

algoritmo "exercício 28"

var

opcao, n1, n2: inteiro

funcao seleciona_opcao():inteiro

var o:caractere

inicio

repita

escreva ("Efetue uma multiplicação ou obtenha o fatorial de um")

escreval (" número natural.")

escreval ("Digite:")

escreval ("* para efetuar uma multiplicação;")

escreval ("! para calcular o fatorial;")

escreval ("S para sair.")

escreva ("Sua opção: ")

leia (o)

escolha (o)

caso "*"

retorne 1

caso "!"

retorne 2

caso "s"

retorne 0

outrocaso

escreval ("Opção inválida!")

fimescolha

ate((o="*") ou (o="!") ou (o="s"))

fimfuncao

funcao multiplicacao (a: inteiro; b: inteiro): inteiro

var res: inteiro

inicio

se (a=0) entao

retorne 0

senao

res <- 0

enquanto (b>0) faca

res <- res + a

b <- b-1

fimenquanto

retorne res

fimse

fimfuncao

funcao fatorial (n: inteiro): inteiro

var i, fat: inteiro

inicio

fat <- 1

para i de 2 ate n faca

fat <- multiplicacao (fat, i)

fimpara

retorne fat

fimfuncao

inicio

repita

opcao <- seleciona_opcao()

escolha (opcao)

caso 1

repita

escreva ("Forneça um número natural para o multiplicando: ")

leia (n1)

se (n1<0) entao

escreval ("Não é aceito um multiplicando negativo.")

fimse

ate (n1>=0)

escreva ("Forneça um número natural para o multiplicador: ")

leia (n2)

enquanto (n2<0) faca

escreval ("Não é aceito um multiplicador negativo.")

escreva ("Forneça um número natural para o multiplicador: ")

leia (n2)

fimenquanto

escreval (n1," * ",n2," = ",multiplicacao(n1,n2))

caso 2

repita

escreva ("Forneça um número natural para o cálculo de seu")

escreva (" fatorial: ")

leia (n1)

se (n1<0) entao

escreval ("Não é aceito um valor negativo.")

fimse

ate (n1>=0)

escreval (n1,"! = ",fatorial(n1))

caso 0

escreva ("Agradecemos a escolha de nosso software!")

fimescolha

ate (opcao=0)

fimalgoritmo

Modularização

Continuaremos nosso estudo de modularização pelo conceito de procedimento.

Um procedimento é um módulo que possui **ou não** um conjunto de entradas, efetua a execução de um conjunto de instruções e **não** gera um valor como saída, ou seja, não apresenta retorno.

Modularização

A estrutura de um procedimento é a seguinte:

```
procedimento          <nome_do_procedimento>
([<sequência-de-declarações-de-parâmetros>])
    // Seção de Declarações Internas
inicio
    // Seção de Comandos
fimprocedimento
```

Obs.: A chamada de um procedimento sem parâmetro segue a mesma regra aplicada à chamada de uma função sem parâmetro.

Modularização

A *<sequência-de-declarações-de-parâmetros>* é uma lista com a seguinte forma geral:

*identificador1: tipo_de_dado; identificador2:
tipo_de_dado; ...; identificadorN: tipo_de_dado*

Assim como na função o *tipo_de_dado* deve ser especificado para cada uma das N variáveis definidas como parâmetros. É na declaração de parâmetros que informamos quais serão as entradas do procedimento.

Como um procedimento é um módulo, também podemos declarar, na **Seção de Declarações Internas**, variáveis que serão utilizadas nas manipulações efetuadas.

As regras para construção do *nome_do_procedimento* são as mesmas aplicadas na definição dos identificadores de variáveis.

Por fim, é na **Seção de Comandos**, também denominado corpo do procedimento, onde as entradas são processadas e demais operações são feitas.

Modularização

Exemplo de procedimento:

algoritmo "exemplo procedimento"

 procedimento frase()

 inicio

 escreval ("Ola! Eu estou vivo.")

 fimprocedimento

 inicio

 frase()

 escreval ("Diga de novo:")

 frase()

 fimalgoritmo

Modularização

Exercício 36:

Construa um procedimento que escreve no monitor a mesma frase 10 vezes. O alinhamento que deve ser obedecido para a escrita das frases é apresentado a seguir:

Sou um procedimento!

Sou um procedimento!

Sou um procedimento!

...

```
procedimento escreve_frase()  
var i, j: inteiro  
inicio  
  para i de 0 ate 9 faca  
    para j de i ate 1 passo -1 faca  
      escreva(" ")  
    fimpara  
    escreval("Sou um procedimento!")  
  fimpara  
fimprocedimento
```

procedimento escreve_frase()

var i: inteiro

frase: caractere

inicio

frase <- "Sou um procedimento!"

para i de 0 ate 9 faca

escreval(frase)

frase <- " " + frase

fimpara

fimprocedimento

Modularização

- Escopo de variáveis

Com a introdução do conceito de módulo passou a ser possível, além de declarar variáveis no algoritmo principal, declarar variáveis nos módulos, seja como parâmetros ou dentro dos mesmos.

