LÓGICA DE PROGRAMAÇÃO

Técnico Integrado em
INFORMÁTICA

Professor Jefferson Chaves
jefferson.chaves@ifc-araquari.edu.br

OBJETIVOS DA AULA

- Definição de funções;
  - Quais problemas as funções resolvem;
- Anatomia de uma função;
- Passagens de parâmetros;
- Retorno de funções;

LÓGICA DE PROGRAMAÇÃO | FUNÇÕES OU PROCEDIMENTOS

Livro desenvolvendo websites com PHP
Páginas 75 a 87

Jefferson de Oliveira Chaves - IFC Araquari

FOCA NA PROGRAMAÇÃO!

- **Sempre** é possível dividir problemas grandes e complicados em problemas **menores** e de solução mais simples;

- A decomposição de um problema é fator determinante para a redução da sua complexidade;

- Um algoritmo que envolve um problema grande **pode ser dividido em um algoritmo principal** e em diversos subalgoritmos ou módulos (tantos quantos forem necessários ou convenientes).

- O algoritmo principal é aquele **por onde começa a execução**, e **chama**, eventualmente, os demais subalgoritmos.

- Subalgoritmo é um algoritmo que, geralmente, resolve um pequeno problema;

  - Geralmente está subordinado a um outro algoritmo que solicitará seu acionamento.

  - É possível que um subalgoritmo **chame outro** subalgoritmo.

- Critérios para orientar o processo de decomposição.
  - **Dividir o problema em suas partes principais.**
  - Analisar a divisão obtida para garantir coerência.
  - **Se alguma parte ainda permanecer complexa, subdividi-la mais.**
  - Analisar o resultado para garantir entendimento e coerência.

- Dividir problemas grandes em vários problemas menores, de baixa complexidade;
- Número pequeno de variáveis;
- Poucos caminhos de controle;
- **Reusabilidade;**
- **Solucionar uma única vez o problema.**

- Todo módulo é constituído por um sequência de comandos que operam sobre um conjunto de variáveis que podem ser globais ou locais:
  - **Variáveis globais:** São visíveis em todo o algoritmo, no entanto não podem ser vistas dentro do escopo local de uma função;
  - **Variáveis locais:** apenas são “visíveis” dentro da função onde foram declaradas. Elas não possuem significado fora desta função.

- Uma variável local é criada (alocada na memória) no momento em que o subalgoritmo que a define **é chamado**;

- Uma variável local é liberada da memória no momento em que o subalgoritmo que a define termina

- Uma variável local somente existe **(só pode ser utilizada)** dentro do subalgoritmo que a define;

- Caso um mesmo identificador (nome de variável) seja declarado em subalgoritmos distintos, esses identificadores são considerados distintos entre si (variáveis distintas)

- **Cuidado com as variáveis globais:** o uso de variáveis locais minimiza a ocorrência de “efeitos colaterais”;

- O programador pode definir e utilizar as variáveis que desejar em um subalgoritmo sem interferir com outros subalgoritmos;

LÓGICA DE PROGRAMAÇÃO | FUNÇÕES

```
//função retorna a soma
function soma($num1, $num2 ) {
        return $num1 + $num2;
}

function media($num1, $num2 ) {
        //usa a função soma() para calcular a média
        return soma($num1, $num2) / 2;
}

media(10,5);
```

PROGRAMAÇÃO DE COMPUTADORES

INCLUDE E REQUIRE

Jefferson de Oliveira Chaves

INCLUDE E REQUIRED

- Durante o desenvolvimento de uma aplicação PHP é comum nos depararmos com situações em que um mesmo texto ou **lógica se repete** em mais de uma página;
- Nesses casos, podemos utilizar os métodos: **include ou required**;

- O que essas funções fazem é que o código de um arquivo seja visível no arquivos onde está sendo incluído, requerido;

```
include   'arquivo.php';      // inclui um arquivo
required  'arquivo.php';      // requer um arquivo
```

- A principal diferença no uso desses dois métodos é que o método include produz um erro do tipo **warning** caso o arquivo não seja encontrado.
  - Esse tipo de erro pode ser "abafado" com @.
  - Isso significa que seu uso é recomendado em casos em que a falta de um arquivo não trará efeitos colaterais para a aplicação;

- Por outro lado o uso do **required** gera um E_COMPILE_ERROR o que encerra a execução da aplicação;
  - Esse método deve ser usado quando a falta de um arquivo afeta a lógica da aplicação como por exemplo um arquivo onde é verificada o saldo de um cliente.

