

Sistema
FIRJAN INFORMA, FORMA, TRANSFORMA.



Algoritmos e Estruturas de Dados

Fabício Curvello Gomes

Sistema
FIRJAN INFORMA, FORMA, TRANSFORMA.

```
graph TD; A[Programa Principal] --> B[Rotina 1]; A --> C[Rotina 2]; B --> D[Rotina 1.1]; B --> E[Rotina 1.2]; B --> F[Rotina 1.3]; C --> G[Rotina 2.1]; E --> H[Rotina 1.2.1]; E --> I[Rotina 1.2.2];
```

Sub-Rotinas do Tipo Procedimento


INFORMA, FORMA, TRANSFORMA.



Introdução

Em linhas gerais, problemas complexos exigem para sua solução algoritmos complexos. No entanto, é possível dividir um problema grande em problemas menores, ou seja, usar o processo de modularidade.

Cada parte menor ou módulo tem um algoritmo mais simples e possibilita maior facilidade para chegar à grande solução.

Um módulo de procedimento (sub-rotina) é um bloco de programa contendo início e fim, identificado por um nome, por meio do qual será referenciado em qualquer parte do programa principal ou do programa chamador da sub-rotina.

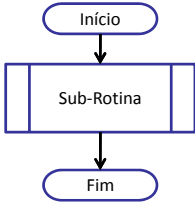
3

INFORMA, FORMA, TRANSFORMA.

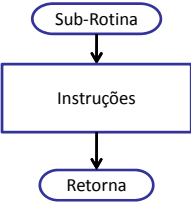
Diagramas de Bloco

As sub-rotinas serão representadas isoladas do programa principal.

a




b




O diagrama de bloco (a) representa o programa chamador do subprograma (veja o uso do símbolo *Processo Predefinido*). O diagrama de bloco (b) representa a ação do módulo de procedimento.

Atente para o uso dos símbolos *terminal* com as identificações início, fim, sub-rotina e retorna.

Obs: Na prática, o termo sub-rotina é substituído pelo seu nome identificador.

4



Português Estruturado (Pseudocódigo)

No pseudocódigo a estrutura fica da seguinte forma:


```
algoritmo "nome-do-algoritmo"  
  procedimento <sub-rotina>  
    var  
      <variáveis>  
  inicio  
    <instruções>  
  fimprocedimento  
inicio  
  <sub-rotina>  
finalgoritmo
```

As sub-rotinas são definidas antes do início do programa principal.

No programa principal, em determinado ponto, a sub-rotina é acionada.

Podem existir várias sub-rotinas no mesmo algoritmo. Quantas forem necessárias.

O programa principal pode acionar uma sub-rotina em diversos pontos diferentes, se for necessário.



5

Exemplo:

