15. Estruturas de repetição 3

DIM0320

2015.1

Sumário

Tipos de laço

Observação

- O comando `enquanto-faca` é um exemplo de **laço condicional**
- Esse tipo de laço é necessário quando é impossível delimitar o número de repetição
- Às vezes, já sabemos com antecedência o número de iterações a ser feito.
- O laço `para-faca` é adaptado a esse tipo de problema.

para-faca-fimpara

Sintaxe

```
para <var> de <expr_1> ate <expr_2> passo <expr_3> faca
   <cmd_1>
   ..
   <cmd_n>
fimpara
```

Observações

- <expr_1>, <expr_2>, <expr_3> são **expressões** do tipo **inteiro**
- <var> é uma **variável** do tipo **inteiro**
- a parte passo <expr_3> é facultativa
  - se não escrito, o valor do passo usado é **1**

Exemplo básico de uso

Contar de 1 a n

```
1 leia(n)
2 para i de 1 ate n passo 1 faca
3   escreval(i)
4 fimpara
```

Contar de n a 1

```
1 leia(n)
2 para i de n ate 1 passo (-1) faca
3   escreval(i)
4 fimpara
```

Semântica

Seja

- `n` o valor da avaliação de `<expr_1>`
- `m` o valor da avaliação de `<expr_2>`
- `p` o valor da avaliação de `<expr_3>`

