdo desta **propriedade** é sempre um **número** (**Integer**).
- **PageSize:** Define o tamanho do passo do ScrollBar.
 - O conteúdo desta **propriedade** é sempre um **número** (**Integer**).
- **Position:** Posição atual do ScrollBar.
 - O conteúdo desta **propriedade** é sempre um **número** (**Integer**).
- **ShowHint:** Habilita a dica.
 - O conteúdo desta **propriedade** é sempre um **boolean** (**true/false**).

TScrollBar (Barra Rolagem): Eventos

- **onChange**: Executa toda vez que o TScrollBar for modificado.
- **onEnter**: Executa toda vez que o TAB entrar dentro do TScrollBar.
- **onExit**: Executa toda vez que o TAB sair dentro do TScrollBar.
- **onMouseEnter**: Executa toda vez que o mouse entrar no TScrollBar.
- **onMouseLeave**: Executa toda vez que o mouse sair no TScrollBar.

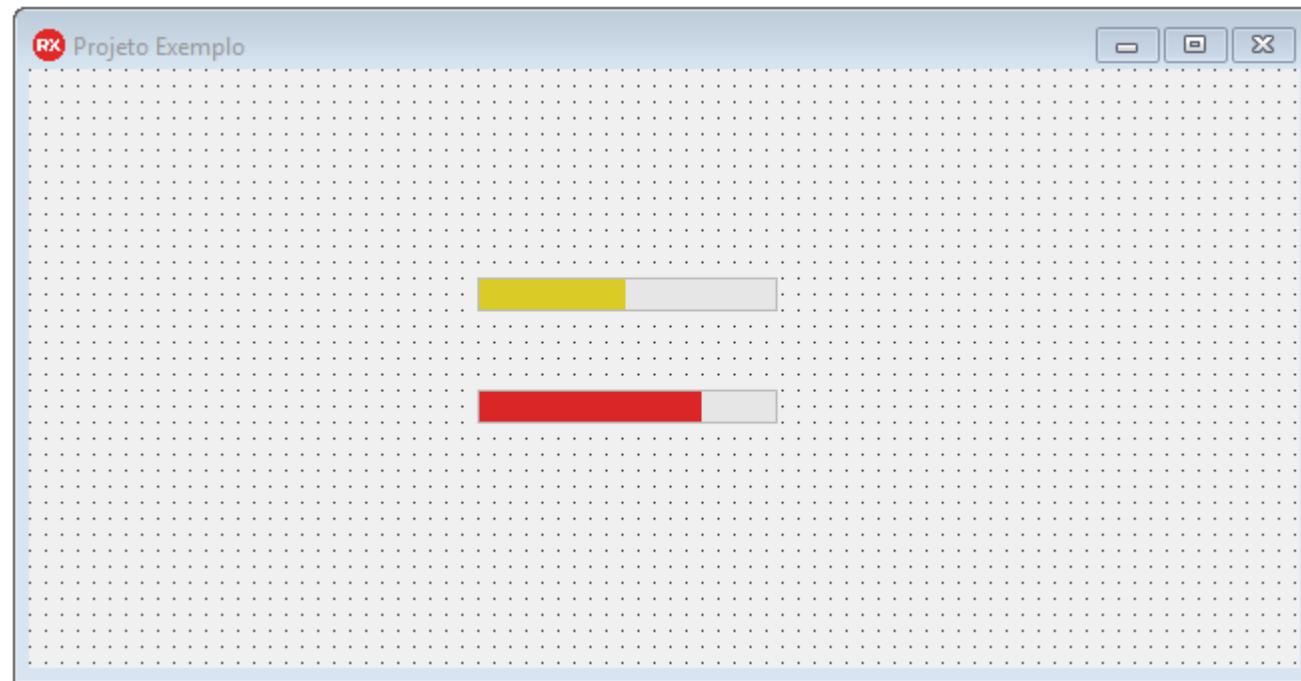


TScrollBar (Barra Rolagem): Exemplo

```
procedure TForm2.SclBarChange(Sender: TObject);  
begin  
  
    SclBar.position := 50;  
    SclBar.min := 1;  
    SclBar.max := 100;  
    SclBar.enabled := TRUE;  
  
end;
```

Tratamento do **evento onChange** da **barra de rolagem**
Define o conteúdo da **barra** com **min** em 1 e **max** em 100 e com **posição** em 50

TProgressBar (Barra Progresso): Definição



Componente `TForm` com um `TProgressBar`

TProgressBar (Barra Progresso): Conceito

- Barras de Progresso permitem indicar progresso:
 - Nós podemos definir o término ou o início de algo ao usuário.
- Utilizaremos a barra de progresso:
 - Para indicar quando a partida de um jogo será concluída.
- Barras de progresso como o TProgressBar:
 - Tornam o programa mais iterativo (dinâmico).



TProgressBar (Barra Progresso): Propriedades

- **Name:** Define o nome do componente na Unit.
 - O conteúdo desta **propriedade** é **sempre** uma **frase** (**String**).
- **Width:** Define a largura do componente.
 - O conteúdo desta **propriedade** é sempre um **número** (**Integer**).
- **Height:** Define a altura do componente.
 - O conteúdo desta **propriedade** é sempre um **número** (**Integer**).
- **Visible:** Torna o componente ocultável.
 - O conteúdo desta **propriedade** é sempre um **boolean** (**true/false**).

TProgressBar (Barra Progresso): Propriedades

- **Enabled:** Habilita o acesso ao componente.
 - O conteúdo desta **propriedade** é sempre um **boolean** (**true/false**).
- **Hint:** Define uma dica para o TProgressBar.
 - O conteúdo desta **propriedade** é **sempre** uma **frase** (**String**).
- **Orientation:** Define se a barra de progresso será horizontal ou vertical.
 - O conteúdo desta **propriedade** é uma **constante** (**pbHorizontal/pbVertical**).
- **ShowHint:** Habilita a dica.
 - O conteúdo desta **propriedade** é sempre um **boolean** (**true/false**).



TProgressBar (Barra Progresso): Propriedades

- **Min:** Intervalo mínimo do TProgressBar.
 - O conteúdo desta **propriedade** é sempre um **número** (**Integer**).
- **Max:** Intervalo máximo do TProgressBar.
 - O conteúdo desta **propriedade** é sempre um **número** (**Integer**).
- **Position:** Posição atual do TProgressBar.
 - O conteúdo desta **propriedade** é sempre um **número** (**Integer**).
- **State:** Define se o estado é normal (verde) ou de erro (vermelho).
 - O conteúdo desta **propriedade** é uma **constante** (**pbsError/pbsNormal/pbsPaused**).



TProgressBar (Barra Progresso): Eventos

- **onEnter**: Executa toda vez que o TAB entrar dentro do TProgressBar.
- **onExit**: Executa toda vez que o TAB sair dentro do TProgressBar.
- **onMouseEnter**: Executa toda vez que o mouse entrar no TProgressBar.
- **onMouseLeave**: Executa toda vez que o mouse sair no TProgressBar.



TProgressBar (Barra Progresso): Exemplo

```
procedure TForm2.BitBtn2Click(Sender: TObject);  
begin  
  
    ProgBar.position := 50;  
    ProgBar.min := 1;  
    ProgBar.max := 100;  
    ProgBar.enabled := TRUE;  
    ProgBar.State := pbsNormal;  
    ProgBar.Orientation := pbVertical;  
  
end;
```

Tratamento do evento **onClick** do botão
Define o **intervalo** da **barra** de **progresso**, seu **estado** e a **orientação**

Funções do DELPHI (Conceito)

- Toda **linguagem** de **programação** possui **funções**.
- Uma **função** nada mais é do que um **bloco de código** que **faz algo**.
 - Por exemplo, **showmessage** é uma **função** que **mostra** um **alerta**.
- Nós **programadores** podemos chamar **funções**:
 - Elas **tornarão** nossos **programas** mais "inteligentes".
 - Graças as **funções**, o **programador** poderá **fazer** mais **coisas**.
 - Graças as **funções**, será mais **fácil desenvolver** nossos **programas**.



Funções do Delphi (Número Aleatório)

- O Delphi possui uma função que gera números aleatórios.
- A função que gera números aleatórios é chamada de RandomRange.
- Para chamar a função, inclua em Uses a biblioteca Math.

```
uses  
  Winapi.Windows, Winapi.Messages, System.SysUtils, Math;
```



Funções do Delphi (Número Aleatório)

- RandomRange necessita receber dois valores:
 - Os valores definem o intervalo (fechado) onde o valor aleatório será gerado.
 - Os valores passados por parâmetro devem sempre ser inteiros (Integer).
- A função RandomRange sempre irá retornar um número Integer:
 - Dentro do intervalo definido pelos parâmetros passados a função.
- Valores aleatórios dentro de um intervalo numérico:
 - Serão utilizados em nossos próximos projetos.



Funções do Delphi (Número Aleatório) : Exemplo

```
procedure TForm1.Button1Click(Sender: TObject);  
var  
    numero: Integer;  
begin  
    // Número aleatório entre 10 e 20  
    numero := RandomRange(10,20)  
end;
```

Tratamento do **evento onChange** do **botão**
Gera um **número integer aleatório** entre **10** e **20**

Funções do DELPHI (Maiúsculo)

- O Delphi possui uma função que torna strings maiúsculas.
- A função que torna strings maiúsculas é chamada de UpperCase.
 - Observe que o valor a ser passado e retornado deve ser string.
- Basta passar a string como parâmetro para UpperCase:
 - Que a mesma irá retornar a String toda em maiúscula.



Funções do DELPHI (Minúsculo)

- O Delphi possui uma função que torna strings minúsculas.
- A função que torna strings minúsculas é chamada de LowerCase.
 - Observe que o valor a ser passado e retornado deve ser string.
- Basta passar a string como parâmetro para LowerCase :
 - Que a mesma irá retornar a String toda em minúscula.



