

**algoritmo "exercício 22 – resposta incorreta"**

**var A, i: inteiro**

**eh\_primo: logico**

**inicio**

**repita**

**escreva ("Entre com um valor inteiro positivo: ")**

**leia (A)**

**ate (A>0)**

**eh\_primo <- verdadeiro**

**para i de 1+1 ate ~~A<sup>0.5</sup>~~ faça**

**se (A%i=0) entao**

**eh\_primo <- falso**

**interrompa**

**fimse**

**fimpara**

**se (eh\_primo e A<>1) entao**

**escreva ("O número ",A," é primo")**

**senao**

**escreva ("O número ",A," não é primo")**

**fimse**

**180 fimalgoritmo**

**algoritmo "exercício 22 – resposta incorreta "**

**var A, i: inteiro**

**eh\_primo: logico**

**inicio**

**repita**

**escreva ("Entre com um valor inteiro positivo: ")**

**leia (A)**

**ate (A>0)**

**eh\_primo <- verdadeiro**

**para i de 1+1 ate ~~(inteiro)A^0.5~~ faça**

**se (A%i=0) entao**

**eh\_primo <- falso**

**interrompa**

**fimse**

**fimpara**

**se (eh\_primo e A<>1) entao**

**escreva ("O número ",A," é primo")**

**senao**

**escreva ("O número ",A," não é primo")**

**fimse**

**fimalgoritmo**

**algoritmo "exercício 22 – resposta incorreta "**

**var A, i: inteiro**

**eh\_primo: logico**

**inicio**

**repita**

**escreva ("Entre com um valor inteiro positivo: ")**

**leia (A)**

**ate (A>0)**

**eh\_primo <- verdadeiro**

**para i de 1+1 ate ~~(inteiro) piso(A^0.5)~~ faça**

**se (A%i=0) entao**

**eh\_primo <- falso**

**interrompa**

**fimse**

**fimpara**

**se (eh\_primo e A<>1) entao**

**escreva ("O número ",A," é primo")**

**senao**

**escreva ("O número ",A," não é primo")**

**fimse**

**fimalgoritmo**

```

algoritmo "exercício 22 – resposta 1"
var A, i, aux: inteiro
       eh_primo: logico
inicio
  repita
    escreva ("Entre com um valor inteiro positivo: ")
    leia (A)
  ate (A>0)
  eh_primo <- verdadeiro
  aux<-1
  enquanto (aux*aux<=a)faca
    aux<-aux+1
  fimenquanto
  para i de 1+1 ate aux-1 faca
    se (A%i=0) entao
      eh_primo <- falso
      interrompa
    fimse
  fimpara
  se (eh_primo e A<>1) entao
    escreva ("O número ",A," é primo")
  senao
    escreva ("O número ",A," não é primo")
  fimse
fimalgoritmo

```

```

algoritmo "exercício 22 - resposta 2"
var A, i, aux1: inteiro
     eh_primo: logico
     aux2:real
inicio
  repita
    escreva ("Entre com um valor inteiro positivo: ")
    leia (A)
  ate (A>0)
  eh_primo <- verdadeiro
  aux1<-1
  aux2<-a^0.5
  enquanto (aux1<=aux2) faça
    aux1<-aux1+1
  fimenquanto
  para i de 1+1 ate aux1-1 faça
    se (A%i=0) entao
      eh_primo <- falso
      interrompa
    fimse
  fimpara
  se (eh_primo e A<>1) entao
    escreva ("O número ",A," é primo")
  senao
    escreva ("O número ",A," não é primo")
  fimse
fimalgoritmo

```

**algoritmo "exercício 22 – resposta 3"**

**var A, i: inteiro**

**eh\_primo: logico**

**inicio**

**repita**

**escreva ("Entre com um valor inteiro positivo: ")**

**leia (A)**

**ate (A>0)**

**eh\_primo <- verdadeiro**

**i <- 1+1**

**enquanto (i<=A^0.5) faça**

**se (A%i=0) entao**

**eh\_primo <- falso**

**interrompa**

**fimse**

**i<-i+1**

**fimenquanto**

**se (eh\_primo e A<>1) entao**

**escreva ("O número ",A," é primo")**

**senao**

**escreva ("O número ",A," não é primo")**

**fimse**

**fimalgoritmo**

## Estruturas de Controle de Fluxo

Para finalizarmos nosso estudo das estruturas de controle de fluxo, vamos tratar do teorema que as originou.

