

ALGORITMOS E PROGRAMAÇÃO DE COMPUTADORES

AULA 5

Subprograma - funções e procedimentos

Vandor Roberto Vilardi Rissoli



APRESENTAÇÃO

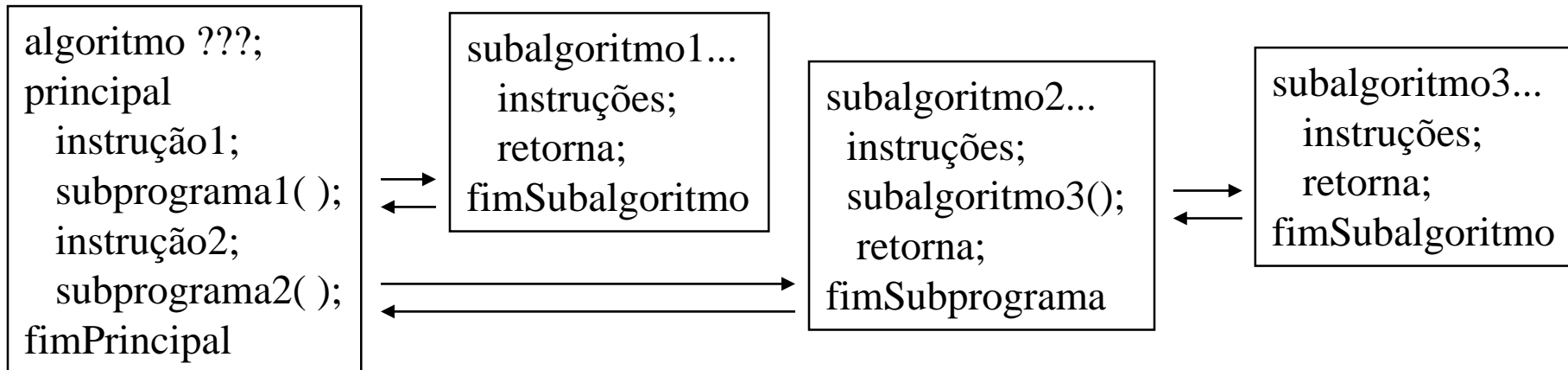
- Programação Modular
- Subprograma ou sub-algoritmo
 - Funções
 - Procedimentos
- Passagem de Parâmetros
- Fluxograma
- Referências



Programação Modular

A identificação de problemas mais complexos resultará no desenvolvimento de algoritmos também mais complexos para resolve-los.

Uma abordagem eficiente para este tipo de problema é a divisão do problema mais complexo em problemas mais simples de serem resolvidos. Este método é conhecido como modularização, onde um grande problema é dividido em problemas menores e a solução destes problemas menores resultará na solução do problema maior (mais complexo).



Programação Modular

Estes programas menores são denominados sub-rotinas, subprogramas ou sub-algoritmos.

Um sub-algoritmo é um trecho de um algoritmo maior (mais complexo) que realiza qualquer operação computacional (entrada, processamento, saída). Ele efetua parte de uma tarefa que um algoritmo maior (algoritmo principal) deverá executar.

SUB-ALGORITMO (características)

- Tarefa bem definida
- Reaproveitamento do código (módulos)
- Execução em diversas situações
- Acionado quantas vezes forem necessárias

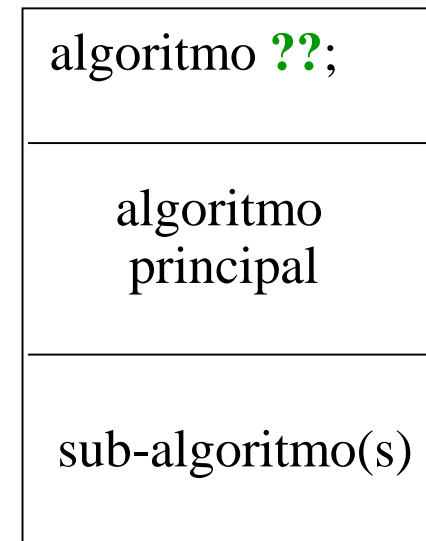


Programação Modular

Um algoritmo possui uma estrutura bem organizada, onde sub-algoritmos executam tarefas bem específicas. Esta divisão facilita o desenvolvimento, a legibilidade e a sua manutenção corretiva e/ou evolutiva.

Um sub-algoritmo só pode ser acionado a partir do algoritmo que o define (cria), podendo este ainda possuir outros sub-algoritmos.

