Lógica de Programação Orientada a Objetos

Aula 04

- Modularização
- Métodos
- Assinatura
- Recursividade

Modularização

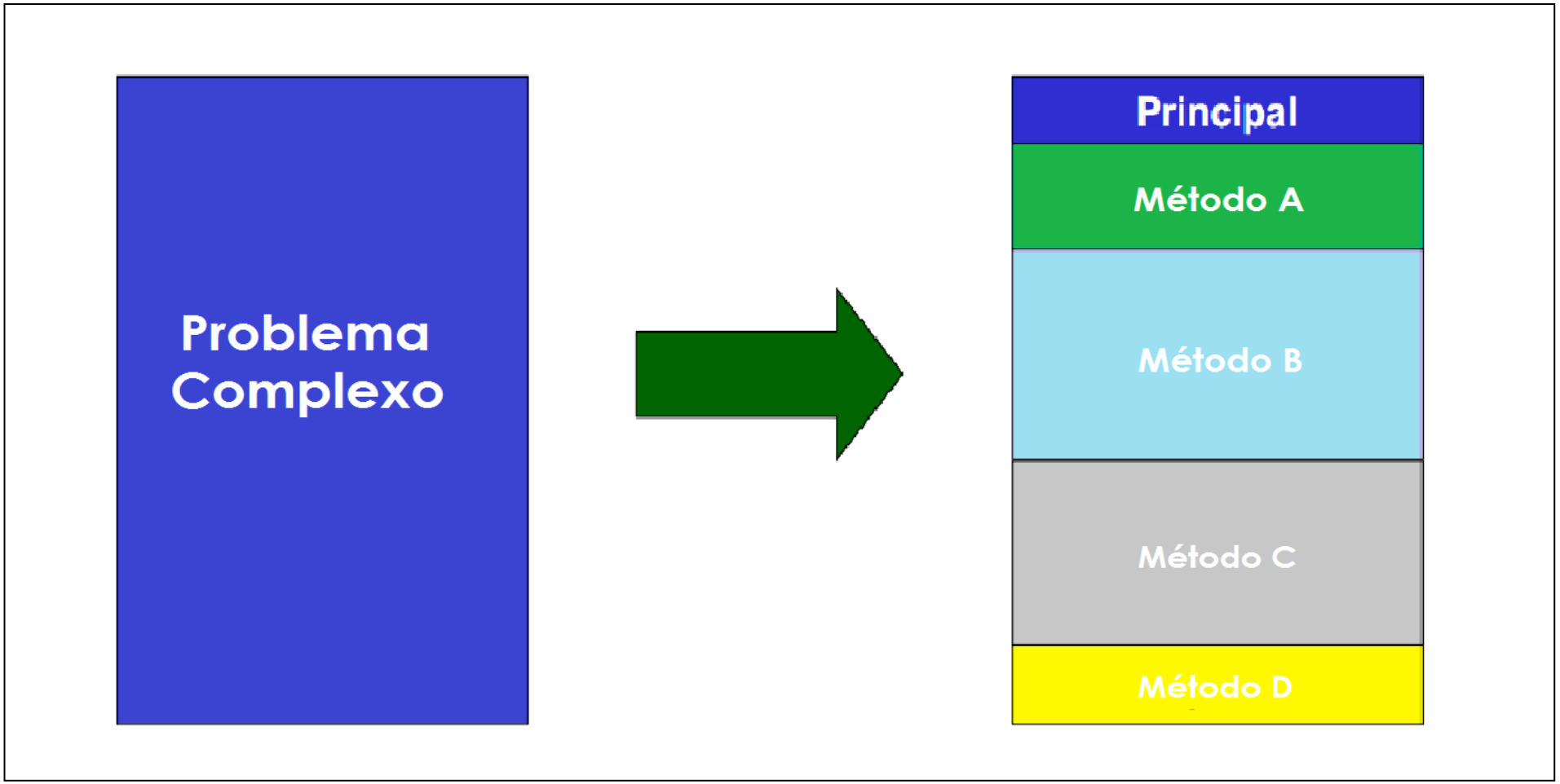
Modularização

- Em problemas complexos passamos a ter uma série de pequenos probleminhas.

- **Modularizar** é dividir o problema em pequenas partes, cada uma com seu papel bem definido na execução do algoritmo.

