

Disciplina BCM0505-15

Processamento da Informação

Comandos de repetição

Profa. Carla Negri Lintzmayer

`carla.negri@ufabc.edu.br`

<http://professor.ufabc.edu.br/~carla.negri>

Centro de Matemática, Computação e Cognição
Universidade Federal do ABC



Agenda

Motivação

Laços ou estruturas de repetição

Repetição do tipo “enquanto”

Repetição do tipo “para”

Mais exemplos

Pratique!

Motivação

- Faça um algoritmo que recebe 4 inteiros e apresenta o maior deles.

- Faça um algoritmo que recebe 4 inteiros e apresenta o maior deles.
- Faça um algoritmo que recebe 50 inteiros e apresenta o maior deles.

- Faça um algoritmo que recebe 4 inteiros e apresenta o maior deles.
- Faça um algoritmo que recebe 50 inteiros e apresenta o maior deles.
- Faça um algoritmo que recebe n inteiros e apresenta o maior deles.

Motivação

- Faça um algoritmo que desenhe uma grade com 4 quadrados 4×4 como a seguinte:

```
+ - - - - + - - - - +  
|           |           |  
|           |           |  
|           |           |  
|           |           |  
+ - - - - + - - - - +  
|           |           |  
|           |           |  
|           |           |  
|           |           |  
+ - - - - + - - - - +
```

Motivação

- Faça um algoritmo que desenhe uma grade com 4 quadrados 4×4 como a seguinte:

```
+ - - - - + - - - - +  
|           |           |  
|           |           |  
|           |           |  
|           |           |  
+ - - - - + - - - - +  
|           |           |  
|           |           |  
|           |           |  
|           |           |  
+ - - - - + - - - - +
```

- Faça um algoritmo que desenhe uma grade com 4 quadrados 20×20 .

Motivação

- Faça um algoritmo que desenhe uma grade com 4 quadrados 4×4 como a seguinte:

```
+ - - - - + - - - - +  
|           |           |  
|           |           |  
|           |           |  
|           |           |  
|           |           |  
+ - - - - + - - - - +  
|           |           |  
|           |           |  
|           |           |  
|           |           |  
|           |           |  
+ - - - - + - - - - +
```

- Faça um algoritmo que desenhe uma grade com 4 quadrados 20×20 .
- Faça um algoritmo que desenhe uma grade com 40 quadrados 20×20 .

Motivação

- Faça um algoritmo que desenhe uma grade com 4 quadrados 4×4 como a seguinte:

```
+ - - - - + - - - - +
|           |           |
|           |           |
|           |           |
|           |           |
+ - - - - + - - - - +
|           |           |
|           |           |
|           |           |
|           |           |
+ - - - - + - - - - +
```

- Faça um algoritmo que desenhe uma grade com 4 quadrados 20×20 .
- Faça um algoritmo que desenhe uma grade com 40 quadrados 20×20 .
- Faça um algoritmo que desenhe uma grade com n quadrados $m \times m$.

- Faça um algoritmo que inverta um inteiro que tem 4 dígitos.

- Faça um algoritmo que inverta um inteiro que tem 4 dígitos.
- Faça um algoritmo que inverta um inteiro que tem 40 dígitos.

- Faça um algoritmo que inverta um inteiro que tem 4 dígitos.
- Faça um algoritmo que inverta um inteiro que tem 40 dígitos.
- Faça um algoritmo que inverta um inteiro que tem n dígitos.

Alice e Bob querem jogar par ou ímpar. Alice sempre aposta no par e Bob no ímpar.

- Faça um algoritmo que mostre quem é o ganhador em um jogo entre os dois.

Alice e Bob querem jogar par ou ímpar. Alice sempre aposta no par e Bob no ímpar.

- Faça um algoritmo que mostre quem é o ganhador em um jogo entre os dois.
- Faça um algoritmo que mostre quem é o ganhador em dez jogos entre os dois.

Alice e Bob querem jogar par ou ímpar. Alice sempre aposta no par e Bob no ímpar.

- Faça um algoritmo que mostre quem é o ganhador em um jogo entre os dois.
- Faça um algoritmo que mostre quem é o ganhador em dez jogos entre os dois.
- Faça um algoritmo que mostre quem é o ganhador em n jogos entre os dois.

Alice e Bob querem jogar par ou ímpar. Alice sempre aposta no par e Bob no ímpar.

