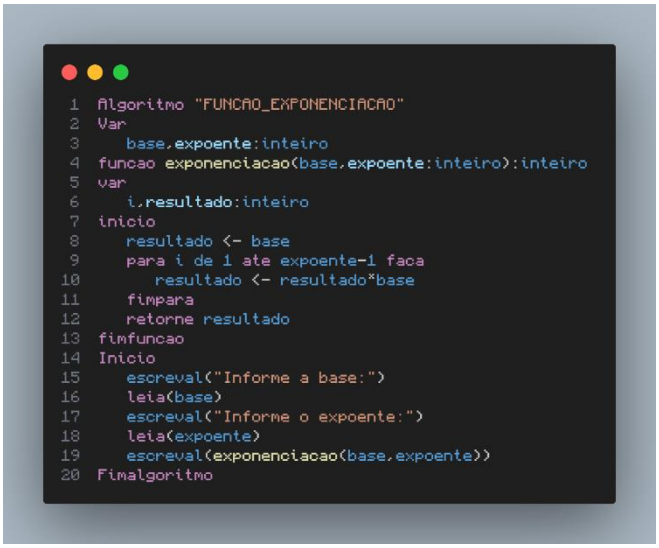
MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DO PIAUÍ
CAMPUS SENADOR HELVÍDIO NUNES DE BARROS – CSHNB
CURSO DE BACHARELADO EM SISTEMAS DE INFORMAÇÃO
DISCIPLINA DE ALGORITMOS E PROGRAMAÇÃO I
PROFESSOR ALAN RAFAEL FERREIRA DOS SANTOS

ATIVIDADE DE FIXAÇÃO VI

1. Escreva um programa em pseudocódigo que solicite ao usuário dois números: a base e o expoente. O programa deve chamar uma função para calcular a potência, utilizando esses dois valores. A função deve retornar o resultado da base elevada ao expoente e o programa principal deve exibir o resultado.



```
Algoritmo "FUNCAO_EXPONENCIACAO"
Var
   base,expoente:inteiro
funcao exponenciacao(base,expoente:inteiro):inteiro
var
   i,resultado:inteiro
inicio
   resultado <- base
   para i de 1 ate expoente-1 faca
      resultado <- resultado*base
   fimpara
   retorne resultado
fimfuncao
Inicio
   escreval("Informe a base:")
   leia(base)
   escreval("Informe o expoente:")
   leia(expoente)
   escreval(exponenciacao(base,expoente))
Fimalgoritmo
```

2. Escreva um algoritmo em pseudocódigo que solicite ao usuário o valor do raio de um círculo e utilize uma função para calcular a área desse círculo. A função deve retornar a área calculada e o programa principal deve exibir o resultado.

```
Algoritmo "FUNCAO_AREA-CIRCULO"
Var

   raio:real
funcao areacirculo(raio:real):real
var
   area:real
inicio
   area <- 3.14*(raio*raio)
   retorne area
fimfuncao


Inicio
     escreval("INFORME O RAIO")
     leia(raio)

     escreval(areacirculo(raio))

Fimalgoritmo
```