- A estas pequenas partes damos o nome de **métodos**.

Modularização



Métodos

Métodos

- Desvio provisório do fluxo de execução do algoritmo principal.

Um método na construção de um algoritmo pode ser de dois tipos:

- **função**
- **procedimento**

Métodos

Função

Uma função sempre retorna um valor !

---

funcao nome_da_funcao (lista_de_parâmetros) : tipo_de_retorno

var

{ variáveis_locais }

inicio

{ instruções_da_função }

fimfuncao

Métodos

Função

Exemplo:

*Uma função **sempre** deve ser definido **antes** do algoritmo principal !*

```
funcao  somarVariaveis ( x, y : inteiro ) : inteiro

var

    soma : inteiro

inicio

    soma ← x + y

    retorne   soma

fimfuncao
```

Métodos

Função

Exemplo:

```
algoritmo "RealizarSoma"

var

    soma : inteiro

Inicio

    soma ← somarVariaveis( 10, 5 )

    escreva ( soma )

fimalgoritmo
```

Saída: 15

Métodos

Procedimento

Um procedimento nunca retorna valor !

```
procedimento  nome_do_procedimento ( lista_de_parâmetros )
var
    { variáveis_locais }
inicio
    { instruções_do_procedimento }
fimprocedimento
```

Métodos

Procedimento

Exemplo:

*Um procedimento **sempre** deve ser definido **antes** do algoritmo principal !*

```
procedimento  somarVariaveis ( x, y : inteiro ; w : real )

var

    soma : real

inicio

    soma ← x + y + w

    escreval ( soma )

fimprocedimento
```

Métodos

Procedimento

Exemplo:

```
algoritmo "RealizarSoma"

var

Inicio

    somarVariaveis( 10 , 5 , 2.5 )

    escreva ( "Fim do algoritmo" )

fimalgoritmo
```