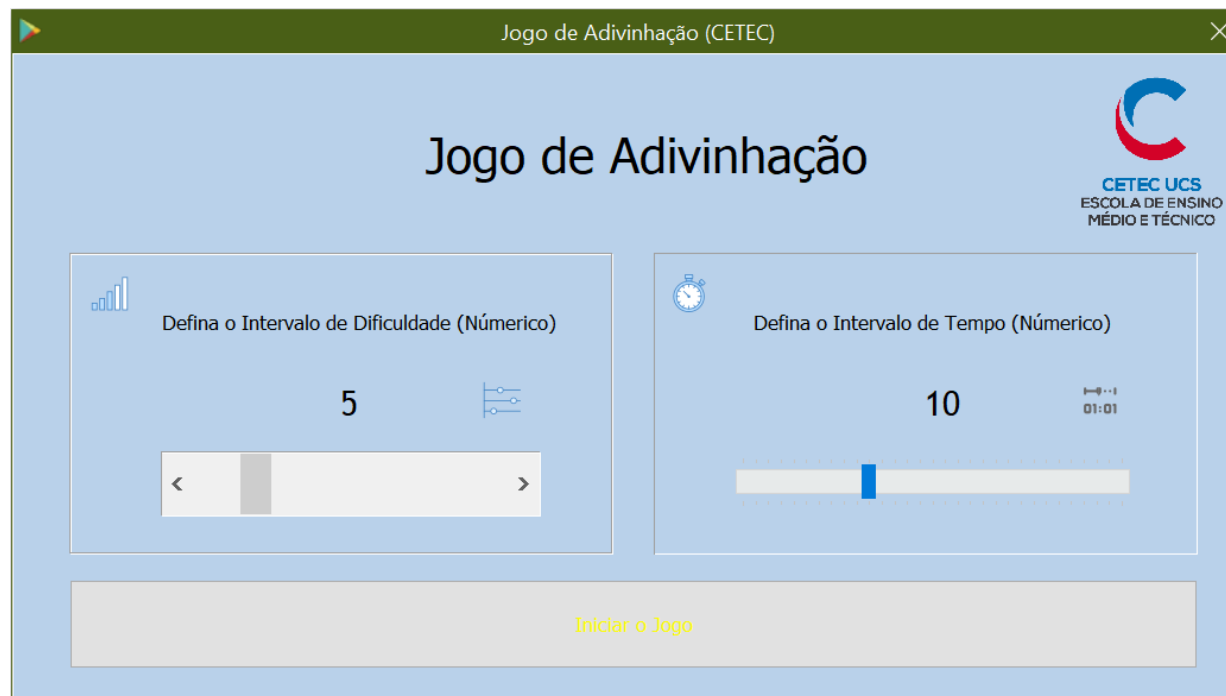
Funções do DELPHI (Maiúsculo/Minúsculo) : Exemplo

```
procedure TForm1.Button1Click(Sender: TObject);  
var  
    nome: string;  
begin  
  
    nome := UpperCase('RaFfAeL');  
    nome := LowerCase('RaFfAeL');  
  
end;
```

Tratamento do evento `onChange` do botão
Converte a `String` para `maiúscula` em `UpperCase` e para `minúscula` em `LowerCase`

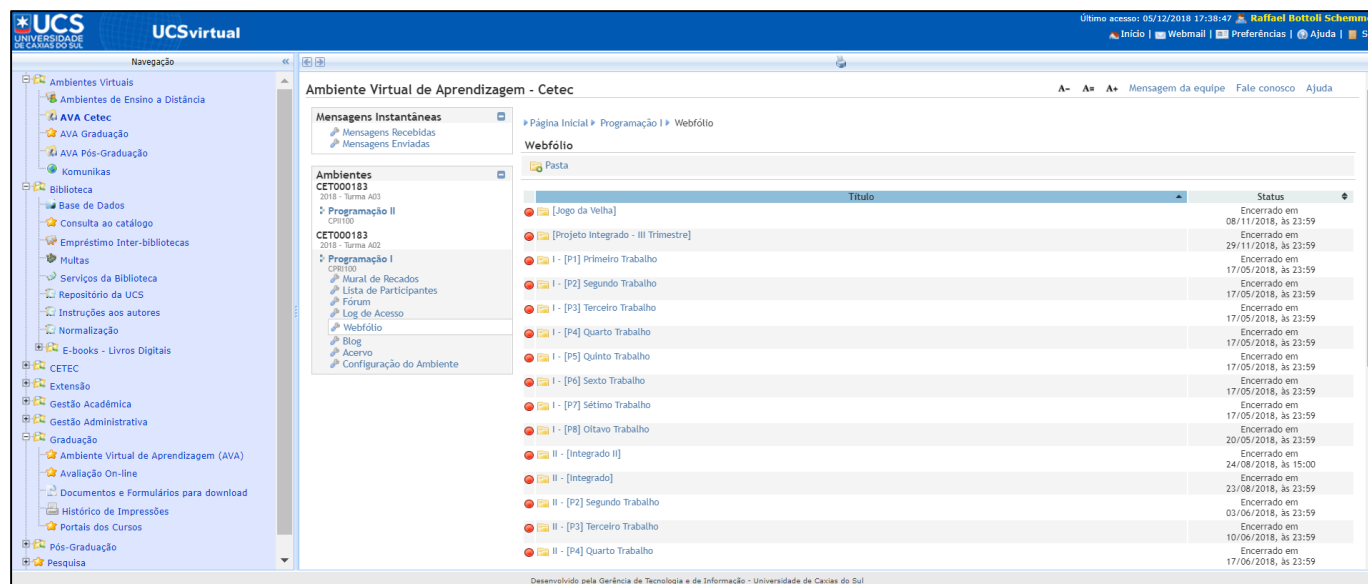
Decimo Segundo Trabalho da Disciplina (2TXII)

- Adivinhador de números:



Utilize o exemplo disponível do GitHub para desenvolver o trabalho

Publicação do TXII no AVA



Último acesso: 05/12/2018 17:38:47 **Rafael Bottoli Schiemmer**
Início | Webmail | Preferências | Ajuda | Sair

UCSvirtual

Ambiente Virtual de Aprendizagem - Cetec

Mensagens Instantâneas
Mensagens Recebidas
Mensagens Enviadas

Ambientes
CET000183
2018 - Turma A03
Programação II
CET000183
2018 - Turma A02
Programação I

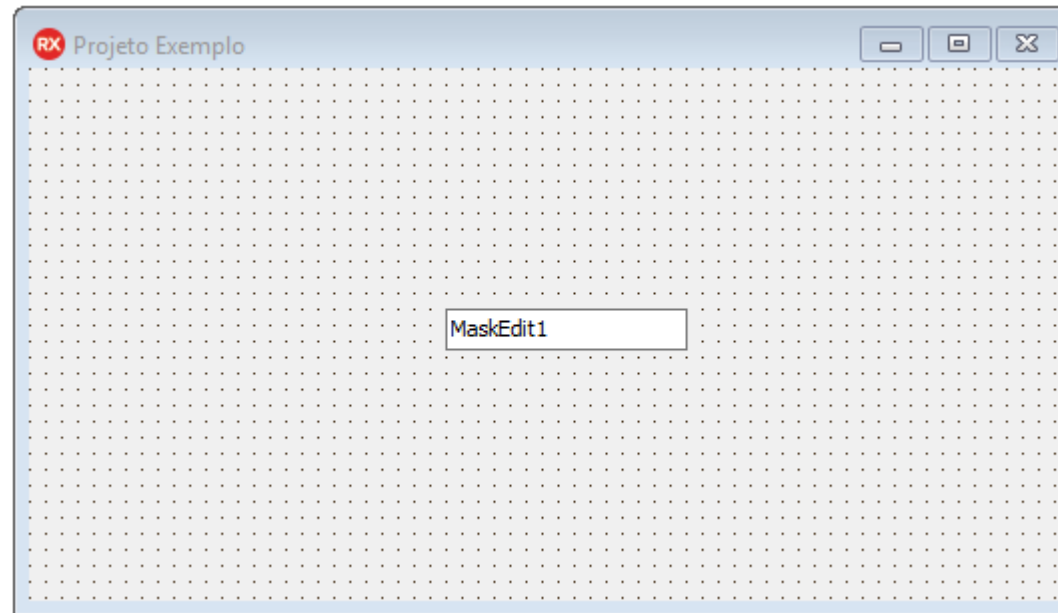
Webfólio

Título	Status
[Jogo da Velha]	Encerrado em 08/11/2018, às 23:59
[Projeto Integrado - III Trimestre]	Encerrado em 29/11/2018, às 23:59
I - [P1] Primeiro Trabalho	Encerrado em 17/05/2018, às 23:59
I - [P2] Segundo Trabalho	Encerrado em 17/05/2018, às 23:59
I - [P3] Terceiro Trabalho	Encerrado em 17/05/2018, às 23:59
I - [P4] Quarto Trabalho	Encerrado em 17/05/2018, às 23:59
I - [P5] Quinto Trabalho	Encerrado em 17/05/2018, às 23:59
I - [P6] Sexto Trabalho	Encerrado em 17/05/2018, às 23:59
I - [P7] Sétimo Trabalho	Encerrado em 17/05/2018, às 23:59
I - [P8] Oitavo Trabalho	Encerrado em 20/05/2018, às 23:59
II - [Integrado II]	Encerrado em 24/08/2018, às 15:00
II - [Integrado]	Encerrado em 23/08/2018, às 23:59
II - [P2] Segundo Trabalho	Encerrado em 03/06/2018, às 23:59
II - [P3] Terceiro Trabalho	Encerrado em 10/06/2018, às 23:59
II - [P4] Quarto Trabalho	Encerrado em 17/06/2018, às 23:59

Desenvolvido pela Gerência de Tecnologia e de Informação - Universidade de Caxias do Sul

Webfólio do Ambiente Virtual de Aprendizagem (CETEC)

TMaskEdit (Caixa Diálogo): Definição



Componente **TForm** com um **TMaskEdit**

TMaskEdit (Caixa Diálogo): Propriedades

- **Text:** Permite manipular o conteúdo do TMaskEdit.
 - O conteúdo desta **propriedade** é **sempre** uma **frase** (**String**).
- **TextHint:** Define um texto de dica para o componente (caso Text não existir).
 - O conteúdo desta **propriedade** é **sempre** uma **frase** (**String**).
- **Height:** Altura do TMaskEdit em pixel.
 - O conteúdo desta **propriedade** é sempre um **número** (**Integer**).
- **Width:** Largura do TMaskEdit em pixel.
 - O conteúdo desta **propriedade** é sempre um **número** (**Integer**).

TMaskEdit (Caixa Diálogo): Propriedades

- **ReadOnly**: Permite que o conteúdo do TMaskEdit seja somente lido.
 - O conteúdo desta **propriedade** é sempre um **boolean** (**true/false**).
- **MaxLength**: Comprimento máximo do TMaskEdit em caracteres.
 - O conteúdo desta **propriedade** é sempre um **número** (**Integer**).
- **PasswordChar**: Permite ocultar o texto com (*).
 - O conteúdo desta **propriedade** é **sempre** uma **frase** (**String**).
- **TabOrder**: Define a ordem que o componente será selecionado pelo TAB.
 - O conteúdo desta **propriedade** é sempre um **número** (**Integer**).