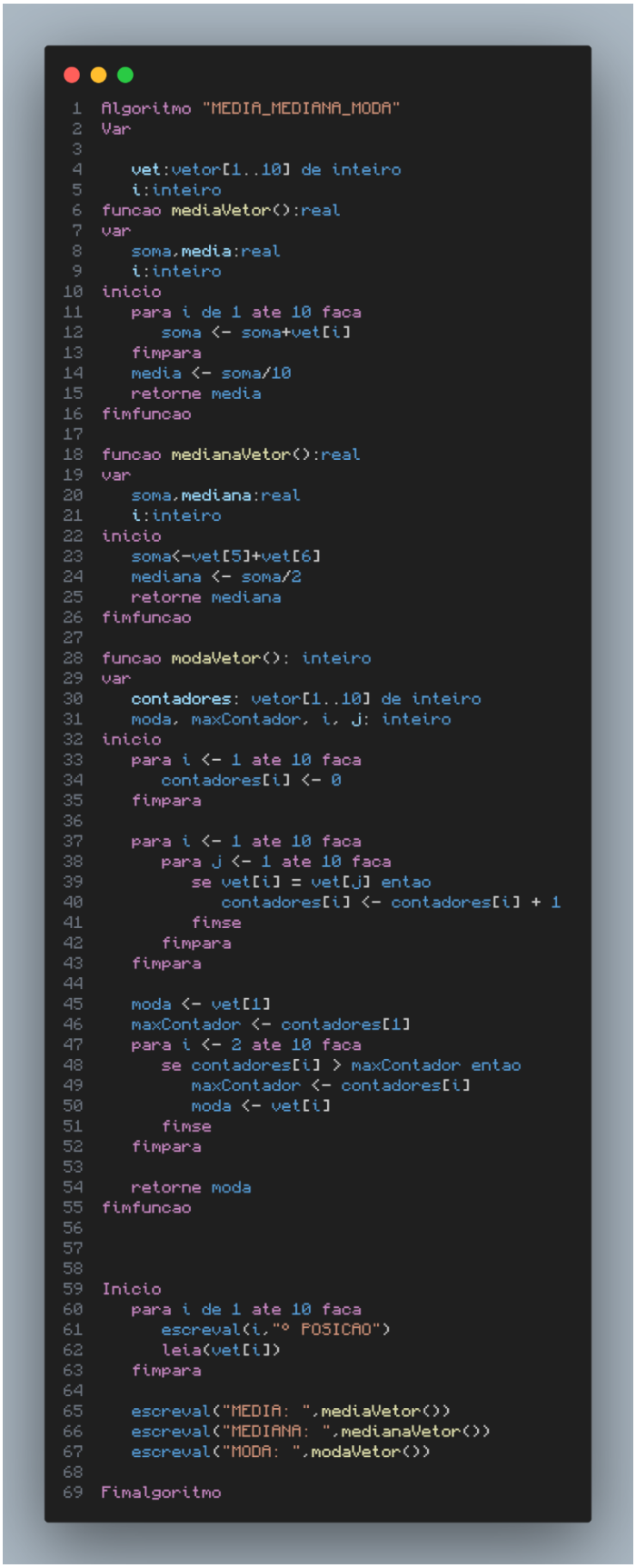
3. Implemente um algoritmo que contenha uma função que verifique se um número é primo. A função deve receber um número inteiro como parâmetro e retornar *true* se o número for primo ou *false* caso contrário.

```
Algoritmo "EHPRIMO"
Var
   num: inteiro

funcao ehPrimo(num: inteiro): logico
var
   primo:logico
   i, divisores:inteiro
inicio
   para i de 1 ate num faca
      se (num % i = 0) entao
         divisores <- divisores + 1
      fimse
   fimpara
   se (divisores = 2) entao
      primo <- verdadeiro
   senao
      primo <- falso
   fimse
   retorne primo
fimfuncao

Inicio
   escreval("Informe o número:")
   leia(num)
   escreval(ehPrimo(num))
Fimalgoritmo
```

4. Desenvolva um algoritmo que faça a chamada de uma função para calcular a média de um vetor de 10 números fornecidos pelo usuário. O programa deve seguir o seguinte fluxo:
   - O algoritmo principal deve solicitar que o usuário insira 10 números, que serão armazenados em um vetor.
   - Em seguida, o algoritmo deve chamar uma função que recebe o vetor como parâmetro e calcula a média dos números presentes nele.
   - O valor da média deve ser retornado pela função e exibido no programa principal.