Saída:

```
17.5
Fim do algoritmo
```

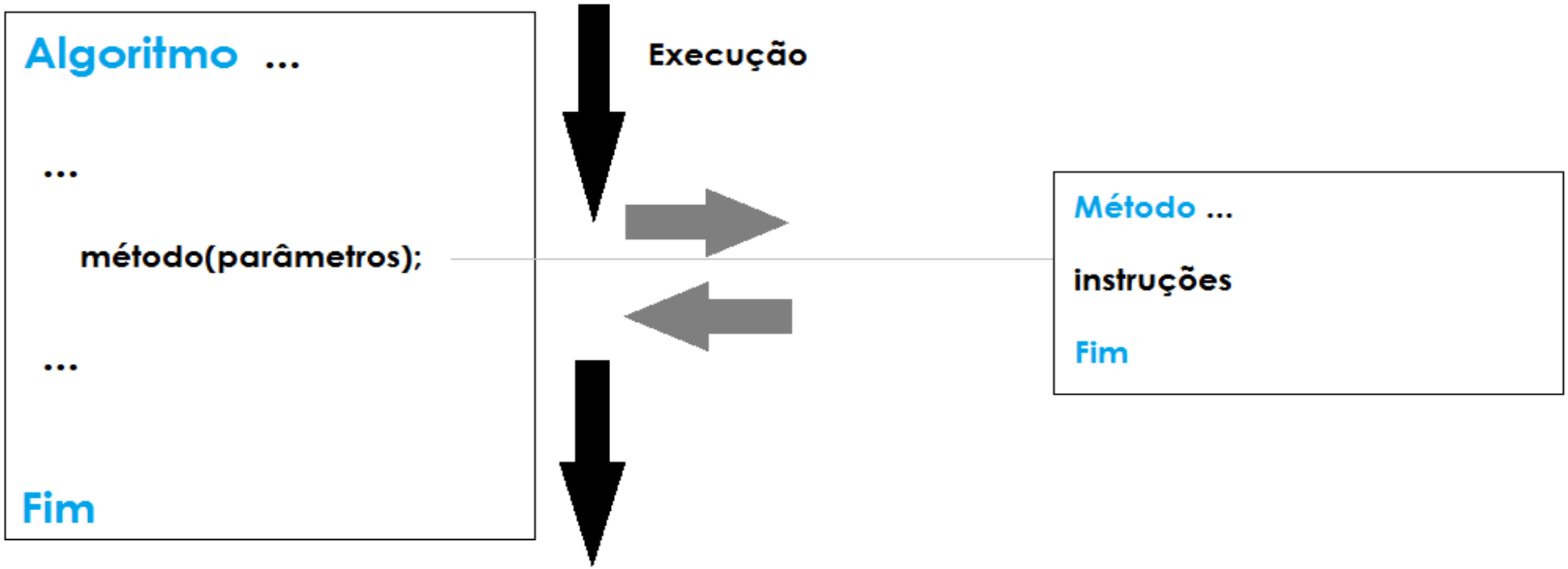
www.3way.com.br

Métodos

- As variáveis declaradas em um método só podem ser usadas dentro dele. Dizemos então que a variável tem escopo local.
- As variáveis declaradas no algoritmo principal, podem ser acessadas por qualquer método. Dizemos então que a variável tem escopo global.
- Um método pode receber nenhum, um ou mais parâmetros.
- Na invocação de um método, a lista de parâmetros enviada deve satisfazer a lista de parâmetros declarados pelo método.

Métodos

Fluxo de execução de um algoritmo modularizado:



Métodos

Considere o seguinte algoritmo:

```
funcao  somarVariaveis ( x, y : inteiro ) : inteiro

var


inicio

    retorne  x + y

fimfuncao
```

www.3way.com.br

Métodos

```
funcao  somarVariaveis ( x, y : real ) : real

var


inicio

    retorne  x + y

fimfuncao
```


Métodos

```
algoritmo " SomandoVariaveis "
var
    soma : real
inicio
    soma ← somarVariaveis ( 10.3 , 10.7 )
    escreva ( soma )
fimalgoritmo
```


Métodos

? Qual método será executado visto que os dois tem o mesmo nome ?

? Como saber qual método será invocado ?

A assinatura do método é que indica ao algoritmo qual é exatamente o método que ele deve invocar.

Assinatura

Assinatura

- A assinatura de um método garante que não haja dois métodos iguais no algoritmo.

- Os elementos do método que fazem parte da sua assinatura são:

  → nome do método.

  → quantidade de parâmetros declarados.

  → tipo de cada parâmetro.

  → ordem desses parâmetros.

Método

Assinatura

O tipo de retorno do método não faz parte da sua assinatura.

Exemplos:

somaVariaveis (literal: num1, literal: num2) : inteiro ;

somaVariaveis (inteiro: num1, real: num2) : real ;

somaVariaveis (real: num1, inteiro: num2) : real ;

imprimeVariavel (inteiro: a) ;

imprimeVariavel (literal: a) ;

Obs: *Todas as declarações acima são de métodos distintos.*

Recursividade

Recursividade

- Técnica aplicada a métodos, para que este possa invocar ele mesmo.

- Um método que invoca ele mesmo é chamado de **método recursivo.**

- Todo cuidado é pouco ao se construir métodos recursivos. Um método recursivo deve seguir duas regras básicas:

    1º **Ter uma condição de parada.**

    2º **Tornar a solução mais simples**.

Recursividade

Exemplo:

Sem Recursividade

```
funcao  fatorial ( n : inteiro ) : inteiro

var

    fatorial, cont : inteiro

inicio

   fatorial ← 1
```

www.3way.com.br

Recursividade

Exemplo:

Sem Recursividade

```
    para cont de 1 ate n faca
        fatorial ← fatorial * cont
    fimpara
    retorne fatorial
fimfuncao
```

Recursividade

Exemplo:

Com Recursividade

```
funcao  fatorial ( n : inteiro ) : inteiro

var

inicio

    se  n = 1  entao

        retorne 1

    senao
```

www.3way.com.br

Recursividade

Exemplo:

Com Recursividade

```
        retorne  (  n *  fatorial ( n – 1 )  )
    fimse

fimalgoritmo
```

Recursividade

Representação Fatorial:

N = 4

```
4! = 4 * 3!
        3! = 3 * 2!
                2! = 2 * 1!
                        1! = 1 * 0!
                                0! = 1
                        1! = 1
                2! = 2
        3! = 6
4! = 24
```

Chamadas recursivas ao método.

Condição de parada

Retorno das chamadas recursivas.

Run Jeliot
Recursividade

Laboratório 04