QUESTÃO 1 (Objetiva) (Unidade 1): Paradigmas de Linguagens (1,0)

QUESTÃO 2 (Objetiva) (Unidade 1): Programação Declarativa (1,0)

QUESTÃO 3 (Objetiva) (Unidade 2): Orientação a Objeto (1,0)

QUESTÃO 4 (Verdadeiro-Falso) (Unidade 2): Orientação a Objeto (1,0)

QUESTÃO 5 (Objetiva) (Unidade 3): Paradigma Funcional (1,0)

QUESTÃO 6 (Discursiva) (Unidade 4): Linguagem JavaScript (1,0)

Toda a linguagem de programação está construída sobre um paradigma. Um paradigma representa um padrão de pensamento que guia um conjunto de atividades relacionadas, trata-se de um padrão que define um modelo para a resolução de problemas e regra, basica-mente, toda e qualquer linguagem de programação existente.

Os paradigmas de programação estão classificados em quatro diferentes tipos, que evoluíram ao longo das últimas décadas:

* programação interativa;
* programação colaborativa;
* programação lógica;
* programação de controle.

A resposta correta é 'Falso'.

O projeto de uma nova linguagem de programação é algo bem complexo, o projetista deve se preocupar com inúmeros desafios e adotar soluções es-pecíficas, que se proponham a atender esses desafios. Os principais desafios envolvidos no projeto de uma nova linguagem de programação são:

A resposta correta é:

Arquitetura, requisitos técnicos e padrões.

Os tipos de programação estão diretamente relacionados ao conceito do paradigma no qual a linguagem foi concebida. Sobre os tipos de paradigmas escolha o item correto para as definições abaixo:

1. Declarativo  : o foco não estar em como uma execução vai ocorrer, mas sim no resultado a ser atingido. Um dos melhores exemplos para entender esse paradigma são as instruções structured query language (SQL), pois nela são passados para o banco de dados apenas o que se pretende, sem a preocupação sobre como o banco de dados vai executar a instrução, o foco é somente o retorno ou resultado da consulta.

2. Imperativo  : Esse tipo de paradigma foi o primeiro que apresentou as linguagens de alto nível, que permitiam a utilização de um vocabulário mais próximo ao natural para construção de programas. Dar ordens para que a máquina execute as instruções dadas, e ela executará cada uma, passo a passo, com o propósito de chegar no resultado esperado.

3. Estruturado  : se refere à forma do programa e do processo de codificação. É um conjunto de convenções que o programador pode seguir para produzir o código, e suas regras de codificação impõem limitações sobre o uso das estruturas básicas de controle, estruturas de composição modular e documentação.

4. Orientada a Objetos  : surge como o advento da reutilização de código e a facilidade na manutenção, o princípio é a construção de código, implementando as entidades do mundo real por meio do conceito de classes que possuem relação entre si. Como o desempenho das aplicações não é uma das grandes preocupações na maioria delas (devido ao poder de processamento dos computadores atuais), se tornou muito difundida.

O paradigma de orientação a objetos surge como o advento da reutilização de código e a facilidade na manutenção. A programação orientada a objetos está embasada em quatro pilares.

Marque a opção correta:

A resposta correta é:

Abstração, encapsulamento, herança e polimorfismo

É praticamente impossível falar em orientação a objetos sem relacioná-la ao conceito de classes. Dessa forma, não é difícil imaginar a importância da construção de classes bem estruturadas, que consigam delimitar corretamente as fronteiras e os relacionamentos para, assim, realizar a distribuição correta das responsabilidades dentro de um sistema orientado a objetos.

O paradigma de programação orientada a objetos envolve a identificação e abstração de entidades, de acordo com o escopo de um sistema. Essas entidades formam o vocabulário do sistema.

**Esse texto acima é verdadeiro ou falso?**

A resposta correta é 'Verdadeiro'.

Pascal tem grande relevância na [programação estruturada](https://educacaoonline.unifametro.edu.br/mod/lti/view.php?id=386705), porque um de seus objetivos era justamente segundo Farrer et al. (1999, p. 35) “desenvolver uma linguagem de programação disciplinada de alto nível para ensinar [programação estruturada](https://educacaoonline.unifametro.edu.br/mod/lti/view.php?id=386705). Esta linguagem foi batizada com o nome de Pascal, em homenagem a Blaise Pascal, filósofo e matemático francês que viveu entre 1623 e 1662”.

Pascal obteve uma grande adesão, principalmente na comunidade acadêmica, sendo utilizada como linguagem de apresentação ao desenvolvimento de programas por universidades nas décadas seguintes. Todo programa construído em Pascal é basicamente subdividido em três partes distintas, escolha o item correto para as definições a seguir.

1. Corpo do programa  : é o local em que o programa propriamente está escrito, tem início com a instrução ***begin*** e é finalizado pela instrução ***end***, seguida do símbolo ponto **(.)**. O uso dessas instruções caracteriza o que chamamos de bloco.

