

UNICESUMAR
ENGENHARIA CIVIL
PROGRAMAÇÃO PARA ENGENHARIA (NGER80_271)
ANDRÉ MARTINS OTOMURA

```
<i class= .  
<h3>Happy Clients</h3>  
</div>  
</div>  
<!-- end first count item -->  
<!-- second count item -->  
<div class="col-md-3 col-sm-6 col-xs-12 text-center wow fadeInDown"  
  data-wow-duration="500ms" data-wow-delay="200ms">  
  <div class="counters-item">  
    <div>  
      <span data-speed="3000" data-to="565">565</span>  
    </div>  
  </div>  
</div>
```

2ª AULA

```
80 @media  
81  
82 .navbar-inverse .navbar-nav>li>a {  
83   padding: 30px 10px;  
84 }  
85 .navbar-inverse .smaller .navbar-nav>li>a {  
86   padding: 20px 10px;  
87 }  
88 .navbar-inverse .navbar-nav>li {  
89   padding-right: 0;  
90 }  
91 .carousel-caption h2 {
```

Exercícios

Matrizes e vetores*

Estruturas condicionais (If - Else e Case) e suas implementações*

Estruturas de repetição (While e For) e suas implementações*

Funções e procedimentos*

Manipulação de arquivos

DICAS PARA SOLUCIONAR PROBLEMAS

1. ANOTAR OS DADOS FORNECIDOS PELO ENUNCIADO DO PROBLEMA E QUE O PROGRAMA UTILIZARÁ COMO ENTRADA (EX: NOME, VALOR, QUANTIDADE...);
2. ANOTAR TUDO O QUE PRECISA SER ENTREGUE NO FINAL, O QUE SERÁ APRESENTADO COMO RESPOSTA;
3. ESBOÇAR AS FÓRMULAS OU EQUAÇÕES QUE SERÃO UTILIZADAS PELO PROGRAMA PARA ENCONTRAR CADA UMA DAS RESPOSTAS NECESSÁRIAS;
4. FAZER UM LEVANTAMENTO DAS VARIÁVEIS NECESSÁRIAS PARA UTILIZAR AS FÓRMULAS E TAMBÉM PARA APRESENTAR OS RESULTADOS;
5. LISTAR O PASSO A PASSO DO QUE O PROGRAMA DEVERÁ EXECUTAR, NA ORDEM CERTA QUE POSSIBILITE CHEGAR AO RESULTADO;
6. ANALISAR E LISTAR OS COMANDOS DA LINGUAGEM DE PROGRAMAÇÃO REFERENTES AOS PASSOS LISTADOS ANTERIORMENTE;
- 7.

3. Construa um algoritmo que leia o preço de um produto, o percentual de desconto e calcule o valor a pagar e o valor do desconto.

5. Escreva um algoritmo que leia o valor da hora aula, o número de aulas dadas no mês e o percentual de desconto do INSS. Calcule e apresente o salário líquido e o salário bruto.

#	TÓPICOS	04/jan	05/jan	08/jan	09/jan	10/jan	11/jan	12/jan	15/jan	16/jan	17/jan	18/jan	19/jan	22/jan	23/jan
a01	Conceitos de hardware														
a02	Conceitos e definições de algoritmos														
a03	Diagrama de blocos e pseudocódigos														
a04	Variáveis														
a05	Constantes														
a06	Comandos de atribuição														
a07	Comandos de entrada e saída de dados														
a08	Matrizes e vetores														
a09	Estruturas condicionais (If - Else e Case) e suas implementações														
a10	Estruturas de repetição (While e For) e suas implementações														
a11	Funções e procedimentos														
a12	Manipulação de arquivos														
b01	Introdução ao software MATLAB: comandos e ferramentas básicas do software														
b02	Gráficos 2D e 3D														
b03	Funções e subalgoritmos em MATLAB														
b04	Introdução ao Microsoft Excel: funções de busca e referência e funções lógicas														
b05	Microsoft Excel para engenharia: atingir a meta de solver, cenários e tabela de dados														
b06	Formulários e VBA														
c01	Sistemas de equações lineares														
c02	Zeros de equações algébricas e transcendentais														
c03	Interpolações e aproximações de funções														
c04	Integração numérica														
c05	Soluções numéricas de equações diferenciais														

Horário: de segunda a sexta-feira, das 13h00 até 17h30.
Local: Sala 15, Bloco 06
Intervalo1: 14h30 até 14h40
Intervalo2: 16h00 até 16h10

- Conceitos de hardware;
- Conceitos e definições de algoritmos;
- Diagrama de blocos e pseudocódigos;
- Variáveis;
- Constantes;
- Comandos de atribuição;
- Comandos de entrada e saída de dados.