A estrutura de um algoritmo com sub-algoritmo pode ser assim representada:



Sub-Algoritmo

- O algoritmo que aciona um sub-algoritmo é chamado de algoritmo chamador, ou algoritmo acionador daquele sub-algoritmo
- O funcionamento de um sub-algoritmo corresponde a implementação de uma função, no sentido matemático da expressão, pois um sub-algoritmo também retorna um único valor, de um ou mais valores manipulados em uma ou mais operações realizadas (seno por exemplo)
- A expressão “**funcao**” também é usada na elaboração de um sub-algoritmo, onde ela identifica a disponibilização de um ou mais sub-algoritmos para um algoritmo principal, sendo uma palavra reservada em algoritmo.



Sub-Algoritmo

FUNÇÃO

É um sub-algoritmo iniciado pela palavra reservada **funcao**, seguida por um tipo de dado válido (escalar), referente ao retorno desta função, e seu identificador que fornece o nome à função, além de uma lista de parâmetros

Esta lista de parâmetros consiste nos valores que o acionador possui, mas que são necessários ao sub-algoritmo, para geração dos dados desejados no algoritmo principal.

O acionamento de qualquer sub-algoritmo ocorre por meio da especificação do seu nome, seguido da lista de parâmetros, por qualquer parte do algoritmo ou sub-algoritmo.

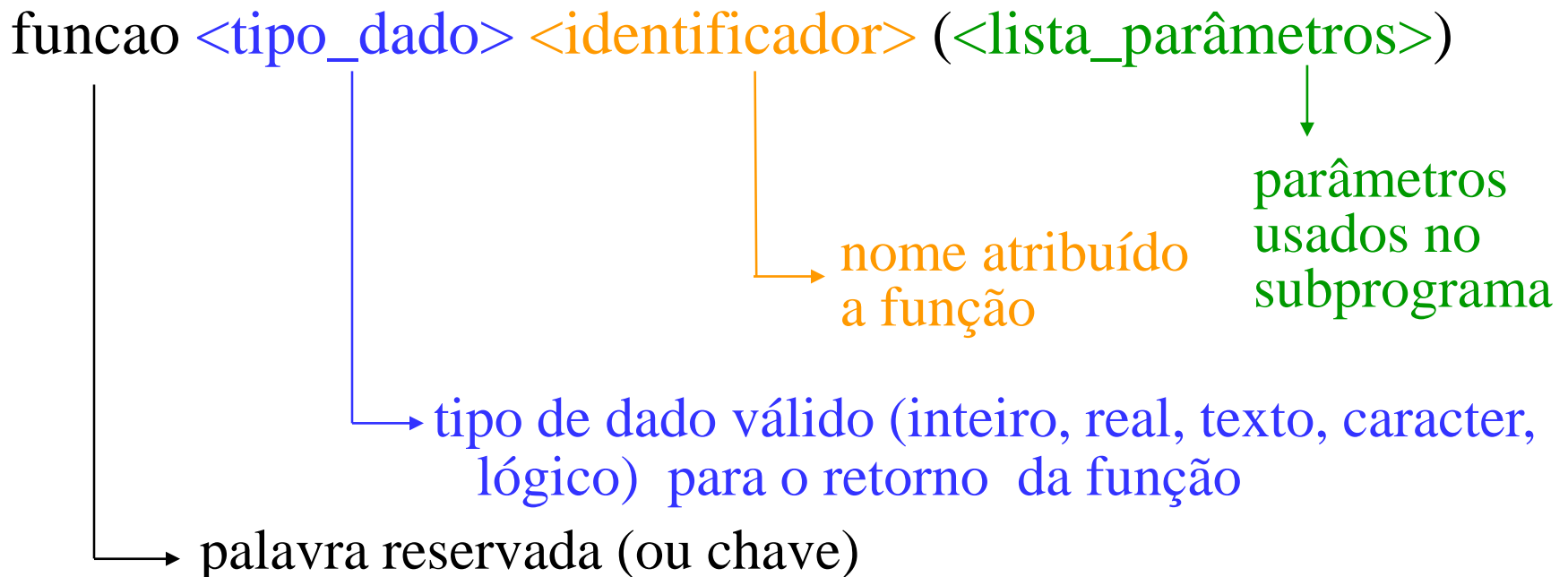
Esta lista é opcional, pois pode não existir nenhum valor no acionador que necessário ao sub-algoritmo.