Desta forma se tornou necessário o estabelecimento de regras que normatizassem a visibilidade das variáveis, estabelecendo-se o conceito de escopo de variáveis.

Modularização

- Escopo de variáveis (continuação)

O escopo de variáveis é o conjunto de regras que determinam a validade de variáveis nas diversas partes do algoritmo, ou seja, onde cada uma das variáveis pertencentes ao algoritmo existe, estando disponível para manipulação.

Veremos agora três tipos de variáveis, no que se refere ao escopo.

Modularização

- Variáveis locais

Variáveis locais são aquelas que só têm validade dentro do módulo no qual são declaradas.

```
algoritmo "exemplo"  
  funcao func1 (...):inteiro  
  var  
    abc,x,z:inteiro  
  inicio  
    ...  
  fimfuncao  
  funcao func2 (...):logico  
  var  
    z:inteiro  
  inicio  
    ...  
  fimfuncao  
inicio  
...  
fimalgoritmo
```

Modularização

- Parâmetros

Os parâmetros são declarados como sendo as entradas de um módulo. Um parâmetro é uma espécie de variável local do módulo que é inicializada com um valor externo ao módulo.

Existem dois tipos de passagem de parâmetro para um módulo:

- A passagem de parâmetro por valor;
- e a passagem de parâmetro por referência.

Analisaremos inicialmente a passagem de parâmetros por valor. Em todos os módulos que desenvolvemos até o momento nos utilizamos da passagem de parâmetro por valor. Ou seja, em todas as aplicações que utilizamos o conceito de módulo sempre ao chamarmos o módulo que pretendíamos executar passávamos para o mesmo variáveis ou constantes que, respectivamente, continham ou representavam o valor que desejávamos inicialmente para cada um dos parâmetros do módulo.

Modularização

- Parâmetros (passagem por valor)

Ao se alterar o valor de um parâmetro, passado por valor, esta alteração não terá efeito na variável que foi passada ao módulo. Isto ocorre, pois quando se passa parâmetros por valor para um módulo, são copiados os valores das variáveis ou constantes para os parâmetros. Isto é, os parâmetros passados por valor existem independentemente das variáveis que foram passadas para o módulo, estes apenas são inicializados com uma cópia dos valores passados para o módulo. É esta característica, da passagem de parâmetro por valor, que possibilita a passagem de uma constante como parâmetro para um módulo.

Modularização

- Parâmetros (passagem por valor)

Exemplo:

```
algoritmo "exemplo 1"
```

```
var
```

```
  j:inteiro
```

```
  procedimento f1(i: inteiro)
```

```
  inicio
```

```
    i<-18
```

```
    escreval (i)
```

```
  fimprocedimento
```

```
inicio
```

```
  j<-3
```

```
  f1(j)
```

```
  escreva (j)
```

```
fimalgoritmo
```

Modularização

- Parâmetros (passagem por valor)

Exemplo:

algoritmo "exemplo 2"

var

 i:inteiro

 procedimento f1(i: inteiro)

 inicio

 i<-18

 escreval (i)

 fimprocedimento

inicio

 i<-3

 f1(i)

 escreva (i)

fimalgoritmo

Modularização

- Parâmetros (passagem por referência)

Em algumas situações torna-se necessário que módulos manipulem posições de memória alocadas fora de seu escopo. Em outras palavras, torna-se necessário que módulos possam manipular variáveis com declarações externas.

Uma situação onde este fato é facilmente visualizado é quando torna-se preciso que uma função retorne mais de um valor.

Para atender esta necessidade existe passagem de parâmetro por referência.

Modularização

- Parâmetros (passagem por referência)

Na passagem de parâmetro por referência, ao invés de ocorrer a declaração de uma variável local ao módulo e esta ser inicializada com um valor externo, ocorre apenas a passagem para o módulo da localização na memória de uma determinada variável, em outras palavras, especifica-se que uma variável externa será visível (manipulável) dentro do módulo.

Obs. Devido à natureza da passagem de parâmetro por referência torna-se impossível a passagem de constantes para módulos que se utilizem desta modalidade de passagem de parâmetro.

algoritmo "Exemplo de passagem de parâmetro por referência"

var

a, b:inteiro

procedimento troca (var x: inteiro; var y:inteiro)

var aux:inteiro

inicio

aux <- x

x <- y

y <- aux

fimprocedimento

inicio

escreva ("entre com a: ")

leia(A)

escreva ("entre com b: ")

leia(B)

troca(a,b)

escreval ("Valor de a: ",a)

escreval ("Valor de b: ",b)

fimalgoritmo

Modularização

Exercício 37:

Construa uma função que receba como parâmetro dois números inteiro A e B , respectivamente, e retorne o quociente e o resto da divisão de A por B . As operações aritméticas de multiplicação, divisão, quociente da divisão inteira e resto da divisão inteira **não** podem ser utilizadas na construção da função. Elabore um algoritmo que se utilize de forma coerente da função construída.