



UNINASSAU

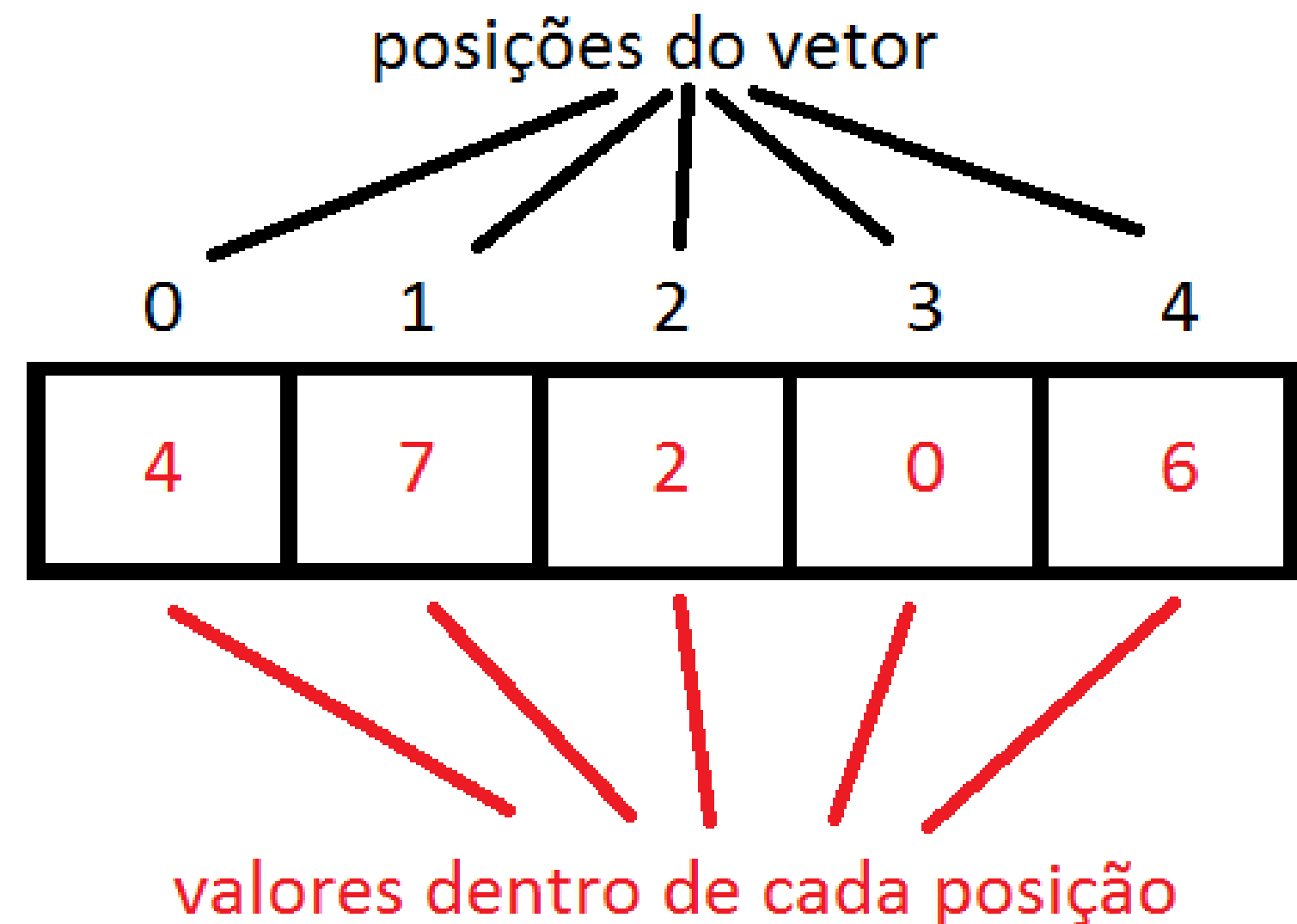
INTRODUÇÃO A VETORES, MATRIZES E ESTRUTURAS DE REPETIÇÃO

Análise e Desenvolvimento de Sistemas
Lógica e Programação Algorítmica
Profº Lindenberg Andrade