Sub-Algoritmo

FORMA GERAL DA FUNÇÃO (português estruturado)

Uma função possui dois momentos no algoritmo:

- **primeiro** momento identifica o que ela faz (seu corpo)
- **segundo** momento é o de utilização (em várias situações)



Sub-Algoritmo

INSTRUÇÃO DE RETORNO

Um sub-algoritmo (função) sempre retorna um valor ao algoritmo que o aciona, por isso ele possui um tipo de dado declarado, retornando ao acionador um dado deste tipo.

Exemplo:

```
funcao real calculaMedia (real valor1, real valor2)
    // Declarações
    real media;
    // Instruções
    media = (valor1 + valor2) / 2;
    retorna media;
fimFuncao
```

- A palavra reservada (ou chave) **retorna** identifica qual valor será retornado ao algoritmo acionador do sub-algoritmo.
- Uma vez executado um retorno o sub-algoritmo encerra suas ações e volta para a próxima instrução do acionador.

EXEMPLO - Sub-Algoritmo

algoritmo mediaComFuncao;

// Síntese

// Objetivo: calcular a média aritmética
// para cada estudante

// Entrada: duas notas por estudante

// Saída: média de cada estudante

principal

// Declarações

real nota1, nota2, resultado;
inteiro qtde, contador;

// Instruções

escreva("Digite quantos estudantes: ");
leia(qtde);

limpaTela(); → **limpa toda a janela de execução**

para (contador de 1 ate qtde passo 1) faca

escreval("Estudante = ", contador);

escreva("Nota 1 = ");

leia(nota1);

escreva("Nota 2 = ");

leia(nota2);

resultado= **calculaMedia(nota1,nota2);**

escreval("Media = ", resultado:3:1);
escreval(""); →

fimPara

fimPrincipal

só salta uma linha na janela de execução

// Subprograma

// Objetivo : obter a média de 2 números

// Parâmetros : dois números

// Retorno : média aritmética

funcao **real** **calculaMedia** (**real** **valor1**,
real **valor2**)

// Declarações locais

real media;

// Instruções

media = (**valor1** + **valor2**) / 2;

retorna media;

fimFuncao

Sub-Algoritmo

ASPECTOS DE EXECUÇÃO DOS SUB-ALGORITMOS

Um algoritmo em execução deve obedecer a sequência das instruções, a menos que existam comandos (instruções) específicos que alterem esta ordem sequencial.

Porém, um acionamento de um sub-algoritmo desloca a execução do algoritmo em execução para o corpo da função, que será executada obedecendo a sequência lógica de execução, respeitando as características de cada instrução existente.

Quando uma função for encerrada, ela retornará um único valor para a instrução que a acionou, executando assim a próxima instrução do algoritmo ou sub-algoritmo acionador.



