Módulos: Procedimentos e funções

Algoritmos e Estruturas de Dados

Prof. Darlon Vasata

Decomposição

Uma das grandes dificuldades presentes no desenvolvimento de algoritmos é como lidar com a complexidade dos problemas. Problemas grandes ou repetitivos podem ser decompostos em pequenos problemas, com partes exclusivas para a resolução de determinadas partes do problema.

A decomposição de códigos implica na construção de módulos, que podem ser chamados durante a execução do algoritmo.

Modularização

A construção de um **módulo** trata do desenvolvimento de um sub-algoritmo, este especializado em resolver determinado problema. É como se refinássemos um problema, e a partir das pequenas soluções chegássemos a uma solução final para todo o problema. Outra vantagem da modularização é a **reutilização de código**, dado que quando o mesmo sub-problema aparece diversas vezes, não é necessário reescrever a solução para o sub-problema diversas vezes, assim reaproveitando soluções já desenvolvidas.

É comum a categorização de módulos em **procedimentos** e **funções**. Nos procedimentos, há a mera execução de trechos de código, enquanto nas funções um valor é **retornado** na linha de código que executou a chamada à determinada função.

É uma boa prática utilizar **verbos no infinitivo** nos nomes das funções e procedimentos, denotando as ações que são realizadas no módulo. Exemplos: somarValores, calcularMedia, preencherFormulario, apresentarValores, lerInteiro, lerReal, entradaAbrir, entradaFechar, etc.

Procedimentos

Um procedimento é um conjunto de linhas de um algoritmo que são agrupados, e quando o procedimento é chamado, todas as linhas são executadas.

Exemplo
Tome como exemplo um sistema em que no início do algoritmo é mostrado um cabeçalho e ao final é mostrado um rodapé, estes compostos por uma sequência de `"="`. O algoritmo consiste em perguntar o nome do usuário e apresentar seu nome na tela, junto com a mensagem "Boa noite".

Uma forma de solução para o problema seria:

```
pubic static void Main(String[] args){

    //variáveis
    String nome;

    // highlight-next-line
    System.out.println("##########");

    //entrada e saída
    entradaAbrir();
    System.out.println("Nome: ");
    nome = entrada.next();
    System.out.printf("Boa noite, %s\n", nome);
    entradaFechar()

    // highlight-next-line
    System.out.println("##########");
}
```

No exemplo observe que há linha que se repete. Esta linha (ou mais que uma linha) poderia ser parte de um procedimento.

```
// highlight-start
public static void mostrarLinhas(){
    System.out.println("##########");
}
// highlight-end

pubic static void Main(String[] args){
```

```
    //variáveis
    String nome;

    // highlight-next-line
    mostrarLinhas();

    //entrada e saída
    entradaAbrir();
    System.out.println("Nome: ");
    nome = entrada.next();
    System.out.printf("Boa noite, %s\n", nome);
    entradaFechar();

    // highlight-next-line
    mostrarLinhas();

    scanner.close();
}
```

Atividade - O que deveria ser feito para modificar o cabeçalho, utilizando *? - É necessário realizar esta modificação em quantas linhas?

As principais vantagens da modularização são: - O mesmo trecho de código pode ser executado mais de uma vez, diminuindo a quantidade de código escrito; - Caso seja necessário atualizar esse trecho de código, este é modificado em apenas um lugar, a atualização será válida para todo o algoritmo; - O teste de código fica mais simplificado, dado que é possível testar trechos isolados de código;

Funções

Os procedimentos consistem na mera execução das linhas de código que estão contidas neles. Já as funções podem **retornar** um valor, ou seja, trazer um valor que foi calculado ou obtido dentro delas para dentro do algoritmo.

Exemplo

Considere um algoritmo que pergunta a três pessoas sua idade, calcula e *retorna* a idade média entre essas pessoas:

```
public static void main(String[] args){

  //variáveis
```

```
  int idade_p1, idade_p2, idade_p3;
  real idade_media;

  //entrada
  entradaAbrir();
  System.out.println("Idade da primeira pessoa: ");
  idade_p1 = entrada.nextInt();
  System.out.println("Idade da segunda pessoa: ");
  idade_p2 = entrada.nextInt();
  System.out.println("Idade da terceira pessoa: ");
  idade_p3 = entrada.nextInt();
  entradaFechar();

  //processamento
  idade_media = (idade_p1 + idade_p2 + idade_p3)/3;

  //saida
  System.out.printf("A idade média é %f\n", idade_media);
}
```

Poderia ser desenvolvida uma função que obtém as entradas e calcula e retorna a idade média.

```
public static void idadeMediaTresPessoas(){
  //variáveis
  int idade_p1, idade_p2, idade_p3;
  real media_idade;

  //entrada
  entradaAbrir();
  System.out.println("Idade da primeira pessoa: ");
  idade_p1 = entrada.nextInt();
  System.out.println("Idade da segunda pessoa: ");
  idade_p2 = entrada.nextInt();
  System.out.println("Idade da terceira pessoa: ");
  idade_p3 = entrada.nextInt();
  entradaFechar();

  //processamento
  media_idade = (idade_p1 + idade_p2 + idade_p3)/3;

  //retorno do valor calculado
```

```
  // highlight-next-line
  return(media_idade);

}

public static void main(String[] args){

  //variáveis
  real idade_media;

  //chamada ao módulo
  // highlight-next-line
  idade_media = idadeMediaTresPessoas();

  //saida
  System.out.printf("A idade média é %f\n", idade_media);
}
```

Java é uma linguagem de programação **orientada a objetos**. Aqui desenvolvemos uma função, porém no contexto de orientação a objetos esta atividade é chamada de **método**. Por ora utilizaremos o termo função e método como sinônimos, porém tenha em mente que estes possuem funcionalidades distintas, e suas diferenças e características serão abordadas futuramente.

