ANOTAÇÕES CURSO TALENTO CLOUD

PROGRAMAÇÃO IMPERATIVA

Sequência de execução 1º: () e funções 2º: Potenciação ^ e radiciação V 3º: Multiplicação * e divisão / 4º: Adição + e subtração - 5º: RELACIONAIS > < >= <= <> = 6º: não 7º: e 8º: ou

Estrutura Condicional <> Estrutura Sequencial

Na estrutura condicional a execução da linha de código vai depender da escolha do usuário. Diferente da sequencial que executará a linha integralmente sem haver a intervenção do usuário.

Estrutura Condicional

tomar uma decisão = estrutura condicional(MESMO QUE ESTRUTURA DE SELEÇÃO) são 4 tipos:

1. SIMPLES;
2. COMPOSTA;
3. ENCADEADAS;
4. MULTIPLA ESCOLHA;

SIMPLES (*se* e *fimse*)

> *CERNE*: **APENAS UMA ALTERNATIVA** É FORNECIDA.SERÁ EXECUTADA SE VERDADEIRO, CASO FALSO, PROGRAMA ENCERRA.
>
> > Uma pergunta para esperar somente 1 resposta

se(CONDIÇÃO) **entao**
(INSTRUÇÃO A SER EXECUTADA SE FOR VERDADE)
Fimse

Var
idade:inteiro
Inicio
escreva("Digite a sua idade: ")
leia(idade)
se(idade>18) **entao**
escreva ("Você é maior de idade")
fimse

se(60>40) **entao**
ESCREVA ("VERDADE")
fimse

COMPOSTA (*se*, *senao* e *fimse*)

> *CERNE*: **duas alternativas são fornecidas**. Se a primeira for falsa, a segunda será executada. Porém, se a primeira for verdadeira, a segunda será descartada e, obrigatoriamente, **uma** das alternativas **será a correta**.

se(condição) **entao**
escreva (instrução executada se verdadeiro)
senao escreva(instrução executada se verdadeiro)
fimse

se(idade>18) **entao**
escreva ("Você é maior de idade")
senao escreva("Você é menor de idade")
fimse

ENCADEADA (*se*, *senao*, *senao*, *senao* e *fimse*)

> *CERNE*: é possível observar um conjunto de estruturas compostas, que **apresenta três ou mais alternativas**. Se a primeira alternativa for falsa, então será comparada com as outras **até que se encontre a resposta correta**. Porém, quando se descobre a verdadeira, o **algoritmo se encerra**.

similiar a composta porém com muitos mais SE e SENAO, finalizando com FIMSE para cada SE em aberto. Diversas instruções lógicas com a espera de apenas 1 resposta;

se(condição) **entao**
escreva (instrução executada se verdadeiro)
senao se (condição) **entao**
(instrução executada se a 1ª condição for falsa)
senao
(instrução se todas anteriores forem falsas)
fimse
fimse

Var
numero1, numero2: inteiro *Inicio*
escreva("Digite o 1º número: ")
leia(numero1); escreva("Digite o 2º número: ")
leia(numero2); **se**(numero1 == numero2) **entao**
escreva ("Os números são iguais")
senao se (numero1 > numero2) **entao** escreva("O maior valor é "+numero1) **senao** escrava("O maior valor é " +numero2) **fimse fimse**

MULTIPLAESCOLHA

> CERNE: também apresenta uma série de alternativas. O resultado da sua operação é baseado na resposta do usuário e **não há a necessidade do uso de uma expressão lógica.**
>
> Assim, a variável, que possui a resposta do usuário, será comparada de forma exata com uma das opções fornecidas pela estrutura.

> Veja a seguir como estruturar um algoritmo em condicional múltipla escolha.

caso1
(estrutura a serem executadas se a condição for verdade) **caso2**
(estrutura a serem executadas se a condição for verdade) **caso3**
(estrutura a serem executadas se a condição for verdade) **Outrocaso** (estrutura a serem executadas se a condição for verdade) **fimescolha**

Var
numero1, numero2, resultado : Real
operacao : caractere
Inicio
escreva ("Digite o primeiro número: ")
leia (numero1)
escreva ("Digite o símbolo da operação: ")
leia (operacao)
escreva ("Digite o segundo número: ")
leia (numero2)
escolha operacao
caso "+"
resultado <- numero1 + numero2
caso "-"
resultado <- numero1 - numero2
caso "x"
resultado <- numero1 x numero2
caso "/"
resultado <- numero1 / numero2
fimescolha
escreva ("Resultado: ", resultado)

escolha EscolhaEsporte
caso "1"
escreva("futebol")
caso "2"
escreva("Vôlei") **caso** "3"
escreva("Basquete")
Outrocaso escreva("Opção errada") **fimescolha**

> As **estruturas condicionais** são fundamentais no desenvolvimento de algoritmos. Pois, com elas, podemos fazer a entrada dos dados a serem processados. Assim, a saída de dados é o retorno de um resultado.