Sub-Algoritmo

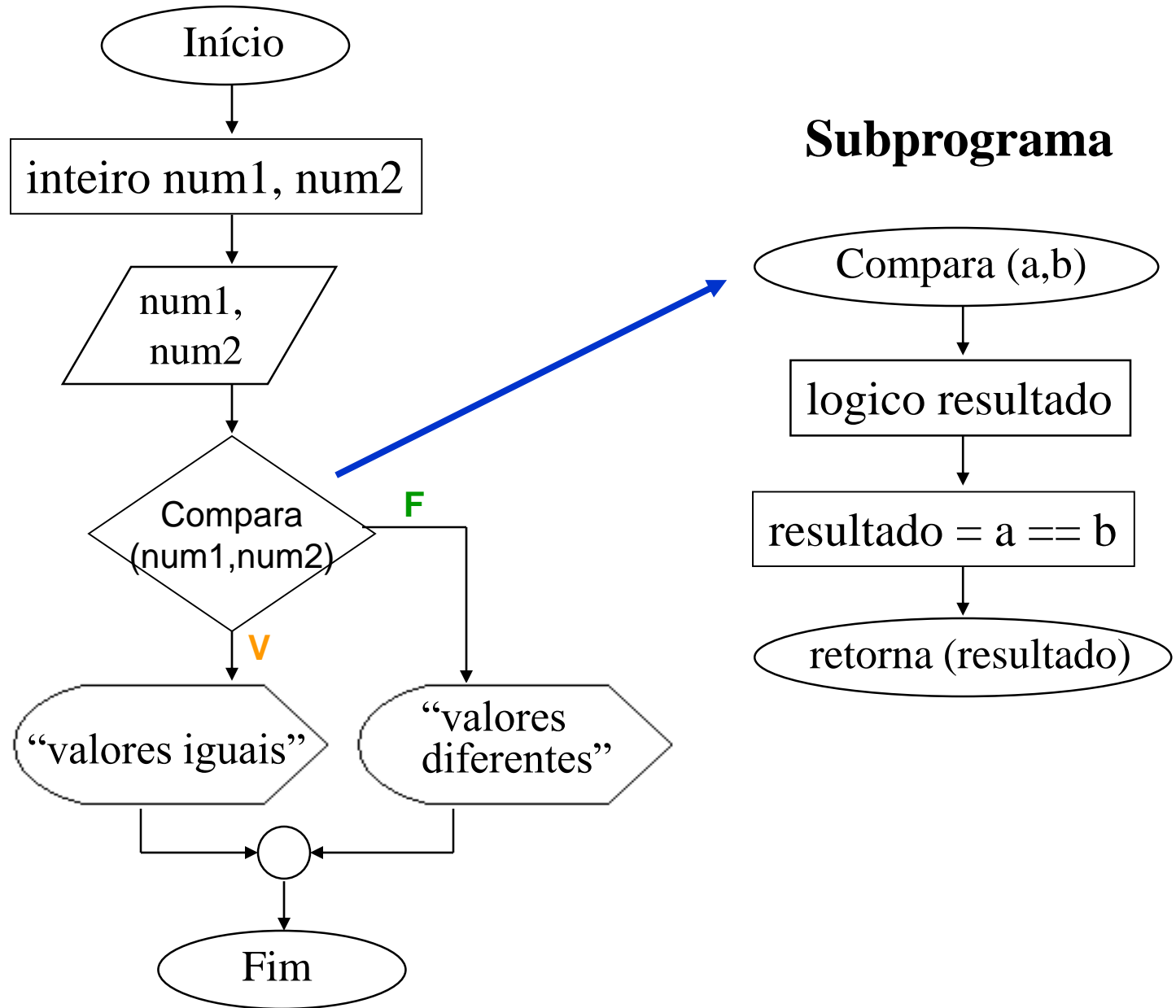
Um algoritmo possui uma estrutura bem organizada, onde subprogramas executam tarefas bem determinadas. Esta divisão facilita a legibilidade e a manutenção corretiva e/ou evolutiva do algoritmo.

Um subprograma só pode ser acionado pelo algoritmo que o contém, podendo este possuir outros subprogramas.

Todo programa com subprograma deve possuir uma estrutura similar a figura abaixo:

algoritmo "??"
declarações
subprograma(s)
algoritmo principal

Fluxograma - função



Fluxograma

- Utilizando o nome do sub-algoritmo e seus parâmetros, se houverem, o local do acionamento do sub-algoritmo é identificado, deslocando a sua execução para um outro fluxograma (fluxograma do sub-algoritmo).
- Este outro fluxograma deverá ser elaborado, representando exatamente o que este sub-algoritmo executa, ou seja, as instruções existentes no corpo do sub-algoritmo.
- As elipses de início e fim do algoritmo principal ou acionador são inseridas da mesma forma, porém, no diagrama que especifica o corpo da função, a elipse inicial é substituída pelo nome do sub-algoritmo e sua lista de parâmetros, enquanto que a elipse final contém a palavra reservada retorna e a informação que será retornada.

Sub-Algoritmo

OBJETOS LOCAIS E GLOBAIS

Aplicando o processo de modularização, os recursos (variáveis, constantes ou subprogramas) que podem ser criados e manipulados por um algoritmo também passam a ter novas características.

Estas novas características permitem que os recursos sejam classificados em LOCAIS e GLOBAIS.

Exemplo:

- Uma variável é considerada GLOBAL quando sua declaração acontece no algoritmo principal, enquanto que uma variável é considerada LOCAL quando a sua declaração acontece em um sub-algoritmo
- Os recursos LOCAIS pertencem ao **escopo** do sub-algoritmo

Sub-Algoritmo

RECURSO GLOBAL

- Declaração feita no algoritmo principal
- Também chamado de escopo global ou recurso público

RECURSO LOCAL

- Declaração feita no sub-algoritmo de um algoritmo principal
- Somente o sub-algoritmo que declara o recurso pode acessá-lo

→ Os recursos declarados no algoritmo principal não devem ser usados por nenhum subprograma, somente se tiverem seus valores passados por **parâmetros formais**


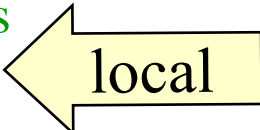


SUB-ALGORITMO

- O correto uso dos recursos resulta na economia de memória, tornando o algoritmo um posterior programa mais eficiente.