UNICESUMAR
ENGENHARIA CIVIL
PROGRAMAÇÃO PARA ENGENHARIA (NGER80_271)
ANDRÉ MARTINS OTOMURA

```
<i class= .  
<h3>Happy Clients</h3>  
</div>  
</div>  
<!-- end first count item -->  
<!-- second count item -->  
<div class="col-md-3 col-sm-6 col-xs-12 text-center wow fadeInDown"  
  data-wow-duration="500ms" data-wow-delay="200ms">  
  <div class="counters-item">  
    <div>  
      <span data-speed="3000" data-to="565">565</span>  
    </div>  
  </div>  
</div>
```

2ª AULA

```
80 @media  
81  
82 .navbar-inverse .navbar-nav>li>a {  
83   padding: 30px 10px;  
84 }  
85 .navbar-inverse .smaller .navbar-nav>li>a {  
86   padding: 20px 10px;  
87 }  
88 .navbar-inverse .navbar-nav>li {  
89   padding-right: 0;  
90 }  
91 .carousel-caption h2 {
```

Exercícios

Matrizes e vetores*

Estruturas condicionais (If - Else e Case) e suas implementações*

Estruturas de repetição (While e For) e suas implementações*

Funções e procedimentos*

Manipulação de arquivos

"MÓDULO, DIREÇÃO E SENTIDO"



$x = [a \ b \ c \ d \ e]$

Ex.:

$x = [1 \ 2 \ 3 \ 4 \ 5]$

$y = [1,1 \ 23,4 \ 143,1]$

$z = ["nome" \ "apelido" \ "sobrenome"]$

! NOTAÇÃO EM DIFERENTES SOFTWARES.

Declaração de vetores:

VISUALG (pseudocódigo)

x: vetor [1..n] de tipo_da_variável

Ex.:

y: vetor [1..3] de INTEIRO

z: vetor [1..10] de CARACTERE

Atribuição de valores no vetor:

VISUALG (pseudocódigo)

x[n] <- a

Ex.:

y[1] <- 25

z[2] <- "meu_nome"

Posteriormente, quando formos utilizar os outros softwares de programação, será disponibilizada uma tabela de comandos básica de cada software.

EXERCÍCIO

Algoritmo "Soma de elementos de vetores"

Var

// Seção de Declarações das variáveis

vetor1: vetor [1..3] de real

vetor2: vetor [1..3] de real

vetor_soma: vetor [1..3] de real

Inicio

// Atribuição de valores para o vetor1

vetor1[1] <- 11

vetor1[2] <- 12

vetor1[3] <- 13

// Atribuição de valores para o vetor2

vetor2[1] <- 21

vetor2[2] <- 22

vetor2[3] <- 23

// Atribuição de valores para o vetor_soma

vetor_soma[1] <- vetor1[1] + vetor2[1]

vetor_soma[2] <- vetor1[2] + vetor2[2]

vetor_soma[3] <- vetor1[3] + vetor2[3]

