

Procedimentos e Funções

Prof: Raffael Bottoli Schemmer
Disciplina: (LPI) Linguagem de Programação I
Ano: 2017/II.
Módulo: Procedimentos e Funções.



O que é um procedimento?

- Para o Delphi (Object Pascal):
 - Todos os objetos visuais da VCL são procedimentos.
- O que é um procedimento?
 - É um subprograma (código) chamado por um programa (código).



Criando um procedimento

- Observe algumas particularidades de um procedure:
 - [1] Não retorna valor.
 - [2] O nome do procedure deve ser único no programa.
 - [3] Não utilize nomes de procedures iguais a palavras reservadas.
 - [4] A primeira linha do código deve (SEMPRE) terminar com ponto e vírgula.
 - [5] Variáveis locais devem ser utilizadas (APENAS) no procedimento.
 - [6] Todo código do procedimento deve estar entre BEGIN e END;
 - [7] Na última linha do procedimento o comando END deve possuir ;

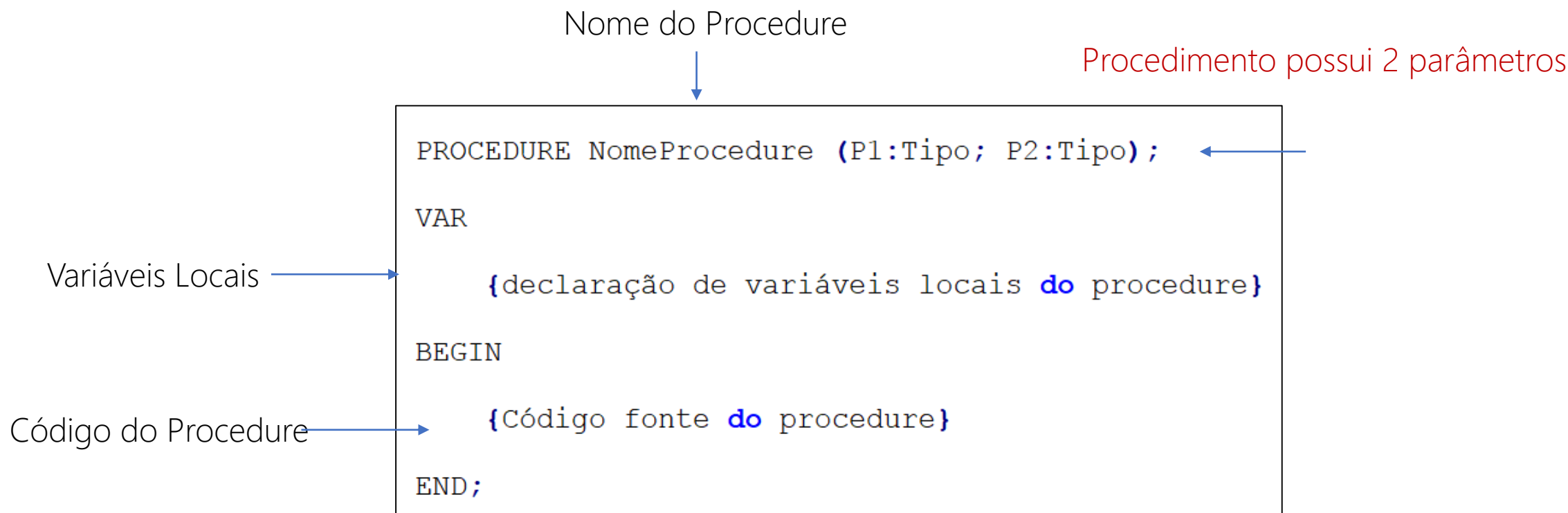


Criando um procedimento

- Qual o objetivo de um parâmetro?
 - Um parâmetro deve ser visto como uma variável local do procedimento.
- O que ele faz?
 - Ele recebe valores passados para o parâmetro em sua chamada.
- Quantos parâmetros eu preciso declarar ao criar um procedimento?
 - O procedimento não é obrigado a ter parâmetros.
 - Você verá que será inevitável (útil) criar procedimentos que recebam parâmetros.
 - O programador é livre para definir quantos parâmetros o procedimento deve ter.