INCLUDE E REQUIRED

- Existem ainda alguns casos em que um algoritmo não pode ser incluído ou requerido mais de uma vez.
- Nestes casos usamos as funções **include_once** e **require_once**.

- Cenário: em uma projeto teremos muitos arquivos que irão compor o nosso aplicativo;
- Funções que realizam **exatamente** o **mesmo algoritmo não devem estar duplicadas** em arquivos diferentes;
- Nesses casos lançamos uso do include ou require, de acordo com regra do projeto.

PROGRAMAÇÃO DE COMPUTADORES

INCLUDE E REQUIRED

```php
$nums = [2, 4, 6, 8];

function somaArray(array $nums){
    $soma = 0;
    for ($i=0; $i < count($nums); $i++){
        $soma = $soma + $nums[$i];
    }
    return $soma;
}

$resultado = somaArray($nums);
```

soma.php

```php
$produtos = [130.90, 8.50, 25.00];

function somaArray(array $nums){
    $soma = 0;
    for ($i=0; $i < count($nums); $i++){
        $soma = $soma + $nums[$i];
    }
    return $soma;
}

$total = somaArray($produtos);
```

carrinho.php

PROGRAMAÇÃO DE COMPUTADORES

INCLUDE E REQUIRED

Lib_funcoes.php

```php
include 'lib_funcoes.php';

$nums = [2, 4, 6, 8];

//função defina no arquivo incluido
$resultado = somaArray($nums);
```

soma.php

```php
include 'lib_funcoes.php';

$produtos = [130.90, 8.50, 25.00];

//função defina no arquivo incluido
$total = somaArray($produtos);
```

carrinho.php

Jefferson de Oliveira Chaves

PROGRAMAÇÃO DE COMPUTADORES | EXERCÍCIOS

1. Implemente um algoritmo que modularizado que a partir de um vetor de **N** inteiros, possibilite:

   a) A digitação de valores no vetor;

   b) Imprimir o somatório de seus elementos;

   c) Imprimir a média de seus elementos;

   d) Imprimir a média e o somatório

   e) Substituir por zero todos os valores negativos e imprimir a média;

   f) Substituir por zero todos os valores repetidos e imprimir a média e o somatório;

2. Implemente um algoritmo que receba o nome e o preço de 5 produtos. Seu algoritmo deve permitir;

   a) A digitação de valores no vetor;

   b) Imprimir o somatório de seus elementos;

   c) Imprimir a média de seus elementos;

PROGRAMAÇÃO DE COMPUTADORES

REFINAMENTO SUCESSIVO

Jefferson de Oliveira Chaves

REFINAMENTO SUCESSIVO

- Afim de diminuir a complexidade podemos utilizar a técnica de refinamentos sucessivos;
- Alguns livros também chamam de refinamento *top-down;*

- A elaboração de cada função pode ser feita de forma independente, e em momentos distintos, permitindo **focar em um problema por vez**;
- Cada função pode ser **testada individualmente**, facilitando a identificação e correção de erros;
- Uma função pode ser reaproveitada diversas vezes no mesmo algoritmo ou em outros algoritmos;

PROGRAMAÇÃO DE COMPUTADORES | REFINAMENTO SUCESSIVO

- Construir um algoritmo que calcule os atrasos e as horas trabalhadas de um dado funcionário;

Dia	Manhã		Tarde	
	Entrada	Saída	Entrada	Saída
1				
2				
...				
29				
30				

- Após digitar os horários, a aplicação deve imprimir:
  - As horas de entrada e saídas de cada dia;
  - O total de horas trabalhadas e de atrasos de cada dia;
  - A média de horas trabalhadas e de atrasos no mês;

PROGRAMAÇÃO DE COMPUTADORES | REFINAMENTO SUCESSIVO

IMPORTANTE!

- Escopo de variáveis;
- Parametrização de funções;
- Complexidade e divisão de algoritmos;