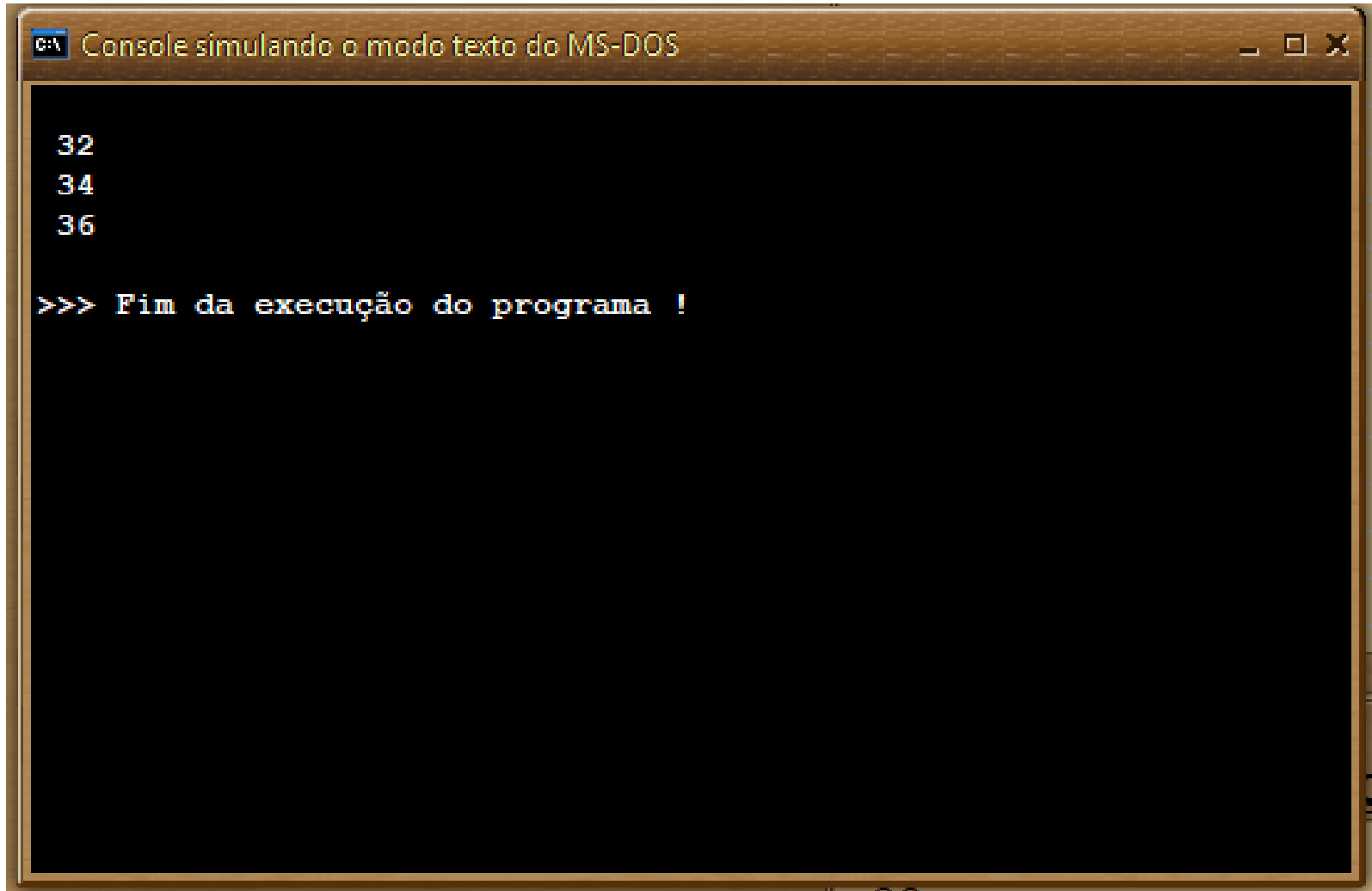
// Mostrar resultados (saída de dados)

escreval(vetor_soma[1])

escreval(vetor_soma[2])

escreval(vetor_soma[3])

Fimalgoritmo



The image shows a screenshot of a console window titled "C:\> Console simulando o modo texto do MS-DOS". The window has a black background with white text. The text displayed is:

```
32  
34  
36  
  
>>> Fim da execução do programa !
```

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}$$

$x = [a \ b \ c \ d \ e]$

Ex.:

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$x = [1 \ 2 \ 3 \ 4 \ 5; 21 \ 22 \ 23 \ 24 \ 25]$

$y = [1,1; \ 23,4; \ 143,1]$

$z = ["nome"; \ "apelido"; \ "sobrenome"; \ "e-mail"]$

Declaração de matrizes:

VISUALG (pseudocódigo)

x: vetor [1..n, 1..m] de tipo_da_variável

Ex.:

y: vetor [1..3, 1..4] de INTEIRO

z: vetor [1..5, 1..5] de CARACTERE

Atribuição de valores na matriz:

VISUALG (pseudocódigo)

x[n,m] <- a

Ex.:

y[1,1] <- 25

z[4,3] <- "meu_nome"

Posteriormente, quando formos utilizar os outros softwares de programação, será disponibilizada uma tabela de comandos básica de cada software.