- Faça um algoritmo que mostre quem é o ganhador em um jogo entre os dois.
- Faça um algoritmo que mostre quem é o ganhador em dez jogos entre os dois.
- Faça um algoritmo que mostre quem é o ganhador em n jogos entre os dois.
- Faça um algoritmo que receba jogos entre Alice e Bob, parando assim que Alice ganhar.

- Faça um algoritmo que recebe 10 números e mostre a soma daqueles que são pares.

- Faça um algoritmo que recebe 10 números e mostre a soma daqueles que são pares.
- Faça um algoritmo que recebe 100 números e mostre a soma daqueles que são pares.

- Faça um algoritmo que recebe 10 números e mostre a soma daqueles que são pares.
- Faça um algoritmo que recebe 100 números e mostre a soma daqueles que são pares.
- Faça um algoritmo que recebe n números e mostre a soma daqueles que são pares.

- Faça um algoritmo que recebe 10 números e mostre a soma daqueles que são pares.
- Faça um algoritmo que recebe 100 números e mostre a soma daqueles que são pares.
- Faça um algoritmo que recebe n números e mostre a soma daqueles que são pares.
- Faça um algoritmo que recebe números positivos e pare de receber números assim que receber um valor negativo, mostrando a soma daqueles que são pares.

- Todas as variações dos problemas que envolvem valores n ou m quaisquer ou que envolvem uma quantidade totalmente desconhecida ficam **impossíveis** de serem resolvidas apenas com as estruturas que vimos até o momento.
- Para resolvê-los, e resolver muitos outros, precisamos das estruturas de repetição.

Laços ou estruturas de repetição

Comandos de repetição

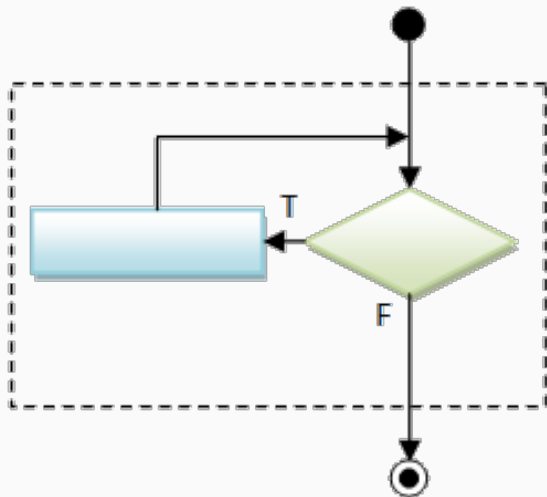
Um comando de repetição é aquele que permite repetir um determinado bloco de comandos por uma quantidade **finita** de vezes.



Comandos de repetição

- Mais comumente chamados de laços ou *loops*.
- Qualquer tipo de laço (veremos 2) têm uma **condição de parada**, que é uma expressão lógica testada após cada repetição.
- O laço para quando essa condição for falsa.
 - Se seu laço não atingir a condição de parada, você tem um *loop infinito*.
 - E nem tem mais um algoritmo...

Estruturas de repetição



Laço (repetição)

Repetição do tipo “enquanto”

Laço “enquanto”

- O comando **Enquanto** permite que haja repetição de um bloco sempre que sua condição for verdadeira.
- Sua sintaxe é:

- 1: **Enquanto** expressão com valor lógico **faça**
- 2: comando(s) executado(s) se a expressão for verdadeira
- 3: comando(s) executado(s) após o término do laço

Laço “enquanto”

- O comando **Enquanto** permite que haja repetição de um bloco sempre que sua condição for verdadeira.
- Sua sintaxe é:

1: Enquanto expressão com valor lógico faça 2: comando(s) executado(s) se a expressão for verdadeira 3: comando(s) executado(s) após o término do laço
--

- Observe a estrutura similar à do comando **Se**: há um cabeçalho com a **condição de parada** e um **corpo**, que contém comandos indentados com relação ao cabeçalho.
- Deve haver ao menos uma instrução no corpo.

Funcionamento do laço “enquanto”

- Verificamos a expressão do cabeçalho (condição de parada).

Funcionamento do laço “enquanto”

- Verificamos a expressão do cabeçalho (condição de parada).
- Se for verdadeira, executamos os comandos no corpo.

Funcionamento do laço “enquanto”

- Verificamos a expressão do cabeçalho (condição de parada).
- Se for verdadeira, executamos os comandos no corpo.
- Após o último comando do corpo, verificamos novamente a condição de parada.