Observe que nas funções existe o **retorno** do valor calculado. Essa é uma das principais características das funções.

Exercício

Considere um algoritmo que pergunta a altura e o peso de uma pessoa, e a partir disso calcula e retorna seu índice de massa corporal (IMC), de acordo com a seguinte fórmula:

$$imc = \frac{peso}{altura^2}$$

Parâmetros

Uma das possibilidades com os procedimentos e as funções é a passagem de parâmetros. Os parâmetros consistem nos conteúdos a serem passados para as variáveis que estão dentro das funções.

O uso dos parâmetros permite que o conteúdo da função seja realizado de acordo com os conteúdos das variáveis a ela passados.

Exemplo - Desenvolva uma função que retorna verdadeiro se o valor passado por parâmetro for par.

```
//módulos
public static boolean testarPar(int numero){
    boolean par;
    par = numero % 2 == 0;
    return par;
}

//main
public static void main(String[] args){
    //variáveis
    int: x;
    boolean valor_par;

    //entrada
    entradaAbrir();
    System.out.println("Entre com um número: ");
    leia(x);
    entradaFechar();

    //processamento e saída
    valor_par = testarPar(x);
    System.out.println("O número " + x + " é par? " + valor_par);
}
```

Atividade - Desenvolva uma função que retorna se um número é ímpar. - Desenvolva uma função que calcula a soma entre dois números. - Desenvolva uma função que calcula a soma entre três números.

Escopo de variáveis

Chamamos de **escopo** ao espaço em que determinada variável pode ser utilizada. Comumente, o escopo pode ser **local** ou **global**.

Variáveis locais

Uma variável é considerada **local** quando esta existe apenas em um módulo específico.

Exemplo

```java
public static float media3Valores(int a, int b, int c){
  //variáveis
  int soma;
  float media;

  //processamento
  soma = a+b+c;
  media = soma/3;

  //retorno
  return media;
}

public static void main(String[] args){
  //variáveis e entrada
  int x,y,z;
  System.out.println("Entre com três valores: ");
  Scanner entrada = new Scanner(System.in);
  x = entrada.nextInt(); y = entrada.nextInt(); z = entrada.nextInt();

  //chamada ao método e saída
  System.out.printf("A média é %f\n", media3Valores(x,y,z));
}
```

Observe que as variáveis `a`, `b`, `c`, `soma` e `media` existem **apenas** dentro do módulo `media3valores`. Com isto, dizemos que o **escopo** destas variáveis é **local** à função `media3Valores`.

Acessar estas variáveis fora de seu escopo ocasiona em um erro.

Exemplo

```java
public static float media3Valores(int a, int b, int c){
  //variáveis
  int soma;
  float media;

  //processamento
  soma = a+b+c;
  media = soma/3;

  //retorno
  return media;
```

```
}

public static void main(String[] args){
  //variáveis e entrada
  int x,y,z;
  System.out.println("Entre com três valores: ");
  Scanner entrada = new Scanner(System.in);
  x = entrada.nextInt(); y = entrada.nextInt(); z = entrada.nextInt();

  //chamada ao método e saída
  System.out.printf("A média é %f\n", media3Valores(x,y,z));

  //erro aqui
  //highlight-next-line
  System.out.printf("A soma dos valores é ", soma);//soma não existe neste escopo
}
```

Em escopos diferentes, variáveis locais podem utilizar os mesmos nomes. Como estão em escopos diferentes são **variáveis diferentes**.

Exemplo

```
  public static void mostraValorMaisDez(int x){
      int v; //variável v local à função
      //highlight-next-line
      v = x+10;  //um valor é atribuído a v dentro da função
      System.out.printf("V dentro do módulo: %d\n", v);
  }

  public static void main(String[] args){
      //variáveis
      int v; //variável v local ao código principal
      //highlight-next-line
      v = 10; //um valor é atribuído a v fora da função
      System.out.printf("V fora do módulo: %d\n", v); //mostra o valor 10
      mostraValorMaisDez(v);                          //mostra o valor 20
      System.out.printf("V fora do módulo: %d\n", v); //mostra o valor 10. O valor original
  }
```

Variáveis globais

Aa variáveis **globais** são aquelas que podem ser acessadas por todos, e portanto, são **compartilhadas** entre todos os módulos.

Para definir uma variável como global, basta declará-la fora do módulo.

```
System.out.println("Olá Mundo");
```