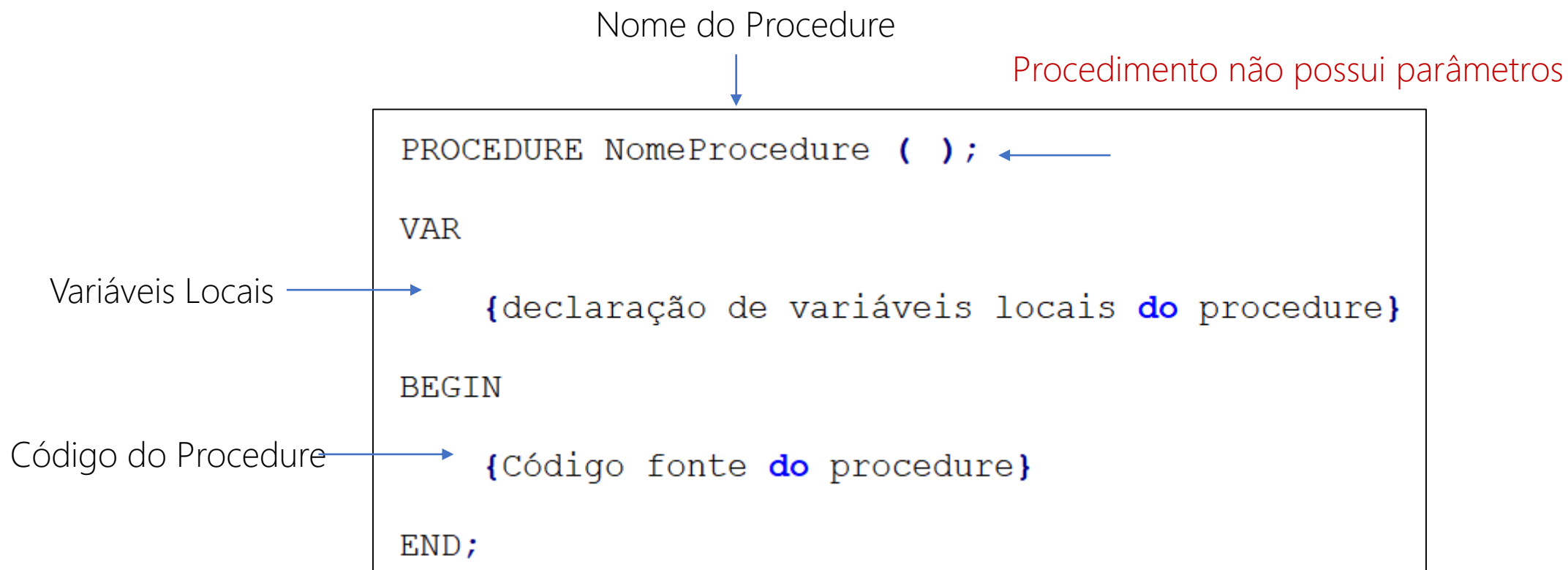


Estrutura de um procedimento



Estrutura mínima de um procedure em Objective Pascal (Delphi)

Estrutura de um procedimento



Estrutura mínima de um procedure em Objective Pascal (Delphi)

Criando um procedimento

- Dicas para nomes e tipos de parâmetros:
 - [0] Alguns programadores chamam **parâmetro** de **argumento** do procedure.
 - [1] Cada identificador (nome) do parâmetro deve possuir um nome único.
 - [2] Não utilize para nome de parâmetro os nomes das palavras da linguagem.
 - [3] Respeite as regras quanto as nomes (não use caracteres especiais).
 - [4] Utilize todos os tipos de dados suportados pela linguagem:
 - idade: INTEGER
 - nota: REAL
 - salario: EXTENDED
 - nome: STRING
 - voto: BOOLEAN



Criando um procedimento

- Onde um procedimento deve ser criado?
 - Existem duas etapas a serem feitas.
 - [1] Declaração:
 - Indica (ao PAS) que estamos criando um novo procedimento.
 - Deve ser feito na seção interface dentro do public.
 - Nosso procedimento será público para todo programa (formulários).
 - Devemos escrever apenas a primeira linha do procedimento.
 - A apostila chama a primeira linha de header (cabeçalho) do procedimento.
 - Ela informa ao Delphi o nome da função e seus parâmetros.