VETOR EM PROGRAMAÇÃO

Em programação, um vetor são arrays unidimensionais que armazenam uma sequência de elementos em uma única linha ou coluna. Ou seja, ele é uma estrutura de dados que armazena uma sequência de elementos do mesmo tipo. Os elementos são organizados em posições consecutivas na memória.

vetor = [1, 2, 3, 4, 5]
Um vetor com 5
elementos

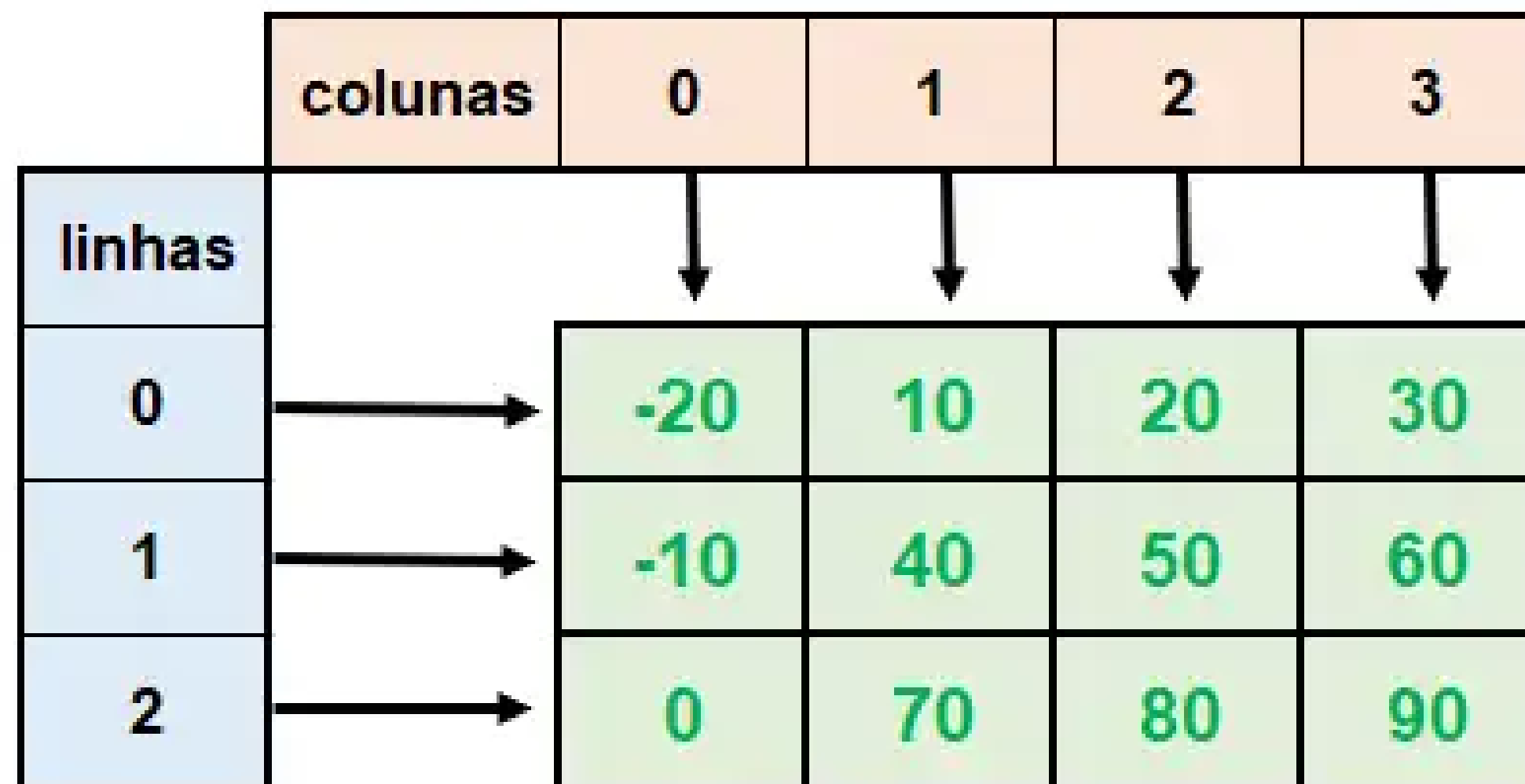


MATRIZ EM PROGRAMAÇÃO

Matrizes são arrays bidimensionais, organizados em linhas e colunas, formando uma tabela.