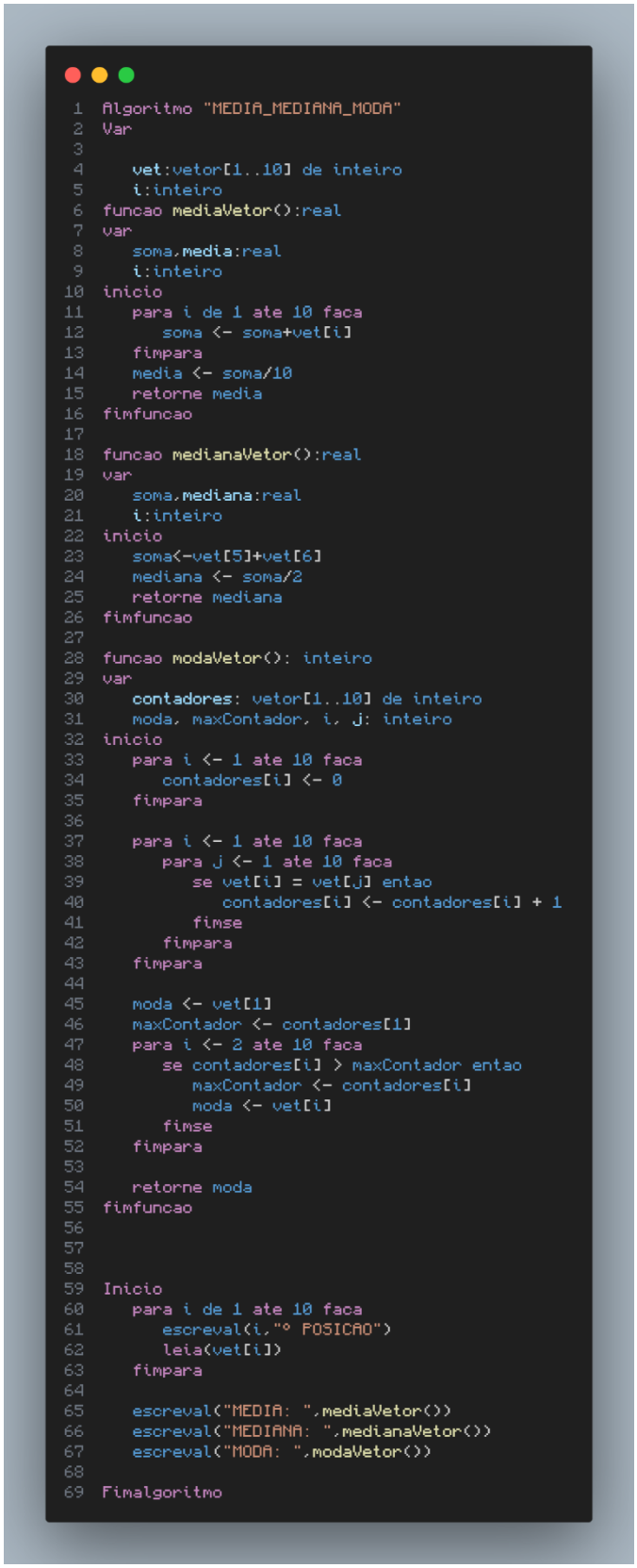
```
Algoritmo "MEDIA_VETOR"
Var
   numeros:vetor[1..10] de inteiro
   i:inteiro
funcao mediaVetor():real
var
   soma,media:real
   i:inteiro
inicio
   para i de 1 ate 10 faca
      soma <- soma+numeros[i]
   fimpara
   media <- soma/10
   retorne media
fimfuncao

Inicio

   para i de 1 ate 10 faca
      escreval(i,"º POSIÇÃO: ")
      leia(numeros[i])
   fimpara

   escreval(mediaVetor())

Fimalgoritmo
```

5. Desenvolva um algoritmo em pseudocódigo que solicite ao usuário um vetor de 10 números e faça a chamada de funções separadas para calcular a média, a mediana e a moda desses números. O algoritmo deve retornar e exibir os três valores.



```
Algoritmo "MEDIA_MEDIANA_MODA"
Var

   vet:vetor[1..10] de inteiro
   i:inteiro
funcao mediaVetor():real
var
   soma,media:real
   i:inteiro
inicio
   para i de 1 ate 10 faca
      soma <- soma+vet[i]
   fimpara
   media <- soma/10
   retorne media
fimfuncao

funcao medianaVetor():real
var
   soma,mediana:real
   i:inteiro
inicio
   soma<-vet[5]+vet[6]
   mediana <- soma/2
   retorne mediana
fimfuncao

funcao modaVetor(): inteiro
var
   contadores: vetor[1..10] de inteiro
   moda, maxContador, i, j: inteiro
inicio
   para i <- 1 ate 10 faca
      contadores[i] <- 0
   fimpara

   para i <- 1 ate 10 faca
      para j <- 1 ate 10 faca
         se vet[i] = vet[j] entao
            contadores[i] <- contadores[i] + 1
         fimse
      fimpara
   fimpara

   moda <- vet[1]
   maxContador <- contadores[1]
   para i <- 2 ate 10 faca
      se contadores[i] > maxContador entao
         maxContador <- contadores[i]
         moda <- vet[i]
      fimse
   fimpara

   retorne moda
fimfuncao



Inicio
   para i de 1 ate 10 faca
      escreval(i,"° POSICAO")
      leia(vet[i])
   fimpara

   escreval("MEDIA: ",mediaVetor())
   escreval("MEDIANA: ",medianaVetor())
   escreval("MODA: ",modaVetor())

Fimalgoritmo
```

