Algoritmos TADS

Algoritmos – Modularização

Professor: Victor Hugo L Lopes

Agenda

- Modularização de programas;
- Procedimentos;
- Funções.

Algoritmos TADS

- Subprogramas

Uma tarefa pode ser realizada com sub-tarefas!!!

Ex.:

A tarefa de fazer um bolo:

-sub-tarefa de separar os ingredientes;

-sub-tarefa de misturar os ingredientes;

-sub-tarefa de bater os ingredientes na batedeira;

-sub-tarefa de untar a forma;

-sub-tarefa de ligar o forno;

- ... sub-tarefa de comer o bolo. \o/

São tarefas, portanto, que auxiliam na realização da tarefa principal.

Podem ter formas genéricas e reusáveis.

Algoritmos TADS

- Subprogramas

Subprograma é um programa que auxilia o programa principal através da realização de uma determinada sub-tarefa.

Também costuma receber os nomes de sub-rotina, procedimento, método ou módulo.

Os subprogramas são chamados dentro do corpo do programa principal como se fossem comandos. Após seu término, a execução continua a partir do ponto onde foi chamado.

É importante compreender que a chamada de um subprograma simplesmente gera um desvio provisório no fluxo de execução.

Algoritmos TADS

- Subprogramas

Os subprogramas são utilizados para a Modularização de um software.

Tem-se duas formas básicas de modularização: Procedimentos e Funções.

A modularização cria a ideia de um corpo principal para o programa.

Algoritmos TADS

- Subprogramas

Algumas regras:

-todo subprograma possui um identificador(nome);

-todo subprograma é um bloco bem definido de instruções, que realizam tarefas bem definidas;

-um subprograma pode ser chamado a executar a sua tarefa;

-um subprograma pode receber dados para auxiliá-lo na execução de sua tarefa;

-um subprograma pode devolver um valor à que lhe chamar.

Algoritmos TADS

- Subprogramas

Procedimentos

Um procedimento é um subprograma que possui um conjunto de instruções que são executadas todas as vezes que o procedimento for chamado, que não retorna valor algum para quem o chamar.

Deve ser declarado (escrito) entre a declaração de Var e o início das instruções do programa.

Algoritmos TADS

- Subprogramas

Procedimentos

sintaxe

procedimento *<nome-de-procedimento>* *[(<seqüência-de-declarações-de-parâmetros>)]*

// Seção de Declarações Internas

inicio

// Seção de Comandos

fimprocedimento

Algoritmos TADS

- Subprogramas

Procedimentos

exemplo.:

Algoritmo “procedimento”

Procedimento escreve

Inicio

    escreval(“Feito pelo procedimento”)

Fimprocedimento

Inicio

    escreve

fimalgoritmo

Algoritmos TADS

- Subprogramas

Procedimentos

Os procedimentos também podem receber valores enviados por quem os chamam: parâmetros.

ex.:

```
procedimento soma (x,y: inteiro)
inicio
    res <- x + y
fimprocedimento
```

Algoritmos TADS

- Subprogramas

Procedimentos

ex. - Algoritmo para somar 2 números

```
Algoritmo “soma_procedimento”
Var A,B,T:inteiro
        procedimento soma (x,y: inteiro)
        inicio
                    T <- x + y
        fimprocedimento
inicio
    leia(A,B)
    soma(A,B)
    escreval(A,”+”,B,”=”,T)
fimalgoritmo
```

Algoritmos TADS

ex. - Algoritmo para somar 2 números – outra forma

```
Algoritmo “soma_procedimento”
Var A,B,T:inteiro
        procedimento leValores
        inicio
                leia(A,B)
        fimprocedimento
        procedimento soma (x,y: inteiro)
        inicio
                        T <- x + y
        fimprocedimento
inicio
    leValores
    soma(A,B)
    escreval(A,”+”,B,”=”,T)
fimalgoritmo
```

Algoritmos TADS

DESAFIO:

- Construir um algoritmo que leia um número, multiplique por 3 e guarde em uma variável e apresente esta variável, utilizando um procedimento, repetindo estes passos 5 vezes.