Representação de uma matriz

	colunas	0	1	2	3
linhas					
0		-20	10	20	30
1		-10	40	50	60
2		0	70	80	90



ESTRUTURA PARA

O **para** é uma estrutura de repetição usada para executar um bloco de código um número determinado de vezes. Ele é conhecido como loop for em muitas linguagens.

Usada quando o número de repetições for conhecido durante a elaboração do algoritmo ou quando puder ser fornecido durante a execução.

```
para ( valor inicial ; <condição >; <valor do incremento> )  
{  
    bloco de comandos  
}
```

ESTRUTURA PARA

< valor inicial>; – é uma expressão do tipo:

```
<identificador> <- <valor inicial> ;
```

Exemplos:

a <- 0;

A variável a recebe o valor inicial 0.

c <- 100;

A variável c recebe o valor inicial 100.

x <- strtam(pal) - 1;

A variável x recebe o valor inicial que é igual ao número de caracteres da variável pal menos 1.

ESTRUTURA PARA

< condição >; – é uma expressão do tipo:

<identificador> <=, <, > ou >= <valor final> ;

Exemplos:

- | | |
|----------|--|
| a <= 10; | A variável a poderá receber valores enquanto forem menores ou iguais a 10. |
| c >= 2; | A variável c poderá receber valores enquanto forem maiores ou iguais a 2. |
| x >= 0; | A variável x poderá receber valores enquanto forem maiores ou iguais a 0. |

ESTRUTURA PARA

< valor do incremento> - é uma expressão do tipo:

<identificador> <- <identificador> + ou - valor

Exemplos:

a <- a + 1; A variável a é incrementada de 1. /* contador */

c <- c - 2; A variável c é decrementada de 2.

x <- x - 1; A variável x é decrementada de 1.

Já vimos que os operadores ++ e -- são equivalentes aos comandos de atribuição quando a variável é incrementada ou decrementada de 1; por isso, o primeiro e o terceiro exemplos poderiam ser escritos assim:

a ++; A variável a é incrementada em 1.

x --; A variável x é decrementada de 1.

ESTRUTURA PARA

Componentes do **para (for)**:

Inicialização → Define a variável de controle
(ex.: `int i = 1;`).

Condição → Mantém o loop rodando
enquanto for verdadeira (ex.: `i <= 5`).

Incremento/Decremento → Atualiza a variável
de controle (ex.: `i++`).

ESTRUTURA PARA

Componentes do **para (for)**:

Inicialização → Define a variável de controle
(ex.: `int i = 1;`).

Condição → Mantém o loop rodando
enquanto for verdadeira (ex.: `i <= 5`).

Incremento/Decremento → Atualiza a variável
de controle (ex.: `i++`).

EXEMPLO PARA

Exemplo 1: ExemploPara

```
1 Algoritmo "ExemploPara"  
2 // Disciplina      : [Lógica de Programação Algorítmica]  
3 // Professor       : José Lindenberg de Andrade  
4 // Descrição       : Exemplo Para  
5 var  
6     i: inteiro  
7 inicio  
8     para i de 1 até 5 faça  
9         escreval("Passo ", i, "\n")  
10    fimpara  
11 fimalgoritmo
```

EXEMPLO PARA

Exemplo 1: ExemploPara

 Console simulando o modo texto do MS-DOS

Passo 1\n

Passo 2\n

Passo 3\n

Passo 4\n

Passo 5\n

>>> Fim da execução do programa !