Exemplo:

```
algoritmo soma;  
// Síntese  
// Objetivo: realizar soma de  
// dois números  
// Entrada: dois números  
// Saída: resultado da soma  
principal  
// Declarações  
real nro1, nro2, soma;  
// Instruções  
escreva("Número 1: ");  
leia(nro1);  
escreva("Número 2 = ");  
leia(nro2);  
soma = nro1 + nro2;  
escreva("Soma = ", soma);  
fimPrincipal
```

```
algoritmo somaComFuncao;  
// Síntese  
// Objetivo: realizar soma de dois números  
// Entrada: dois números  
// Saída: resultado da soma  
principal  
// Declarações  
real nro1, nro2, soma;   
// Instruções  
escreva("Informe dois números: ");  
leia(nro1, nro2);  
soma = fazSoma(nro1, nro2);  
escreva("Soma = ", soma:2:1);  
fimPrincipal  
  
// Subprograma (elaborar a Síntese local)  
funcao real fazSoma(real nro1, real nro2)  
// Declarações locais  
real total;   
// Instruções  
total = nro1 + nro2;  
retorna total;  
fimFuncao
```

Passagem de Parâmetros

A criação de sub-algoritmos nem sempre significa que o desenvolvimento do algoritmo irá diminuir o trabalho do programador.

Exemplo:

Suponha a leitura de 3 variáveis diferentes. A criação de 3 sub-algoritmos, um para cada leitura de variável, poderia vir a resolver o problema, mas e se fossem 100 ou 1000 variáveis diferentes? Desenvolver 100 ou 1000 sub-algoritmos seria uma tarefa trabalhosa para o programador.



Passagem de Parâmetros

CARACTERÍSTICAS DOS SUBPROGRAMAS

- Reutilização de código (aproveitamento do sub-algoritmo)
- Sub-algoritmos devem ser genéricos o bastante para se adaptarem as diversas situações, visando justamente essa reutilização
- Por isso foi criado o conceito de passagem de parâmetros, ou seja, passar dados ou informações para serem usadas ou tratadas dentro do sub-algoritmo.
- A passagem de parâmetros é classificada de duas formas:
 - Passagem **por valor**
 - Passagem **por referência**



Passagem de Parâmetros

PASSAGEM POR VALOR

- O sub-algoritmo cria uma variável que recebe a cópia do valor existente no algoritmo chamador
- Não altera o valor original existente no chamador, somente o valor local do sub-algoritmo
- O valor existente no sub-algoritmo não pode ser acessado pelo chamador, a menos que seja retornado para o algoritmo acionador (conhecido também como chamador)
- Com a finalização do sub-algoritmo o valor do parâmetro passado por valor é perdido pela destruição do parâmetro que existe na memória

Passagem de Parâmetros

PASSAGEM POR REFERÊNCIA

- O sub-algoritmo cria uma variável que usará o valor original existente no algoritmo acionador (na memória)
- A manipulação desse tipo de parâmetro pode alterar o valor existente no algoritmo acionador (valor original)
- Com a finalização do sub-algoritmo o valor do parâmetro passado por referência é destruído (não se perde)
- A expressão usada na declaração de parâmetros, é utilizada para identificar a passagem por referência

Exemplo:

funcao inteiro calcular(*ref* inteiro valor)

Sub-Algoritmo - comparativo

algoritmo sub_algoritmos;

// Síntese

// Objetivo: realizar soma e/ou multiplicação de um inteiro com um real

// Entrada: dois números e a opção de realização das operações

// Saída: resultado da operação ou das operações

principal

// Declarações

inteiro nro1;

real nro2, multiplicacao, resultado ;

caracter opcao, tecla;

// Instruções

faca

limpaTela();

escreva("Número inteiro: ");

leia(nro1);

escreva("Número real: ");

leia(nro2);

escreval(" S - soma número inteiro e real");

escreval(" M - multiplica número inteiro pelo numero real");

escreval(" T - soma e multiplica número inteiro e real");

escreval(" E - encerra algoritmo");

Sub-Algoritmo

leiaCaracter(opcao);

opcao = maiusculoCaracter(opcao);

escolha(opcao)

caso 'S'

resultado = calculaSoma(nro1,nro2);

escreval("Número1= ", nro1, " Número2= ", nro2);

escreval("soma= ", resultado);

interrompa;

caso 'M'

resultado = calculaMulti(nro1,nro2);

escreval("Número1= ", nro1, " Número2= ", nro2);

escreval("Multiplicação= ", resultado);

interrompa;

caso 'T'

resultado = calculaOperacoes(nro1,nro2,multiplicacao);

escreval("Número1 = ",nro1," Número2 = ", nro2);

escreval("Soma = ", resultado);

escreval("Multiplicação = ", multiplicacao);

interrompa;