EXERCÍCIO

Algoritmo "Soma de elementos de matrizes"

Var

// Seção de Declarações das variáveis

matriz1: vetor [1..2, 1..2] de real

matriz2: vetor [1..2, 1..2] de real

matriz_soma: vetor [1..2, 1..2] de caractere

Inicio

// Atribuição de valores para a matriz1

matriz1[1,1] <- 111

matriz1[1,2] <- 112

matriz1[2,1] <- 121

matriz1[2,2] <- 122

// Atribuição de valores para a matriz2

matriz2[1,1] <- 211

matriz2[1,2] <- 212

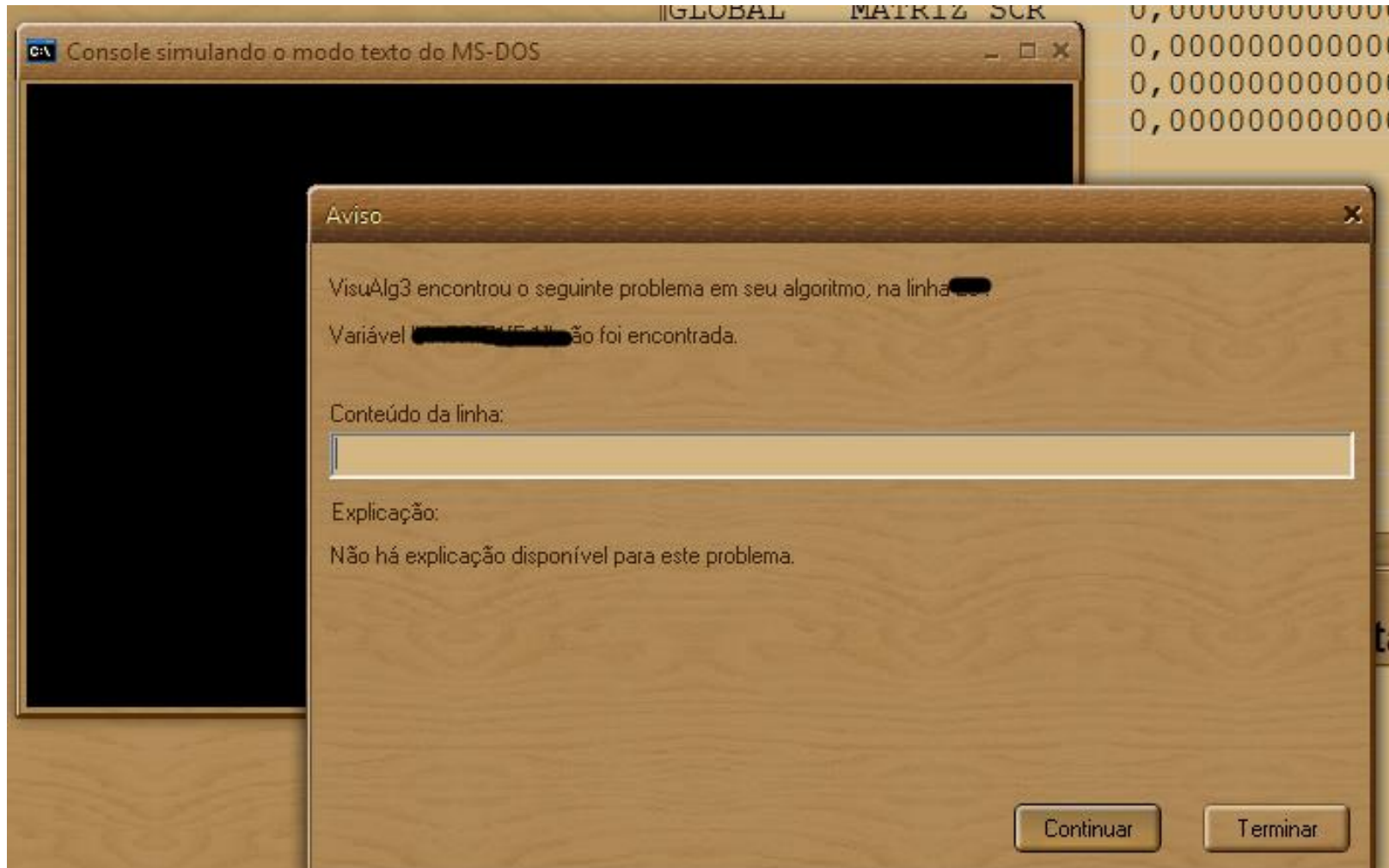
matriz2[2,1] <- 221

matriz2[2,2] <- 222

```
// Atribuição de valores para a matriz_soma  
matriz_soma[1,1] <- matriz1[5,1] + matriz2[5,1]  
matriz_soma[1,2] <- matriz1[1,2] + matriz2[1,2]  
matriz_soma[2,1] <- matriz1[1,3] + matriz2[1,3]  
matriz_soma[2,2] <- matriz1[2,4] + matriz2[2,4]
```

```
// Mostrar resultados (saída de dados)  
escreval(matriz_soma[1,1])  
escreval(matriz_soma[1,2])  
escreval(matriz_soma[2,1])  
escreval(matriz_soma[2,2])
```