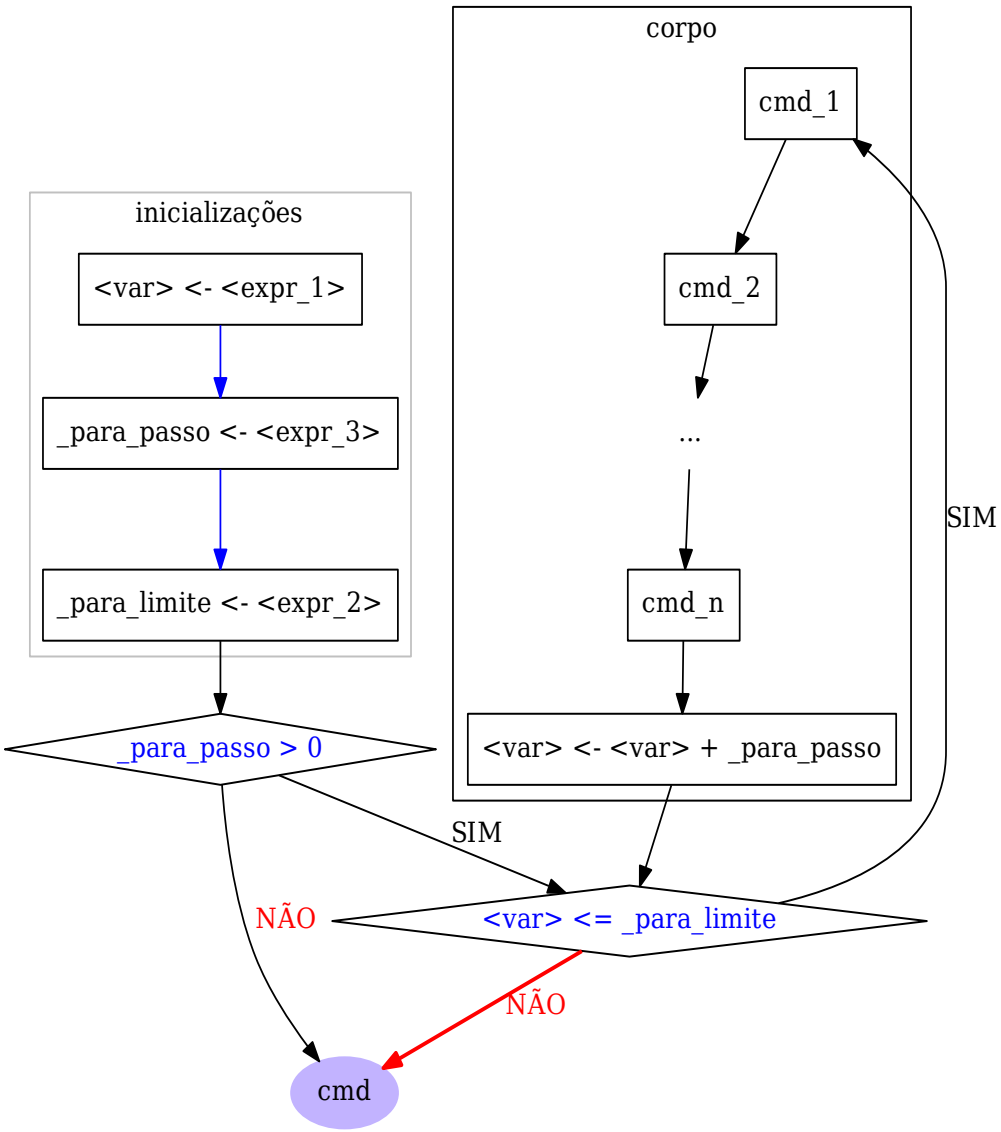
Então

- A variável `<var>` é inicializada a `n`
- O corpo do laço `para` é executado vezes até
  - `<var> > m`, se `m >= n`
  - `<var> < m`, se `m < n`
- Após cada execução do corpo, mas antes de voltar ao teste da condição, a variável `<var>` é incrementada de `p`