6. Crie uma função chamada ocorre que receba um vetor (lista) de números como entrada. A função deve verificar se há algum número repetido no vetor. Se houver repetições, a função deve retornar o número que se repete e todas as posições em que ele ocorre no vetor.



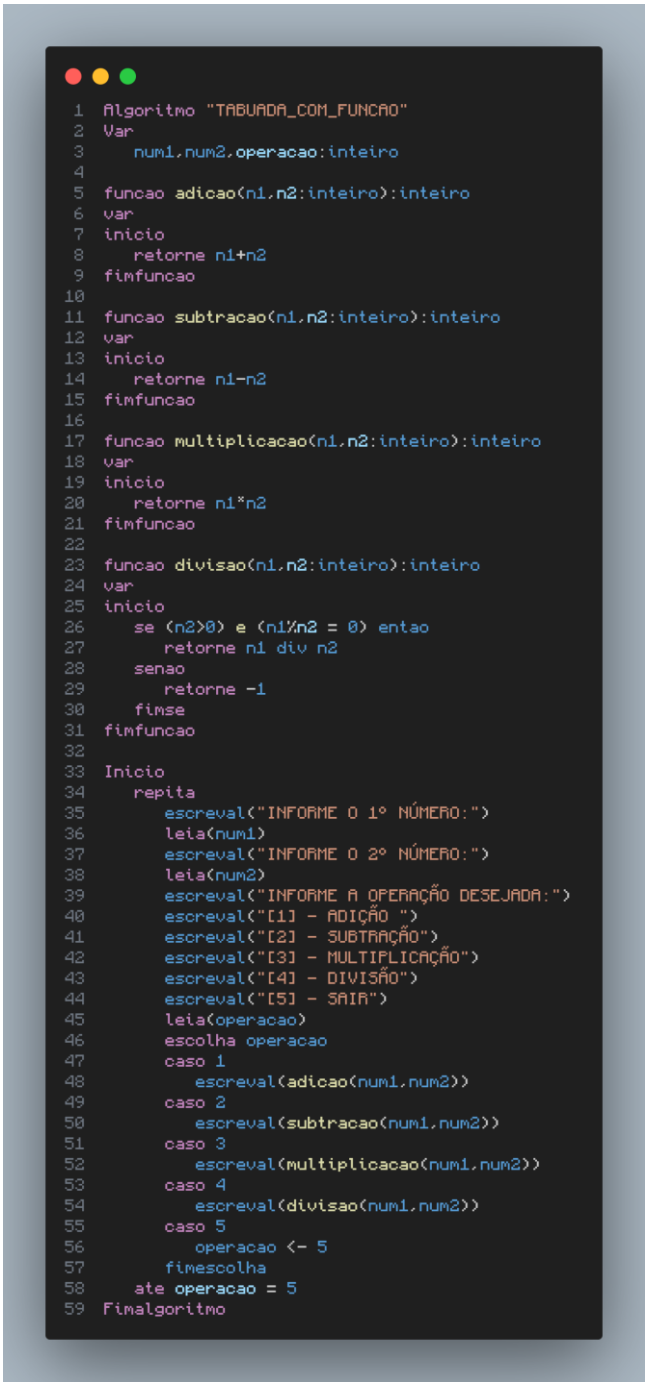
```
Algoritmo "MEDIA_MEDIANA_MODA"
Var

   vet:vetor[1..10] de inteiro
   i:inteiro
funcao mediaVetor():real
var
   soma,media:real
   i:inteiro
inicio
   para i de 1 ate 10 faca
      soma <- soma+vet[i]
   fimpara
   media <- soma/10
   retorne media
fimfuncao

funcao medianaVetor():real
var
   soma,mediana:real
   i:inteiro
inicio
   soma<-vet[5]+vet[6]
   mediana <- soma/2
   retorne mediana
fimfuncao

funcao modaVetor(): inteiro
var
   contadores: vetor[1..10] de inteiro
   moda, maxContador, i, j: inteiro
inicio
   para i <- 1 ate 10 faca
      contadores[i] <- 0
   fimpara

   para i <- 1 ate 10 faca
      para j <- 1 ate 10 faca
         se vet[i] = vet[j] entao
            contadores[i] <- contadores[i] + 1
         fimse
      fimpara
   fimpara

   moda <- vet[1]
   maxContador <- contadores[1]
   para i <- 2 ate 10 faca
      se contadores[i] > maxContador entao
         maxContador <- contadores[i]
         moda <- vet[i]
      fimse
   fimpara

   retorne moda
fimfuncao



Inicio
   para i de 1 ate 10 faca
      escreval(i,"° POSICAO")
      leia(vet[i])
   fimpara

   escreval("MEDIA: ",mediaVetor())
   escreval("MEDIANA: ",medianaVetor())
   escreval("MODA: ",modaVetor())

Fimalgoritmo
```