TMaskEdit (Caixa Diálogo): Propriedades

- **Visible:** Torna o componente ocultável.
 - O conteúdo desta **propriedade** é sempre um **boolean** (**true/false**).
- **Name:** Troca o nome do TMaskEdit (Dentro do Código Delphi).
 - O conteúdo desta **propriedade** é **sempre** uma **frase** (**String**).
- **Color:** Define a cor de fundo de um TMaskEdit.
 - O conteúdo desta **propriedade** é sempre uma **frase** (**String**) definida pelo **DELPHI**.
- **Hint:** Define uma caixa de dica para o componente.
 - O conteúdo desta **propriedade** é **sempre** uma **frase** (**String**).

TMaskEdit (Caixa Diálogo): Propriedades

- **ShowHint:** Habilita a dica.
 - O conteúdo desta **propriedade** é sempre um **boolean** (true/false).
- **Bevel:** (Inner/Kind/Outer): Permite estilizar as bordas do TMaskEdit.
 - **bkTile/bkSoft/bkFlat:** Define as **bordas** do **componente visual**
- **CharCase:** Permite definir se as letras serão em CAPS ou não.
 - O conteúdo desta **propriedade** é uma **constante** (ecNormal/ecLowerCase/ecUpperCase).
- **EditMask:** Define uma máscara fixa para o TMaskEdit.
 - Define uma **máscara** (**padrão**) para o **componente**.

TMaskEdit (Caixa Diálogo): Eventos

- **onChange**: Executa toda vez que o TMaskEdit for modificado.
- **onEnter**: Executa toda vez que o TAB entrar dentro do TMaskEdit.
- **onExit**: Executa toda vez que o TAB sair dentro do TMaskEdit.
- **onClick**: Executa toda vez que um clique for feito no TMaskEdit.



TMaskEdit (Caixa Diálogo): Eventos



- **onDblClick**: Executa toda vez que dois clicks forem feitos no TMaskEdit.
- **onMouseEnter**: Executa toda vez que o mouse entrar no TMaskEdit.
- **onMouseLeave**: Executa toda vez que o mouse sair no TMaskEdit.

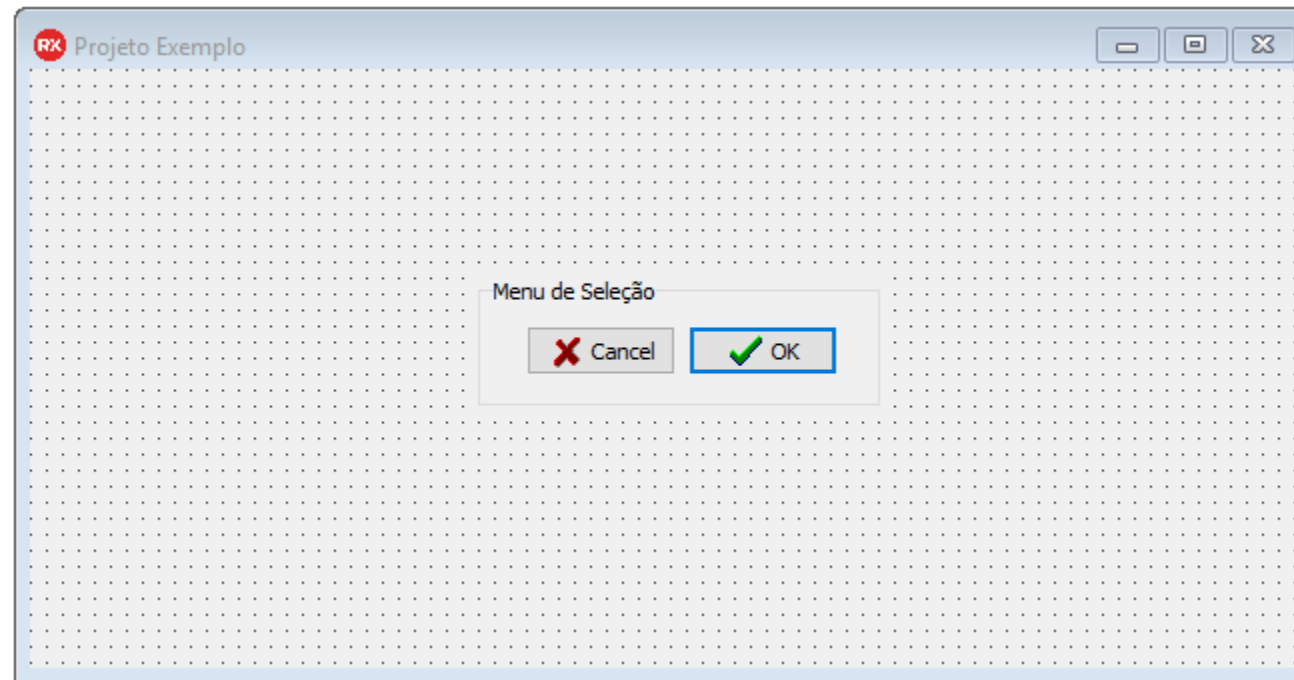


TMaskEdit (Caixa Diálogo): Exemplo

```
procedure TForm2.BtnClick(Sender: TObject);  
begin  
  
    MskEdt.MaxLength := 5;  
    MskEdt.ReadOnly := TRUE;  
    MskEdt.Width := 10;  
    MskEdt.Height := 50;  
    MskEdt.PasswordChar := '*';  
  
end;
```

Tratamento do evento `onClick` do botão
Define o conteúdo do `TMaskEdit` com tamanho de 5 posições e `PasswordChar` com '*'

TGroupBox (Container): Definição



Componente **TForm** com um **TGroupBox**

TGroupBox (Container): Propriedades

- **Name:** Define o nome do componente na Unit.
 - O conteúdo desta **propriedade** é **sempre** uma **frase** (**String**).
- **Caption:** Define o rótulo do TGroupBox.
 - O conteúdo desta **propriedade** é **sempre** uma **frase** (**String**).
- **Font:** Define a fonte e o tamanho do texto do TGroupBox.
 - Para **estilizar** esta **propriedade** você deve **pressionar** (..).
 - Nesta **propriedade** você pode **estilizar** a **fonte**, o **tamanho** da **fonte** e a **cor** da **fonte**.
- **Enabled:** Define se o TGroupBox será habilitado.
 - O conteúdo desta **propriedade** é sempre um **boolean** (**true/false**).
- **TabOrder:** Define a ordem de chamada do TAB no TGroupBox.
 - O conteúdo desta **propriedade** é sempre um **número** (**Integer**).

TGroupBox (Container): Propriedades

- **Hint:** Define uma dica para o TGroupBox.
 - O conteúdo desta **propriedade** é **sempre** uma **frase** (String).
- **ShowHint:** Define se a dica será mostrada ou não.
 - O conteúdo desta **propriedade** é sempre um **boolean** (true/false).
- **Visible:** Define se o botão será visível ou não.
 - O conteúdo desta **propriedade** é sempre um **boolean** (true/false).
- **Height:** Altura do TGroupBox em pixel.
 - O conteúdo desta **propriedade** é sempre um **número** (Integer).
- **Width:** Largura do TGroupBox em pixel.
 - O conteúdo desta **propriedade** é sempre um **número** (Integer).

TGroupBox (Container): Eventos

- **onEnter**: Executa toda vez que o TAB entrar dentro do TGroupBox.
- **onExit**: Executa toda vez que o TAB sair dentro do TGroupBox.
- **onClick**: Executa toda vez que um clique for feito no TGroupBox.

TGroupBox (Container): Eventos

- **onDbClick**: Executa toda vez que dois clicks forem feitos no TGroupBox.
- **onMouseEnter**: Executa toda vez que o mouse entrar no TGroupBox.
- **onMouseLeave**: Executa toda vez que o mouse sair no TGroupBox.

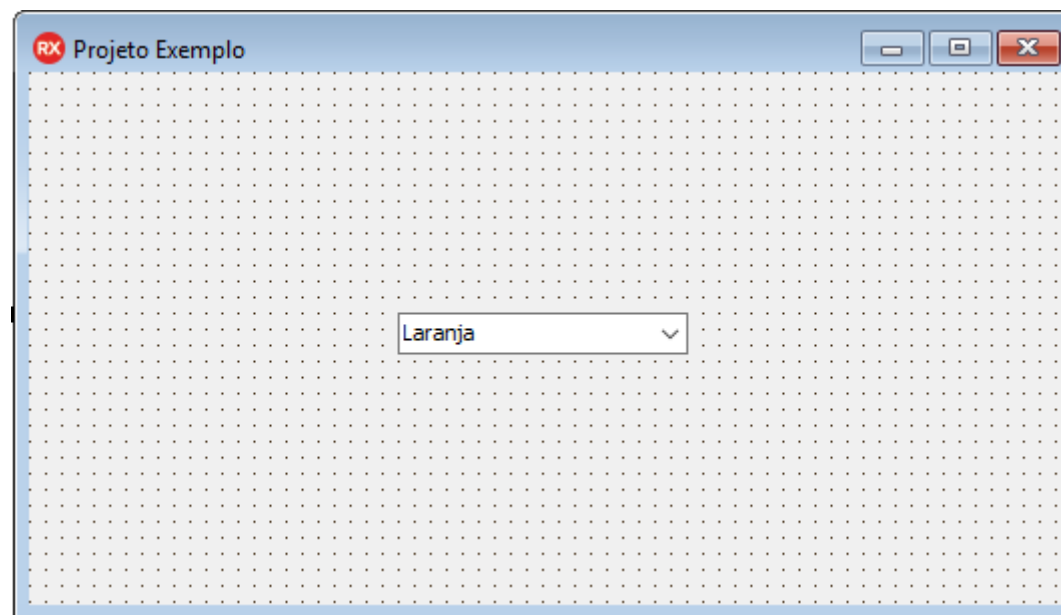


TGroupBox (Container): Exemplo

```
procedure TForm18.BtnClick(Sender: TObject);  
begin  
  
    GrpBox.Caption := 'Grupo das Frutas';  
    GrpBox.Font.Size := 10;  
    GrpBox.Font.Color := clBlue;  
    GrpBox.TabOrder := 2;  
    GrpBox.Visible := TRUE;  
  
end;
```

Tratamento do evento do **onClick** do **botão**
Define o conteúdo do **TGroupBox** como o **Caption** e a **cor** e o **tamanho** da **fonte**

TComboBox (Lista Combinada): Definição



Componente **TForm** com um **TComboBox**

TComboBox (Lista Combinada): Propriedades

- **Name:** Define o nome do componente na Unit.
 - O conteúdo desta **propriedade** é **sempre** uma **frase** (**String**).
- **Color:** Define a cor interna do componente.
 - O conteúdo desta **propriedade** é sempre uma **frase** (**String**) definida pelo **DELPHI**.
- **Width:** Define a largura do componente.
 - O conteúdo desta **propriedade** é sempre um **número** (**Integer**).
- **Height:** Define a altura do componente.
 - O conteúdo desta **propriedade** é sempre um **número** (**Integer**).
- **Font:** Define a fonte e o tamanho da lista.
 - Para **estilizar** esta **propriedade** você deve **pressionar** (..).
 - Nesta **propriedade** você pode **estilizar** a **fonte**, o **tamanho** da **fonte** e a **cor** da **fonte**.