Fimalgoritmo



Algoritmo "Soma de elementos de matrizes"

Var

```
// Seção de Declarações das variáveis  
matriz1: vetor [1..2, 1..2] de real  
matriz2: vetor [1..2, 1..2] de real  
matriz_soma: vetor [1..2, 1..2] de real
```

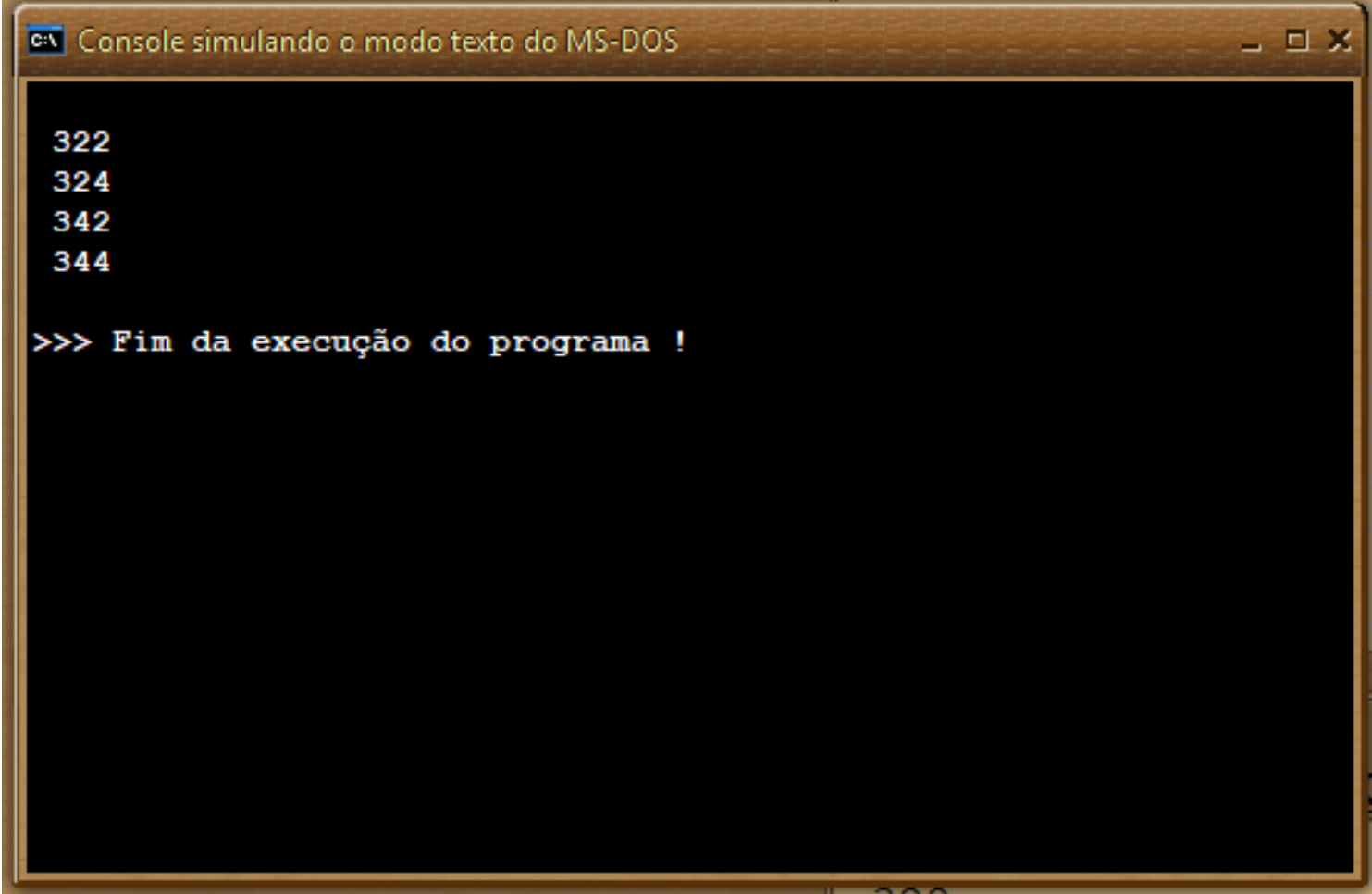
Inicio

```
// Atribuição de valores para a matriz1  
matriz1[1,1] <- 111  
matriz1[1,2] <- 112  
matriz1[2,1] <- 121  
matriz1[2,2] <- 122  
  
// Atribuição de valores para a matriz2  
matriz2[1,1] <- 211  
matriz2[1,2] <- 212  
matriz2[2,1] <- 221  
matriz2[2,2] <- 222
```

```
// Atribuição de valores para a matriz_soma  
matriz_soma[1,1] <- matriz1[1,1] + matriz2[1,1]  
matriz_soma[1,2] <- matriz1[1,2] + matriz2[1,2]  
matriz_soma[2,1] <- matriz1[2,1] + matriz2[2,1]  
matriz_soma[2,2] <- matriz1[2,2] + matriz2[2,2]
```

```
// Mostrar resultados (saída de dados)  
escreval(matriz_soma[1,1])  
escreval(matriz_soma[1,2])  
escreval(matriz_soma[2,1])  
escreval(matriz_soma[2,2])
```

