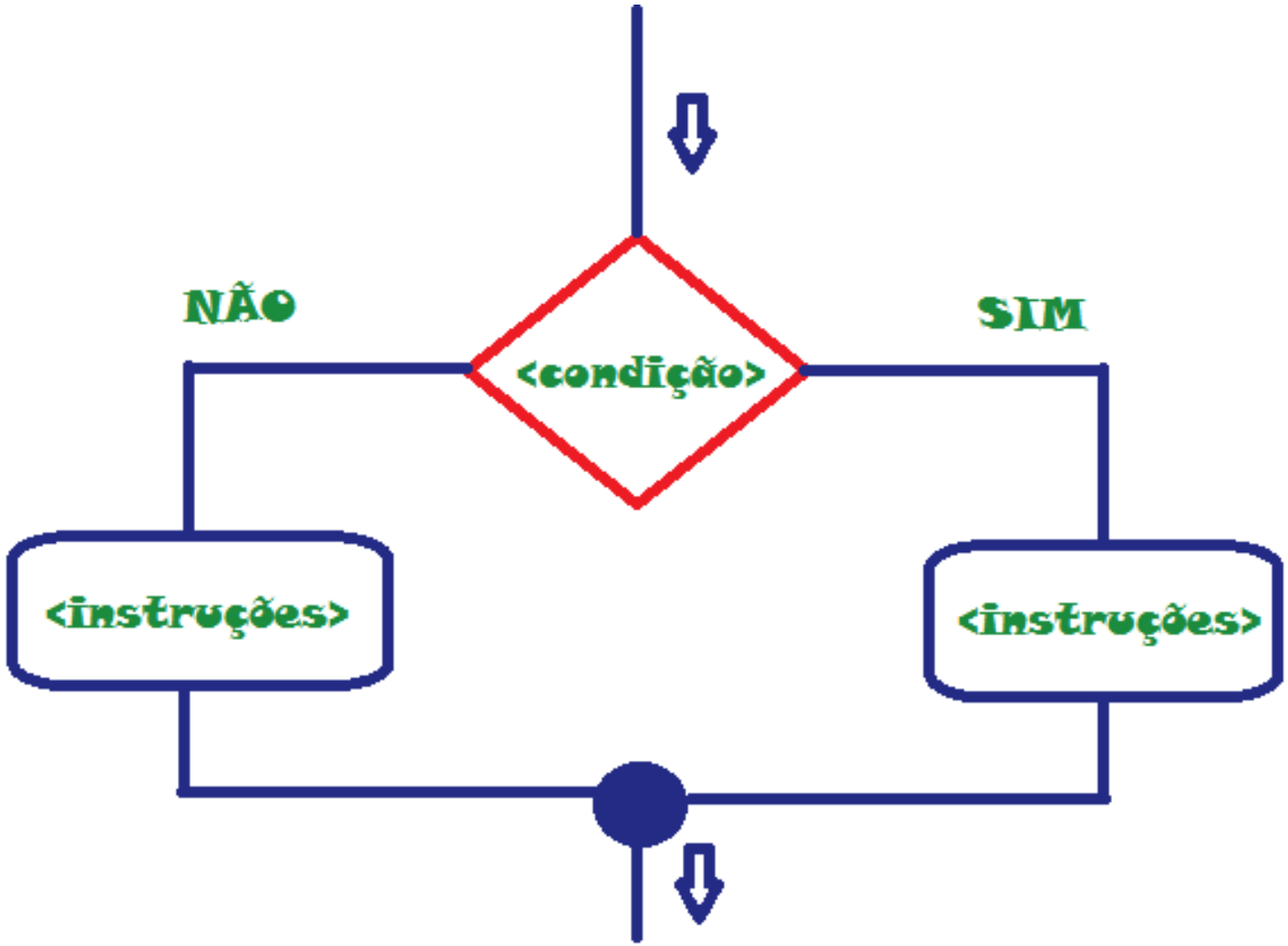
Introdução ao Mundo da Computação e Lógica de Programação

Lógica de Programação

Introdução ao Mundo da Computação e Lógica de Programação

Estrutura de Decisão

Estrutura de Decisão



Estrutura de Decisão

A maioria dos algoritmos precisam tomar decisões ao longo de sua As principais estruturas de decisão são:

SE...ENTAO

SE...ENTAO...SENAO

SE...ENTAO...SENAO (encadeado)

CASO

Operadores Relacionais

Serão utilizados nas estruturas de decisão fazendo a comparação entre valores/informações em uma condição.