TComboBox (Lista Combinada): Propriedades

- **Enabled**: Define se a lista será habilitada.
 - O conteúdo desta **propriedade** é sempre um **boolean** (**true/false**).
- **TabOrder**: Define a ordem de chamada do TAB no botão.
 - O conteúdo desta **propriedade** é sempre um **número** (**Integer**).
- **Hint**: Define uma dica para o botão.
 - O conteúdo desta **propriedade** é **sempre** uma **frase** (**String**).
- **ShowHint**: Define se a dica será mostrada ou não.
 - O conteúdo desta **propriedade** é sempre um **boolean** (**true/false**).

TComboBox (Lista Combinada): Propriedades

- **Visible**: Define se o botão será visível ou não.
 - O conteúdo desta **propriedade** é sempre um **boolean** (**true/false**).
- **Style**: Define o estilo do ComboBox.
 - O **conteúdo** é uma **constante** como (**csDropDownList/csDropDown**).
- **DropDownCount**: Define o tamanho da lista do ComboBox.
 - O conteúdo desta **propriedade** é sempre um **número** (**Integer**).
- **Sorted**: Ordena em ordem alfabética os itens.
 - O conteúdo desta **propriedade** é sempre um **boolean** (**true/false**).

TComboBox (Lista Combinada): Eventos



- **onEnter**: Executa toda vez que o TAB entrar dentro do TComboBox.
- **onExit**: Executa toda vez que o TAB sair dentro do TComboBox.
- **onClick**: Executa toda vez que um clique for feito no TComboBox.



TComboBox (Lista Combinada): Funções

- **cbox.Items.add(item:string)**
 - Permite adicionar um valor string na lista combinada (sempre no final).
- **cbox.Items.Insert(indice:integer, item:string):**
 - Permite adicionar um valor em um índice específico.
- **cbox.Items.Delete(indice:integer):**
 - Permite deletar um valor em um índice (Integer) específico.
- **cbox.Items.Move(PosCorrente:integer, NovaPos:integer):**
 - Move os itens entre posições.

TComboBox (Lista Combinada): Funções

- **cbox.Items.Count:**
 - Retorna a quantidade de itens presentes dentro da lista combinada.
- **cbox.Clear:**
 - Limpa toda a lista combinada.
- **cbox.Items[I]:**
 - Captura o conteúdo (string) do índice I da lista combinada.
- **cbox.itemIndex:**
 - Captura o índice selecionado pelo click do usuário na lista combinada.



TComboBox (Lista Combinada): Eventos

- **onSelect**: Executa toda vez que um valor for selecionado no TComboBox.
- **onDbClick**: Executa toda vez que dois clicks forem feitos no TComboBox.
- **onMouseEnter**: Executa toda vez que o mouse entrar no TComboBox.
- **onMouseLeave**: Executa toda vez que o mouse sair no TComboBox.

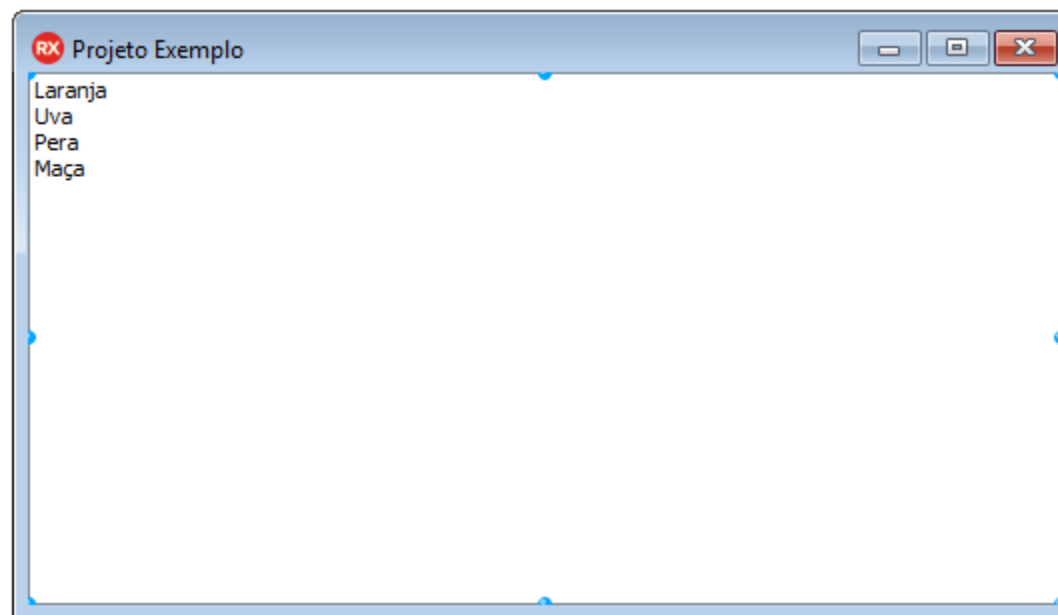


TComboBox (Lista Combinada): Exemplo

```
procedure TFrmPrincipal.ComboSelect(Sender: TObject);  
begin  
    showmessage('Selecionou : ' + combo.Items[combo.ItemIndex]);  
end;
```

Tratamento do **evento onSelect** do **ComboBox**
Captura o **conteúdo** de **Items** utilizando o **ItemIndex** selecionado

TListBox (Lista): Definição



Componente `TForm` com um `TListBox`

TListBox (Lista): Propriedades

- **Name:** Define o nome do componente na Unit.
 - O conteúdo desta **propriedade** é **sempre** uma **frase** (**String**).
- **Color:** Define a cor interna do componente.
 - O conteúdo desta **propriedade** é sempre uma **frase** (**String**) definida pelo **DELPHI**.
- **Width:** Define a largura do componente.
 - O conteúdo desta **propriedade** é sempre um **número** (**Integer**).
- **Height:** Define a altura do componente.
 - O conteúdo desta **propriedade** é sempre um **número** (**Integer**).
- **Font:** Define a fonte e o tamanho da lista.
 - Para **estilizar** esta **propriedade** você deve **pressionar** (..).
 - Nesta **propriedade** você pode **estilizar** a **fonte**, o **tamanho** da **fonte** e a **cor** da **fonte**.

TListBox (Lista): Propriedades

- **Enabled**: Define se a lista será habilitada.
 - O conteúdo desta **propriedade** é sempre um **boolean** (**true/false**).
- **TabOrder**: Define a ordem de chamada do TAB no botão.
 - O conteúdo desta **propriedade** é sempre um **número** (**Integer**).
- **Hint**: Define uma dica para o botão.
 - O conteúdo desta **propriedade** é **sempre** uma **frase** (**String**).
- **ShowHint**: Define se a dica será mostrada ou não.
 - O conteúdo desta **propriedade** é sempre um **boolean** (**true/false**).
- **Visible**: Define se o botão será visível ou não.
 - O conteúdo desta **propriedade** é sempre um **boolean** (**true/false**).

TListBox (Lista): Funções

- `lista.Items.add(item:string)`
 - Permite adicionar um valor `string` na `lista` (sempre no `final`).
- `lista.Items.Insert(indice:integer, item:string):`
 - Permite adicionar um valor em um índice específico.
- `lista.Items.Delete(indice:integer):`
 - Permite deletar um valor em um índice (`Integer`) específico.
- `lista.Items.Move(PosCorrente:integer, NovaPos:integer):`
 - Move os itens entre posições.

TListBox (Lista): Funções

- **lista.Items.Count:**
 - Retorna a quantidade de itens presentes dentro da lista.
- **lista.Clear:**
 - Limpa toda a lista.
- **lista.Items[I]:**
 - Captura o conteúdo (string) do índice I da lista.
- **lista.itemIndex:**
 - Captura o índice selecionado pelo click do usuário na lista.



TListBox (Lista): Eventos

- **onEnter**: Executa toda vez que o TAB entrar dentro do TListBox.
- **onExit**: Executa toda vez que o TAB sair dentro do TListBox.
- **onClick**: Executa toda vez que um clique for feito no TListBox.

TListBox (Lista): Eventos

- **onData**: Executa toda vez que um dado for selecionado no TListBox.
- **onDbClick**: Executa toda vez que dois clicks forem feitos no TListBox.
- **onMouseEnter**: Executa toda vez que o mouse entrar no TListBox.
- **onMouseLeave**: Executa toda vez que o mouse sair no TListBox.



TListBox (Lista): Exemplo

```
var
  I: Integer;
begin
  Lista.Clear;

  // Adiciona 10 números a lista
  FOR I := 1 TO 10 DO
  BEGIN
    Lista.Items.Add(inttostr(I));
  END;

  // Mostra o conteúdo da lista
  I := 0;
  WHILE (I < Lista.Items.Count) DO
  BEGIN
    showmessage(Lista.Items[I]);
    I := I + 1;
  END;
end;
```

Adiciona 10 valores na lista
Consulta os elementos da lista e mostra na tela

TStringList (Lista de String)

- TStringList é um objeto não visual:
 - Ele representa um objeto Delphi.
 - Ele implementa uma estrutura de dados vetorial.
- Observe que a propriedade item de ComboBox, ListBox e RadioGroup:
 - Utiliza esta estrutura de dados (similar a um vetor de strings).
- Um TStringList pode ser declarado de forma similar a uma variável:
 - Nós devemos criar um objeto antes de utilizá-lo.
 - Em seguida, você poderá utilizar as funções clássicas para inicializar e acessar a estrutura.
- O slide a seguir, demonstra um exemplo de como utilizar a estrutura.

TStringList (Lista de String)

Declarando Estrutura



Criando Estrutura



Atribuindo Valores



ComboBox
Recebe TStringList



```
procedure TForm1.FormCreate(Sender: TObject);  
  
var  
  
    bancodedados: TStringList;  
  
begin  
  
    bancodedados := TStringList.Create();  
    bancodedados.Clear;  
    bancodedados.Add('laranja');  
    bancodedados.Add('melancia');  
    bancodedados.Add('pera');  
  
    fruteira.Items := bancodedados;  
  
end;
```


TStringList (Lista de String): Funções

- `lista.Items.add(item:string)`
 - Permite adicionar um valor string na lista (sempre no final).
- `lista.Items.Insert(indice:integer, item:string):`
 - Permite adicionar um valor em um índice específico.
- `lista.Items.Delete(indice:integer):`
 - Permite deletar um valor em um índice (Integer) específico.
- `lista.Items.Move(PosCorrente:integer, NovaPos:integer):`
 - Move os itens entre posições.