EXEMPLO PARA

Exemplo 2: Contagem Regressiva

```
1 Algoritmo "ExemploPara"  
2 // Disciplina      : [Lógica de Programação Algorítmica]  
3 // Professor       : José Lindenberg de Andrade  
4 // Descrição        : Contagem Regressiva  
5 var  
6     i: inteiro  
7 inicio  
8     para i de 10 até 1 passo -1 faça  
9         escreval(i)  
10    fimpara  
11    escreva("FIM!")  
12 fimalgoritmo
```

EXEMPLO PARA

Exemplo 2: Contagem Regressiva

 Console simulando o modo texto do MS-DOS

```
10  
9  
8  
7  
6  
5  
4  
3  
2  
1
```

```
FIM!
```

```
>>> Fim da execução do programa !
```

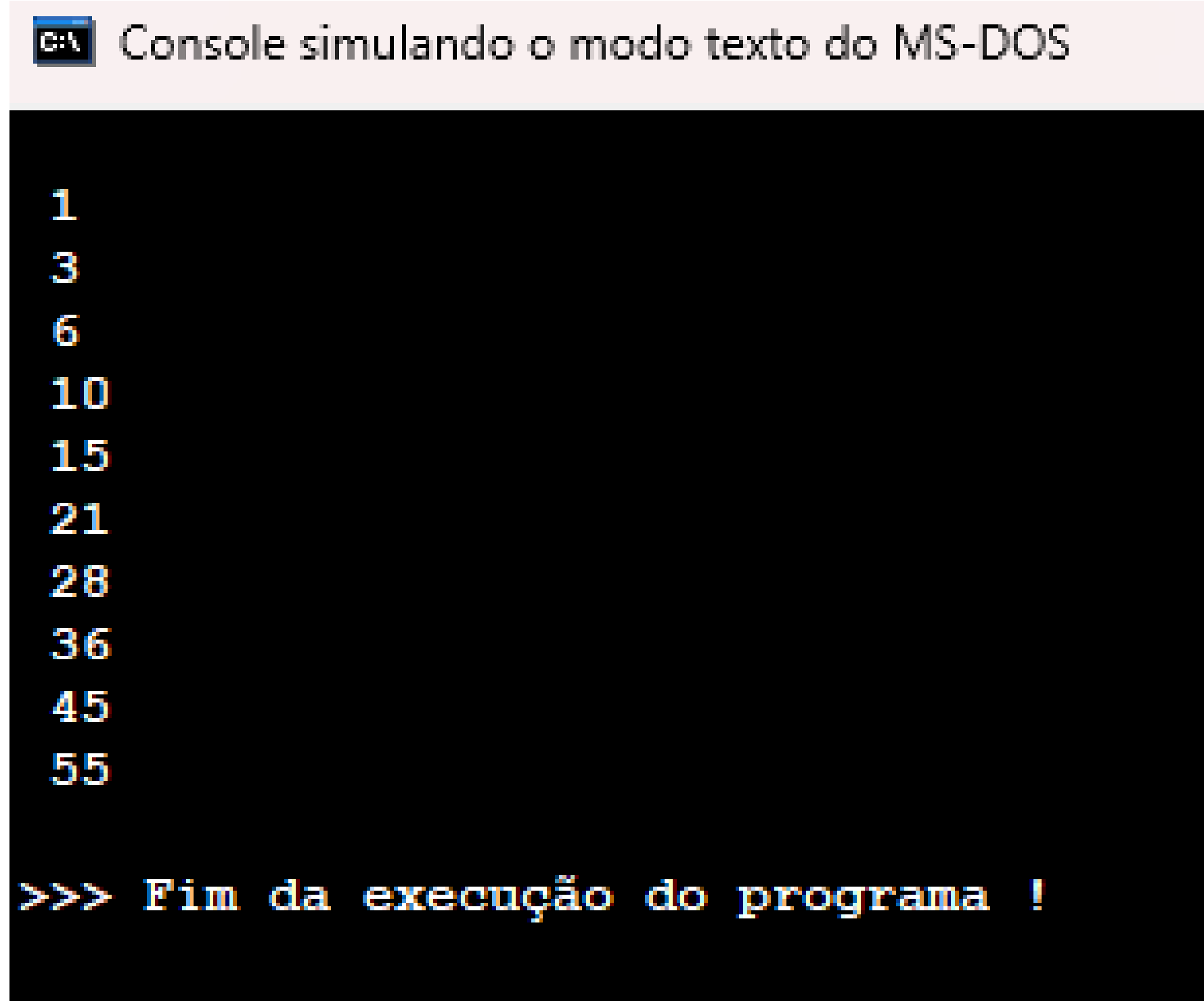
EXEMPLO PARA

Exemplo 3: Exemplo com Vetores

```
1 Algoritmo "ExemploPara"
2 // Disciplina   : [Lógica de Programação Algorítmica]
3 // Professor    : José Lindenberg de Andrade
4 // Descrição    : Exemplo com Vetores
5 var
6     cont: inteiro
7     vet: vetor [1..10] de inteiro
8 inicio
9     para cont de 1 até 10 passo 1 faça
10         se cont = 1 entao
11             vet[cont] <- cont
12         senao
13             vet[cont] := vet[cont-1] + cont
14         fimse
15         escreval (vet[cont])
16     fimpara
17 fimalgoritmo
18
19
```

EXEMPLO PARA

Exemplo 3: Exemplo com Vetores



```
BN Console simulando o modo texto do MS-DOS

1
3
6
10
15
21
28
36
45
55

>>> Fim da execução do programa !
```

ESTRUTURA REPITA

O comando repita é uma estrutura de repetição que executa um bloco de código até que uma determinada condição seja atendida.

```
enquanto ( condição)
{
    bloco de comandos
}
```