Símbolo	Função
<>	Diferente
>	Maior
<	Menor
<=	Menor igual
>=	Maior igual
==	Igual
Mod ou %	Resto da divisão entre números inteiros
x = (a > b) ? a:b	Ternário

Operadores Lógicos

Serão utilizados nas estruturas de decisão junto com os operadores relacionais para tratar mais de uma condição.

Símbolo	Função
&&	And (e)
||	Or (ou)

Operadores Lógicos

Tabela verdade - AND		
Condição 1	Condição 2	Resultado
FALSO	FALSO	FALSO
VERDADEIRO	FALSO	FALSO
FALSO	VERDADEIRO	FALSO
VERDADEIRO	VERDADEIRO	VERDADEIRO

Operadores Lógicos

Tabela verdade - OR		
Condição 1	Condição 2	Resultado
FALSO	FALSO	FALSO
VERDADEIRO	FALSO	VERDADEIRO
FALSO	VERDADEIRO	VERDADEIRO
VERDADEIRO	VERDADEIRO	VERDADEIRO

Estrutura de Decisão

SE ENTÃO

SE <condição lógica> **ENTÃO**
    <ações>
FIMSE

Significado: Se a <operação lógica> resultar em verdadeiro, então executar as <ações>.
Senão, simplesmente ignorar as <ações> e seguir para a próxima instrução no algoritmo.

Porque usar? Usada para decidir se um conjunto de ações opcionais deve ser executado ou não, dependendo do valor de algum dado ou de algum resultado que já tenha sido calculado no algoritmo. O valor do dado ou do resultado anterior será testado na operação lógica.

Estrutura de Decisão

Exemplo:

Faça um algoritmo que leia um valor qualquer, e se o valor for negativo mostre uma mensagem dizendo o valor digitado é negativo.

```
algoritmo "numero negativo"
var
n : inteiro
inicio
leia(n)
se (n<0) entao
     escreval ("o número digitado é negativo")
fimse
fimalgoritmo
```

Estrutura de Decisão

SE ENTÃO SENÃO

A maioria dos algoritmos precisam tomar decisões ao longo de sua execução. Para isso existem as estruturas de decisão, e a mais utilizada é a estrutura **SE-ENTÃO-SENÃO**. O funcionamento é simples: com base no resultado de uma expressão booleana (**VERDADEIRO** ou **FALSO**), o fluxo do algoritmo segue para um bloco de instruções ou não. Observe o esquema da estrutura **SE-ENTÃO-SENÃO:**

Sintaxe:

SE <expressão booleana> **ENTÃO**
<instruções a serem executadas caso a expressão booleana resulte em **VERDADEIRO**>
SENÃO
<instruções a serem executadas caso a expressão booleana resulte em **FALSO**>
FIMSE

Estrutura de Decisão

Exemplo:

Faça um algoritmo que leia um valor qualquer, e exiba uma mensagem dizendo se o valor é positivo ou negativo.

```
algoritmo "saber numero"
var
n : inteiro
inicio
leia(n)
se (n<0) entao
      escreval ("o número digitado é negativo")
senão
      escreval ("o número digitado é positivo")
fimse
fimalgoritmo
```

Estrutura de Decisão

SE ENTAO SENAO (encadeado)

```
SE <operação lógica> ENTAO
      <ação 1>
       SENAO
            SE <operação lógica> ENTAO
                  <ação 2>
            SENAO
                  <ação 3>
            FIMSE
FIMSE
```

Significado: Se a primeira <operação lógica> resultar em verdadeiro, então executar <ações 1>. Senão, ignorar <ações 1> e testar a segunda <operação lógica>. Se a segunda operação lógica for verdadeira então executar <ações 2>. Senão, ignorar <ações 2> e executar <ações 3>.

Estrutura de Decisão

Exemplo:

Faça um algoritmo que leia um valor qualquer, e mensagem dizendo se o valor digitado é negativo, positivo ou igual a 0.

```
algoritmo "positivo, negativo e 0"
var n : inteiro inicio leia(n)
se (n<0) então
    escreval (" o número digitado é negativo")
    senão
            se  (n>0) então
                escreval ("o número digitado é positivo")
      senão
                escreval ("o número é igual a 0")
           fimse
fimse
fimalgoritmo
```

Exercícios com o uso do VisuAlg

1) Faça um algoritmo que receba a idade do usuário e verifique se ele tem 18 anos ou mais. Se a resposta for positiva escrever “maior de idade”, senão “menor de idade”.

2) Faça um algoritmo que receba três números inteiros e mostre o maior deles. Considere que os números sempre serão diferentes.

3) Faça um algoritmo que simule um caixa eletrônico quando vamos sacar dinheiro. O caixa eletrônico verifica se o valor que desejamos sacar é menor que o saldo disponível. Assumiremos que há R$ 1000 de saldo disponível para o saque.

