CI1055: Algoritmos e Estruturas de Dados I

Profs. Drs. Marcos Castilho, Bruno Müller Jr, Carmem Hara

Departamento de Informática/UFPR

25 de março de 2022

Resumo

Funções e procedimentos

Objetivos da aula

- Apresentar os conceitos de funções e procedimentos
- Motivar para:
  - Modularidade
  - Reaproveitamento de código
  - Legibilidade e demais bons princípios de programação

Motivação

- Já sabemos resolver pequenos problemas, alguns bem complexos
- Usamos os elementos e técnicas básicas
- Lidamos com as limitações do computador
- Já percebemos que alguns aspectos subjetivos são relacionados com o ser humano que programa

Motivação

- Problemas mais complexos $\Rightarrow$ códigos com muitas linhas
- Escrever o programa é mais trabalhoso
- Dar manutenção neles é pior

Motivação

- Construir programas para facilitar a vida do programador
- Elaborar melhor o código, separando algumas partes em módulos bem definidos
- Pascal oferece as funções e procedimentos

Modularidade

- Capacidade de separar programas em pedaços que executam operações bem definidas
- Cada módulo possui variáveis e estruturas próprias
- Modificações em trechos de código não devem causar reflexos no resto do programa

No mínimo três partes

- Entrada de dados
- Cálculos propriamente ditos
  - que também podem ser divididos em partes se for necessário ou conveniente
- Saída dos dados

Modularidade em *Pascal*

- *Pascal* não fornece todos os meios para se implementar um programa totalmente modular
- Suficiente para primeiro curso de programação
- Evolução: Paradigma de Programação Orientada a Objetos

Reaproveitamento de código

- Frequentemente temos códigos idênticos ou muito parecidos em alguns trechos do mesmo programa
- Este trechos podem, ou devem, constituir subprogramas
- Resultado:
  - economia de código escrito
  - facilidade de manutenção do programa

Legibilidade

- Consequência direta de modularidade e reaproveitamento de código
- Além disso:
  - Nomes de identificadores
  - Indentação
  - Uso racional de comentários no código
- Implicação: programa legível para o programador ou para quem dará manutenção
- É comum o programador não conseguir ler seu próprio código após algum tempo, isto é péssimo

Noções fundamentais

- Funções ou procedures?
- Passagem de parâmetros por valor ou por referência?
- Usar variáveis locais ou globais?

Exemplo básico

Ler uma sequência de valores inteiros terminada por zero e que imprima somente aqueles que são pares.

```
1  program imprime_pares_v0;
2  var a: integer;
3  begin
4        read (a);
5        while a <> 0 do
6        begin
7              if a mod 2 = 0 then
8                    writeln (a);
9              read (a);
10       end;
11 end.
```

Conceitos básicos

- Programa principal
- Variáveis globais

```
1  program imprime_pares_v0_comentado;
2  var a: integer;        (* a eh variavel global *)
3
4  (* do begin ate o end. eh o programa principal *)
5  begin
6        read (a);
7        while a <> 0 do
8        begin
9              if a mod 2 = 0 then
10                   writeln (a);
11             read (a);
12       end;
13 end.
```

Funções

- Observar:
  - $a \bmod 2 = 0$ na linha 9
  - Esta expressão significa "a é par?"
- E se fosse para avaliar se "a é primo" ?
- Importante:
  - O `if` avalia um valor booleano
  - Este valor pode ser o retorno de uma função!
- Devemos fazer um subprograma que resulta em um valor booleano e que ao mesmo tempo calcula corretamente se o número é par

Assinatura, ou protótipo, da função

É uma espécie de "*contrato*" entre o programa principal e o subprograma.

- Constituído por três elementos:
  - O nome da função;
  - A lista de parâmetros, que são identificadores tipados (no caso da linguagem *Pascal*);
  - O tipo do valor de retorno contendo o cálculo feito na função.
- Evidentemente, se espera que a função compute corretamente o subproblema em questão, isso é a *semântica* da função

Exemplo de protótipo

```
1  function a_eh_par: boolean;
```

- Função que calcula se um dado número é par retornando *true* em caso positivo e *false* em caso contrário
  - `function` – palavra reservada
  - `a_eh_par` – identificador
  - `: boolean` – tipo do retorno

Código da função?