Sub-Algoritmo

caso 'E'

escreval("Encerrar algoritmo");

interrompa;

outroCaso

escreval("Opção inválida");

fimEscolha

escreval("Pressione qualquer tecla para continuar");

leiaCaracter(tecla);

enquanto (opcao != 'E');

fimPrincipal

// Subprogramas

// Objetivo : calcular a soma de 2 númros

// Parâmetros: dois números

// Retorno : soma dos 2 números

funcao real calculaSoma(**inteiro valor1, real valor2**) → **passagem por valor**

// Declarações locais

real total;

// Instruções

total = valor1 + valor2;

retorna total;

fimFuncao

Sub-Algoritmo

// Objetivo : calcular a multiplicação de 2 números

// Parâmetros: dois números

// Retorno : multiplicação dos números

→ passagem por valor

funcao real calculaMulti(**inteiro valor1**, **real valor2**)

// Instruções

retorna (valor1 * valor2);

fimFuncao

// Objetivo : calcular a soma e multiplicação de 2 números

// Parâmetros: dois números, resultado da multiplicação (referência)

// Retorno : soma de 2 números

funcao real calculaOperacoes(**inteiro valor1**, **real valor2** , **ref real vezes**)

// Declarações locais

passagem por valor

passagem por
referência

 real total;

// Instruções

 vezes = valor1 * valor2;

 total = valor1 + valor2;

retorna total;

fimFuncao

Sub-Algoritmo

SÍNTESE DO SUB-ALGORITMO

O lógica bem definida de um sub-algoritmo deve ser representada, primeiramente, em sua síntese, assim como acontece na construção do algoritmo.

Exemplo:

// Objetivo: calcular média de dois números

// Parâmetros: dois números

// Retorno : média

funcao **real** **calcularMedia**(**real** valor1, **real** valor2)

// Instrução

retorna (valor1 + valor2) / 2 ;

fin

Está Síntese é específica a cada sub-algoritmo e tem que ser elaborada para cada um destes que venham a existir em sua solução.

- A lógica de soma aritmética de dois números pode ser usada para calcular a média das alturas, de distâncias, dos salários, etc.

Exercícios de Fixação

- 1) Faça a leitura da altura de **até** cinco pessoas e apresente a média aritmética entre elas, calculada por um sub-algoritmo que receberá só dados por parâmetros.
- 2) Faça um algoritmo que calcule, por meio de uma função, o valor de x elevado a n , onde x e n podem ser somente inteiros negativos, positivos ou zero.
- 3) Desenvolva um algoritmo que armazene as duas notas de 20 alunos e calcule as possíveis médias finais para cada um deles, usando um sub-algoritmo diferente para cada possível cálculo de média abaixo e apresente os resultados para cada um dos alunos em cada média, recebendo dados só como parâmetro.
 - a) Média aritmética das duas notas de cada aluno
 - b) Média ponderada com peso 4 e 6 respectivamente
 - c) Média ponderada com peso 3 e 7 respectivamente

Exercícios de Fixação

- 4) Solicite a um professor a quantidade de alunos matriculados em sua disciplina, a quantidade de aprovados e reprovados no final do semestre. Apresente, por meio da função denominada *fazPorcentagem*, as porcentagens de alunos reprovados e aprovados no semestre corrente. Mostre ao final também a quantidade total de alunos analisados.
- 5) Elabore duas funções, além do algoritmo principal, que calcule duas operações aritméticas consecutivas sobre três valores inteiros e duas operações inteiras aritméticas, lidas no algoritmo principal. Na outra função verifique se o resultado final das operações produziram um número par ou ímpar. Use parâmetros sempre.

Exemplo: $5 + 3 * 2 = ?$



Exercícios de Fixação

- 6) Elabore um algoritmo que armazene o peso de 5 pessoas e depois organize este armazenamento de forma que os valores sejam armazenados e apresentados em ordem crescente de peso. Esta ordenação deverá ser feita e apresentada por um sub-algoritmo que só poderá trabalhar com parâmetros na manipulação dos pesos.



Procedimento - sem retorno

A realização de uma atividade por um sub-algoritmo pode não ter a necessidade de retornar um valor para o acionador (sub-algoritmo denominado **procedimento**).

Nestas situações o sub-algoritmo desenvolve a tarefa, definida no seu corpo, satisfazendo a necessidade existente na resolução do problema desejado.

