Programação de Computadores I

Procedimentos e Funções

PROFESSORA CINTIA CAETANO

Introdução

- Dividir o programa em subprogramas é útil para deixar mais fácil de depurá-lo e de se reutilizar código.
- Temos dois tipos de subprogramas:
  - Procedimentos
  - Funções

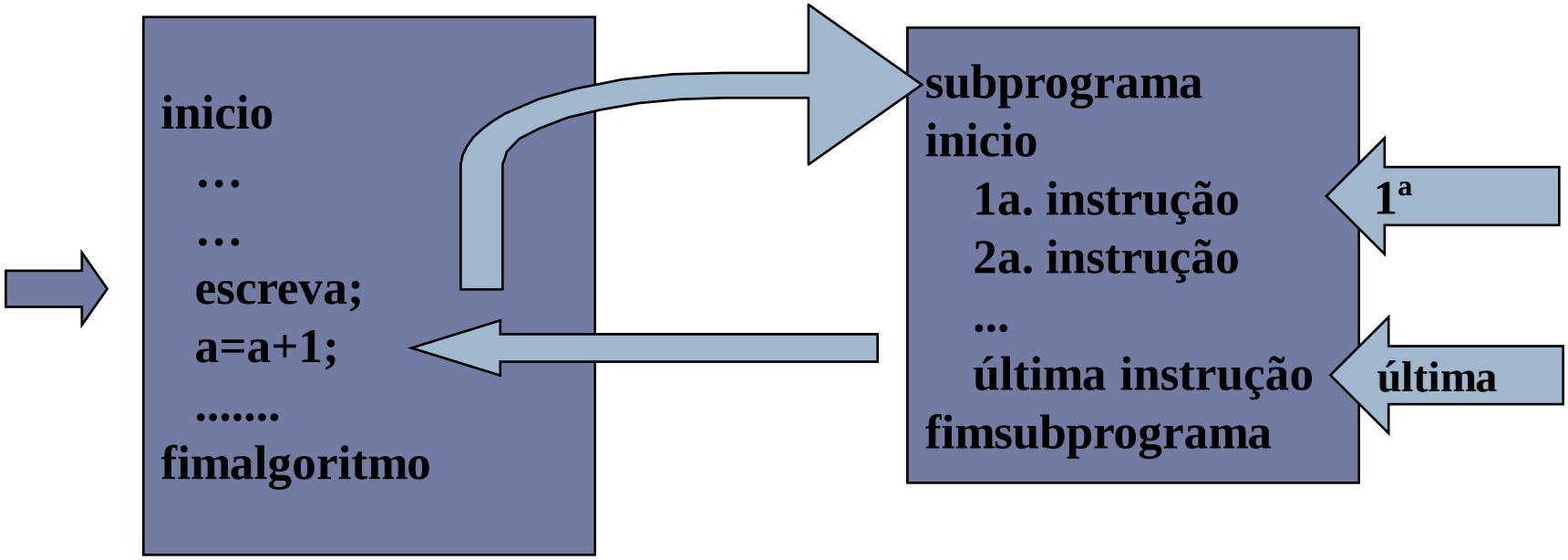
Introdução

- Procedimentos e funções são trechos de código que são separados do fluxo principal do programa e podem ser chamadas uma ou mais vezes.
- O funcionamento de um procedimento e uma função é muito similar.
- A diferença entre procedimento e função está no fato de que a *função* retorna explicitamente um valor e a procedimento não. **VEREMOS ISSO MAIS A FRENTE!**

Introdução

- Quando um procedimento ou uma função é chamada durante a execução de um programa:
  1. A execução do programa desvia para o subprograma (procedimento ou função).
  2. Executa o subprograma (procedimento ou função) chamado.
  3. E recomeça a execução do programa na linha após a chamada do subprograma (procedimento ou função).

Introdução


inicio
…
…
escreva;
a=a+1;
.....
fimalgoritmo
subprograma
inicio
1a. instrução
2a. instrução
...
última instrução
fimsubprograma
1ª
última

Introdução

- Vantagens na utilização de procedimentos e funções.
  - Melhorar a clareza e compreensão do programa;
  - Diminui o tamanho do código;
  - Facilita a alteração;
  - Diminui a quantidade de erros;
  - Diminui a complexidade;
  - Facilita o gerenciamento;
  - Independência;
  - Reutilização;
  - Etc.