Algoritmos TADS

DESAFIO:

- E pra não perder o costume, que tal desenvolver um algoritmo para calcular o fatorial de 5 utilizando procedimento?!

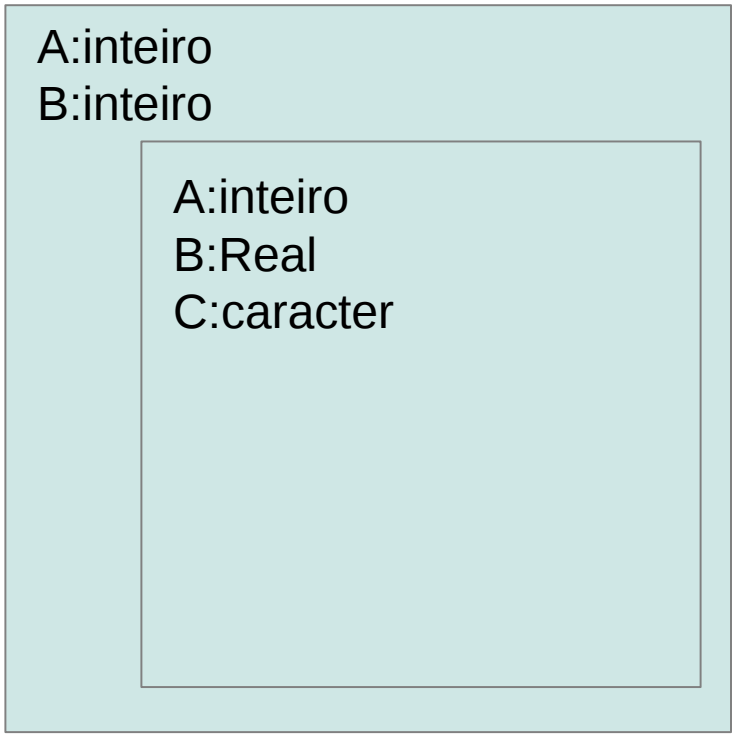
Algoritmos TADS

Subprogramas

Escopo de variável:

Global

Local



Algoritmos TADS

```
Algoritmo “global_local”
Var A,B,C:inteiro
        procedimento soma
        var A,B,C:inteiro
        inicio
                A<-100
                B<-10
                C<-A+B
        fimprocedimento
inicio
        A<-1
        B<-2
        C<-A+B
        soma
        escreval(A,”+”,B,”=”,C)
fimprocedimento
```

Algoritmos TADS

- Subprogramas

Procedimentos

Exercício:

Construa uma calculadora básica, que realize as 4 operações básicas, utilizando procedimentos.

Algoritmos TADS

- Subprogramas

Funções

função é um subprograma que retorna um valor.

De modo análogo aos procedimentos, sua declaração deve estar entre o final da declaração de variáveis e a linha inicio do programa principal, e segue a sintaxe abaixo:

funcao <nome-de-função> [(<seqüência-de-declarações-de-parâmetros>)]: <tipo-de-dado>

    // Seção de Declarações Internas

inicio

    // Seção de Comandos

fimfuncao

Algoritmos TADS

- Subprogramas

Funções

O <nome-de-funcao> obedece às mesmas regras de nomenclaturas de variáveis.

ex.:

```
Funcao calcula(A:inteiro):inteiro
Var T:inteiro
Inicio
    T<- A *3
    Retorne T
fimfuncao
```

Algoritmos TADS

- Subprogramas

Funções

A <sequencia-de-declaracoes-de-parametros> é uma sequência parâmetros de entrada na função. São declarações de variáveis no estilo <nome-da-variavel>:tipo de dados, separando com vírgula em caso de mais de uma variável. A declaração do parâmetro pode utilizar a palavra Var, que é opcional, e indica passagem de parâmetro por referência.

ex.:

```
Funcao soma(A,B:inteiro):inteiro
Inicio
    Retorne A+B
fimfuncao
```

Algoritmos TADS

- Subprogramas

Funções

O comado Retorne indica que a função deve retornar o valor para quem a chamou. Ao encontrar o comando Retorne, a função retorna o valor e termina a execução*.