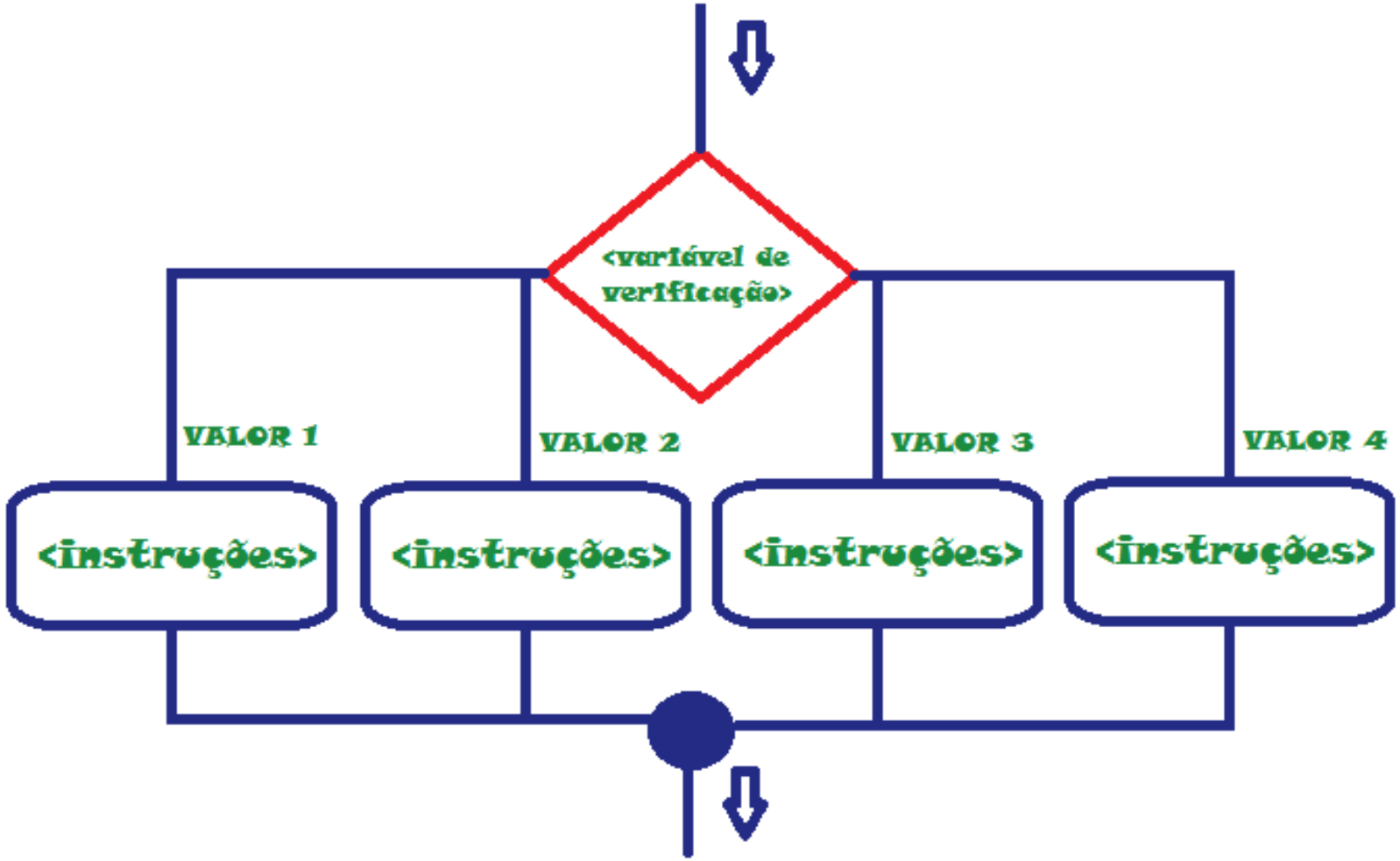
Gabarito dos Exercícios com o uso do VisuAlg

```
algoritmo "SacarDinheiro"
var SaldoDisponivel : REAL ValorDoSaque : REAL inicio SaldoDisponivel := 1000
//Assumimos que há 1000 reais de saldo na conta disponível para saque

ESCREVA ("Informe o valor do Saque: ")
LEIA (ValorDoSaque)

SE ValorDoSaque <= SaldoDisponivel ENTAO
   SaldoDisponivel := SaldoDisponivel - ValorDoSaque
   ESCREVAL ("Sacando R$ ", ValorDoSaque, ".")
SENAO
   ESCREVAL ("O valor solicitado é maior que o valor disponível para saque!")
FIMSE
ESCREVAL ("Saldo disponível: R$ ", SaldoDisponivel)
fimalgoritmo
```