Fimalgoritmo



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

322
324
342
344

>>> Fim da execução do programa !
```



TESTES

DESVIOS NO FLUXO DO CÓDIGO

Se (<Condição>) entao

 <instruções para condição verdadeira>

fimse

Algoritmo "exemplo"

Var

 A : inteiro

Inicio

 Escreva("Digite um valor qualquer: ")

 Leia (A)

 Se (A > 10) então

 Escreva ("A é maior que 10")

 Fimse

Fimalgoritmo

ESTRUTURAS CONDICIONAIS COMPOSTAS

Algoritmo "parimpar"

Var

 n: inteiro

Inicio

 Escreva ("Digite um número:")

 Leia (n)

 Se (n mod 2 = 0) entao

 Escreva ("O número é par")

 Senao

 Escreva ("O número é ímpar")

 Fimse

Fimalgoritmo

ESTRUTURAS CONDICIONAIS ANINHADAS

```
Algoritmo "maior"
Var
    a, b, c, max: inteiro
Inicio
    Escreva ("Digite o primeiro número inteiro:")
    Leia (a)
    Escreva ("Digite o segundo número inteiro:")
    Leia (b)
    Escreva ("Digite o terceiro número inteiro:")
    Leia (c)
    Se (a > b) entao
        Se (a > c) entao
            max ← a
        Senao
            max ← c
        Fimse
    senao
        Se (b>c) entao
            max ← b
        senao
            max ← c
        Fimse
    Fimse
    Escreva("O maior número é:", max)
Fimalgoritmo
```

VISUALG:

```
escolha <variável>  
    caso <valor 1>  
        <instrução 1>  
    caso <valor 2>  
        <instrução 2>  
    caso <valor 3>  
        <instrução 3>  
  
    outrocaso  
        < instrução >  
fimescolha
```

OUTRO TIPO DE ESCRITA:

```
caso <variável>  
    seja <valor 1> faça <instrução 1>  
    seja <valor 2> faça <instrução 2>  
    seja <valor 3> faça <instrução 3>  
    senao < instrução >  
Fimcaso
```

```
Algoritmo "Mês"
Var
    num: inteiro
Inicio
    Escreva ("Digite um número de 1 a 12:")
    Leia (num)
    Escolha (num)
        caso 1
            Escreva ("Janeiro")
        caso 2
            Escreva ("Fevereiro")
        caso 3
            Escreva ("Março")
        caso 4
            Escreva ("Abril")
        caso 5
            Escreva ("Maio")

        outrocaso
            Escreva ("Mês inválido")
    Fimescolha
Fimalgoritmo
```

```
Algoritmo "Mês"
Var
    num: inteiro
Inicio
    Escreva ("Digite um número de 1 a 12:")
    Leia (num)
    Escolha (num)
        caso 1
            Escreva ("Janeiro")
        caso 2
            Escreva ("Fevereiro")
        caso 3
            Escreva ("Março")
        caso 4
            Escreva ("Abril")
        caso 5
            Escreva ("Maio")
        caso 6
            Escreva ("Junho")
        caso 7
            Escreva ("Julho")
        caso 8
            Escreva ("Agosto")
        caso 9
            Escreva ("Setembro")
        caso 10
            Escreva ("Outubro")
        caso 11
            Escreva ("Novembro")
        caso 12
            Escreva ("Dezembro")

        outrocaso
            Escreva ("Mês inválido")
    Fimescolha
Fimalgoritmo
```



PROCESSOS ITERATIVOS
CÁLCULOS LONGOS

WHILE/ENQUANTO

FOR/PARA

DO WHILE/REPITA

FOR