ex.:

```
Funcao soma(A1,B1:inteiro):inteiro
Inicio
    (3)Retorne(A1+B1)
fimfuncao
[...]
    (1)leia(A,B)
    (4)TOTAL<-(2)soma(A,B)
    (5)escreval(TOTAL)
[...]
```

```
algoritmo “saudacao_funcao”
Var NOME, SOBRENOME, SEXO:caracter
Funcao saudacao(N,S,SX:caracter):caracter
Var TEXTO:caracter
Inicio
    se (SX = “F”)entao
        TEXTO<-”Senhora ” + N + “ “ + S + “, Bom Dia”
    Senao
        TEXTO<-”Senhor ” + N + “ “ + S + “, Bom Dia”
    fimse
    retorne(TEXTO)
Fimfuncao
Inicio
    leia(NOME,SOBRENOME,SEXO)
    //variavel<-saudacao(NOME,SOBRENOME,SEXO)
    //escreval(variavel)
    escreval(saudacao(NOME,SOBRENOME,SEXO))
fimalgoritmo
```

Algoritmos TADS

- Subprogramas

Funções

Uma função também pode trabalhar sem a necessidade de receber parâmetros.

ex.:

```
Funcao soma:inteiro
Var T:inteiro
Inicio
    T<- A+B
    Retorne T
fimfuncao
```

```
Algoritmo “funcao_sem_parametro”
Var TOTAL:inteiro

    Funcao calcula:inteiro
    Var A,B:inteiro
    Inicio
        A<-2
        B<-3
        retorne(A^B)
    Fimfuncao

Inicio
    TOTAL<- calcula
    escreval(TOTAL)
fimalgoritmo
```

```
Algoritmo “funcao_sem_parametro”
Var TOTAL,A,B:inteiro

    Funcao calcula:inteiro
    Inicio
        retorne(A^B)
    Fimfuncao

Inicio
    A<-2
    B<-3
    TOTAL<- calcula
    escreval(TOTAL)
fimalgoritmo
```

```
Algoritmo “funcao_sem_parametro”
Var TOTAL:inteiro

    Funcao calcula:inteiro
    Var A,B:inteiro
    Inicio
        A<-2
        B<-3
        retorne(A^B)
    Fimfuncao

Inicio
    TOTAL<- calcula
    escreval(TOTAL)
fimalgoritmo
```

```
Algoritmo “funcao_sem_parametro”
Var TOTAL,A,B:inteiro

    Funcao calcula:inteiro
    Inicio
        retorne(A^B)
    Fimfuncao

Inicio
    A<-2
    B<-3
    TOTAL<- calcula
    escreval(TOTAL)
fimalgoritmo
```

Algoritmos TADS

- Subprogramas

Funções

Vale ressaltar que a variável que irá receber o valor retornado de uma função deve ser do mesmo tipo de dados do “dado” retornado. O tipo de retorno não precisa ter relação com o tipo dos parâmetros.

ex.:

```
Funcao teste(A,B:real):caracter
Var TEXTO:caracter
Inicio
        se (A>B)entao
                TEXTO<- “A é maior do que B”
                senao
           TEXTO<-”A não é maior do que B”
        fimse
        retorne(TEXTO)
fimfuncao
[...] varivalcaracter <- teste(2,3) [...]
```

Algoritmos TADS

- Implementando:

```
algoritmo “testar_valores”
var A,B,C: inteiro
funcao teste(A,B,C:inteiro):logico
inicio
        se ((A>B)E(B>C)) entao
                retorne verdadeiro
    senao
                retorne falso
    fimse
fimfuncao
inicio
        leia(A,B,C)
        se (teste(A,B,C))entao
                escreval(A,“ é o maior valor”)
        fimse
fimalgoritmo
```

Algoritmos TADS

- Subprogramas

Funções

exercitando:

1) Construir um algoritmo para uma calculadora básica, utilizando funções.

2) Construir um algoritmo para ler 3 números inteiros, utilize funções para ordená-los de forma decrescente e descrever os pares e ímpares.

3) Construir um algoritmo para escrever na tela, utilizando funções, os números pares e ímpares entre 1 e 100