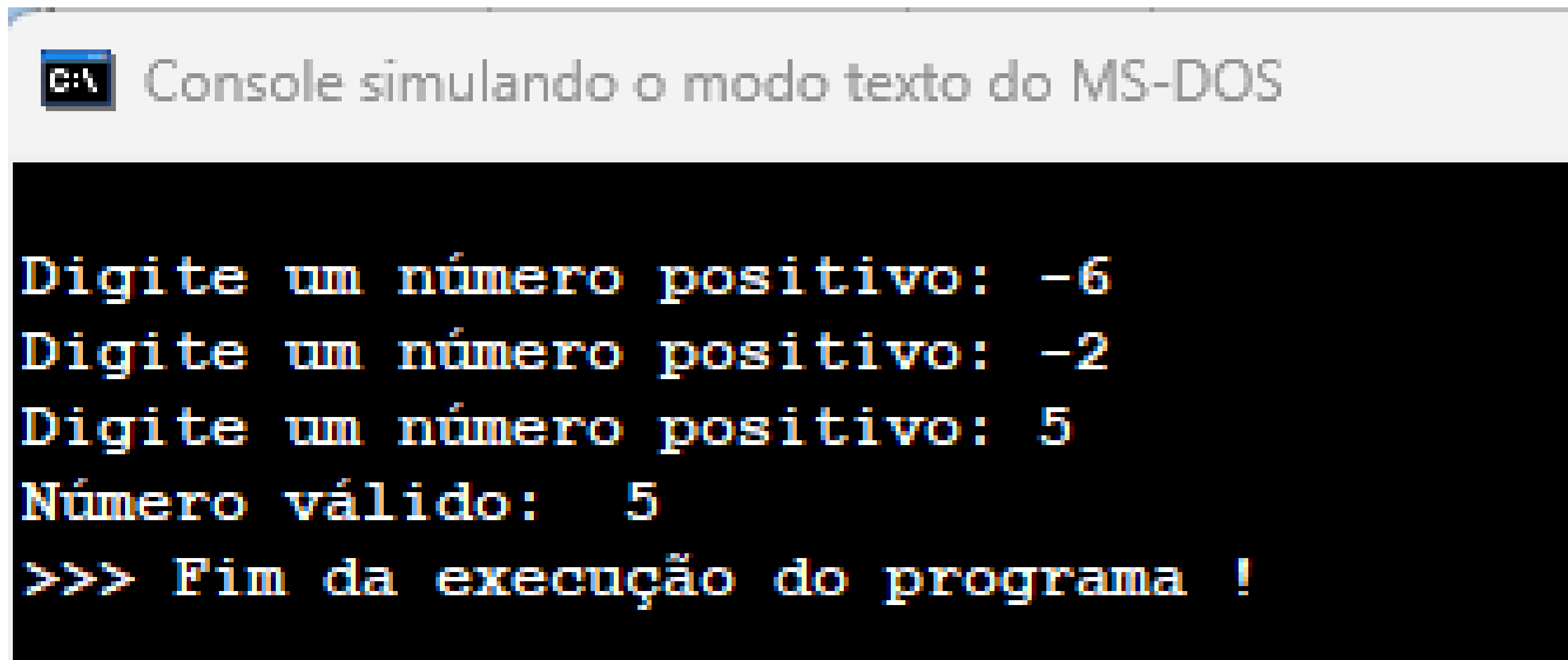

EXEMPLO REPITA

Exemplo 4: Números Positivos

```
1 algoritmo "NumeroPositivo"  
2 // Disciplina      : [Lógica de Programação Algorítmica]  
3 // Professor       : José Lindenberg de Andrade  
4 // Descrição        : Verificar se o numero é positivo  
5  
6 var  
7     num: inteiro  
8 inicio  
9     repita  
10         escreva("Digite um número positivo: ")  
11         leia(num)  
12         até num > 0  
13         escreva("Número válido: ", num)  
14 fimalgoritmo  
15
```

EXEMPLO REPITA

Exemplo 4: Números Positivos



```

C:\> Console simulando o modo texto do MS-DOS

Digite um número positivo: -6
Digite um número positivo: -2
Digite um número positivo: 5
Número válido: 5
>>> Fim da execução do programa !