7. Implemente um algoritmo em pseudocódigo que utiliza um procedimento para imprimir a tabuada completa de um número fornecido pelo usuário. O procedimento deve realizar as quatro operações matemáticas básicas: adição, subtração, multiplicação e divisão. Para cada operação, o programa deve mostrar os resultados da operação do número fornecido com os números de 1 a 10.

```
Algoritmo "PROCEDIMENTO_TABUADA"
Var
   num:inteiro
procedimento tabuada(num:inteiro):inteiro
var
   i:inteiro
inicio
   escreval("ADIÇÃO:")
   para i de 1 ate 10 faca
      escreval(num," +",i," =",num+i)
   fimpara

   escreval("SUBTRAÇÃO:")
   para i de 1 ate 10 faca
      escreval(num," -",i," =",num-i)
   fimpara

   escreval("MULTIPLICAÇÃO:")
   para i de 1 ate 10 faca
      escreval(num," *",i," =",num*i)
   fimpara

   escreval("DIVISÃO:")
   para i de 1 ate num*10 faca
      se i%num = 0 entao
         escreval(i," /",num," =",i div num)
      fimse
   fimpara
fimprocedimento
Inicio
   escreval("INFORME O NUMERO:")
   leia(num)

   tabuada(num)

Fimalgoritmo
```

8. Implemente um algoritmo em pseudocódigo que utilize funções e estruturas condicionais para realizar operações aritméticas básicas (adição, subtração, multiplicação e divisão) entre dois números fornecidos pelo usuário. O algoritmo deve seguir a seguinte lógica:

    - O usuário deve fornecer dois números e a operação desejada.
    - O algoritmo deve verificar qual operação foi solicitada pelo usuário.
    - Para cada operação (adição, subtração, multiplicação e divisão), crie uma função específica.
    - O algoritmo deve chamar e executar apenas a função correspondente à operação desejada, após a verificação condicional.
    - O algoritmo deve tratar possíveis erros, como divisão por zero.



```
Algoritmo "TABUADA_COM_FUNCAO"
Var
   num1,num2,operacao:inteiro

funcao adicao(n1,n2:inteiro):inteiro
var
inicio
   retorne n1+n2
fimfuncao

funcao subtracao(n1,n2:inteiro):inteiro
var
inicio
   retorne n1-n2
fimfuncao

funcao multiplicacao(n1,n2:inteiro):inteiro
var
inicio
   retorne n1*n2
fimfuncao

funcao divisao(n1,n2:inteiro):inteiro
var
inicio
   se (n2>0) e (n1%n2 = 0) entao
      retorne n1 div n2
   senao
      retorne -1
   fimse
fimfuncao

Inicio
   repita
      escreval("INFORME O 1º NÚMERO:")
      leia(num1)
      escreval("INFORME O 2º NÚMERO:")
      leia(num2)
      escreval("INFORME A OPERAÇÃO DESEJADA:")
      escreval("[1] - ADIÇÃO ")
      escreval("[2] - SUBTRAÇÃO")
      escreval("[3] - MULTIPLICAÇÃO")
      escreval("[4] - DIVISÃO")
      escreval("[5] - SAIR")
      leia(operacao)
      escolha operacao
      caso 1
         escreval(adicao(num1,num2))
      caso 2
         escreval(subtracao(num1,num2))
      caso 3
         escreval(multiplicacao(num1,num2))
      caso 4
         escreval(divisao(num1,num2))
      caso 5
         operacao <- 5
      fimescolha
   ate operacao = 5
Fimalgoritmo
```