TStringList (Lista de String): Funções

- **lista.Items.Count:**
 - Retorna a quantidade de itens presentes dentro da lista.
- **lista.Clear:**
 - Limpa toda a lista.
- **lista.Items[I]:**
 - Captura o conteúdo (string) do índice I da lista.
- **lista.itemIndex:**
 - Captura o índice selecionado pelo click do usuário na lista.

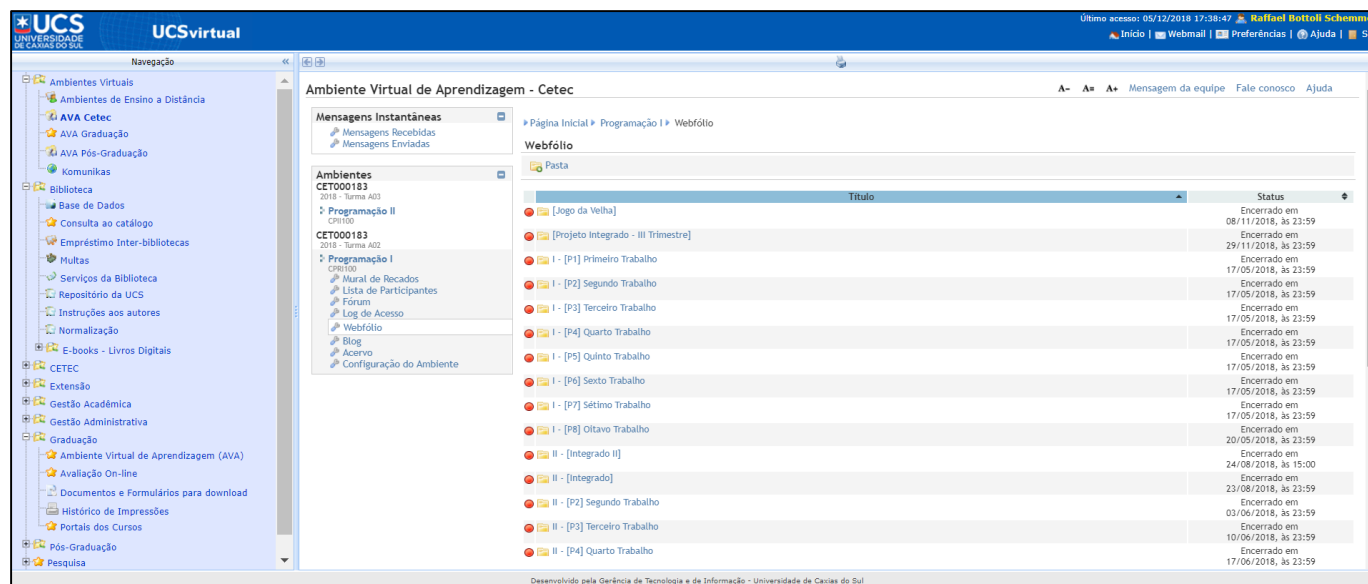
Decimo Terceiro Trabalho da Disciplina (2TXIII)

- Carrinho de Compras:



Utilize o exemplo disponível do GitHub para desenvolver o trabalho

Publicação do TXIII no AVA



Último acesso: 05/12/2018 17:38:47 **Rafael Bottoli Schiemmer**
Início | Webmail | Preferências | Ajuda | Sair

UCSvirtual

Ambiente Virtual de Aprendizagem - Cetec

Mensagens Instantâneas
Mensagens Recebidas
Mensagens Enviadas

Ambientes
CETO00183
2018 - Turma A03
Programação II
CETO00183
2018 - Turma A02
Programação I

Webfólio

Título	Status
[Jogo da Velha]	Encerrado em 08/11/2018, às 23:59
[Projeto Integrado - III Trimestre]	Encerrado em 29/11/2018, às 23:59
I - [P1] Primeiro Trabalho	Encerrado em 17/05/2018, às 23:59
I - [P2] Segundo Trabalho	Encerrado em 17/05/2018, às 23:59
I - [P3] Terceiro Trabalho	Encerrado em 17/05/2018, às 23:59
I - [P4] Quarto Trabalho	Encerrado em 17/05/2018, às 23:59
I - [P5] Quinto Trabalho	Encerrado em 17/05/2018, às 23:59
I - [P6] Sexto Trabalho	Encerrado em 17/05/2018, às 23:59
I - [P7] Sétimo Trabalho	Encerrado em 17/05/2018, às 23:59
I - [P8] Oitavo Trabalho	Encerrado em 20/05/2018, às 23:59
II - [Integrado II]	Encerrado em 24/08/2018, às 15:00
II - [Integrado]	Encerrado em 23/08/2018, às 23:59
II - [P2] Segundo Trabalho	Encerrado em 03/06/2018, às 23:59
II - [P3] Terceiro Trabalho	Encerrado em 10/06/2018, às 23:59
II - [P4] Quarto Trabalho	Encerrado em 17/06/2018, às 23:59

Webfólio do Ambiente Virtual de Aprendizagem (CETEC)

Avaliação Kahoot! (Junho)



Prova Mensal de Conceitos!

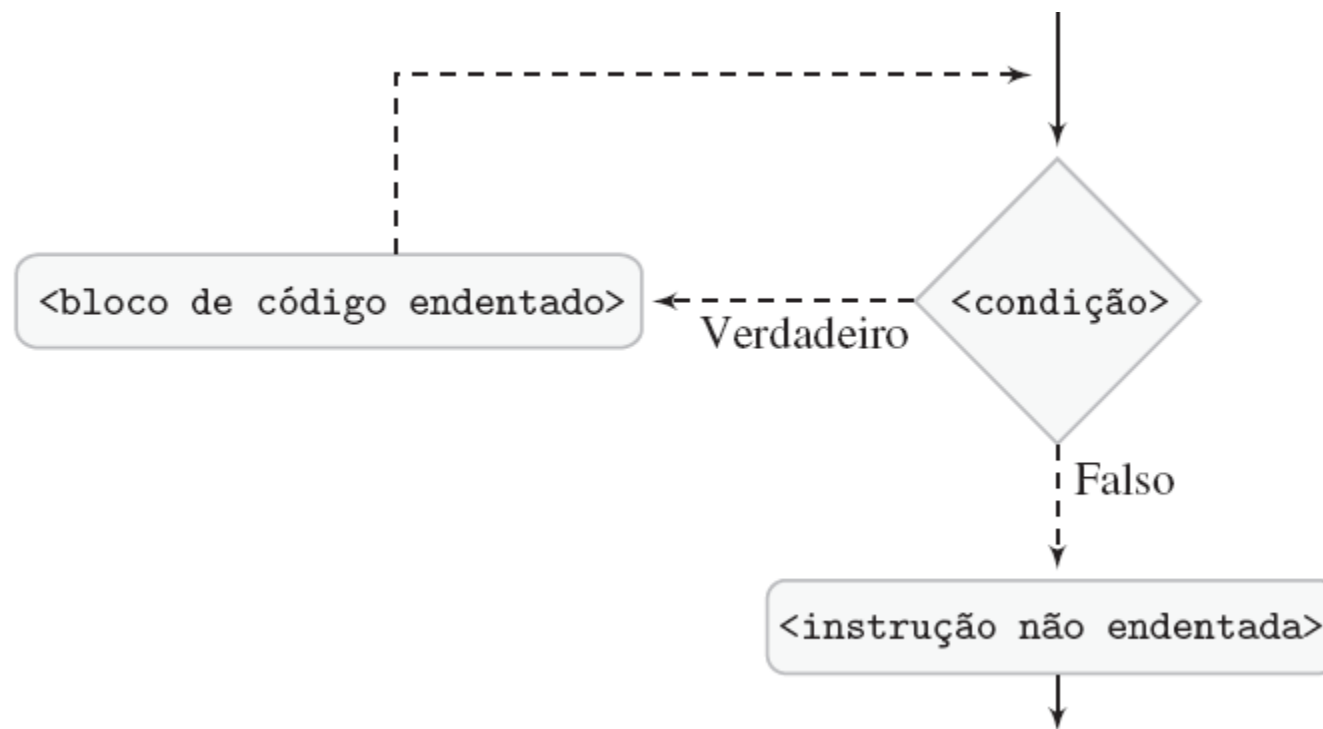


(10 Questões sobre as 4 Semanas de Junho)



Estruturas Iterativas

- Permitem **executar** um **bloco** de **código** N vezes:
 - A quantidade de **repetições** pode ser **definida** pelo **usuário**.

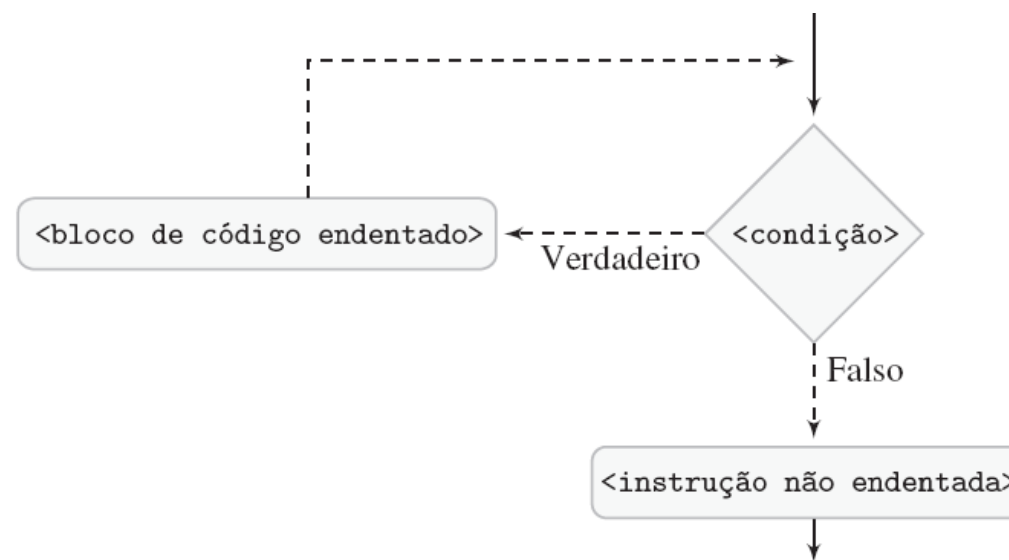


Estruturas Iterativas

- Laço **WHILE** (Estrutura Pré-Testada):
 - Executa o bloco **enquanto** condição for **TRUE**.

```
var cont : Integer;  
begin  
    cont := 1;  
    WHILE (cont <= 10) DO  
    BEGIN  
        cont := cont + 1;  
    END;  
end;
```