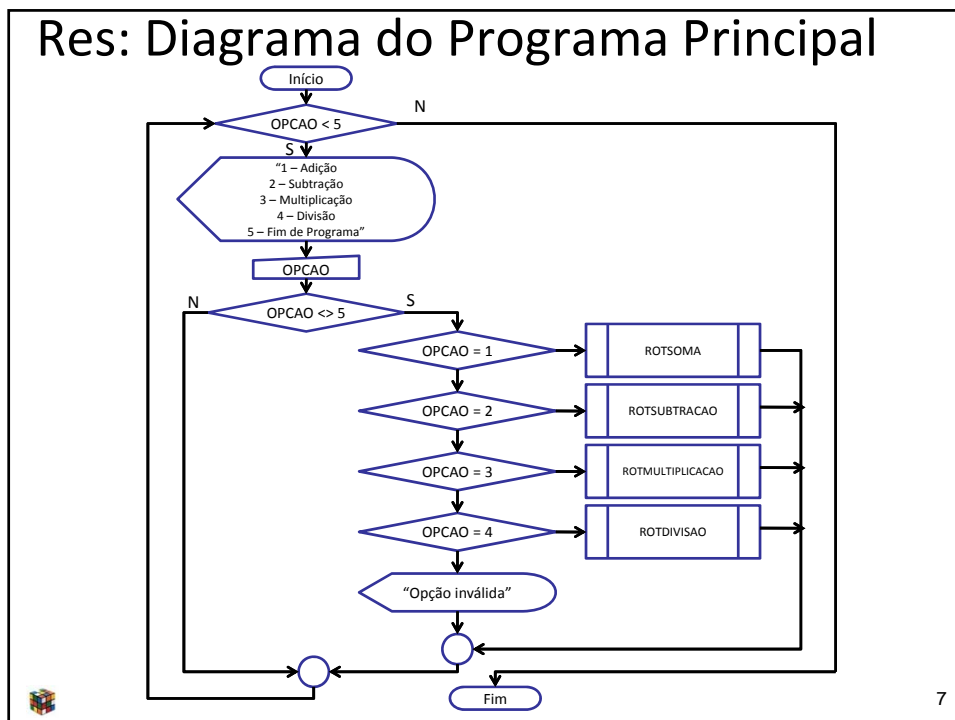
Desenvolver um programa que simule as operações básicas de uma calculadora, que opere com a entrada de dois valores do tipo real. O programa deve apresentar uma lista de opções (menu) com as operações disponíveis. Após a operação, o programa deve apresentar o resultado, e após o resultado, o programa deverá exibir novamente o menu.

Vamos resolver este exercício passo a passo.

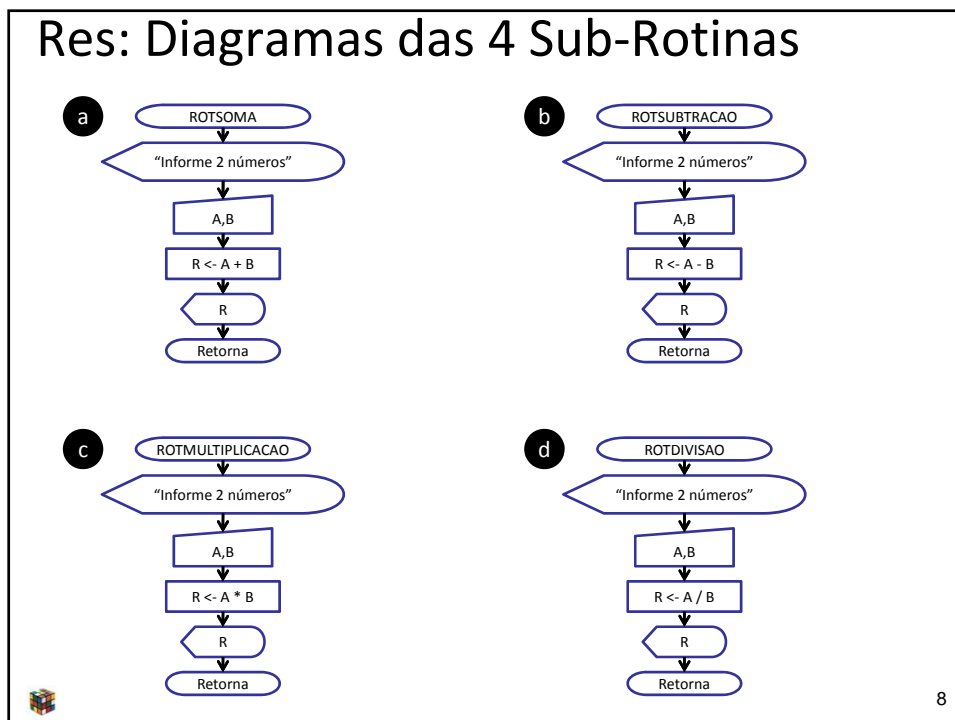
Observe que existem 4 rotinas dentro deste programa, que são as 4 operações básicas que a calculadora realiza. Deve-se incluir uma outra rotina, que é a principal, que aciona uma das 4 rotinas secundárias, de acordo com a opção que o usuário escolher. A hierarquia das rotinas deste algoritmo fica assim:



Res: Diagrama do Programa Principal



Res: Diagramas das 4 Sub-Rotinas



Res: Pseudocódigo Completo

a

```
algoritmo "SUB-ROTINAS"
procedimento ROTSOMA
var
  R, A, B: real
inicio
  escreval ("Informe 2 números")
  leia (A,B)
  R <- A + B
  escreval (R)
fimprocedimento
procedimento ROTSUBTRACAO
var
  R, A, B: real
inicio
  escreval ("Informe 2 números")
  leia (A,B)
  R <- A - B
  escreval (R)
fimprocedimento
procedimento ROTMULTIPLICACAO
var
  R, A, B: real
inicio
  escreval ("Informe 2 números")
  leia (A,B)
  R <- A * B
  escreval (R)
fimprocedimento
procedimento ROTDIVISAO
var
  R, A, B: real
inicio
  escreval ("Informe 2 números")
  leia (A,B)
  R <- A / B
  escreval (R)
fimprocedimento
```


b

c



d

Programa Principal

```
var
  opcao: inteiro
inicio
  // Seção de Comandos
  enquanto OPCA0 < 5 faça
    escreval ("1 - Adição")
    escreval ("2 - Subtração")
    escreval ("3 - Multiplicação")
    escreval ("4 - Divisão")
    escreval ("5 - Fim de Programa")
    leia (OPCA0)
    Se OPCA0 <> 5 entao
      escolha OPCA0
      caso 1
        ROTSOMA
      caso 2
        ROTSUBTRACAO
      caso 3
        ROTMULTIPLICACAO
      caso 4
        ROTDIVISAO
      outrocaso
        escreval ("Opção inválida")
      fimsecolha
    fimse
  fimenquanto
finalgoritmo
```



9




INFORMA, FORMA, TRANSFORMA.

Variáveis Globais e Locais

Algo muito importante a ser observado ao utilizar programação com sub-rotinas do tipo procedimento é com relação às variáveis utilizadas.

Uma variável é considerada **Global** quando é declarada no início do programa principal. Esta variável poderá ser utilizada pelo programa principal e por qualquer sub-rotina subordinada ao programa principal.

Uma variável é considerada **Local** quando é declarada dentro de uma sub-rotina e é somente válida dentro da sub-rotina à qual está declarada. As demais sub-rotinas e o programa principal não visualizam sua existência.



10

Refinamento Sucessivo

É o ato de tentar gerar o máximo de sub-rotinas.

No pseudocódigo da calculadora, repare que cada sub-rotina possui a leitura das variáveis A e B.

Esta leitura poderia se tornar outra sub-rotina.

```
procedimento COLETADADOS
  inicio
    escreval ("Informe 2 números")
    leia (A,B)
  fimprocedimento
```

E cada sub-rotina apresentada anteriormente ficaria como esta abaixo:

```
procedimento ROTSOMA
  inicio
    COLETADADOS
    R <- A + B
    escreval (R)
  fimprocedimento
```

OBS: Como foi descrito no slide anterior, a declaração das variáveis R, A e B não poderá acontecer dentro do procedimento COLETADADOS e nem dentro do procedimento ROTSOMA, pois se isto for feito, as variáveis se tornarão LOCAIS, exclusivas do procedimento onde foram declaradas. Elas terão que ser declaradas dentro do programa principal, para que sejam GLOBAIS e com isso, possam ser utilizadas também pelos procedimentos.



11



Utilização de Parâmetros

Parâmetros têm por finalidade servir como um ponto de comunicação bidirecional entre uma sub-rotina e o programa principal, ou com uma outra sub-rotina hierarquicamente de nível mais alto.

Desta forma, é possível passar valores de uma sub-rotina ou rotina chamadora a outra sub-rotina e vice-versa.

