Exercício prático 01
Gustavo Ribeiro de Figueiredo

1- Explique o motivo de existem tantas linguagens de programação, sendo que não existe uma linguagem geralmente considerada melhor que as outras.
Resposta → As inúmeras linguagens de programação que existem não implica em uma melhor pelo fato de que cada linguagem cumpre com um propósito. Seguindo esse raciocínio, cada linguagem se encaixa em um paradigma, e como consequência, cada uma se propõe a solucionar um problema diferente. Não existe a "melhor linguagem" porque não existe uma que resolva todos os problemas.

2- O tratamento de erros é algo que dificulta a escrita de programas. Quais são as vantagens de se usar manipulação de exceções?
Resposta → O tratamento de exceções permite a continuidade da execução do programa, mesmo após um erro que "quebraria a execução" aconteça. Além disso, esse erro pode ser salvo em um arquivo de log ou no próprio banco de dados, de forma que, posteriormente, possa ser analisado e corrigido pelos desenvolvedores.

3- Qual a diferença entre uma linguagem tipificação forte e uma com tipificação fraca. Cite um exemplo de linguagem de programação de cada categoria.
Resposta →

. Linguagens fortemente tipadas são aquelas que o tipo das variáveis deve ser informado antes da execução do programa, e esse tipo não muda ao longo da execução.

. Linguagens com tipificação fraca são aquelas que os tipos das variáveis não precisa ser informado para que o programa seja executado. Além disso, ao longo da execução, o próprio interpretador da linguagem define o tipo de dado daquela variável.

- Exemplos:
  . Tipagem fraca: PHP, Python, Javascript
  . Tipagem forte: C, C++, Java

* 4- Explique o que é o recurso de sobrecarga.

* 5- Subprogramas não são valores. Ainda assim, é útil que sejam passados como parâmetros ou armazenados em variáveis. Explique.

* 6- Algumas linguagens têm um recurso de tipos genéricos para criação de subprogramas genéricos ou estruturas de dados genéricas. Explique as situações em que tal recurso ajuda na programação.

7- Discuta a passagem de parâmetros de subprogramas em linguagens de programação, apresentando elementos de diferenciação entre as mesmas.
Resposta→A passagem de parâmetros para subprogramas pode ser feita de 2 formas. Por cópia ou por referência.
A passagem por cópia significa que dentro do subprograma, será criado uma cópia da variável que foi passada, a qual será alocada em um espaço de memória diferente. Dessa forma, qualquer alteração na variável cópia, não será refletido na variável externa.
A passagem por referência não cria uma cópia, é passado a própria variável da chamada, o mesmo endereço de memória. Portanto, quaisquer alterações nessa variável dentro do escopo da função, será refletido na variável da chamada.
Exemplo:

cópia: function calculaDesconto($compra);
referência: function calculaDesconto(& $compra);

8- Discuta sobrecarga de subprogramas em linguagens de programação, apresentando elementos de diferenciação entre as mesmas.
Resposta → Sobrecarga diz respeito a um recurso das linguagens de programação que permite comportamento (implementação) diferente em funções com o mesmo nome, baseado nos parâmetros que lhe foi passado. O exemplo a seguir indica isso:

function int calcula(int numero1, int numero2);
function float calcula(float numero1, float numero2);
No exemplo anterior, embora as duas funções possuam o mesmo nome, elas possuem retorno diferente, e portanto, suas implementações são distintas.

Vale ressaltar que o seguinte exemplo não é permitido:
function int calcula(int numero1, int numero2);
function float calcula(int numero1, int numero2);
Note que, como os parâmetros são iguais, não é possível realizar essa sobrecarga.

9-
Respostas → implementações realizadas em PHP
a)

```
<?php
function somaCincoPrimeirosNumerosInteirosPositivos() {
	$primeiroInteiro = 1;
	$segundoInteiro = 2;
	$terceiroInteiro = 3;
	$quartoInteiro = 4;
	$quintoInteiro = 5;

	$somaPrimeirosInteiros = $primeiroInteiro + $segundoInteiro + $terceiroInteiro + $quartoInteiro + $quintoInteiro;

	return $somaPrimeirosInteiros;
}
somaNumerosInteiros();
```

b)

```
<?php
function calculaMediaIdade() {
	$idadeSara = 23;
	$idadeMark = 19;
	$idadeFatima = 31;

	$somaIdades = $idadeSara + $idadeMark + $idadeFatima;
	$quantidadePessoas = 3;

	$mediaIdades = $ somaIdades / $quantidadePessoas;
	return $mediaIdades;
}
calculaMediaIdade();
```

c)
```
<?php
function calculaQuantidadePorcoes() {
	$numeroMaior = 403;
	$porcao = 73;

	$quantidadePorcoes = $numeroMaior/ $porcao;
	return intval($quantidadePorcoes);
}
calculaQuantidadePorcoes();
```

d)
```
<?php
function calculaResto() {
	$numero = 403;
	$divisor = 73;

	return $numero % $divisor;
}
calculaResto();
```

e)
```
<?php
function calculaPotencia() {
	$numero = 2;
	$expoente = 10;

	return $numero ** $expoente;
}
calculaPotencia();
```

f)
```
<?php
function calculaValorAbsoluto() {
	$numero1 = 54;
	$numero2 = 57;

	$diferenca = $numero1 - $numero2;

	return ($diferenca **2) **(1/2);

}
calculaValorAbsoluto();
```