Procedimento

- É um subprograma que não retorna nenhum valor (corresponde ao procedure do Pascal).
- Deve ser definido após a declaração das variáveis.
- É ativado ao ser chamado no programa principal.
- Pode ou não ter parâmetros.

Sintaxe

```
procedimento <identificador> ([var]<parâmetros>)

var

<declaração de variáveis locais>

inicio

<lista de comandos>
fimprocedimento
```

Sintaxe

- **Identificador:** nome do procedimento também tratado como identificador.
- **Passagem de parâmetros por referência:** utiliza-se a construção VAR antes dos identificadores para indicar a passagem por referência. Os identificadores são separados por vírgula.
- **Parâmetros:** Entre um mesmo tipo de dados são separados por vírgula. Entre tipos de dados a separação é feita com ponto-e-vírgulas ';'.

Sintaxe

- Toda vez que precisarmos chamar o procedimento devemos chamar por esse nome/identificador.
- As declarações de variáveis dentro do bloco do procedimento, são opcionais.
- O nome do procedimento obedece as mesmas regras de nomenclatura das variáveis.

Parâmetro

- **Parâmetro** é um valor que é fornecido à função quando ela é chamada.
- É passado um valor para que a função execute um determinado cálculo.
- A lista de parâmetros e tipos, delimitada pelos parênteses no cabeçalho do procedimento, também é opcional.
- A declaração de parâmetros é feita de maneira semelhante à declaração de variáveis.

Exemplo 1

```
algoritmo "procedimento_soma“
var
    m,n,res: inteiro // variáveis globais
inicio //inicio do programa principal
// Procedimento
procedimento soma
var
    aux: inteiro //variável local
Inicio //inicio do procedimento
aux <- n + m
res <- aux
fimprocedimento

//Programa Principal
n <- 4
m <- 6
soma
escreva(res)

fimalgoritmo
```

Exemplo 2

```
Algoritmo “Ola”
var
    nome: literal

procedimento elogio
inicio
    Escreva("Bom dia “ ,  nome , " seja bem vindo!")
fimprocedimento

//Programa Principal
inicio
    Escreva(" Informe o seu nome:  ")
    Leia(nome)
    elogio
fimalgoritmo
```

Funções

- A estrutura de uma função é muito parecida com um procedimento.
- Pode-se imaginar que uma função é um procedimento com características especiais quanto ao retorno de valores.
- A função é um subprograma que pode agir sobre os dados e **retornar um único valor** para o programa principal.

Funções

- Uma característica que distingue uma função de um procedimento é que a função pode ser impressa, atribuida ou participar de cálculos como se fosse uma variável qualquer.
- Isso por que ela tem um valor retorno.

Sintaxe

```
funcao <identificador> ([var]<parâmetros>) <tipo de retorno>

var

<declaração de variáveis locais>

inicio

<lista de comandos>

retorne <variável de retorno>
fimfuncao
```

Exemplo 3

```
Algoritmo "Soma"
var
    a, b, result: inteiro

funcao ImprimeSoma(x:inteiro; y:inteiro):inteiro
Var
    soma: inteiro
inicio
    soma <- x + y
    retorne soma
fimfuncao

{Programa Principal}
inicio
    Escreva("Entre com o primeiro valor: " )
    Leia(a)
    Escreva("Entre com o segundo valor: " )
    Leia(b)
    result <- ImprimeSoma(a,b)
    Escreval("A soma dos valores é igual a: ", result)
fimalgoritmo
```

Exemplo 4

```
Algoritmo "Produto"
Var
    A,B,C: real

Funcao PROD(X,Y : real): real
inicio
    retorne X *Y
fimfuncao

{--* PROGRAMA PRINCIPAL *--}
Inicio
    A <- 2
    B <- 10
    C <- PROD(A,B)
    Escreva("O produto de A e B é: ", C)
Fimalgoritmo
```

Variáveis locais e Globais

- Quando declaramos as variáveis, nós podemos fazê-las;

1. **Local** - declaradas dentro de um procedimento ou função - só têm validade dentro do escopo ao qual foram declaradas.
2. **Global** - são as variáveis do programa principal. São acessíveis a todos os subprogramas e ao programa principal.