O funcionamento deste tipo de sub-algoritmo respeita todas as normas da modularização, mas a sua declaração e desenvolvimento possuem algumas diferenças muito relevantes em seu uso e no seu corpo. Observe o exemplo:

palavra reservada ↵

↵ lista de parâmetros

procedimento calculaMedia(real nota1, real nota2)

identificador do procedimento ↵

Procedimento

EXEMPLO:

Suponha um algoritmo que deseje classificar todos os participantes de uma prova de corrida por idade, identificando as pessoas de maior e menor risco de problemas de saúde (coração, respiração, etc.), baseando-se na idade de cada participante.



Procedimento

algoritmo avaliacao_medica;

// Síntese

// Objetivo: classificar o risco médico

// Entrada: nome e idade

// Saída: descrição do risco

principal

// Declarações

inteiro idade;

texto nome;

// Instruções

faca

 escreva("Informe sua idade: ");

 leia(idade);

enquanto ((idade == 0) ou (idade > 150));

enquanto (idade > 0) faca

 faca

 escreva("Qual seu nome: ");

 leia(nome);

 enquanto(comparaTexto(nome, "") == 0);

 alerta(idade);

 escreval("");

faca

 escreval("Qual a nova idade: ");

 leia(idade);

 enquanto ((idade==0)ou (idade>150));

 fimEnquanto

fimPrincipal

// Subprograma

// Objetivo: classificar o risco médico

// Parâmetros : idade

// Retorno: nenhum

procedimento alerta (inteiro qtdeAnos)

 se (qtdeAnos > 35) entao

 escreval(nome,"- Alto risco");

 senao

 escreval(nome,"- Baixo risco");

 fimSe

fimProcedimento

não existe retorno
ao principal

Sub-algoritmo com grave erro para
programação mais segura e correta

Procedimento

O desenvolvimento de subprogramas pode ser realizado de duas formas: *procedimento* ou *função*. As principais diferenças entre estes subprogramas são:

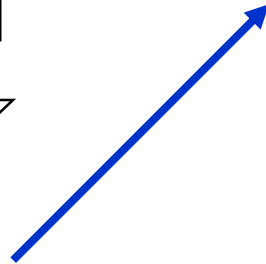
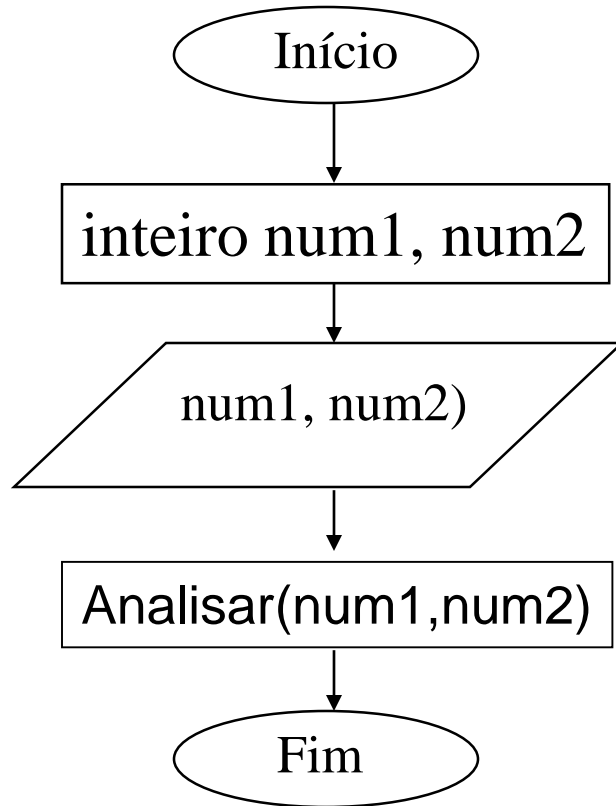
FUNÇÃO

- Sempre retorna um valor para o "acionador"
- Pode fornecer valor ao acionador pela passagem por referência
- É ativada por meio do seu acionamento através de seu nome