10)
Respostas →
a) Fracamente Tipada
b)
c) Sim, o uso dos parênteses tornaria a leitura da expressão muito mais facilitada
$c = (x^2) * (y/4)$

d) Sim, a variável c na linha 6 receberia apenas a parte inteira da operação. Isso ocorreria em razão da variável b ser um inteiro;

11- Com relação ao uso de subprogramas, discuta as seguintes afirmações:
Respostas →
a) Sim, verdadeiro. É possível interromper a execução do programa principal dentro de um subprograma, caso um erro aconteça ou alguma condição seja satisfeita, por exemplo.
b) Sim, verdadeiro. O uso de subprogramas faz com que a leitura do programa como um todo seja facilitada. É essencial ter subprogramas o mais sucintos possível, de forma que tenham apenas uma responsabilidade, e seus nomes devem evidenciar exatamente o que será feito em seu interior.
c) Não, falso. O uso de subprogramas não afeta na confiabilidade do código, muito pelo contrário. Quanto melhor for a modularização, mais legível é, visto que cada bloco será responsável por uma coisa. Dessa maneira, o desenvolvedor tem mais facilidade para realizar manutenção e desenvolver novas funcionalidades.
d) Sim, verdadeiro. Cada subprograma possui o seu escopo, e esse escopo diz respeito ao ciclo de vida das variáveis.
e) Sim verdadeiro. Quando recebemos um parâmetro por referência, não é criada uma cópia, é a própria variável referenciada que é passada. Portanto ela pode ter seu valor alterado.

12) Cite pelo menos uma linguagem de programação de cada um dos paradigmas:
Respostas →
a) Imperativo: PHP, C++,
b) Funcional: Haskell
c) Lógico: Prolog
d) Orientado a Objetos: Java, PHP

13- Sobre os tipos de dados, discuta as seguintes afirmações:
Respostas →
a) Falso. Não é possível representar aproximações desses 2 tipos.
b)
c) Verdadeiro. Podemos pensar em um vetor como uma estrutura que armazena várias variáveis do mesmo tipo (pode variar de linguagem para linguagem) em um espaço alocado de memória.
d) Verdadeiro. Podemos pensar em uma string como um vetor de char. Dessa forma, cada letra da string é um char, ou seja, uma posição do vetor. Por isso é possível acessar uma posição da string, como é feito em um vetor. Exemplo:
string nome = “Paradigmas”;
nome[0] == ”P” //true
e) Falso. Na linguagem C, não existe o tipo primitivo Bool. Nela, o zero é tratado como false, e tudo que seja diferente de zero é true;

14)
Resposta →
a = (b == 0) ? (a – 1) : (a + 1);

15)
Resposta → antes da iteração pelas linhas do arquivo, poderia ter sido verificado se o arquivo foi aberto corretamente, de forma que não ocorressem erros dentro do loop.
Segestão:
arquivo = abrir(“c:\abc.txt”, “leitura”)
se !arquivo.existe => interromper_fluxo_de_leitura()
linhas = arquivo.ler_linhas()
2 para_cada linha em linhas

3 faça
4 numero = int(linha)
5 print(numero)
16)
Resposta → Pode ocorrer um loop infinito. Como a variável index não está sendo incrementada a cada iteração, a condição de parada nunca será satisfeita.

17)
Resposta → código implementado em PHP

```
<?php
function exibeNumerosMaioresQueMedia($vetorNumeros) {
	$somaNumeros = 0;
	$quantidadeNumeros = count($vetorNumeros);

	foreach($vetorNumeros as $numero) {
		$somaNumeros += $numero;
	}

	$media = $somaNumeros / $quantidadeNumeros;

	 foreach($vetorNumeros as $numero) {
		if($numero > $media) {
		echo $numero . ' | ';
	}
}
$lista = [3,6,23,67,-98,8,90,-8,0];
exibeNumerosMaioresQueMedia($lista);
```

18- Por que existem vários paradigmas computacionais? Liste os principais, apresentando suas características gerais e dê exemplos de aplicações para cada paradigma.
Resposta → A existência de vários paradigmas vem para oferecer diferentes propostas de solução para diferentes problemas. Dessa forma, certos paradigmas são mais indicados para certas situações. Principais paradigmas: imperativo, lógico, funcional, orientado a objetos.