```

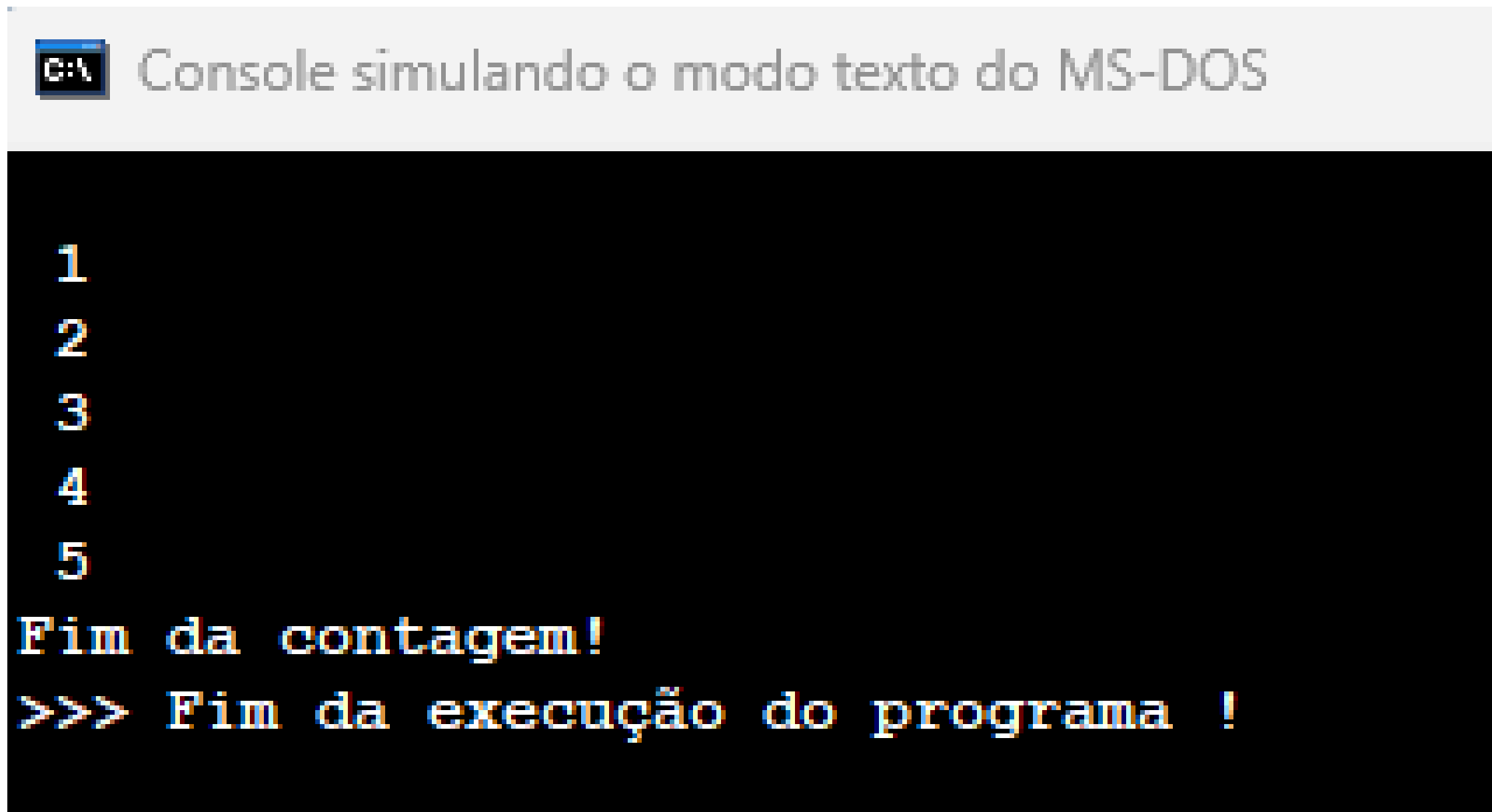
EXEMPLO REPITA

Exemplo 5: Contagem Progressiva

```
1 algoritmo "Contagem"  
2 // Disciplina    : [Lógica de Programação Algorítmica]  
3 // Professor     : José Lindenberg de Andrade  
4 // Descrição     : Contagem Progressiva  
5  
6 var  
7     i: inteiro  
8 inicio  
9     i <- 1  
10    repita  
11        escreval(i)  
12        i <- i + 1  
13    até i > 5  
14    escreva("Fim da contagem!")  
15 fimalgoritmo
```

EXEMPLO REPITA

Exemplo 5: Contagem Progressiva



```
C:\ Console simulando o modo texto do MS-DOS

1
2
3
4
5
Fim da contagem!
>>> Fim da execução do programa !
```

ESTRUTURA ENQUANTO

É uma estrutura de repetição que executa um bloco de código pelo menos uma vez, e continua executando enquanto uma condição for verdadeira.

É semelhante ao repita... até, mas com a condição invertida:

repita ... até → executa até que a condição seja verdadeira.

faça ... enquanto → executa enquanto a condição for verdadeira.

```
faca
{
    bloco de comandos
}
enquanto (condição)
```

EXEMPLO ENQUANTO

Exemplo 6: Contagem Progressiva

```
1 algoritmo "Contagem2"  
2 // Disciplina   : [Lógica de Programação Algorítmica]  
3 // Professor    : José Lindenberg de Andrade  
4 // Descrição    : Contagem Progressiva  
5  
6 var  
7     i: inteiro  
8 inicio  
9     i <- 1  
10    enquanto i <= 5 faça  
11        escreval(i)  
12        i <- i + 1  
13    fimenquanto  
14    escreva("Fim da contagem!")  
15 fimalgoritmo  
16  
17
```

 Console simulando o modo texto do MS-DOS

```
1  
2  
3  
4  
5  
Fim da contagem!  
>>> Fim da execução do programa !
```