Funcionamento do laço “enquanto”

- Verificamos a expressão do cabeçalho (condição de parada).
- Se for verdadeira, executamos os comandos no corpo.
- Após o último comando do corpo, verificamos novamente a condição de parada.
- Se for verdadeira, executamos os comandos no corpo.

Funcionamento do laço “enquanto”

- Verificamos a expressão do cabeçalho (condição de parada).
- Se for verdadeira, executamos os comandos no corpo.
- Após o último comando do corpo, verificamos novamente a condição de parada.
- Se for verdadeira, executamos os comandos no corpo.
- ...

Funcionamento do laço “enquanto”

- Verificamos a expressão do cabeçalho (condição de parada).
- Se for verdadeira, executamos os comandos no corpo.
- Após o último comando do corpo, verificamos novamente a condição de parada.
- Se for verdadeira, executamos os comandos no corpo.
- ...
- Se a condição for falsa, então o comando de repetição termina e passa-se à linha seguinte ao corpo.

Funcionamento do laço “enquanto”

- Verificamos a expressão do cabeçalho (condição de parada).
- Se for verdadeira, executamos os comandos no corpo.
- Após o último comando do corpo, verificamos novamente a condição de parada.
- Se for verdadeira, executamos os comandos no corpo.
- ...
- Se a condição for falsa, então o comando de repetição termina e passa-se à linha seguinte ao corpo.
- Obs.: Se já na primeira vez a condição for falsa, então nenhum comando do corpo é executado nenhuma vez.

Funcionamento do laço “enquanto”

- Verificamos a expressão do cabeçalho (condição de parada).
- Se for verdadeira, executamos os comandos no corpo.
- Após o último comando do corpo, verificamos novamente a condição de parada.
- Se for verdadeira, executamos os comandos no corpo.
- ...
- Se a condição for falsa, então o comando de repetição termina e passa-se à linha seguinte ao corpo.
- Obs.: Se já na primeira vez a condição for falsa, então nenhum comando do corpo é executado nenhuma vez.
- Cada execução do corpo é chamada de **iteração** do laço.

Exemplo

- 1: $i \leftarrow 1$
- 2: **Enquanto** $i \leq 5$ **faça**
- 3: $i \leftarrow i + 1$
- 4: ESCREVA("O valor de i é", i)

comando	i
1	
2	
3	
4	
5	
6	
7	
8	
9	
10	

Exemplo

- 1: $i \leftarrow 1$
- 2: **Enquanto** $i \leq 10$ **faça**
- 3: $i \leftarrow i \times 2$
- 4: **ESCREVA**("O valor de i é", i)

comando	i
1	
2	
3	
4	
5	
6	
7	
8	
9	
10	

Exemplo

- 1: LEIA(i)
- 2: **Enquanto** não EH_PAR(i) **faça**
- 3: LEIA(i)
- 4: ESCREVA("O valor de i é", i)

comando	i
1	
2	
3	
4	
5	
6	
7	
8	
9	
10	

Exemplo

- 1: LEIA(i)
- 2: $cont \leftarrow 0$
- 3: **Enquanto** não EH_PAR(i) **faça**
- 4: LEIA(i)
- 5: $cont \leftarrow cont + 1$
- 6: ESCREVA("O valor de i é", i)

comando	i	$cont$
1		
2		
3		
4		
5		
6		
7		
8		
9		
10		

Repetição do tipo “para”

Laço “para”

- O comando **Para** permite que haja repetição de um bloco um número predeterminado de vezes.
- Sua sintaxe é:

- 1: **Para** *var* $\leftarrow$ *ini* **até** *fim*, **com** “expressão que muda *var*” **faça**
- 2: comando(s) executado(s) dependendo da comparação de *var* com *fim*
- 3: comando(s) executado(s) após o término do laço

Laço “para”

- O comando **Para** permite que haja repetição de um bloco um número predeterminado de vezes.
- Sua sintaxe é:

1: Para <i>var</i> $\leftarrow$ <i>ini</i> até <i>fim</i>, com “expressão que muda <i>var</i>” faça 2: comando(s) executado(s) dependendo da comparação de <i>var</i> com <i>fim</i> 3: comando(s) executado(s) após o término do laço

- Observe a estrutura similar à do comando **Se**: há um cabeçalho com as **condições** de execução e um **corpo**, que contém comandos indentados com relação ao cabeçalho.
- Deve haver ao menos uma instrução no corpo.

Funcionamento do laço “para”

Suponha que “expressão que muda *var*” aumenta o valor de *var*.

- Executamos a atribuição $var \leftarrow ini$.