Quando se utilizam variáveis locais em sub-rotinas, é necessário muitas vezes passar um valor dessa sub-rotina para outra, o que ocorre pela passagem de parâmetro que pode acontecer de duas formas:

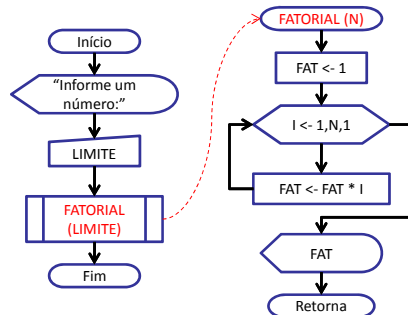
- 1) Passagem de Parâmetro por Valor
- 2) Passagem de Parâmetro por Referência



12

1) Passagem de Parâmetro por Valor

Observe o algoritmo abaixo:



```

algoritmo "PARAMETROS"
procedimento FATORIAL (N:inteiro)
var
    I, FAT: inteiro
inicio
    FAT <- 1
    para I de 1 ate N passo 1 faca
        FAT <- FAT * I
    fimpara
    Escreval (FAT)
fimprocedimento

var
    LIMITE: inteiro
inicio
    Escreval ("Informe um número:")
    Leia (LIMITE)
    FATORIAL (LIMITE)
fimalgoritmo

```

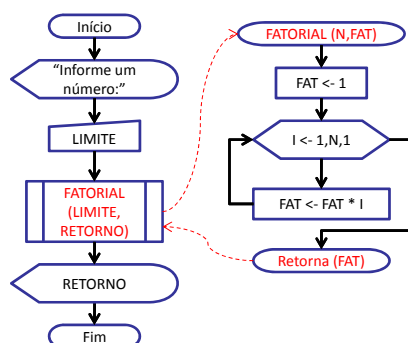
A seta vermelha pontilhada está chamando a atenção para o fato do conteúdo da variável **LIMITE** estar passando para a variável **N** quando se solicita a execução da sub-rotina pela linha **FATORIAL (LIMITE)**.

A sub-rotina **FATORIAL** recebe o valor passado pela variável **LIMITE** e executa o processamento e a apresentação do resultado obtido, através do comando **Escreval (FAT)**, que somente é válido dentro da sub-rotina, por isso fica “preso” nela.

13

2) Passagem de Parâmetro por Referência

Observe o algoritmo abaixo:



```

algoritmo "PARAMETROS"
procedimento FATORIAL (var N, FAT:inteiro)
var
    I: inteiro
inicio
    FAT <- 1
    para I de 1 ate N passo 1 faca
        FAT <- FAT * I
    fimpara
fimprocedimento

var
    LIMITE, RETORNO: inteiro
inicio
    Escreval ("Informe um número:")
    Leia (LIMITE)
    FATORIAL (LIMITE, RETORNO)
    Escreval (RETORNO)
fimalgoritmo

```

O comando **var** precisa existir aqui

As 2 setas vermelhas pontilhadas indicam que o conteúdo da variável **LIMITE** passa para a variável **N** quando se solicita a execução da sub-rotina pela linha **FATORIAL (LIMITE)**, e o resultado disto que é acumulado na variável **FAT** e retorna ao programa principal na variável **RETORNO**.

Na passagem de parâmetro por referência, o valor não fica “preso” na sub-rotina, pois ele retorna ao programa principal.


14

 INFORMA, FORMA, TRANSFORMA.




Dúvidas?



15

 INFORMA, FORMA, TRANSFORMA.

Bibliografia

	<p>Estudo Dirigido de Algoritmos José Augusto N. G. Manzano e Jayr Figueiredo de Oliveira Ed. Érica</p>
	<p>Introdução aos Algoritmos Bruno Tonet e Cristian Koliver (Acompanha o software VisuAlg)</p>
	<p>A Linguagem de Programação do VisuAlg (Acompanha o software VisuAlg)</p>

16