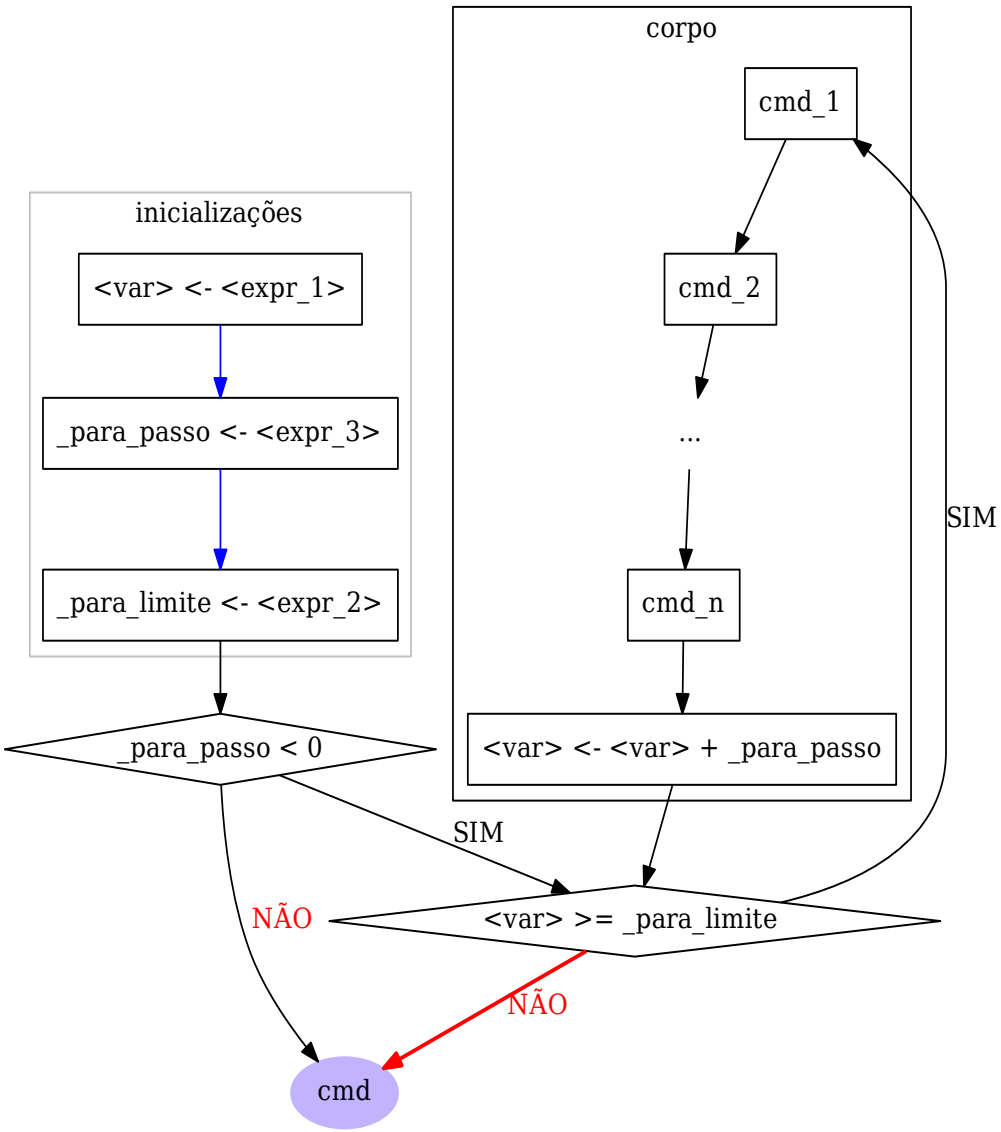
Observação

- Numa execução finita, o laço é executado $(|m - n| + 1)/|p|$ vezes

Fluxograma (<expr$_2$> >= <expr$_1$>=)



Fluxograma (<expr$_2$> < <expr$_1$>=)



Saber escrever um laço para-faca

1. Determinar quantas vezes o laço será executado

Saber escrever um laço para-faca

1. Determinar quantas vezes o laço será executado
   - Esse número depende geralmente de uma ou várias variável(eis)

Saber escrever um laço para-faca

1. Determinar quantas vezes o laço será executado
   - Esse número depende geralmente de uma ou várias variável(eis)
2. Escolher uma variável para o contador de laço

Saber escrever um laço para-faca

1. Determinar quantas vezes o laço será executado
   - Esse número depende geralmente de uma ou várias variável(eis)
2. Escolher uma variável para o contador de laço
   - Não esqueça a declaração

Saber escrever um laço para-faca

1. Determinar quantas vezes o laço será executado
   - Esse número depende geralmente de uma ou várias variável(eis)
2. Escolher uma variável para o contador de laço
   - Não esqueça a declaração
   - Identificar se ela é usada no corpo do laço

Saber escrever um laço para-faca

1. Determinar quantas vezes o laço será executado
   - Esse número depende geralmente de uma ou várias variável(eis)
2. Escolher uma variável para o contador de laço
   - Não esqueça a declaração
   - Identificar se ela é usada no corpo do laço
3. Escrever o corpo

Saber escrever um laço para-faca

1. Determinar quantas vezes o laço será executado
   - Esse número depende geralmente de uma ou várias variável(eis)
2. Escolher uma variável para o contador de laço
   - Não esqueça a declaração
   - Identificar se ela é usada no corpo do laço
3. Escrever o corpo
4. Prever inicializações necessárias é pós-tratamentos adequados.

Exemplo: fatorial

```
1  algoritmo "Fatorial para"
2  var i, n, fat: inteiro
3  inicio
4    escreval("Entre com um número > 0")
5    leia(n)
6    enquanto n < 0 faca
7      escreval("Entre com um número > 0")
8      leia(n)
9    fimenquanto
10
11   fat <- 1
12   para i de 1 ate n faca
13     fat <- i * fat
14   fimpara
15
16   escreval("fatorial(", n, ") = ", fat)
17 fimalgoritmo
```

Tradução em laço enquanto

```
para <var> de <expr_1> ate <expr_2> passo <expr_3> faca
   <cmd_1>
   ..
   <cmd_n>
fimpara
```

- se <expr_1> <= <expr_2> (nesse caso, verifique que <expr_3> >= 0)

```
<var> <- <expr_1>
m <- <expr_2>
p <- <expr_3>
enquanto <var> <= m faca
   <cmd_1>
   ..
   <cmd_n>
   <var> <- <var> + p
fimenquanto
```

Tradução em laço enquanto

```
para <var> de <expr_1> ate <expr_2> passo <expr_3> faca
   <cmd_1>
   ..
   <cmd_n>
fimpara
```

- se <expr_1> >= <expr_2> (nesse caso, verifique que <expr_3> < 0)

```
<var> <- <expr_1>
m <- <expr_2>
p <- <expr_3>
enquanto <var> >= m faca
   <cmd_1>
   ..
   <cmd_n>
   <var> <- <var> + p
fimenquanto
```

Como escolher o tipo de laço ?