ESTRUTURA DE REPETIÇÃO

Permitem repetir instruções sem que haja a necessidade de duplicação do algorítmo. Tornando o código mais enchuto.

> Link material complementar. https://github.com/drianoaz/visualg/blob/master/docs/04-estruturas-de-repeticao.md

ETAPAS DA ESTRUTURA DE REPETIÇÃO

Toda estrutura de repetição precisa ter:

1. **uma variável contadora** com valor inicial.
2. **uma condição** que indica o início e o término da repetição. instrução **se repetirá até atingir o limite**. Limite que é definido pelo usuáiro.
3. **incremento**, que especifica um valor a ser incrementado na variável contadora.

As estruturas de repetição **são divididas em três categorias.** Elas se diferenciam por um teste, que é a condição para executar a repetição.

- Estrutura com Teste **no ínicio**
- Estrutura com Teste **no final**
- Estrutura com variável de controle. Onde o teste é **previamente** determinado através da variável.

ESTRUTURA DE REPETIÇÃO COM TESTE NO INÍCIO - 'ENQUANTO() FAÇA'

Enquanto a condição for verdade, o algoritmo repete e novamente analisa a condição no início, se for verdade, novamente repete e assim fica no looping até a condição ter um resultado falso.

A condição, se trata de uma expressão lógica, que é avaliada antes de uma nova repetição ocorrer, **enquanto o resultado da condição for verdadeiro, o algoritmo será executado**, e ficará repetindo suas instruções.

Sintaxe:

```
contador <- variável contadora
enquanto (contador < condição) faca
(instruções a serem realizadas durante a repetição)
    contador <- contador + incremento
fimenquanto
```

Neste exemplo, a variável contador inicia com o valor 1 e passa pela condição. Enquanto o contador for menor ou igual a 10 a frase "**Eu amo lógica!**" será mostrada na tela. Assim, essa frase será emitida na tela dez vezes.

```
Var
contador: inteiro
Inicio
contador <- 1
enquanto (contador <= 10) faca
   Escreval ("Eu amo lógica!")
   contador <- contador + 1
fimenquanto
```

ESTRUTURA DE REPETIÇÃO COM TESTE NO FINAL - 'REPITA() ATÉ'

Já a estrutura de repetição com teste no final tem sua condição avaliada apenas no final da estrutura, logo, **os comandos são executados pelo menos uma vez**, já que a verificação da condição se encontra no final.

Com essa estrutura, as instruções são repetidas até que uma determinada condição seja verdadeira.

Sintaxe:

```
contador <- inicialização
Repita
(instruções a serem realizadas durante a  repetição)
   contador <- contador + incremento
ate (contador > condição)
```

Exemplo de algoritmo com teste no final:

```
var
  numero : inteiro
  contador : inteiro
inicio
    contador <- 0
    REPITA
      ESCREVA ("Digite um número para a somar:  ")
      LEIA (numero)
      contador <- contador + numero
      ESCREVAL ("Total: ", contador)
    ATE numero = 0
```

ESTRUTURA DE REPETIÇÃO COM VARIÁVEL DE CONTROLE - PARA'

Essa estrutura de repetição é **utilizada quando sabemos quantas vezes as instruções precisam ser repetidaS**, e a própria estrutura funciona como o contador. "laço contado"

```
para contador de inicialização ate condição passo incremento faca
        (instruções a serem realizadas durante a  repetição)
fimpara
```

Exemplo da estrutura com variável de controle

O algoritmo recebe um número de entrada digitado pelo usuário e mostra a tabuada da soma desse número. Existem duas variáveis do tipo inteiro: a variável número guarda o valor digitado pelo usuário e a variável contador é utilizada para ir do número 0 até o 10.

Para finalizar, é apresentado na tela a soma dos números do contador com o número digitado pelo usuário. A seguir, vamos observar a representação do algoritmo.

Exemplo da estrutura com variável de controle

Este algoritmo usa a variável contador, e o algoritmo se repetirá 10 vezes, ou seja, já existe um valor estabelecido de repetições.

```
var
numero, contador: inteiro
Inicio
escreva ("Digite o número que deseja verificar a tabuada de soma: ")
leia(numero)
para contador <- 0 ate 10  faca
Escreval (numero, " + ", contador, " = ", numero + contador)
fimpara
```