EXEMPLO ENQUANTO

Exemplo 7: Número Positivo

```
1 algoritmo "NumeroPositivo"
2 // Disciplina      : [Lógica de Programação Algorítmica]
3 // Professor       : José Lindenberg de Andrade
4 // Descrição       : Contagem Progressiva
5
6 var
7     num: inteiro
8 inicio
9     enquanto num <= 0 faça
10         escreval("Digite um número positivo: ")
11         leia(num)
12     fimenquanto
13     escreval("Número válido: ", num)
14 fimalgoritmo
```

EXEMPLO ENQUANTO

Exemplo 7: Número Positivo



Console simulando o modo texto do MS-DOS

```
Digite um número positivo:
```

```
-2
```

```
Digite um número positivo:
```

```
-50
```

```
Digite um número positivo:
```

```
3
```

```
Número válido: 3
```

```
>>> Fim da execução do programa !
```


DICA DE FILME: A CONSPIRAÇÃO DA LÂMPADA (2010)



A obsolescência programada é um mecanismo que provoca o encurtamento da vida de um produto para garantir uma demanda contínua. Ela está no âmago da sociedade de consumo, movida pelo desperdício. A economia moderna, baseada no crescimento econômico, conseguiria se sustentar sem a obsolescência programada? Essa é uma das questões chave colocada pelo filme A Conspiração da Lâmpada.

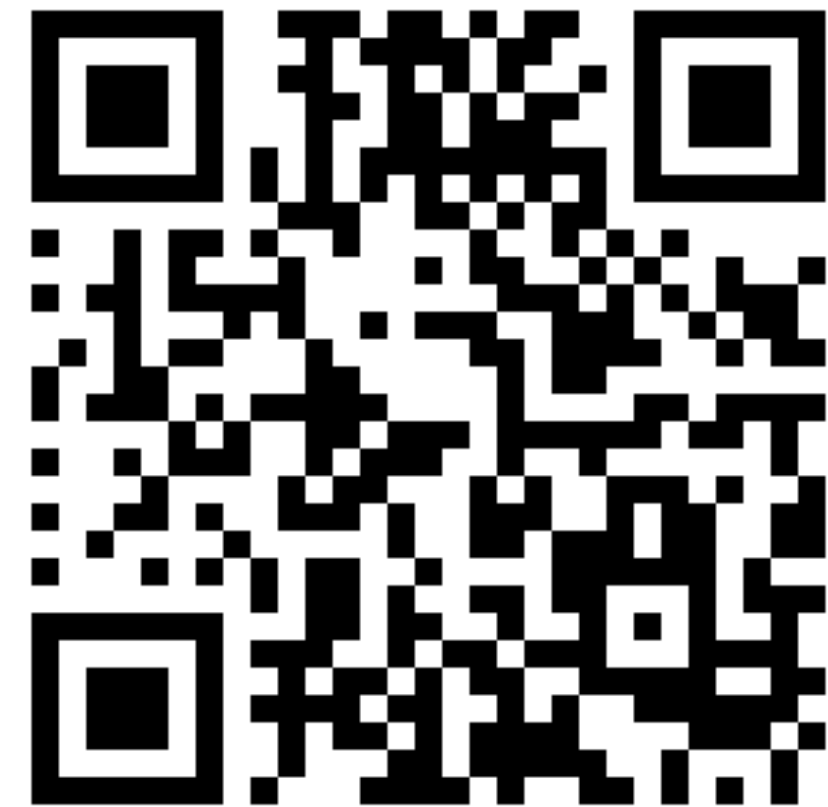
REFERÊNCIAS

- ALENCAR FILHO, E. de. Iniciação à lógica matemática. São Paulo: Nobel, 2002.
- VAZ, R. M. Formalização do raciocínio lógico baseada na lógica matemática. Dissertação (Mestrado Profissional em Matemática) – Universidade Federal do Mato Grosso do Sul, Três Lagoas, 2014. Disponível em <https://repositorio.ufms.br/handle/123456789/2333> .

Dúvidas?



Profº Lindenberg Andrade
E-mail: linndemberg1@gmail.com



Additional contacts via QR code