Entre `enquanto` e `para`

- Use `para` se
  - o número de repetição é conhecido
- Use `enquanto` se
  - a condição de parada é necessariamente um **teste**

Primeiras funções

Funções matemáticas

- **abs(r : real)** : valor absoluto de r
- **log(r : real)** : logaritmo natural (neperiano)
- **logn(r, n : real)** : logaritmo de base n, i.e. `log(r) / log(n)`
- **exp(r1, r2 : real)** : exponenciação, i.e. $r_1^{r_2}$
- **raizq(r : real)** : raiz quadrada de r, i.e. $r^{0.5}$
- **quad(r : real)** : quadrado de r, i.e. r^2
- **pi** : valor da constante pi, i.e. $4 * arctan(1)$

Funções de geração de valores

- **rand()** : valor real aleatório do intervalo $[0, 1[$
- **randi(n)** : valor inteiro aleatório do intervalo $[0, n]$

Funções trigonométricas

- **cos(r : real)**
- **sen(r : real)**
- **tan(r : real)**
- **arccos(r : real)**
- **arcsen(r : real)**
- **arctan(r : real)**

Funções de manipulação de texto

- + permite a concatenação de texto : "a" + "bc" + "de" $\rightarrow$ "abcde"
- **asc (s : caractere)** retorna um inteiro com o código ASCII do primeiro caractere da expressão.
- **carac (c : inteiro)** retorna o caractere cujo código ASCII corresponde à expressão.
- **caracpnum (c : caractere)** retorna o inteiro ou real representado pela expressão.

Funções de manipulação de texto

- **compr (c : caractere)** retorna um inteiro contendo o comprimento (quantidade de caracteres) da expressão.
- **copia (c : caractere ; p, n : inteiro)** retorna um valor do tipo caracter 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.
- **maiusc (c : caractere)** retorna um valor caractere contendo a expressão em maiúsculas.
- **minusc (c : caractere)** retorna um valor caractere contendo a expressão em minúsculas.
- **numpcarac (n : inteiro ou real)** retorna um valor caractere contendo a representação de n como uma cadeia de caracteres.
- **pos (subc, c : caractere)** retorna um inteiro que indica a posição em que a cadeia subc se encontra em c, ou zero se subc não estiver contida em c.

Conversão binário -> decimal

Assunto

Escrever um algoritmo que:

- lê um texto
- verifica que desse texto inclui só 0 e 1 (esse texto é uma expressão binária)
- calcula o valor inteiro (decimal) que corresponde a essa expressão binária

Considere que o bit mais a esquerda é bit de menor peso, i.e. ele é o dígito $d_0 * 2^0$.

Geração de valores aleatórios

Assunto

A sequência definida por indução como

$$\begin{cases} u_0 = 13 \\ u_{n+1} = (16805 * u_n + 1) \ mod \ 32768 \end{cases}$$

parece aleatoria

1. Escrever um algoritmo que escrever os 10 000 primeiros termos da sequência.
2. Para simular um sequência aleatória tipo "cara ou coroa", observamos o bit 9 da representação binária de cada elemento dessa sequência. Se o nono bit for 1 dizemos a moeda caiu do lado "cara", se for 0, ela caiu do lado "coroa".
   - Alterar o seu programa para escrever os 10000 primeiros sorteios.
3. A qualidade desse gerador pode ser testado contando o número de vezes que a moeda cai do lado "cara" e do lado "coroa"
   - Alterar seu programa para contar o número de vezes que a moeda cai do lado "cara".
4. O que obtemos se usarmos o primeiro bit em vez do nono ? Explicar por que.