O teorema da programação estruturada, conhecido como **Teorema de Böhm-Jacopini**. Enunciado em 1966 por Corrado Böhm e Giuseppe Jacopini sendo resultado da teoria das linguagens de programação. O qual define que cada rotina computável pode ser descrita por um algoritmo que combine as instruções utilizando apenas três maneiras específicas:

1. Executar uma instrução, depois outra instrução (sequência);
2. Executar uma ou duas sequências de instruções de acordo com um valor booleano (condição);
3. Executar uma sequências de instruções até que um valor booleano seja verdadeiro (iteração).

## Estruturas de Controle de Fluxo

### 3. Laços de repetição (continuação)

#### Exercício 23:

Com base nos conceitos estudados solucione o problema de receber um número natural e retornar o seu fatorial. Gerar três soluções, utilizando em cada uma, uma das estruturas de repetição vistas. As entradas devem ser validadas. **Obs.:** Os algoritmos gerados devem ser representados através de pseudocódigos.



```
algoritmo " exercício 23 - enquanto"  
var num, fat: inteiro  
Inicio  
  num <- -1  
  enquanto (num < 0) faça  
    escreva ("Digite um número natural: ")  
    leia (num)  
  fimenquanto  
  se (num=0 ou num=1) entao  
    fat <- 1  
  senao  
    fat <- num  
    num <- num - 1  
    enquanto (num>1) faça  
      fat <- fat * num  
      num <- num - 1  
    fimenquanto  
  fimse  
  escreva ("O fatorial é ", fat)  
fimalgoritmo
```

algoritmo " exercício 23 - repita"

var num, fat: inteiro

Inicio

  repita

    escreva ("Digite um número natural: ")

    leia (num)

  ate (num>=0)

  se (num=0 ou num=1) entao

    fat <- 1

  senao

    fat <- num

    repita

      num <- num - 1

      fat <- fat \* num

    ate (num=1)

  fimse

  escreva ("O fatorial é ", fat)

189 fimalgoritmo

**algoritmo "exercício 23 - para"**

**var**

**i,num,fat: inteiro**

**inicio**

**para i de 1 ate 1 faça**

**escreva ("Digite um número natural: ")**

**leia (num)**

**se (num<0) entao**

~~**i <- i-1**~~

**escreval ("Não foi fornecido um valor válido!")**

**fimse**

**fimpara**

**fat <- 1**

**para i de 2 ate num faça**

**fat <- fat \* i**

**fimpara**

**escreva ("O fatorial de ", num, " é: ", fat)**

**fimalgoritmo**

```
algoritmo "exercício 23 - para"  
var  
    i,num,fat: inteiro  
inicio  
    repita  
        escreva ("Digite um número natural: ")  
        leia (num)  
    ate (num>=0)  
    fat <- 1  
    para i de 2 ate num faça  
        fat <- fat * i  
    fimpara  
    escreva ("O fatorial de ", num, " é: ", fat)  
fimalgoritmo
```

algoritmo "exercício 23 – para – resposta alternativa"

var

num,aux: inteiro

inicio

repita

escreva ("Digite um número natural: ")

leia (num)

ate (num>=0)

se (num=0) entao

escreva ("O fatorial é: 1")

senao

para aux de num-1 ate 2 passo -1 faca

num <- num \* aux

fimpara

escreva ("O fatorial é: ", num)

fimse

192 fimalgoritmo

## Estruturas de Controle de Fluxo

### 3. Laços de repetição (continuação)

Exercício 24:

Escreva um algoritmo para calcular o valor da série, para 5 termos.

$$S = -\frac{1}{2!} + \frac{2}{4!} - \frac{3}{6!} + \dots$$

**algoritmo "exercício 24a"**

**var**

**ind, i, fat: inteiro**

**s: real**

**inicio**

**s <- 0**

**para ind de 1 ate 5 faca**

**fat <- 1**

**para i de 2 ate ind\*2 faca**

**fat <- fat \* i**

**fimpara**

**se (ind%2=0) entao**

**s <- s + ind / fat**

**senao**

**s <- s - ind / fat**

**fimse**

**fimpara**

**escreval ("O valor do somatório é: ", s)**

**fimalgoritmo**

**algoritmo "exercício 24b"**

**var**

**ind, i, fat: inteiro**

**s: real**

**inicio**

**s <- 0**

**para ind de 1 ate 5 faca**

**fat <- 1**

**para i de 2 ate ind\*2 faca**

**fat <- fat \* i**

**fimpara**

**s <- s + -1<sup>ind</sup> \* ind / fat**

**fimpara**

**escreval ("O valor do somatório é: ", s)**

**fimalgoritmo**



## Estruturas de Controle de Fluxo

### 3. Laços de repetição (continuação)

Exercício 25:

Escreva um algoritmo para calcular o valor da série, para N termos. Onde o valor de N será fornecido pelo usuário.

$$S = -\frac{1}{2!} + \frac{2}{4!} - \frac{3}{6!} + \dots$$