- Obs: Parâmetros são variáveis locais.

Exemplo 5

```
Algoritmo "Variaveis Locais"
procedimento valores
Var
   a, b, c: inteiro  // Variáveis locais
Inicio
     a <- 1
     b <- 2
     c <- 3
     Escreva("a = ", a, " b = ", b," c = ", c)
fimprocedimento

Inicio
    valores
Fimalgoritmo
```

Exemplo 6

```
Algoritmo "Variaveis Globais"
Var
   a, b, c:  inteiro  // Variaveis globais

Procedimento valores(a: inteiro; b: inteiro; c: inteiro)
Inicio
     Escreva("a = ", a, " b = ", b," c = ", c)
Fimprocedimento

Inicio
    a <- 1
    b <- 2
    c <- 3
    valores(a,b,c)
Fimalgoritmo
```

Passagem de Parâmetros

- SABEMOS QUE: Os programas passam informações para procedimentos e funções usando parâmetros.

- Essa passagem de parâmetros podem ser feita de 2 formas:
  1. Por valor
  2. Por referência

Passagem de Parâmetros por Valor

- Fornece uma cópia dos valores dos parâmetros para a função chamada.
- Quaisquer modificações que a função fizer nos valores ocorrerá apenas dentro da função.
- Ou seja, não são alterados os valores dos parâmetros fora da função.

Passagem de Parâmetros por Valor

```
Algoritmo "Parametros por Valor"
Var
    a, b: inteiro

procedimento exibe_altera(x: inteiro; y:inteiro)
Inicio
   Escreval("Valor original: ", x, " ", y)
    x <- x + 100
   y <- y + 100
   Escreval("Valor alterado: ", x, " ", y)
Fimprocedimento

{Programa Principal}
Inicio
    a <- 1
    b <- 2
    exibe_altera(a,b)
    Escreval("Valor final: ", a, " ", b)
Fimalgoritmo
```

Resultado do código

Valor original: 1 2
Valor alterado: 101 102
Valor final: 1 2

Passagem de Parâmetros por Referência

- Na chamada do subprograma é passado o **endereço** de uma variável, onde se pode dar um nome alternativo a variável no subprograma.
- Obs: deve-se utilizar o nome var na frente do identificador do parâmetro

- **Exemplos:**

  Procedimento soma (var x,y : integer)

  funcao imprime (var aluno: literal)

PASSAGEM DE PARÂMETROS POR REFERÊNCIA

- Quando a função termina, as variáveis dentro do **programa principal** estão alteradas.

```
Algoritmo "Parametros por Referencia"
Var
     a, b: inteiro

procedimento exibe_altera(var x: inteiro; var y:inteiro)
Inicio
    Escreval("Valor original: ", x, " ", y)
      x <- x + 100
    y <- y + 100
    Escreval("Valor alterado: ", x, " ", y)
Fimprocedimento

{Programa Principal}
Inicio
     a <- 1
     b <- 2
     exibe_altera(a,b)
     Escreval("Valor final: ", a, " ", b)
Fimalgoritmo
```

Resultado do código

Valor original 1 2
Valor alterado 101 102
Valor final: 101 102

PASSAGEM DE PARÂMETROS POR REFERÊNCIA

- Quando a função termina, as variáveis dentro de **programa principal** estão alteradas.

```
Algoritmo "Parametros por Referencia"
var
     a, b: inteiro

Procedimento troca(var a: inteiro; var b:inteiro)
Var
   aux: inteiro
Inicio
    aux <- a
    a <- b
    b <- aux
Fimprocedimento

{Programa Principal}
Inicio
     a <- 10
     b <- 20
     Escreval("Antes da troca: a = ", a, " b = ", b)
     troca(a,b)
     Escreval("Depois da troca: a = ", a, " b = ", b)
Fimalgoritmo
```

Resultado do código

Antes da troca: a = 10 b = 20
Depois da troca: a = 20 b = 10

DÚVIDAS!?!?