27. Modularização

DIM0320

2015.1

Sumário

Sufixo

Um número b é dito ser *sufixo* de um número a se o número formado pelos últimos dígitos de a são iguais a b. Assim:

a	b		
567890	890	$\rightarrow$	sufixo
1234	1234	$\rightarrow$	sufixo
2457	245	$\rightarrow$	não é sufixo
457	2457	$\rightarrow$	não é sufixo

- Construa uma função *sufixo* que dados dois números inteiros a e b verifica se b é um sufixo de a.
- Utilizando a função do item anterior, escreva um algoritmo que leia dois números inteiros a e b e verifica se o menor deles é subsequência do outro. Exemplo:

a	b		
567890	678	$\rightarrow$	b é subsequência de a
1234	2212345	$\rightarrow$	a é subsequência de b
235	236	$\rightarrow$	um não é subsequência do outro

Vetores como parâmetros ?

Problemas

Em VisuAlg Portugol:

- Vetores (e matrizes) **não** podem ser **usados** como parâmetros formais de função
- Vetores (e matrizes) **não** podem ser **retornados** como valores de função

Exemplo (Elementos proibidos no VisuAlg)

```
procedimento f(v: vetor [1..10] de inteiro)

funcao f(i, j: inteiro) : vetor [1..8, 2..75] de real
```

Como fazer nesse caso ?

Regras de escopo + variáveis globais

```
algoritmo "Preenchimento"
var v : vetor [1..10] de real
    n : inteiro

funcao checked_size(): inteiro
var n : inteiro
inicio
  repita
    escreval("Entre com um inteiro: ")
    leia(n)
  ate (n >= 1) e (n <= 10)
  retorne n
fimfuncao

procedimento fill(size: inteiro)
var i : inteiro
inicio
  para i de 1 ate size faca
    escreval("Valorda célula ", i, "? ")
    leia(v[i])
  fimpara
fimprocedimento
```

```
procedimento write(size: inteiro)
var i : inteiro
inicio
  para i de 1 ate size faca
    escreval(v[i])
  fimpara
fimprocedimento

inicio
  n <- checked_size()
  fill(n)
  write(n)
fimalgoritmo
```

Em sala de aula / na outra ferramenta

- Vetores podem e **devem** ser passados em parâmetro das funções
- Células do vetor são **referências**.
- Eles **não** podem ser retornados

```
algoritmo "Preenchimento"
var v, w : vetor [1..10] de real
    n : inteiro
funcao checked_size(): inteiro
var n : inteiro
inicio
  repita
    escreval("Entre com um inteiro: ")
    leia(n)
  ate (n >= 1) e (n <= 10)
  retorne n
fimfuncao

procedimento fill(size: inteiro;
                  v : vetor[1..10] de real)
var i : inteiro
inicio
  para i de 1 ate size faca
    escreval("Valor da célula ", i, "? ")
    leia(v[i])
  fimpara
fimprocedimento
```

```
procedimento write(size: inteiro;
                   v : vetor[1..10] de real)
var i : inteiro
inicio
  para i de 1 ate size faca
    escreval(v[i])
  fimpara
fimprocedimento

inicio
  n <- checked_size()
  fill(n, v)
  write(n, v)
  fill(n, w)
  write(n, w)
fimalgoritmo
```

Escrever a declaração duma rotina

Rotina

- Toda função é uma rotina
- Todo procedimento é uma rotina

1. Escolher um nome que indica claramente o que a rotina faz
2. Identificar os parâmetros que variam nas diferentes chamadas de rotina no módulo principal. Dar um nome a cada um desses parâmetros formais.
3. Identificar o tipo adequado para cada um dos parâmetros formais.
4. Identificar se a rotina retorna um valor e, se sim, o tipo desse valor.

Resumo

1. Exercício (Correção)
2. Vetores e modularização
3. Exercícios

Perguntas ?

http://dimap.ufrn.br/~richard/dim0320

Cadeia de caracteres (leitura)

```
funcao sub(s: caractere; i, j:inteiro):caractere
var mys: caractere
    k: inteiro
inicio
  mys <- ""
  para k de i ate j faca
      mys <- mys + car(s,k)
  fimpara
  retorne mys
fimfuncao
```

- "a" + "bcde" + "f" $\rightarrow$ "abcdef"
- `car(s,i)` retorna o i-ésimo caractere de `s` como cadeia de caracteres

Assunto

- O que faz essa função ?
- Identificar problemas potenciais.