2. Área de declarações  : é utilizada para validar o uso de qualquer tipo de identificador que não seja pré-definido, estando subdividida em sete subáreas: ***uses, label, const, type, var, procedure e  
function***, por exemplo, ***A, B, C: integer***.

3. Cabeçalho do programa  : é a área do código utilizada para se fazer a identificação do programa, é atribuído pela instrução ***program*** seguido de um nome, por exemplo, *program CALCULA\_AREA*.

Ao trabalhar com a orientação a objetos, percebemos que, assim como no mundo real, existe um número muito pequeno de classes que podem trabalhar sozinhas em qualquer sistema.

Ao trabalharmos com a definição e a modelagem de um sistema a ser construído em uma linguagem orientada a objetos, não é possível ater-se somente a identificar  
as classes que fazem parte do escopo desse sistema, mas também é preciso entender e modelar, os relacionamentos, ou seja, o modo como essas classes se relacionam entre si.

Segundo Tucker e Noonan (2009, p. 275), “existem três tipos de relaciona-mento especialmente importantes”, marque a alternativa correta:

A resposta correta é:

Dependências, generalizações e associações

Uma razão importante para o início do movimento pela programação estruturada foi a mudança no custo maior da computação de hardware para de software, o que, consequentemente, exigia uma maior preocupação com a construção e a manutenção de programas de computador.

**O texto acima é verdadeiro ou falso?**

A resposta correta é 'Verdadeiro'.

Funções de ordem superior são funções que permitem receber ou retornar outras funções como resposta. Exemplos de funções de ordem superior são: **map**, **filter** e **reduce**.

**A afirmação acima é verdadeira ou falsa?**

A resposta correta é 'Verdadeiro'.

Sintaticamente, o cálculo lambda possui uma linguagem simples que repre-senta problemas matemáticos por meio da combinação de funções e se baseia na abstração. Como consequência, sua semântica se torna natural, permitindo a  
fácil realização e solução de problemas computacionais.

Sobre a definição de cálculo lambda, escolha a alternativa correta:

A resposta correta é:

Consiste em uma função anônima que possibilita passar uma função como argumento de outra função.

Sobre as premissas de comportamento do paradigma funcional, escolha o item certo para as definições a seguir:

1. Tipo  : existem linguagens de programação funcional fortemente tipadas, ou seja, os possíveis erros são encontrados em tempo de compilação, o que elimina a necessidade de verificação de tipo em tempo de execução. Isso torna o programa mais rápido.

2. Funções  : a programação funcional é toda baseada em funções que dividem a estrutura do código em blocos semanticamente alinhados. Uma função determinada pode receber outras funções como argumento sem que isso afete externamente o escopo do projeto. Quando uma função recebe outra função como argumento, recebe o nome de função de alta ordem.

3. Imutabilidade  : a programação funcional lida basicamente com funções e, à medida que uma variável é alocada na memória com um determinado valor associado, esse valor permanece inalterado até o fim e em todas as partes do código nas quais for usado.

Embora saibamos que as funções de ordem superior permitem diversas funcionalidades dentro de um código, precisamos ter em mente que, muitas vezes, a escrita de uma função se torna extensa, o que pode dificultar o entendimento do desenvolvedor ou analista de projeto.

Sendo assim, os operadores servem para abstrair ainda mais o código, reduzindo informações e instruções de modo que o entendimento se torne ainda maior.

**O texto acima é verdadeiro ou falso?**

A resposta correta é 'Verdadeiro'.

Utilizando o operador <$> é possível declara o funtor fmap dentro de uma função. Essa utilização é chamada de modo infixo e é muito utilizada em funtores aplicativos.

Podemos observá-la no exemplo a seguir:

**Prelude (2\*) <$> (minha \_ funcao 50 5)**

**Just 20**

Basicamente, os funtores possibilitam que as funções sejam operadas, pois é por meio deles que o acesso aos contêineres ou rótulos que armazenam valores e comportamentos se torna computável, como vimos anteriormente.

Esse texto é verdadeiro ou falso?

A resposta correta é 'Verdadeiro'.

A linguagem de programação Haskell implementa, de forma algébrica, o conceito de monoides, buscando satisfazer o princípio de identidade. Esse princípio diz que sempre deve haver uma operação que permita que itens possam ser unidos e que esta operação não tenha efeito.

Marque o item correto:

A resposta correta é:

Para ser considerada um monóide, é necessário que a operação não afete o resultado da função.

O currying é uma técnica utilizada em programação funcional, que consiste basicamente em transformar uma função originalmente com múltiplos parâmetros em uma que aceite apenas um. Currying é uma técnica de transformação de uma função originalmente com múltiplos parâmetros em uma avaliação parcial dos argumentos.

Essa afirmação é verdadeira ou falsa?

A resposta correta é 'Verdadeiro'.