CASO


<variável de verificação>
VALOR 1
VALOR 2
VALOR 3
VALOR 4
<instruções>
<instruções>
<instruções>
<instruções>

CASO

A estrutura ESCOLHA-CASO (em inglês SWITCH-CASE), é uma solução elegante quanto se tem várias **estruturas de decisão** (SE-ENTÃO-SENÃO) aninhadas. Isto é, quando outras verificações são feitas caso a anterior tenha falhado (ou seja, o fluxo do algoritmo entrou no bloco SENÃO). A proposta da estrutura ESCOLHA-CASO é permitir ir direto no bloco de código desejado, dependendo do valor de uma variável de verificação. Veja o esquema abaixo.

Sintaxe:

```
ESCOLHA <variável de verificação>
CASO <valor1>
"instruções a serem executadas caso <variável de verificação> = <valor1>"
CASO <valor2>
"instruções a serem executadas caso <variável de verificação> = <valor2>"
CASO <valor3>
"instruções a serem executadas caso <variável de verificação> = <valor3>"
FIMESCOLHA
```

CASO

Exemplo:

Faça um algoritmo que informa o estado de um determinado time de futebol.

```
algoritmo "Times"
var time: caractere
inicio
escreva ("Entre com o nome de um time de futebol: ")
leia (time)
escolha time
caso "Flamengo"
   escreval ("É um time carioca")
caso "Corinthians"
   escreval ("É um time paulista")
outrocaso
   escreval ("É de outro estado")
fimescolha

fimalgoritmo
```

Exercício de CASO com o VisuAlg

1) Faça um algoritmo que simule uma calculadora. O usuário irá digitar um operador, uma valor inicial, um valor final para que o cálculo seja realizado.

Gabarito do Exercício de CASO com o VisuAlg

```
algoritmo "CalculadoraBasicaComSE"
var
   numero1 : REAL
   numero2 : REAL
   operacao : CARACTERE
   resultado : REAL
inicio

      ESCREVA ("Digite o primeiro número: ")
      LEIA (numero1)
      ESCREVA ("Digite a operação: ")
      LEIA (operacao)
      ESCREVA ("Digite o segundo número: ")
      LEIA (numero2)

      ESCOLHA operacao
         CASO "+"
            resultado := numero1 + numero2
         CASO "-"
            resultado := numero1 - numero2
         CASO "*"
            resultado := numero1 * numero2
         CASO "/"
            resultado := numero1 / numero2
      FIMESCOLHA

      ESCREVA ("Resultado: ", resultado)

fimalgoritmo
```

Introdução ao Mundo da Computação e Lógica de Programação

Estrutura de Repetição

Estrutura de Repetição

As **estruturas de repetição** são utilizadas para repetir uma série de operações semelhantes que são executadas para todos os elementos de uma lista ou de uma tabela de dados, ou simplesmente para repetir um mesmo processamento até que uma certa condição seja satisfeita.

REPITA - ATE

ENQUANTO - FACA

PARA - FACA

Estrutura de Repetição

ENQUANTO – FAÇA

- Verifica primeiro, executa depois;
- Repete somente enquanto <clausula> = verdade;

Sintaxe:

```
enquanto <clausula> faca
      <ações>
fimenquanto
```

Estrutura de Repetição

ENQUANTO - FAÇA

Faça um algoritmo para contar um número até 10.

Exemplo:

```
x: inteiro
x <-1
enquanto (x <= 10) faca
      escreval(x)
      x <-x + 1
fimenquanto
```

Estrutura de Repetição

REPITA - ATE

- Executa primeiro, verifica depois.
- Repete somente enquanto: <clausula> = falso.

Sintaxe:

```
repita
      <ações>
ate <clausula>
```

Estrutura de Repetição

REPITA - ATE

Faça um algoritmo para contar um número até 10.

Exemplo:

```
x: inteiro
x <-1
Repita
      escreval(x)
      x <-x + 1
ate (x > 10)
```

Estrutura de Repetição

PARA – FACA

Sintaxe:

```
para <variável> de <valor inicial> ate <valor final>
[passo<incremento>]faca
      <ações>
fimpara
```

Opcional "passo <incremento>"

Estrutura de Repetição

PARA - FACA

Faça um algoritmo para contar um número até 10.

Exemplo:

```
x: inteiro
para x de 1 ate 10 faca
     escreval(x)
fimpara
```

```
x: inteiro
para x de 1 ate 10 passo 1 faca
     escreval(x)
fimpara
```