Estrutura **iterativa** que executa 10 vezes

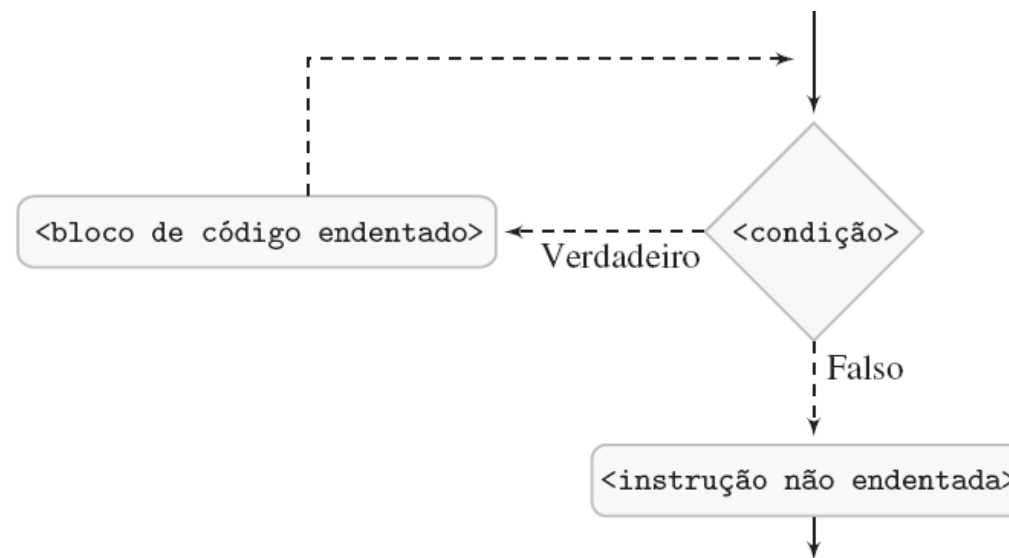


Estruturas Iterativas

- Laço **FOR** (Pré-Testada):
 - Executa o **bloco** dentro do intervalo de **TO** ou **DOWNTO**

```
var cont : Integer;  
  
begin  
    for cont := 1 TO 10 DO  
        BEGIN  
        END;  
    for cont := 10 DOWNT0 0 DO  
        BEGIN  
        END;  
end;
```

Estrutura **iterativa** que executa **10 vezes**

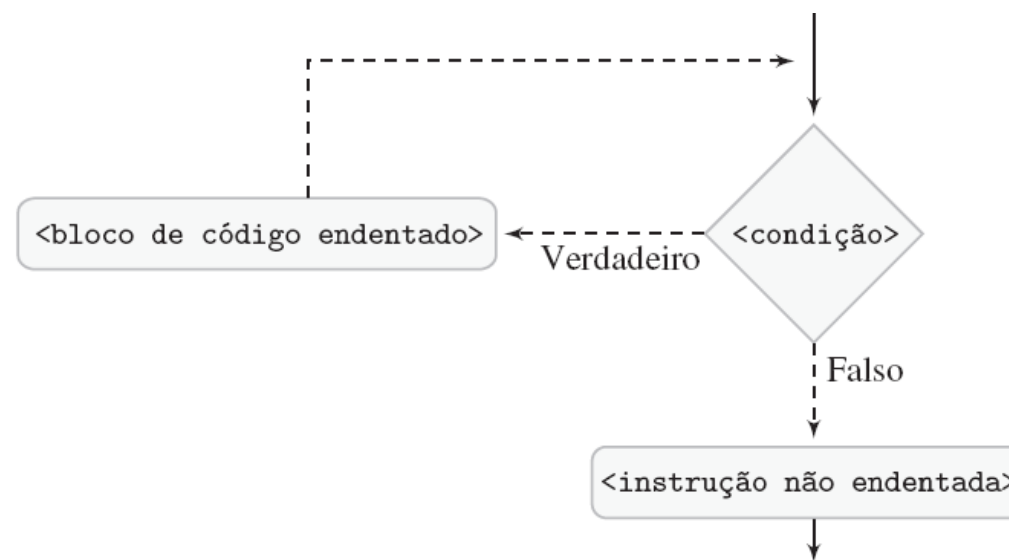


Estruturas Iterativas

- Laço **REPEAT** (Estrutura Pós-Testada):
 - Executa o **bloco** enquanto condição for **FALSE**.

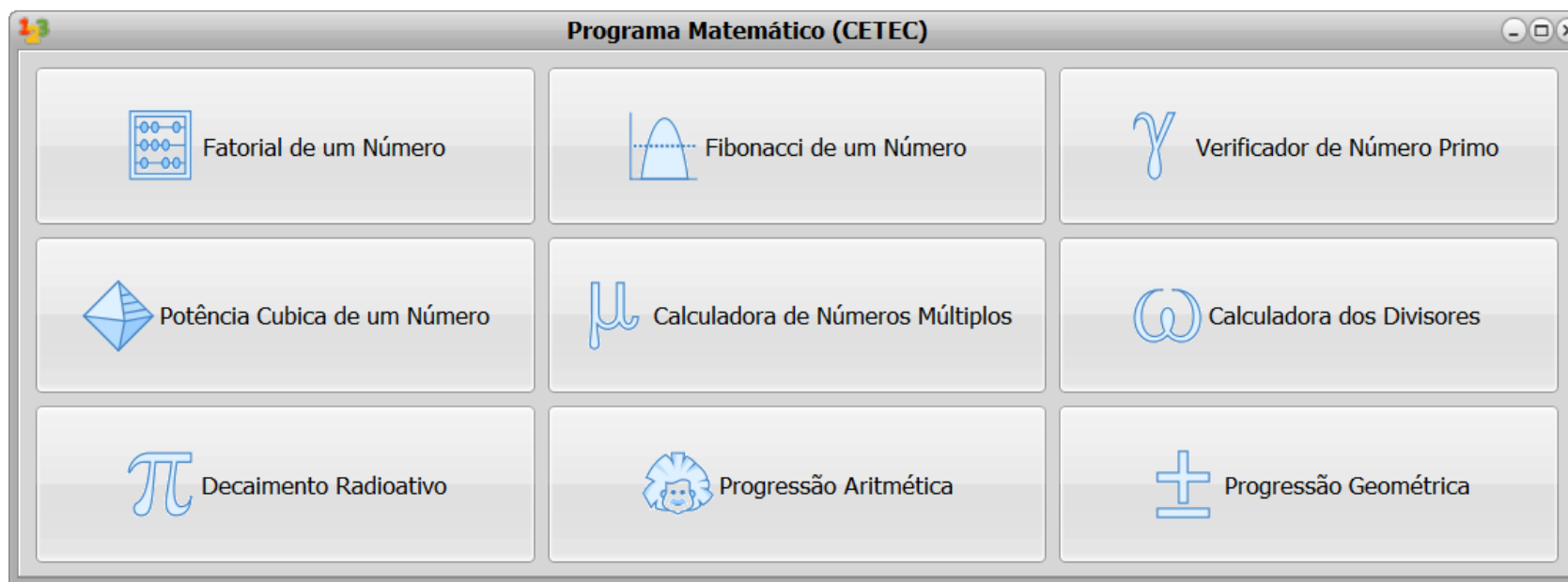
```
var cont : Integer;  
  
begin  
    cont := 1;  
    REPEAT  
        cont := cont + 1;  
    UNTIL cont = 5;  
end;
```

Estrutura **iterativa** que executa **5 vezes**



Decimo Quarto Trabalho da Disciplina (2TXIV)

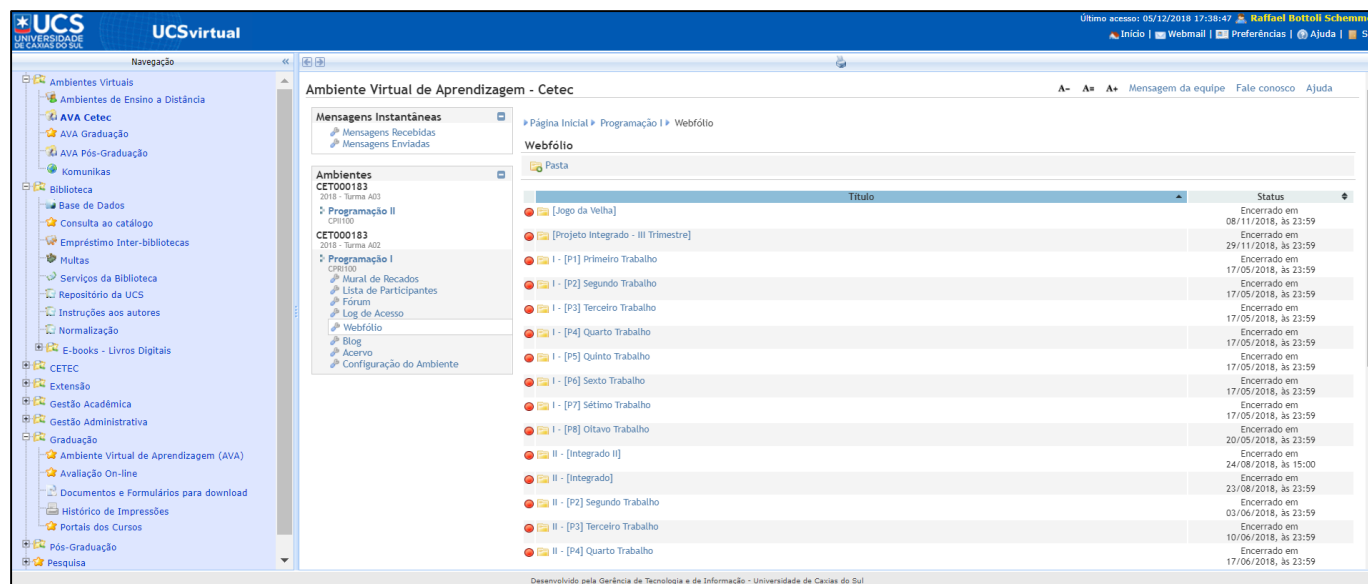
- Programa Matemático:



Utilize o exemplo disponível do GitHub para desenvolver o trabalho



Publicação do TXIV no AVA



Último acesso: 05/12/2018 17:38:47 **Rafael Bottoli Schiemer**
Início | Webmail | Preferências | Ajuda | Sair