Neste momento não nos interessa *como* os cálculos são feitos! Assumindo que são feitos corretamente, então:

```
1  program imprime_pares_v1;
2  var a: integer;
3
4  (* funcao que calcula se a variavel global a eh par *)
5  function a_eh_par: boolean;
6  begin
7      (* codigo da funcao *)
8  end;
9
10 begin (* programa principal *)
11       read (a);
12       while a <> 0 do
13       begin
14             if a_eh_par then  (* chamada da funcao *)
15                   writeln (a);
16             read (a);
17       end;
18 end.
```

Observações

- Programa mais legível
- Onde está o obscuro $a \bmod 2 = 0$?
- Em algum momento deverá se escrever o código da função

Diferentes maneiras de implementar a função

O identificador da função deve aparecer pelo menos uma vez no lado esquerdo de uma atribuição. Esta atribuição deve ser executada.

```
1  (* primeira versao da funcao *)
2  function ah_eh_par: boolean;
3  begin
4      if a mod 2 = 0 then
5          a_eh_par:= true
6      else
7          a_eh_par:= false;
8  end;
```

```
1  (* segunda versao da funcao *)
2  function ah_eh_par: boolean;
3  begin
4      if a mod 2 <> 0 then
5          a_eh_par:= false
6      else
7          a_eh_par:= true;
8  end;
```

Diferentes maneiras de implementar a função

O identificador da função deve aparecer pelo menos uma vez no lado esquerdo de uma atribuição. Esta atribuição deve ser executada.

```
1  (* terceira versao da funcao *)
2  function ah_eh_par: boolean;
3  begin
4      if a mod 2 <> 1 then
5          a_eh_par:= true
6      else
7          a_eh_par:= false;
8  end;
```

```
1  (* quarta versao da funcao *)
2  function ah_eh_par: boolean;
3  begin
4      if a mod 2 = 1 then
5          a_eh_par:= false
6      else
7          a_eh_par:= true;
8  end;
```

Diferentes maneiras de implementar a função

O identificador da função deve aparecer pelo menos uma vez no lado esquerdo de uma atribuição. Esta atribuição deve ser executada.

```
1  (* quinta versao da funcao *)
2  function ah_eh_par: boolean;
3  begin
4      a_eh_par:= false;
5      if a mod 2 = 0 then
6          a_eh_par:= true
7  end;
```

```
1  (* sexta versao da funcao *)
2  function ah_eh_par: boolean;
3  begin
4      a_eh_par:= true;
5      if a mod 2 = 1 then
6          a_eh_par:= false
7  end;
```

Diferentes maneiras de implementar a função

O identificador da função deve aparecer pelo menos uma vez no lado esquerdo de uma atribuição. Esta atribuição deve ser executada.

```
1  (* setima versao da funcao *)
2  function ah_eh_par: boolean;
3  begin
4      a_eh_par:= true;
5      if a mod 2 <> 1 then
6          a_eh_par:= true
7  end;
```

```
1  (* oitava versao da funcao *)
2  function ah_eh_par: boolean;
3  begin
4      a_eh_par:= true;
5      if a mod 2 <> 0 then
6          a_eh_par:= false
7  end;
```

Observações

- O identificador da função deve aparecer pelo menos uma vez no lado esquerdo de uma atribuição
- Esta atribuição deve ser executada pelo menos uma vez
- Em *Pascal*, a função executa até o final
- Não tem como sair antes

Versão final

```
1  program imprime_pares_final;
2  var a: integer;
3
4  (* funcao que calcula se a variavel global a eh par *)
5  function a_eh_par: boolean;
6  begin
7      a_eh_par:= true;
8      if a mod 2 <> 0 then
9          a_eh_par:= false
10 end;
11
12 begin (* programa principal *)
13       read (a);
14       while a <> 0 do
15       begin
16             if a_eh_par then  (* chamada da funcao *)
17                   writeln (a);
18             read (a);
19       end;
20 end.
```

Valor semântico das funções

- Funções são utilizadas em expressões aritméticas ou booleanas
- Portanto, podem estar no lado direito de atribuições, em comandos de impressão ou como parte das expressões aritiméticas ou booleanas
- Exemplos:
  - `raiz1:= (-b - sqrt (b*b - 4*a*c))/(2*a)`
  - `writeln((-b - sqrt (b*b - 4*a*c))/(2*a));`
  - `if ah_eh_par and (a >= 20)`

Fim do tópico

- este material está no livro no capítulo 8, seções 8.1 e 8.2 (de 8.2.1 até 8.2.4)

Licença

- Slides feitos em LaTeX usando beamer
- Licença

  Creative Commons Atribuição-Uso Não-Comercial-Vedada a Criação de Obras Derivadas 2.5 Brasil License.`http://creativecommons.org/licenses/by-nc-nd/2.5/br/`

  Creative Commons Atribuição-Uso Não-Comercial-Vedada a Criação de Obras Derivadas 2.5 Brasil License.`http://creativecommons.org/licenses/by-nc-nd/2.5/br/`