Declaração de um procedimento

- Onde um procedimento deve ser criado?

Nome do Procedimento Argumentos (Parâmetros)

```
private  
    procedure media3Valores(n1: INTEGER; n2: INTEGER; n3: INTEGER);  
public
```

;

Local e formato para definição de um procedimento no PAS

Definição de um procedimento

- Onde um procedimento deve ser criado?
 - Existem duas etapas a serem feitas.
 - [2] Definição:
 - Deve ser feito após {\$R *.dfm} e antes de end.
 - O local não é importante mas o programador deve seguir uma estrutura lógica.
 - A organização ajuda outros programadores (professor) a entender o código.
 - A definição nada mais é do que o procedimento em si (código fonte).
 - Como nossos procedimentos irão utilizar componentes (VCL) do form:
 - Precisaremos utilizar uma cláusula "TForm." antes do nome do procedimento.



Definição de um procedimento

- Onde um procedimento deve ser criado?

Vinculação ao Form

Nome do Procedimento

Argumentos (Parâmetros)

```
PROCEDURE TForm.media3Valores(n1:INTEGER; n2:INTEGER; n3:INTEGER);  
VAR  
BEGIN  
    ..  
    ..  
    ..  
END;
```

Definição de um procedimento

- Como chamar um procedimento?
 - Basta chamar o nome do procedimento e passar os parâmetros.
 - Se uma função possui 2 parâmetros você é obrigado a passar 2 valores.
 - Sempre que possível, passe os valores via variáveis.

Chamando procedure media3Valores
Que possui 3 parâmetros →

```
procedure TForm4.Button1Click(Sender: TObject);  
VAR  
  
    N1:= 2;  
    N2:= 4;  
    N3:= 8;  
  
BEGIN  
    media3Valores (N1,N2,N3) ;  
END;
```

Exemplo Condicional

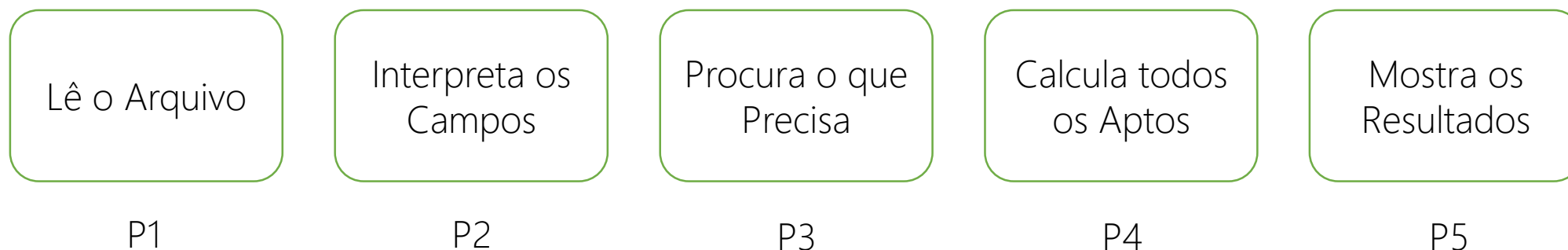
- Vamos realizar um exemplo simples (condicional):
 - Faça um procedimento que **calcule a média de 3 notas e mostre na saída padrão do Delphi**. As notas são valores não inteiros (REAL) que devem ser passadas para o procedimento. O programa só deve calcular a média caso o somatório das notas seja maior do que zero (verifique e informe na saída padrão caso o cálculo não seja possível ser realizado). Faça um programa em Delphi para testar o procedimento. O programa deve receber (do usuário) valores via InputBox ou via TEdit e deve chamar o procedimento acima.