9. Escreva um algoritmo que solicite ao usuário um vetor de 5 números e faça a chamada de funções para calcular a variância e o desvio padrão dos números. O desvio padrão é uma medida da dispersão dos valores em relação à média, e a variância é o quadrado do desvio padrão.

```
Algoritmo "VARIANCIA_E_DESVIO-PADRAO"
Var
   vet:vetor[1..5] de inteiro
   i:inteiro

funcao desvioPadrao():real
var
   res,soma,media,desvio:real
   i:inteiro
inicio
   para i de 1 ate 5 faca
      soma<-soma+vet[i]
   fimpara

   media <- soma/5
   soma<-0
   para i de 1 ate 5 faca
      soma<- soma+abs((vet[i] - media)*(vet[i] - media))
   fimpara
   media <- soma/5

   res <- raizq(media)

   retorne res
fimfuncao

funcao variancia(num:real):real
var
inicio
   retorne num^2
fimfuncao

Inicio

   para i de 1 ate 5 faca
      escreval(i,"º POSIÇÃO:")
      leia(vet[i])
   fimpara

   escreval(desvioPadrao())
   escreval(variancia(desvioPadrao()))

Fimalgoritmo
```

10. Desenvolva um algoritmo em pseudocódigo que solicita ao usuário uma matriz 3x3 de números inteiros e utilize funções para realizar as seguintes operações:

    - Função de Soma de Linhas: Implemente uma função que receba a matriz como parâmetro e retorne a soma dos elementos de cada linha.
    - Função de Soma de Colunas: Crie uma função que receba a matriz e retorne um a soma dos elementos de cada coluna.
    - Função para Calcular o Traço: Escreva uma função que calcule o traço da matriz (a soma dos elementos da diagonal principal).

```
algoritmo "OPERACOES_MATRIZ_3-3"
Var

   mat:vetor[1..3,1..3] de inteiro
   i,j:inteiro

funcao somaTraco():inteiro
var
   soma:inteiro
inicio
   para i de 1 ate 3 faca
      para j de 1 ate 3 faca
         se i = j entao
            soma <- soma+mat[i,j]
         fimse
      fimpara
   fimpara
   retorne soma
fimfuncao

funcao somaLinhas():inteiro
var
   linha1,linha2,linha3:inteiro
   frase:caractere
inicio
   para i de 1 ate 3 faca
      para j de 1 ate 3 faca
         se i = 1 entao
            linha1 <- linha1+mat[i,j]
         senao
            se i = 2 entao
               linha2 <- linha2+mat[i,j]
            senao
               linha3 <- linha3+mat[i,j]
            fimse
         fimse
      fimpara
   fimpara
   escreval("LINHA 1:",linha1," LINHA 2:",linha2," LINHA 3:",linha3,".")

   retorne 0
fimfuncao

funcao somaColunas():inteiro
var
   coluna1,coluna2,coluna3:inteiro
   frase:caractere
inicio
   para i de 1 ate 3 faca
      para j de 1 ate 3 faca
         se j = 1 entao
            coluna1 <- coluna1+mat[i,j]
         senao
            se j = 2 entao
               coluna2 <- coluna2+mat[i,j]
            senao
               coluna3 <- coluna3+mat[i,j]
            fimse
         fimse
      fimpara
   fimpara
   escreval("COLUNA 1:",coluna1," COLUNA 2:",coluna2," COLUNA 3:",coluna3,".")

   retorne 0
fimfuncao

Inicio

   para i de 1 ate 3 faca
      para j de 1 ate 3 faca
         escreval("POSIÇÃO [",i,j," ]")
         leia(mat[i,j])
      fimpara
   fimpara

   escreval(somaTraco())
   escreval("")
   escreval(somaLinhas())
   escreval("")
   escreval(somaColunas())


Fimalgoritmo
```