```
para <variável> de <início> ate <fim> passo <incremento> faca  
    <instruções>  
fimpara
```

ESTRUTURAS DE REPETIÇÃO: FOR (PARA)

Algoritmo "tabuada"

Var

Num, i, mult: inteiro

Inicio

Escreva ("Digite um número:")

Leia (num)

Para i de 1 ate 10 passo 1 faca

mult \leftarrow num*i

Escreva (num, "x", i, "=", mult)

fimpara

Fimalgoritmo

WHILE

```
Enquanto <condição> faça
```

```
    <instruções>
```

```
fimenquanto
```

ESTRUTURAS DE REPETIÇÃO: WHILE (ENQUANTO)

Algoritmo "conta"

Var

Num, cont: inteiro

Inicio

Escreva ("Digite um número:")

Leia (num)

cont \leftarrow 0

Enquanto (num \neq 0) faça

 Se (num \geq 100) e (num \leq 300) entao

 cont \leftarrow cont + 1

 fimse

 Escreva ("Digite um número:")

 Leia (num)

 fimenquanto

Escreva ("A quantidade de números entre 100 e 300 é:", cont)

Fimalgoritmo

do while

Repita

<instruções>

Ate <condição>

ESTRUTURAS DE REPETIÇÃO: DO WHILE (REPITA)

Algoritmo "conta"

Var

Num, cont: inteiro

Inicio

cont \leftarrow 0

Repita

Escreva ("Digite um número:")

Leia (num)

Se (num \geq 100) e (num \leq 300) entao

cont \leftarrow cont + 1

fimse

Ate (num = 0)

Escreva ("A quantidade de números entre 100 e 300 é:", cont)

Fimalgoritmo

ESCREVA UM ALGORITMO QUE LEIA O NÚMERO DE VEZES QUE SE DESEJA IMPRIMIR A PALAVRA "ALGORITMOS" E IMPRIMIR.

Objetivo do algoritmo: ler um número de vezes que se deseja imprimir a palavra "ALGORITMOS".

Entrada: ler um número inteiro.

Processamento: não há.

Saída: imprimir a palavra "Algoritmos" o número de vezes informado.

ESTRUTURAS DE REPETIÇÃO - EXEMPLOS

Algoritmo "palavra"

Var

num, i: inteiro

Inicio

Escreva ("Informe o número de vezes que deseja imprimir:")

Leia (num)

Para i de 1 ate num passo 1 faca

Escreva ("ALGORITMOS")

fimpara

Fimalgoritmo

CONSTRUA UM ALGORITMO QUE ENTRE COM NÚMEROS INTEIROS ENQUANTO FOREM POSITIVOS E IMPRIMA QUANTOS NÚMEROS FORAM DIGITADOS.

Objetivo do algoritmo: ler vários números enquanto forem positivos e contar quantos foram digitados.

Entrada: ler números enquanto forem positivos.

Processamento: contar a quantidade de números digitada.

Saída: imprimir a quantidade de números positivos digitadas.

ESTRUTURAS DE REPETIÇÃO - EXEMPLOS

Algoritmo "conta"

Var

num, qtdade: inteiro

Inicio

qtdade \leftarrow 0

Escreva ("Informe um número:")

Leia (num)

Enquanto (num > 0) faça

qtdade \leftarrow qtdade + 1

Escreva ("Informe um número:")

Leia (num)

Fimenquanto

Escreva ("O total de números positivos informado é:", qtdade)

Fimalgoritmo

**FAÇA UM ALGORITMO QUE CALCULE O FATORIAL DE UM NÚMERO
DIGITADO PELO USUÁRIO.**

Ex.:

$$1! = 1$$

$$2! = 2 \times 1 = 2$$

$$3! = 3 \times 2 \times 1 = 6$$

$$4! = 4 \times 3 \times 2 \times 1 = 24$$

$$5! = 5 \times 4 \times 3 \times 2 \times 1 = 120$$

ESTRUTURAS DE REPETIÇÃO - EXEMPLOS

Algoritmo "fatorial"

Var

num, fat, cont: inteiro

Inicio

Escreva ("Digite o número que deseja calcular o fatorial:")

Leia (num)

fat \leftarrow 1

Para cont de num ate 1 passo -1 faca

fat \leftarrow fat*cont

fimpara

Escreva ("O fatorial é: ", fat)

Fimalgoritmo



DIVIDIR PARA CONQUISTAR!
CÓDIGO MODULADO

PROCEDIMENTOS: NÃO RETORNAM VALOR