Exemplo Iterativo

- Vamos realizar um exemplo avançado (iterativo):
 - Faça um procedimento que **calcule se um número é primo ou não e mostre na saída padrão do Delphi a mensagem (PRIMO) ou (NÃO PRIMO)**. O número passado ao procedimento deve ser inteiro (INTEGER) e maior do que zero (verifique e informe na saída padrão caso o cálculo não seja possível ser realizado). Números primos são números divisíveis por 1 e por ele mesmo. Por exemplo, 3 é primo pois só é capaz de ser dividido por 1 e por 3. Já 6 não é primo pois é capaz de ser dividido por 1, 2, 3 e por 6. Faça um programa em Delphi para testar o procedimento. O programa deve receber (do usuário) valores via InputBox ou via TEdit e deve chamar o procedimento acima.

Vantagens no uso de Procedimentos

- Desenvolvimento de programas modulares:
 - Parte da lógica do código é encapsulada em um procedimento.
 - Imagine um programa com 500 linhas de código.
 - Graças a análise e o projeto modular 5 funções de 100 linhas:
 - Organizam o que precisa ser programado e resolvido.

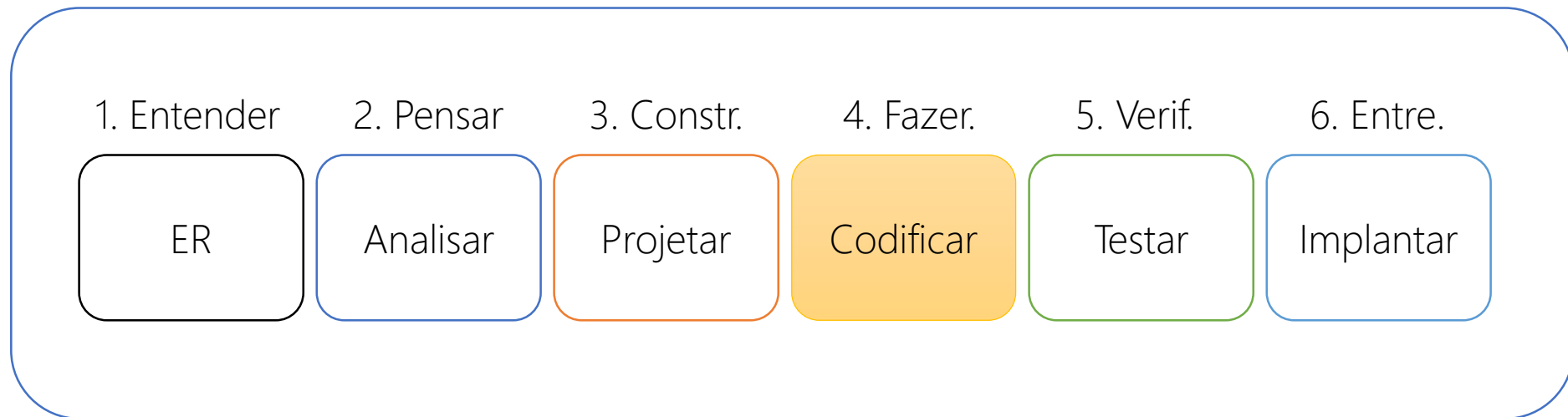


Vantagens no uso de Procedimentos

- Como saber a hora certa de criar um procedimento?
 - Antes de começar a programar o problema.
 - Realize um pouco de **engenharia** de **software**.
 - Leia com cuidado o **enunciado** do **problema** (requisitos do programa).
 - Faça uma breve **análise** do enunciado.
 - Se possível, divida o que precisa ser feito em pedaços.
 - Em seguida, realize o **projeto** dos pedaços de forma estruturada.
 - Caso seja possível, divida os pedaços em sub pedaços.
 - Só então, de início a programação.
 - Neste estágio você conseguirá enxergar o problema em sub problemas.