Declarações e implementações

Declarações

Escrever as declarações das funções abaixo

1. Uma função que calcula a velocidade média dum elemento dado o tempo de percurso e a distância percorrida.

Declarações e implementações

Declarações

Escrever as declarações das funções abaixo

1. Uma função que calcula a velocidade média dum elemento dado o tempo de percurso e a distância percorrida.
   - `funcao vmedia(distancia, tempo: real): real`

Declarações e implementações

Declarações

Escrever as declarações das funções abaixo

1. Uma função que calcula a velocidade média dum elemento dado o tempo de percurso e a distância percorrida.
   - `funcao vmedia(distancia, tempo: real): real`
2. Uma função que verifica se é possível construir um triângulo a partir de 3 segmentos de medição dados.

Declarações e implementações

Declarações

Escrever as declarações das funções abaixo

1. Uma função que calcula a velocidade média dum elemento dado o tempo de percurso e a distância percorrida.
   - `funcao vmedia(distancia, tempo: real): real`
2. Uma função que verifica se é possível construir um triângulo a partir de 3 segmentos de medição dados.
   - `funcao e_triangulo(s1, s2, s3: real): logico`

Declarações e implementações

Declarações

Escrever as declarações das funções abaixo

1. Uma função que calcula a velocidade média dum elemento dado o tempo de percurso e a distância percorrida.
   - `funcao vmedia(distancia, tempo: real): real`
2. Uma função que verifica se é possível construir um triângulo a partir de 3 segmentos de medição dados.
   - `funcao e_triangulo(s1, s2, s3: real): logico`
3. Uma função que calcula o mdc de 2 inteiros.

Declarações e implementações

Declarações

Escrever as declarações das funções abaixo

1. Uma função que calcula a velocidade média dum elemento dado o tempo de percurso e a distância percorrida.
   - `funcao vmedia(distancia, tempo: real): real`
2. Uma função que verifica se é possível construir um triângulo a partir de 3 segmentos de medição dados.
   - `funcao e_triangulo(s1, s2, s3: real): logico`
3. Uma função que calcula o mdc de 2 inteiros.
   - `funcao mdc(m, n: inteiro): inteiro`

Declarações e implementações

Declarações

Escrever as declarações das funções abaixo

1. Uma função que calcula a velocidade média dum elemento dado o tempo de percurso e a distância percorrida.
   - `funcao vmedia(distancia, tempo: real): real`
2. Uma função que verifica se é possível construir um triângulo a partir de 3 segmentos de medição dados.
   - `funcao e_triangulo(s1, s2, s3: real): logico`
3. Uma função que calcula o mdc de 2 inteiros.
   - `funcao mdc(m, n: inteiro): inteiro`
4. Um procedimento que escreve na tela as iniciais duma pessoa cujo nome completo foi dado.

Declarações e implementações

Declarações

Escrever as declarações das funções abaixo

1. Uma função que calcula a velocidade média dum elemento dado o tempo de percurso e a distância percorrida.
   - `funcao vmedia(distancia, tempo: real): real`
2. Uma função que verifica se é possível construir um triângulo a partir de 3 segmentos de medição dados.
   - `funcao e_triangulo(s1, s2, s3: real): logico`
3. Uma função que calcula o mdc de 2 inteiros.
   - `funcao mdc(m, n: inteiro): inteiro`
4. Um procedimento que escreve na tela as iniciais duma pessoa cujo nome completo foi dado.
   - `procedimento esc_iniciais(nome: caractere)`

Implementações

- Implementar as rotinas 2 e 4
- Para 4, pode-se usar
  - **copia (c : caractere ; p, n : inteiro)** : retorna um valor do tipo caractere contendo uma cópia parcial da expressão, a partir do caractere p, contendo n caracteres. Os caracteres são numerados da esquerda para a direita, começando de 1.
  - **compr (c : caractere)** : retorna um inteiro contendo o comprimento (quantidade de caracteres) da expressão.

Pesquisa num vetor

Assunto

- Escreva uma função que procura um elemento inteiro num vetor (de inteiros) de até 100 elementos.
- A função retorna o número do índice com o elemento se existir, -1 senão.

Observação

- O número efetivo de elementos do vetor deve também ser passado como parâmetro.

Assunto 2

- Escreva uma nova versão mais eficiente da sua função a partir da seguinte hipótese: o vetor passado é ordenado.