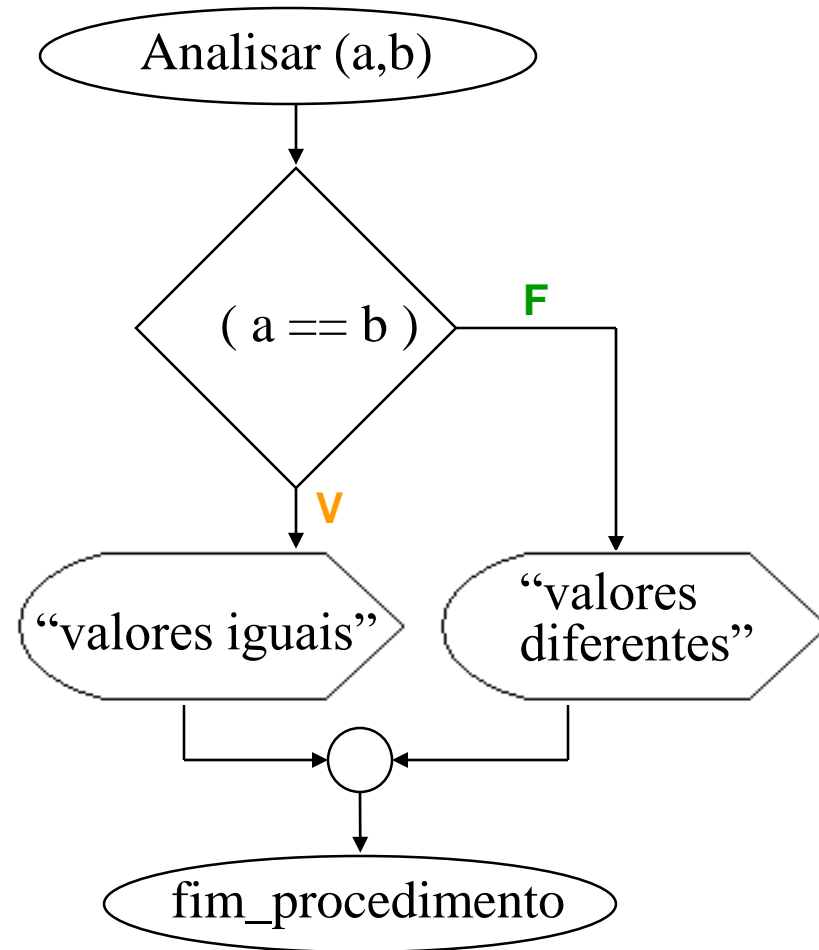
PROCEDIMENTO

- Não retorna nenhum valor para o "acionador"
- É ativado na avaliação (ou execução) de uma expressão que a contém, sendo acionada pelo seu nome
- É ativado por meio do seu comando de acionamento ou chamada através de seu nome

Fluxograma - procedimento



Subprograma



Exercícios de Fixação

- 7) Desenvolva um algoritmo que leia um peso em quilogramas (kg) e apresente, por meio de outros dois sub-algoritmos (**fazGrama**, **fazTonelada**), o valor destes pesos informados em gramas e em toneladas respectivamente. Valide este peso na **validaPeso**.
- 8) Faça um algoritmo que classifique o tamanho de uma organização populacional respeitando a escala a seguir:

Classificação	DE	Até
VILA	1	5000 hab.
DISTRITO	5001	15000 hab.
CIDADE	15001	1000000
METROPOLE	1000001	∞

Apresente por meio de um sub-algoritmo o nome e a classificação para cidade informada. As cidades deverão ser classificadas enquanto o usuário desejar.

Exercícios de Fixação

- 9) Elabore um algoritmo que solicite ao usuário o seu ano de nascimento e o ano atual. Em seguida, ele acionará um procedimento, denominado *calculaIdade*, que calculará a idade provável, em anos, deste usuário, além de acionar uma função, chamada *verificaNivel*, que identificará qual o nível da idade deste usuário, classificando-o em:

IDADE \leq 10	→ CRIANÇA
10 < IDADE < 18	→ ADOLESCENTE
18 \leq IDADE < 25	→ JOVEM
25 \leq IDADE \leq 58	→ ADULTO
IDADE > 58	→ IDOSO

Esta classificação só será apresentada, juntamente com a idade calculada, no procedimento que será acionado enquanto o usuário desejar informar idades.

Referências de Criação para Apoio ao Estudo

Material para Consulta e Apoio ao Conteúdo

- FARRER, H. *et al.* Algoritmos Estruturados, Editora LTC, 3ª . edição, 1999.
 - Capítulo 3
- MANZANO, J.; OLIVEIRA, J. Algoritmos, Lógica para desenvolvimento de programação, Editora Ética, 1996.
 - Capítulo 10, 11, 12 e 13
- Universidade de Brasília (UnB Gama)
 - <https://sae.unb.br/cae/conteudo/unbfga>
(escolha a disciplina **Algoritmo Prog Computadores**)