Engenharia de Software



Fluxo de execução para o desenvolvimento modular

Passagem de Parâmetros

- Os valores passados aos procedimentos:
 - São copiados para os mesmos.
 - Chama-se esta técnica de passagem de parâmetros por valor (cópia).
 - Nossos procedimentos não retornam valores.
 - Quando a palavra END; é encontrada toda a estrutura do procedure acaba.
 - As variáveis locais ao procedimento são descartadas pelo Delphi.



Passagem de Parâmetros

- Os valores passados aos procedimentos:
 - Existe outra técnica para realizar a passagem de parâmetros.
 - Ela é conhecida como passagem de parâmetro por referência.
 - O uso da técnica é simples e o impacto no resultado é significativo.
 - Todo parâmetro por referência deve vir antecedido da palavra VAR.
- Isso significa que o parâmetro irá possuir o endereço da variável original:
 - Se este parâmetro for modificado pelo procedimento o valor será modificado.
 - Através desta técnica é possível retornar valores pelos parâmetros.
 - O usuário da função deverá conhecer se o parâmetro é por cópia ou por referência.
 - Pois alguns procedimentos poderão retornar valores (via parâmetro).



Passagem de Parâmetros

- Os valores passados aos procedimentos:

Parâmetros por cópia



Parâmetro referência



```
PROCEDURE Media (Nota1: REAL; Nota2: REAL; VAR Media:REAL) ;  
VAR  
BEGIN  
  
    Media:= (Nota1 + Nota2)/2;  
  
END;
```

Exemplo Condicional

- Vamos realizar um exemplo simples (condicional):
 - Utilizando passagem de parâmetro por referência.
 - Faça um procedimento que **calcule a média de 3 notas e mostre na saída padrão do Delphi**. As notas são valores não inteiros (REAL) que devem ser passadas para o procedimento. O procedimento deve possuir um quarto parâmetro que deve possuir o retorno do procedimento (por referência). O programa só deve calcular a média caso o somatório das notas seja maior do que zero (verifique e retorne 0 via parâmetro caso o cálculo não seja possível). Faça um programa em Delphi para testar o procedimento. O programa deve receber (do usuário) valores via InputBox ou via TEdit e deve chamar o procedimento acima. O retorno via referência deve ser capturado e mostrado na tela.

O que é uma função?

- Para o Delphi (Object Pascal):
 - Uma função é um *procedimento* que retorna um valor (apenas isso).
 - Sendo assim, todos os conhecimentos vistos anteriormente são os mesmos.
 - Exceto pelo retorno da função (aprenderemos como fazer a seguir).
 - Toda função pode retornar um valor (caso o programador queira retornar).
 - O retorno é feito com a palavra reservada RETORNE
 - Exceto pelo uso da palavra **function** e não **procedure**.



Como criar uma função?

- Tudo o que aprendemos se aplica em função:

FUNCTION



```
FUNCTION FAT(N: INTEGER;) : INTEGER;  
  
VAR F,I: INTEGER;  
  
BEGIN  
  
    F := 1;  
    FOR I:= 1 to N DO  
  
        BEGIN  
            F:= F * I;  
        END;  
  
    RESULT := F;  
  
END;
```



Tipo de Retorno



Função que calcula o fatorial de um número (N)

Retorno é feito com RESULT



RESULT := F;

O que é uma função?

- Para o Delphi (Object Pascal):
 - A palavra RESULT pode aparecer várias vezes na função.
 - Na primeira vez que for chamada encerrará a função.
 - Você é livre para adicionar a palavra dentro das condições e/ou laços.
 - **Tudo** o que foi visto para **procedure** se **aplica** para **function**:
 - Inclusive a passagem de parâmetro por valor e por referência.
 - Você não é obrigado a utilizar RESULT podendo retornar valor por referência.