UCSvirtual

Ambiente Virtual de Aprendizagem - Cetec

Mensagens Instantâneas
Mensagens Recebidas
Mensagens Enviadas

Ambientes
CET000183
2018 - Turno A02
Programação II
CET000183
2018 - Turno A02
Programação I

Webfólio

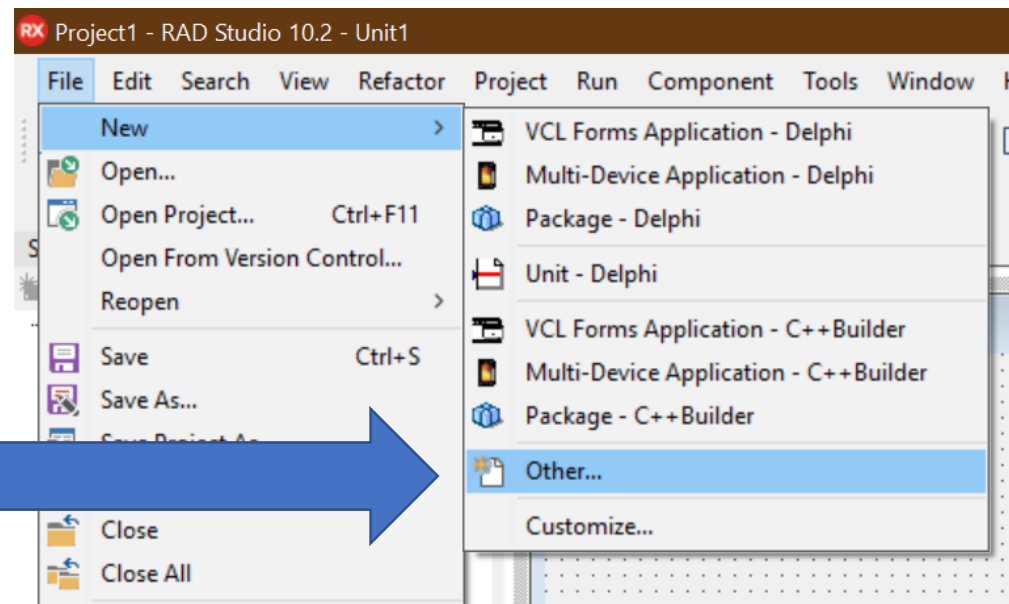
Título	Status
[Jogo da Velha]	Encerrado em 08/11/2018, às 23:59
[Projeto Integrado - III Trimestre]	Encerrado em 29/11/2018, às 23:59
I - [P1] Primeiro Trabalho	Encerrado em 17/05/2018, às 23:59
I - [P2] Segundo Trabalho	Encerrado em 17/05/2018, às 23:59
I - [P3] Terceiro Trabalho	Encerrado em 17/05/2018, às 23:59
I - [P4] Quarto Trabalho	Encerrado em 17/05/2018, às 23:59
I - [P5] Quinto Trabalho	Encerrado em 17/05/2018, às 23:59
I - [P6] Sexto Trabalho	Encerrado em 17/05/2018, às 23:59
I - [P7] Sétimo Trabalho	Encerrado em 17/05/2018, às 23:59
I - [P8] Oitavo Trabalho	Encerrado em 20/05/2018, às 23:59
II - [Integrado II]	Encerrado em 24/08/2018, às 15:00
II - [Integrado]	Encerrado em 23/08/2018, às 23:59
II - [P2] Segundo Trabalho	Encerrado em 03/06/2018, às 23:59
II - [P3] Terceiro Trabalho	Encerrado em 10/06/2018, às 23:59
II - [P4] Quarto Trabalho	Encerrado em 17/06/2018, às 23:59

Webfólio do Ambiente Virtual de Aprendizagem (CETEC)

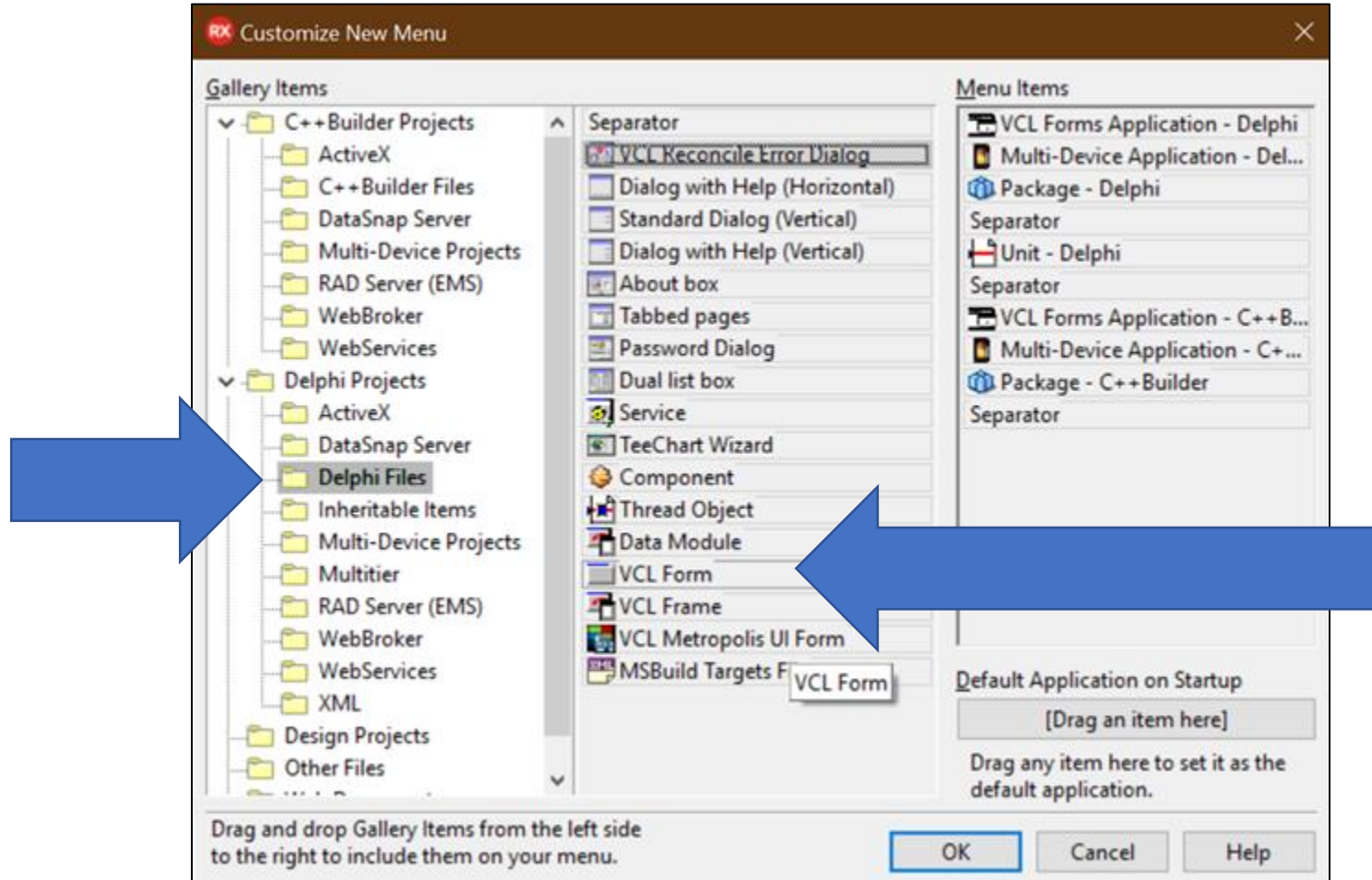
Multi Form (Projeto Multi Formulários)

- Todos os projetos podem possuir vários TFormMs
 - Cada formulário precisa ser tratado de forma individual.
- Criando um novo formulário:
 - File > New > Other ..

Criando um novo VCL Form



Multi Form (Projeto Multi Formulários)



Multi Form (Projeto Multi Formulários)

- Conceitos importantes:
 - Cada **TFORM** possui sua **estrutura própria Unt/Dfm**.
- Cabe a nós **programadores pensarmos**:
 - Na **estrutura** do **programa** em **questão** das **telas** da **aplicação**.
- Devemos ter em **mente** a **estrutura** de chamada das **telas**:
 - Aprenderemos a definir um dos **TFORMs** como o **FORM principal**.
 - Ele será **chamado** quando o **EXE** for **disparado**.

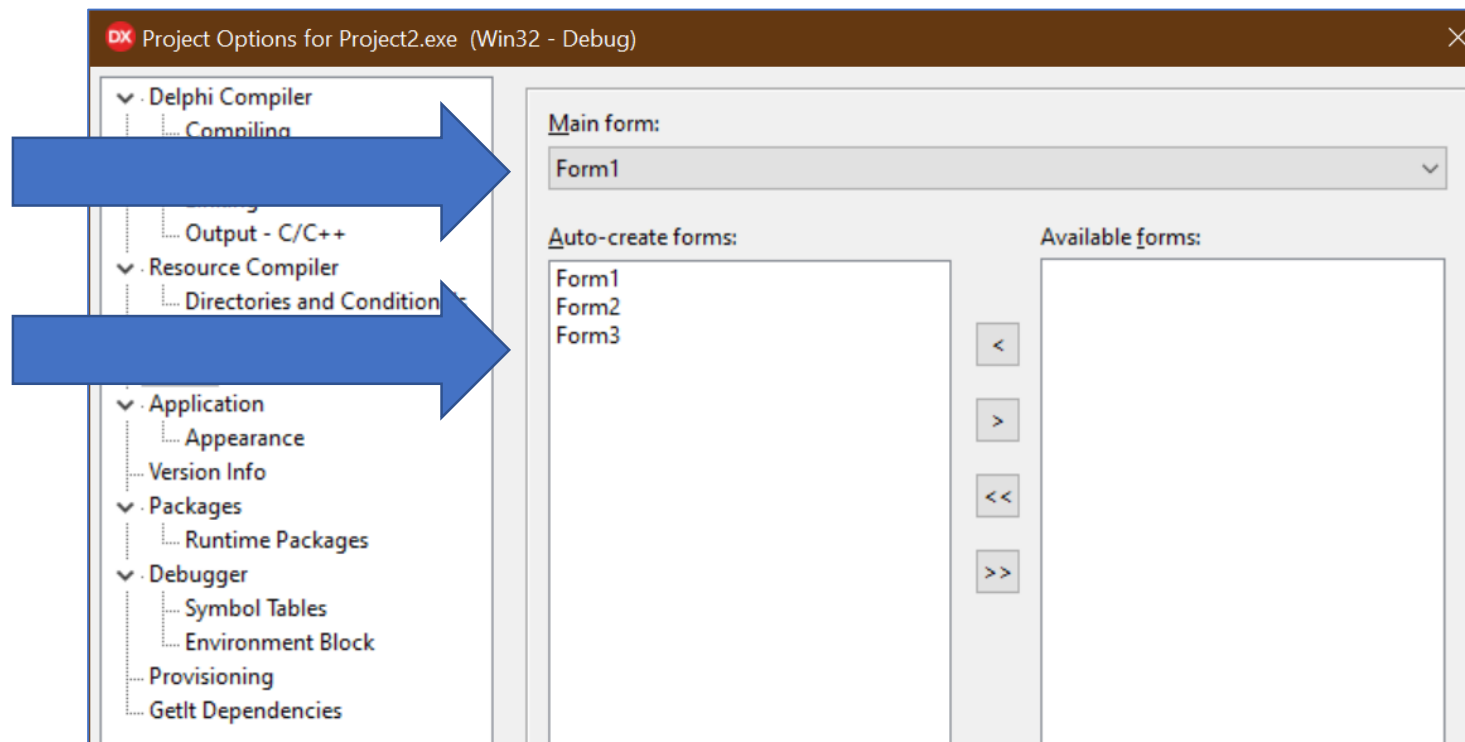


Multi Form (Projeto Multi Formulários)

- Definindo o formulário principal:
 - Project > Options > Form ..