Funcionamento do laço “para”

Suponha que “expressão que muda var ” aumenta o valor de var .

- Executamos a atribuição $var \leftarrow ini$.
- Se $var \leq fim$, executamos os comandos no corpo.

Funcionamento do laço “para”

Suponha que “expressão que muda var ” aumenta o valor de var .

- Executamos a atribuição $var \leftarrow ini$.
- Se $var \leq fim$, executamos os comandos no corpo.
- Após o último comando do corpo, executamos “expressão que muda var ”.

Funcionamento do laço “para”

Suponha que “expressão que muda var ” aumenta o valor de var .

- Executamos a atribuição $var \leftarrow ini$.
- Se $var \leq fim$, executamos os comandos no corpo.
- Após o último comando do corpo, executamos “expressão que muda var ”.
- Se $var \leq fim$, executamos os comandos no corpo.

Funcionamento do laço “para”

Suponha que “expressão que muda var ” aumenta o valor de var .

- Executamos a atribuição $var \leftarrow ini$.
- Se $var \leq fim$, executamos os comandos no corpo.
- Após o último comando do corpo, executamos “expressão que muda var ”.
- Se $var \leq fim$, executamos os comandos no corpo.
- Após o último comando do corpo, executamos “expressão que muda var ”.

Funcionamento do laço “para”

Suponha que “expressão que muda var ” aumenta o valor de var .

- Executamos a atribuição $var \leftarrow ini$.
- Se $var \leq fim$, executamos os comandos no corpo.
- Após o último comando do corpo, executamos “expressão que muda var ”.
- Se $var \leq fim$, executamos os comandos no corpo.
- Após o último comando do corpo, executamos “expressão que muda var ”.
- ...

Funcionamento do laço “para”

Suponha que “expressão que muda var ” aumenta o valor de var .

- Executamos a atribuição $var \leftarrow ini$.
- Se $var \leq fim$, executamos os comandos no corpo.
- Após o último comando do corpo, executamos “expressão que muda var ”.
- Se $var \leq fim$, executamos os comandos no corpo.
- Após o último comando do corpo, executamos “expressão que muda var ”.
- ...
- Se $var > fim$, então o comando de repetição termina e passa-se à linha seguinte ao corpo.

Funcionamento do laço “para”

Suponha que “expressão que muda var ” aumenta o valor de var .

- Executamos a atribuição $var \leftarrow ini$.
- Se $var \leq fim$, executamos os comandos no corpo.
- Após o último comando do corpo, executamos “expressão que muda var ”.
- Se $var \leq fim$, executamos os comandos no corpo.
- Após o último comando do corpo, executamos “expressão que muda var ”.
- ...
- Se $var > fim$, então o comando de repetição termina e passa-se à linha seguinte ao corpo.
- Obs.: Se já na primeira vez $var > fim$, então nenhum comando do corpo é executado nenhuma vez.

Funcionamento do laço “para”

Suponha que “expressão que muda var ” aumenta o valor de var .

- Executamos a atribuição $var \leftarrow ini$.
- Se $var \leq fim$, executamos os comandos no corpo.
- Após o último comando do corpo, executamos “expressão que muda var ”.
- Se $var \leq fim$, executamos os comandos no corpo.
- Após o último comando do corpo, executamos “expressão que muda var ”.
- ...
- Se $var > fim$, então o comando de repetição termina e passa-se à linha seguinte ao corpo.
- Obs.: Se já na primeira vez $var > fim$, então nenhum comando do corpo é executado nenhuma vez.
- Cada execução do corpo é chamada de **iteração** do laço.

Funcionamento do laço “para”

Suponha que “expressão que muda *var*” diminui o valor de *var*.

- Executamos a atribuição $var \leftarrow ini$.
- Se $var \geq fim$, executamos os comandos no corpo.
- Após o último comando do corpo, executamos “expressão que muda *var*”.
- Se $var \geq fim$, executamos os comandos no corpo.
- Após o último comando do corpo, executamos “expressão que muda *var*”.
- ...
- Se $var < fim$, então o comando de repetição termina e passa-se à linha seguinte ao corpo.
- Obs.: Se já na primeira vez $var < fim$, então nenhum comando do corpo é executado nenhuma vez.
- Cada execução do corpo é chamada de **iteração** do laço.