Exemplo Condicional

- Vamos realizar um exemplo simples (condicional):
 - Escreva uma função no Delphi responsável por **receber 3 lados (REAL) de um triângulo e por retornar o tipo de triângulo informado**. Os tipos de retorno podem ser: (1) **Triângulo isóceles**; (2) **Triângulo escaleno**; (3) **Triângulo equilátero**. Triângulos escalenos possuem 3 lados com tamanhos diferentes. Triângulos equiláteros possuem 3 lados com o mesmo tamanho. Triângulos isóceles possuem 2 lados com medidas iguais e um com medida diferente. O cálculo só deve ser feito com números positivos e maiores que zero. Para entradas inválidas, a função deve retornar (-1). Faça ainda um programa VCL (mínimo) que receba da entrada padrão (InputBox) ou de um TEdit o número de cada um dos 3 lados e retorne na saída padrão (ShowMessage) ou TLabel as mensagens (TRIÂNGULO EQUILÁTERO) (TRIÂNGULO ISÓCELES) (TRIÂNGULO ESCALENO) (ENTRADAS NÃO SUPORTADAS).

Exemplo Condicional

- Vamos realizar um exemplo avançado (iterativo):
 - Escreva uma função no Delphi responsável por **receber 2 números (INTEGER) e por verificar se os mesmos são amigos ou não**. Números são amigos se a soma dos divisores de N1 (excluindo o próprio N1) é igual a N2 e se a soma dos divisores de N2 (excluindo o próprio N2) é igual a N1. Sua função deve retornar (1) se os 2 números informados para a função forem amigos (possuírem a propriedade anterior). Caso contrário deve retornar (0). Se uma das entradas forem negativas a função deve retornar (-1). Faça ainda um programa VCL (mínimo) que receba da entrada padrão (InputBox) ou de um TEdit os dois números e retorne na saída padrão (ShowMessage) ou TLabel as mensagens (NÚMEROS AMIGOS) (NÚMEROS NÃO AMIGOS) (ENTRADAS NÃO SUPORTADAS).

O Delphi possui funções?

- Sim, como você já deve imaginar:
 - Todos os elementos VCL do Delphi são procedimentos e funções.
 - Conforme aprendemos, funções são blocos de código reutilizáveis.
 - Quando você arrasta (chama) um InputBox você está reusando o código.
- Assuma sempre ao criar funções:
 - Que no futuro você poderá **reutilizar** as **funções**.
 - Que **outras pessoas** poderão **utilizar** estas **funções**.
 - Desde que você **disponibilize** o **código** (**publicamente**).
 - Funções também podem ser **vendidas** para **outros programadores**.
 - Encare um **programa** como **funções** que **chamam funções**.



O Delphi possui funções?

- Vamos estudar algumas funções clássicas:
 - $x := \text{sqr}(x)$ Quadrado de x (INTEGER) : (INTEGER)
 - $x := \text{sqrt}(x)$ Raiz quadrada de x (INTEGER) : (INTEGER)
 - $x := \text{sin}(x)$ Seno de x (INTEGER) : (INTEGER)
 - $x := \text{cos}(x)$ Cosseno de x (INTEGER) : (INTEGER)
 - $x := \text{tan}(x)$ Tangente de x (INTEGER) : (INTEGER)
 - $x := \text{int}(x)$ Parte inteira de x (REAL) : (INTEGER)
 - $x := \text{frac}(x)$ Parte fracionária de x (REAL) : (INTEGER)
 - $x := \text{power}(x,y)$ x na potencia y (xy) (INTEGER,INTEGER) : (INTEGER)
 - $x := \text{abs}(x)$ Valor absoluto de x (INTEGER) : (INTEGER)

O Delphi possui funções?