Form1 será o
primeiro a ser chamado

Todos os Forms
Serão criados no início

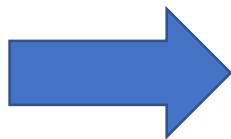


Multi Form (Projeto Multi Formulários)

- Observe um **aspecto importante**:
 - Todos os **TForms** serão **criados** e **inicializados**.
 - Porém apenas o **Form1** irá **aparecer** na tela.
- A partir deste **Form** você **chamará** os **demaís**:
 - Conforme sua **aplicação** fizer **uso** dos **demaís Forms**.
- Para que um **TForm** consiga **enxergar** o **outro**:
 - Você terá que **incluir** a clausula **uses** depois de **implementation**.
 - **Chamando** o nome da **Unit** do **TForm** desejado.

Multi Form (Projeto Multi Formulários)

FrmLogin passa a ter
acesso a FrmCadastro



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

var
    FrmLogin: TFrmLogin;

implementation

uses
    UntCadastro;

{$R *.dfm}
```

Multi Form (Projeto Multi Formulários)

- Para chamar um **TForm vizinho**:
 - Você pode utilizar o **Visible** tornando o **TForm** atual **Visible falso**.
 - E **tornando** o outro **TForm Visible true**.
- Porém, os eventos **onHide** e **onShow** são mais **úteis**:
 - Chamando **onHide** o **TForm** será **apagado** (e chamará o **onHide**).
 - Chamando **onShow**, o **TForm** será **ligado** (e chamará o **onShow**).
- Existe ainda o **showModal**, que liga o **próximo TForm**:
 - E deixa o **anterior visível**, porém **inacessível**.

Multi Form (Projeto Multi Formulários)

- Entenda apenas que:
 - Se você tornar tudo invisível, o EXE continuará executando.
 - O usuário não terá como realizar uma operação para fechar o programa.
- O TForm principal é pai dos demais TForms:
 - Chamando onClose dele, todos os demais TForms serão finalizados.
 - E o executável (EXE) sairá do gerenciador de tarefas.
- Caso você chame onClose de um TForm filho:
 - Este TForm irá morrer, porém seu pai, continuará vivo (executando).
 - Cabe a você programador, evitar que isso aconteça.



Multi Form (Projeto Multi Formulários)

- Caso **you** **mate** um **filho** com **onClose**:
 - Não poderá chamar novamente o TForm do **pai**, pois ele **estará morto**.
- Desta forma, **you** **deverá** **realizar**:
 - `FrmFilho := TFrmFilho.Create(Self);`
- Isso **recria** o **objeto** do **FrmFilho**:
 - Permitindo que o **mesmo** seja **chamado** com **onShow**;

Multi Form (Projeto Multi Formulários)

```
procedure TFrmLogin.Button1Click(Sender: TObject);  
begin  
  
    FrmLogin.Hide;  
    FrmCadastro := TFrmCadastro.Create(Self);  
    FrmCadastro.show;  
  
end;
```

Procedimento que **oculta** o **FrmLogin** (Pai) e **cria/chama** o **FrmCadastro** (Filho)

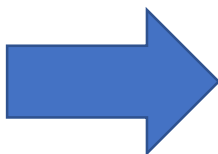
Multi Form (Projeto Multi Formulários)

- Como **permitir** que **dois TForms troquem mensagens?**
 - Utilizando **variáveis globais** entre **dois TForms**.
- As **variáveis globais** devem ser **declaradas** entre **uses** e **type**.
- O **TForm** que for **acessar** a **variável** deve ter **acesso** ao **TForm**:
 - Adicionando a **clausula uses** após **implementation**.



Multi Form (Projeto Multi Formulários)

Variável Global



```
uses
    Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants,
    System.Classes, Vcl.Graphics,
    Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.StdCtrls;

var vglobal : string;

type
    TFrmLogin = class(TForm)
        fruteira: TComboBox;
        Button1: TButton;
        procedure FormCreate(Sender: TObject);
        procedure Button1Click(Sender: TObject);
    end;
```

Avaliação Kahoot! (Julho)



Prova Mensal de Conceitos!



(10 Questões sobre as 2 Semanas de Julho)



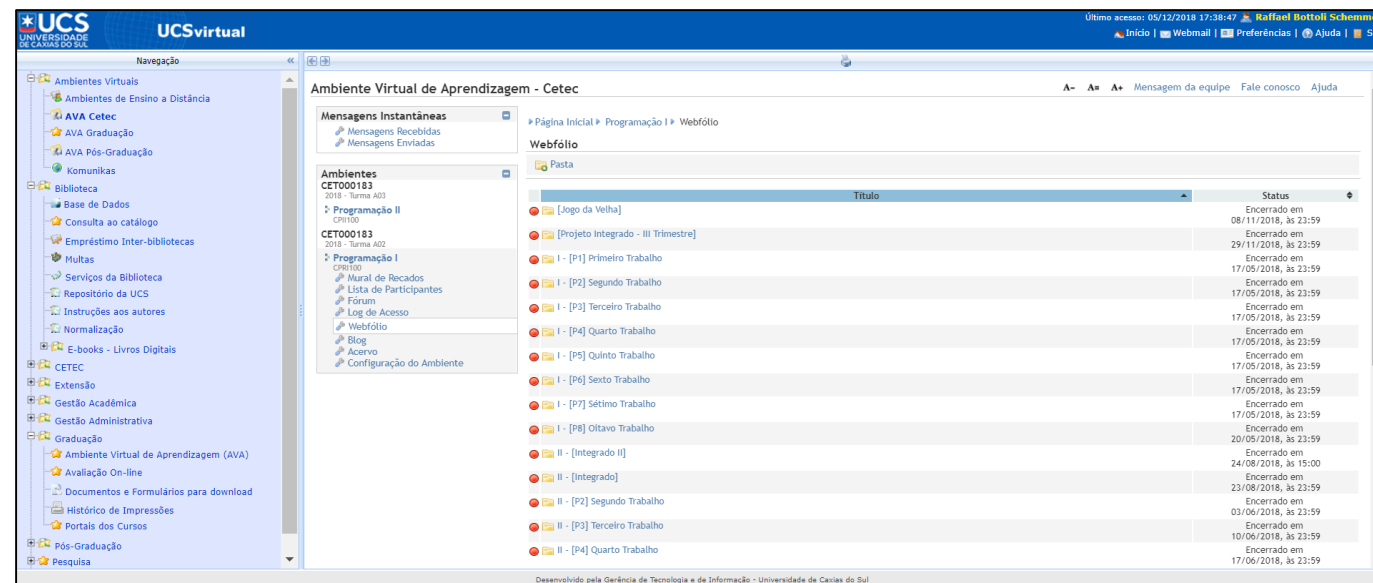
Decimo Quinto Trabalho da Disciplina (2TXV)



Alunos devem definir/executar um projeto Delphi
(Utilizando todos os componentes VCL da disciplina)



Publicação do TXV no AVA



UCSvirtual

Último acesso: 05/12/2018 17:38:47 **Rafael Bottoli Schiemmer**

[Início](#) | [Webmail](#) | [Preferências](#) | [Ajuda](#) | [Sair](#)

Ambiente Virtual de Aprendizagem - Cetec

Mensagens Instantâneas

- Mensagens Recebidas
- Mensagens Enviadas

Ambientes

- CET000183
- 2018 - Turma A03
- Programação II
- CET000183
- 2018 - Turma A02
- Programação I
- CR00100
- Mural de Recados
- Lista de Participantes
- Fórum
- Log de Acesso
- Webfólio
- Blog
- Acervo
- Configuração do Ambiente

Webfólio

Pasta

Título	Status
[Jogo da Velha]	Encerrado em 08/11/2018, às 23:59
[Projeto Integrado - III Trimestre]	Encerrado em 29/11/2018, às 23:59
I - [P1] Primeiro Trabalho	Encerrado em 17/05/2018, às 23:59
I - [P2] Segundo Trabalho	Encerrado em 17/05/2018, às 23:59
I - [P3] Terceiro Trabalho	Encerrado em 17/05/2018, às 23:59
I - [P4] Quarto Trabalho	Encerrado em 17/05/2018, às 23:59
I - [P5] Quinto Trabalho	Encerrado em 17/05/2018, às 23:59
I - [P6] Sexto Trabalho	Encerrado em 17/05/2018, às 23:59
I - [P7] Sétimo Trabalho	Encerrado em 17/05/2018, às 23:59
I - [P8] Oitavo Trabalho	Encerrado em 20/05/2018, às 23:59
II - [Integrado II]	Encerrado em 24/08/2018, às 15:00
II - [Integrado]	Encerrado em 23/08/2018, às 23:59
II - [P2] Segundo Trabalho	Encerrado em 03/06/2018, às 23:59
II - [P3] Terceiro Trabalho	Encerrado em 10/06/2018, às 23:59
II - [P4] Quarto Trabalho	Encerrado em 17/06/2018, às 23:59

Desenvolvido pela Gerência de Tecnologia e de Informação - Universidade de Caxias do Sul

Webfólio do Ambiente Virtual de Aprendizagem (CETEC)

Avaliação Kahoot! (Agosto)



Prova Mensal de Conceitos!



(10 Questões sobre as 4 Semanas de Agosto)