```
procedimento <nome do procedimento>
```

```
var
```

```
    <variáveis>
```

```
inicio
```

```
    <instruções>
```

```
fimprocedimento
```

PROCEDIMENTOS: NÃO RETORNAM VALOR

```
algoritmo "procedimento para soma"
```

```
var
```

```
    n, m, res: inteiro
```

```
procedimento soma
```

```
    var
```

```
        aux: inteiro
```

```
    inicio
```

```
        // n, m e res são variáveis globais
```

```
        aux <- n + m
```

```
        res <- aux
```

```
fimprocedimento
```

```
inicio
```

```
    n <- 4
```

```
    m <- -9
```

```
    soma
```

```
    escreva(res)
```

```
fimalgoritmo
```

```
funcao <nome-de-função> [(<sequência-de-declarações-de-  
parâmetros>)] : <tipo-de-dado>
```

```
    inicio
```

```
    // Seção de Comandos
```

```
fimfuncao
```

FUNÇÕES: RETORNAM VALOR!

```
algoritmo "procedimento para soma"

var
    n, m, res: inteiro

funcao soma (n, m: inteiro): inteiro

    inicio

        retorne n + m

fimfuncao

inicio
    n <- 4
    m <- -9
    res <- soma (n,m)
    escreva(res)

fimalgoritmo
```

LER, COPIAR, GUARDAR
ALTERAR, INSERIR DADOS


```
algoritmo "lendo do arquivo"
```

```
arquivo "teste.txt"
```

```
var
```

```
    x,y: inteiro
```

```
inicio
```

```
    para x de 1 ate 5 faca
```

```
        leia (y)
```

```
    fimpara
```

```
fimalgoritmo
```

CONSTRUA UM ALGORITMO QUE ENTRE COM OS COEFICIENTES DE UMA EQUAÇÃO DO SEGUNDO GRAU E IMPRIMA OS VALORES DAS RAÍZES.

QUANDO NÃO FOR POSSÍVEL ENCONTRAR RAÍZES REAIS, O ALGORITMO DEVE AVISAR QUE NÃO HÁ SOLUÇÃO.

ALÉM DISSO, CASO NÃO SEJA DIGITADO UM VALOR VÁLIDO PARA 'a', O ALGORITMO DEVE AVISAR QUE NÃO SE TRATA DE UMA EQ. DO SEGUNDO GRAU.

A FORMA PADRÃO DE UMA EQUAÇÃO DO SEGUNDO GRAU É:

$$a.x^2 + b.x + c = 0$$

DICA: "BHASKARA"

OBSERVAÇÃO: AGORA QUE JÁ TEMOS UMA CALCULADORA DE RAÍZES PARA EQUAÇÃO DO SEGUNDO GRAU, ENCONTRE AS RAÍZES DA SEGUINTE EQUAÇÃO:

$$x^2 - 4x - 5 = 0$$

BOM TRABALHO!

SUPER EXERCÍCIO

Algoritmo "raiz"

Var

a, b, c, delta, x1, x2: real

Início

Escreva ("Digite o valor de a:")

Leia (a)

Escreva ("Digite o valor de b:")

Leia (b)

Escreva ("Digite o valor de c:")

Leia (c)

Se (a = 0) então

Escreva ("Não é uma equação do segundo grau")

Senão

delta <- $\text{sqr}(b^2 - 4 * a * c)$

Se (delta < 0) então

Escreva ("Não existe raiz real")

Senão

Se (delta = 0) então

Escreva ("Existe raiz real")

x1 <- $(-b) / (2 * a)$

Escreva ("A raiz é: ", x1)

Senão

Se (delta > 0) então

Escreva ("Existem duas raízes reais")

x1 <- $(-b + \text{sqrt}(\text{delta})) / (2 * a)$

Escreva ("A raiz x1 é: ", x1)

x2 <- $(-b - \text{sqrt}(\text{delta})) / (2 * a)$

Escreva ("A raiz x2 é: ", x2)

Fim_se

Fim_se

Fim_se

Fim_se

Fim