- Vamos estudar algumas funções clássicas:
 - Todos os componentes VCL são procedimentos e funções.
 - Todos eles devem possuir acesso ao FORM para mudar o valor das propriedades.
- Seguem os procedimentos/funções mais clássicos:
 - NomePais := **InputBox**('Escolha de país', 'Digite o nome do país:', 'Brasil');
 - ClickOK := **InputQuery**('Escolha de País', 'Digite o nome do país ', NovaString);
 - mrYes := **MessageDlg**('Deseja sair agora?', mtConfirmation, [mbYes, mbNo], 0);
 - **ShowMessage**('Olá mundo');
 - Dia := **DateToStr**(Date);
 - **Dec**(X);
 - **Inc**(X);
 - Nome := **LowerCase**('RAFAEL');
 - Num := **round**(23,4);
 - ADate := **StrToDate**(Edit1.Text);
 - X:= **trunc**(23,4);

Boas práticas de programação

- Entre programadores:
 - É legal definir algumas regras comuns.
 - Regras baseadas no que diz o coletivo (consenso entre a comunidade).
 - Vamos aprender algumas convenções que tornam o código mais agradável.
 - Respeitando as regras do Object Pascal (Delphi) e do RAD (VCL).
- **Regra [1] : Comentários**
 - Lembre que o Delphi possui 3 tipos de comentários sendo eles.
 - // Sou um comentário de linha
 - (* Comentário de bloco *) ou { Comentário de bloco }

Boas práticas de programação

- Onde eu utilizo um comentário?
 - Sempre que estiver "definindo" um procedimento ou função.
 - Com o comentário deve ser escrito para um procedimento?
 - Faça um comentário de bloco como o descrito abaixo
- ```
{
 Autor: Raffael Bottoli Schemmer
 Objetivo: Procedimento que calcula e retorna no 2 parâmetro a média
 PAR: (real, VAR real) Notas a serem calculadas a média
}
```



# Boas práticas de programação

- Onde eu utilizo um comentário?
    - Sempre que estiver "definindo" um procedimento ou função.
    - Com o comentário deve ser escrito para uma função?
      - Faça um comentário de bloco como o descrito abaixo
- ```
{  
    Autor: Raffael Bottoli Schemmer  
    Objetivo: Função que calcula e retorna a média de 2 parâmetros  
    RET: (real) Retorna a média dos números  
    PAR: (real, real) Notas a serem calculadas a média  
}
```



Boas práticas de programação

- Onde eu utilizo um comentário?
 - Sempre que estiver "definindo" um procedimento ou função.
 - Com o comentário deve ser escrito para uma função?
 - Faça um comentário de bloco como o descrito abaixo
- ```
{
 Autor: Raffael Bottoli Schemmer
 Objetivo: Função que calcula e retorna no 2 parâmetro a média
 PAR: (real, VAR real) Notas a serem calculadas a média
}
```



# Boas práticas de programação

- Onde eu utilizo um comentário?
  - Sempre que implementar algo relativamente importante e complexo.
  - Comentários de linha são importantes para detalhar coisas simples.
  - Exemplo:
    - $senha = a*b + !a^3*2/1+2\backslash 3 + !a*b$
    - Observe como a linha acima necessita de um comentário para facilitar o entendimento.
- Afinal, porque comentar?
  - Pois outras pessoas (ou você mesmo) irá querer estudar o programa no futuro.
  - Os comentários auxiliam a entender um código complexo (ou feito no passado).



# Boas práticas de programação

- Regra [2]: Nome de variável ou parâmetro:
  - Nunca utilize nomes similares do Delphi.
  - Como uma variável BEG ou IND.
  - Convencione em utilizar as palavras da linguagem em letra maiúscula.
  - Nome das variáveis em letra minúscula.
  - Nomes simples (idade, data, salario, hora)
  - Nomes composto (diaSemana, salarioMinimo, valorMedio).
  - Este padrão é conhecido como camelCase.
  - Linguagens modernas como Java/C# possuem como regra este padrão.
  - Linguagens antigas não se dão bem com o CapsLock (CASE).
    - FUNCTION / function / Function são 3 coisas diferentes para o C e para o C++.
    - O case sensitive (CAPS) muda o sentido das coisas.