Note que

- 1: **Para** $var \leftarrow ini$ **até** fim , **com** expressão que muda var **faça**
- 2: comando(s) executado(s) dependendo da comparação de var com fim

faz exatamente o mesmo que

- 1: $var \leftarrow ini$
- 2: **Enquanto** compara var com fim **faça**
- 3: comando(s) executado(s) se a comparação for verdadeira
- 4: expressão que muda var

Exemplo

- 1: **Para** $i \leftarrow 1$ **até** 5, **com** $i \leftarrow i + 1$ **faça**
- 2: ESCREVA("O valor de i é", i)
- 3: ESCREVA("O valor de i é", i)

comando	i
1	
2	
3	
4	
5	
6	
7	
8	
9	
10	

Exemplo

- 1: **Para** $i \leftarrow 1$ **até** 10, **com** $i \leftarrow i \times 2$ **faça**
- 2: ESCREVA("O valor de i é", i)
- 3: ESCREVA("O valor de i é", i)

comando	i
1	
2	
3	
4	
5	
6	
7	
8	
9	
10	

Mais exemplos

Exemplo 1

Faça um algoritmo que recebe n números e mostre a soma daqueles que são pares.

Exemplo 1

Faça um algoritmo que recebe n números e mostre a soma daqueles que são pares.

```
1: LEIA( $n$ )
2:  $soma \leftarrow 0$ 
3: Para  $i \leftarrow 1$  até  $n$ , com  $i \leftarrow i + 1$  faça
4:     LEIA( $num$ )
5:     Se EH_PAR( $num$ ) então
6:          $soma \leftarrow soma + num$ 
7: ESCREVA("A soma dos valores pares é",  $soma$ )
```

Exemplo 1 - Outra versão

Faça um algoritmo que recebe n números e mostre a soma daqueles que são pares.

```
1: LEIA( $n$ )
2:  $soma \leftarrow 0$ 
3:  $i \leftarrow 1$ 
4: Enquanto  $i \leq n$  faça
5:     LEIA( $num$ )
6:     Se EH_PAR( $num$ ) então
7:          $soma \leftarrow soma + num$ 
8:      $i \leftarrow i + 1$ 
9: ESCREVA("A soma dos valores pares é",  $soma$ )
```

Exemplo 2

Faça um algoritmo que recebe n inteiros e apresenta o maior deles.

Exemplo 2

Faça um algoritmo que recebe n inteiros e apresenta o maior deles.

```
1: LEIA( $n$ )
2: LEIA( $maior$ )
3: Para  $i \leftarrow 2$  até  $n$ , com  $i \leftarrow i + 1$  faça
4:     LEIA( $num$ )
5:     Se  $num > maior$  então
6:          $maior \leftarrow num$ 
7: ESCREVA("O maior número lido é",  $maior$ )
```

Exemplo 3

Faça um algoritmo que calcule o conceito final dos n alunos dessa disciplina.

Exemplo 3

Faça um algoritmo que calcule o conceito final dos n alunos dessa disciplina.

```
1: LEIA( $n$ )
2: Para  $i \leftarrow 1$  até  $n$ , com  $i \leftarrow i + 1$  faça
3:   LEIA( $nome$ )
4:   LEIA( $P$ ,  $EP$ ,  $ET$ )
5:    $MF \leftarrow 0.45 \times P + 0.45 \times EP + 0.1 \times ET$ 
6:    $conceito \leftarrow \text{CONCEITOPreRec}(MF)$ 
7:   ESCREVA("O conceito de",  $nome$ , "é",  $conceito$ )
```

Pratique!

Exercício 1

Qual o resultado da chamada MAGIC(10)? E MAGIC(1)?

- 1: **Função** MAGIC(INTEIRO: x)
- 2: **Enquanto** ($x > 0$ e $x < 10$) ou ($x > 10$ e $x < 20$) **faça**
- 3: $x \leftarrow x + 5$
- 4: **Devolve** x

Exercício 2

Qual o resultado da chamada `ABACATE(3)`?

```
1: Função ABACATE(INTEIRO:  $x$ )  
2:   INTEIRO:  $s$ ,  $p$ ,  $i$   
3:    $i \leftarrow 0$   
4:    $p \leftarrow 1$   
5:   Enquanto  $p \leq x$  faça  
6:      $s \leftarrow 1$   
7:     Enquanto  $s \leq x$  faça  
8:        $i \leftarrow i + 1$   
9:        $s \leftarrow s + 1$   
10:     $p \leftarrow p + 1$   
11:  Devolve  $i$ 
```

Exercício 3

Faça uma função que calcule $n!$, para qualquer inteiro $n \geq 0$.

Exercício 4

Faça uma função que encontre os divisores de um inteiro n .