Estruturas de repetição aninhadas

Patrícia de Siqueira Ramos

UNIFAL-MG, *campus* Varginha

22 de Setembro de 2020

Estruturas encadeadas

- Já vimos que é possível ter estruturas condicionais (Se) encadeadas (umas dentro das outras)
- E, para usá-las, é necessário tomar cuidado com a indentação
- É possível encadear quantas estruturas quisermos, umas dentro das outras
- Um esboço de um Se dentro de outro Se em pseudocódigo seria:

Estrutura Se encadeada

```
Início
   ---
   ---
   Se ----- Então
      ---------
   Senão
      Se ----- Então
         Se ----- Então
            ---------
         Senão
            ---------
         FimSe
      Senão
         ---------
      FimSe
   FimSe
   ---
   ---
Fim
```

Estrutura de repetição dentro de outra estrutura de repetição (*loop* dentro de *loop*)

- Problemas complexos às vezes exigem uma estrutura de repetição dentro de outra (um *loop* dentro de outro *loop*)
- E também é preciso tomar cuidado com a indentação
- Veja um exemplo simples:

```
Início
    Inteiro: i, j
    Para i de 0 até 2 faça
        Escreva(i)
        Para j de 4 até 5 faça
            Escreva(j)
        FimPara
    FimPara
Fim
```

Estrutura de repetição dentro de outra estrutura de repetição (*loop* dentro de *loop*)

- Ao executar esse algoritmo, a saída seria

```
0
4
5
1
4
5
2
4
5
```

Estrutura de repetição dentro de outra estrutura de repetição (*loop* dentro de *loop*)

- Isso iria acontecer porque i começa valendo 0
- Dentro do primeiro `Para`, após mostrar i, acontece outro `Para`, com a variável j variando de 4 até 5, cujos valores são mostrados
- Quando o `Para` menor termina, o `Para` maior continua, com a variável i assumindo o próximo valor, que é $i = 1$
- Após mostrar esse valor de i, o `Para` menor é executado de novo, mostrando o 4 e 5 novamente
- Finalmente, o mesmo acontece quando $i = 2$

Ex. simples no python

- Implementando no python vamos deixar a saída mais clara, informando qual a variável está sendo mostrada (i ou j):

```
[13] # for aninhado - saída melhor
     for i in range(3):
       print('i =', i, end=' ')
       for j in range(4, 6):
         print('j =', j, end=' ')
```

```
i = 0 j = 4 j = 5 i = 1 j = 4 j = 5 i = 2 j = 4 j = 5
```

- Veja que fica claro qual variável está sendo mostrada de cada vez

Estrutura de repetição dentro de outra estrutura de repetição (*loop* dentro de *loop*)

- Um bom exemplo de uso de estruturas de repetição aninhadas seria para andar dentro de uma tabela, por exemplo:

O *loop* maior avança nas linhas da tabela e o *loop* menor avança nas colunas da mesma tabela

Exemplo de *loop* dentro de *loop*

loop menor

loop maior	n = 1	n = 2	n = 3	n = 4
	x ** n	x ** n	x ** n	x ** n
x = 1	1	1	1	1
x = 2	2	4	8	16
x = 3	3	9	27	81
...	...	...	...	...
x = 10	10	100	1000	10000

Exemplo de *loop* dentro de *loop*

- Para resolver esse problema de mostrar as potências x ** n com n variando de 1 a 4,
  x variando de 1 a 10,
  podemos usar as estruturas `Enquanto` ou `Para`.
- Primeiro usaremos a estrutura `Para` (e o correspondente *for* no `python`):

Ex.: *loop* dentro de *loop* no python - estrutura Para

```
Início
    Inteiro: x, n
    Para x de 1 até 10 faça
        Para n de 1 até 4 faça
            Escreva(x ** n)
        FimPara
        Escreva(‘ ’)
    FimPara
Fim
```

Ex.: *loop* dentro de *loop* no python - estrutura Para

```
Início
   Inteiro: x, n
   Para x de 1 até 10 faça
      Para n de 1 até 4 faça
         Escreva(x ** n)
      FimPara
      Escreva(‘ ’)
   FimPara
Fim
```

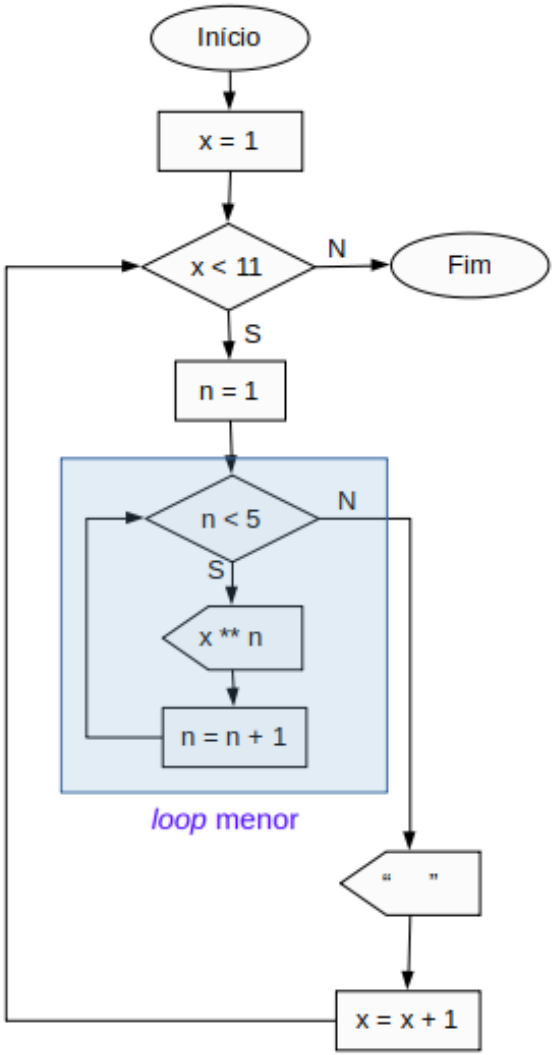
```
for x in range(1, 11):
  for n in range(1, 5):
    print(x ** n, end=' ')
  print()
```

Ex.: *loop* dentro de *loop* - estrutura `Enquanto`

Agora usando a estrutura `Enquanto`:

```
Início
   Inteiro: x, n
   x = 1
   Enquanto x < 11 faça
      n = 1
      Enquanto n < 5 faça
         Escreva(x ** n)
         n = n + 1
      FimEnquanto
      Escreva(‘ ’)
      x = x + 1
   FimEnquanto
Fim
```

Ex.: *loop* dentro de *loop* - estrutura `Enquanto`



Ex.: *loop* dentro de *loop* no python - estrutura Enquanto

```
x = 1
while x < 11:
  n = 1
  while n < 5:
    print(x ** n, end=' ')
    n = n + 1
  print()
  x = x + 1
```