# Boas práticas de programação

- Regra [3]: Uso de procedimento e função:
  - Tente SEMPRE que puder utilizar os parâmetros e variáveis locais.
  - Imagine a função de forma autocontida (independente) do programa.
  - Imagine você vendendo a função no futuro.
  - O que você explicaria ao usuário da função (cliente/programador)?
- Pergunta: O cliente ficaria feliz se fosse necessário declarar 10 variáveis globais, sendo elas com nomes específicos (Ex: contadorLaco3) do tipo INTEGER para utilizar a função?
- Pela pergunta acima, você deve procurar criar uma função modular (encapsulada), que possua tudo o que precisa dentro de si mesma.
- Isso facilita que a mesma seja copiada e vendida a outros códigos/programadores.



# Boas práticas de programação

- Regra [4]: Variáveis locais e globais
  - Procure utilizar o mínimo possível as variáveis globais.
  - Projete as funções para que as mesmas consigam trocar (copiar) os dados.
  - Se a informação for muito grande (matriz 100x100) passe a matriz por referência a cada chamada de função.
  - Utilize variáveis globais apenas em último caso.
  - Considere que a variável global faz parte do programa e não das funções.



# Boas práticas de programação

- Regra [5]: Últimas dicas
  - Indentação e uso de TABs.
    - Não mudam a lógica mas facilitam o entendimento e o debug do código.
  - Utilize um espaçamento de linhas correto entre cada função e cada bloco.
  - Utilize BEGIN e END; sempre que possível:
    - Isso estrutura os blocos condicionais (if) e os laços de repetição (while/for).
  - Crie códigos estruturados:
    - A legibilidade favorece o entendimento das coisas.



# Variáveis Heterogêneas

- O Delphi permite que você crie variáveis heterogêneas:
  - Uma variável heterogênea é composta por outras variáveis homogêneas (INTEGER/REAL).
- Graças a isso você será capaz de criar variáveis:
  - Com características similares ao mundo real.
- Como uma variável do tipo pessoa:
  - Toda pessoa possui atributos como nome, idade, altura, rg, cpf.
  - O tipo heterogêneo terá (neste caso) 5 variáveis INTEGER/REAL/STRING.
  - Cada variável irá representar cada atributo.
  - Sempre que precisar manipular uma pessoa você poderá utilizar o novo tipo de dado.
  - Que será tipo pessoa.
  - Toda variável pessoa irá possuir 5 atributos (valores).





# Variáveis Heterogêneas

- Definindo um novo tipo de dado:
  - Deve ser especificado antes de VAR.

TYPE

```
 pessoa = record
 nome : STRING;
 idade: INTEGER
 altura: REAL
 rg: INTEGER
 cpf: STRING
```

END;

- Declarando uma variável (ou vetor) com o tipo de dado criado acima:
  - aluno: pessoa;
  - turma[50]: pessoa;

# Variáveis Heterogêneas

- Como utilizar/manipular uma variável heterogênea?
  - Cada atributo deve ser manipulado com o uso do .atributo.
  - Utilizando o exemplo anterior:
    - aluno.nome := 'Raffael';
    - aluno.idade := 15;
    - aluno.altura := 1.80;
  - Utilize o máximo que puder esta tecnologia:
    - Ela ajuda a criar programas complexos.



# O que fazer agora?



Lista de exercícios de Funções

$$M.T = 4 \text{ (LISTAS)} + 0.5 \text{ (DOC)} + 5.5 \text{ (INTEGRADO)}$$



# O que fazer agora?

Onde o **integrado** deve ser **publicado** e **documentado**?  
Onde **publicar** a **resolução** dos **exercícios**?



GITHUB.com



# O que fazer agora?

Onde encontrar o slide e as listas de programação?

[GitHub.com/RaffaelSchemmer/A2](https://github.com/RaffaelSchemmer/A2)



[GitHub.com](https://github.com)

