Universidade Federal do Rio Grande do Sul – UFRGS

Disciplina: Modelos de Linguagem de Programação

Prof. Dr. Jéferson Campos Nobre

Prova P1

Aluna: Letícia dos Santos

---

1. (1,5) Acerca de paradigmas de linguagens de programação, julgue os itens a seguir:

I. Linguagens procedurais facilitam a legibilidade e a documentação do software.

II. Elementos de linguagens funcionais têm sido introduzidos em linguagens do paradigma imperativo.

III. Linguagens orientadas a objeto permitem reduzir custos de desenvolvimento e manutenção.

Explique sua posição sobre os itens, utilizando fundamentos relacionados com a disciplina.

A afirmação I é verdadeira, pois esse paradigma é voltado a chamadas de procedimentos. Logo, os procedimentos podem ser nomeados de forma conveniente e reutilizados quando necessário. A separação em procedimentos facilita a documentação de cada um.

II também é verdade, porque linguagens muito utilizadas atualmente, como C e Python, permitem construções em cálculo lambda, que é a base do paradigma funcional. Nesses casos, o cálculo lamba poupa trabalho do programador abstraindo o como (foco do paradigma imperativo) o cálculo em si deve ser feito. Por exemplo, uma soma entre vetores que seria feita em c tradicionalmente com um laço pode ser descrita em uma linha com cálculo lambda.

III é falsa, pois os custos de projeto dependem de vários fatores, inclusive a adequação da linguagem ao problema. A grande diversidade de paradigmas de programação existe justamente para endereçar problemas diferentes da forma mais eficiente possível. A eficiência também considera reduzir custos. Um dos objetivos dessa cadeira é justamente conhecer vários paradigmas para utilizar o mais adequado para resolver o problema.

2. (1,5) Diferencie os paradigmas imperativo e declarativo. Cite e explique um(sub)modelo em cada um dos paradigmas. Quais são os formalismos computacionais considerados?

O paradigma imperativo utiliza comandos ou ações para mudar o estado do programa, ou seja, foca em como o algoritmo funciona. Se assemelha ao imperativo, que expressa ordem, da linguagem natural. Formalmente, é o paradigma das linguagens projetadas considerando a arquitetura de von Neumann. Submodelo: O paradigma procedural se baseia em chamadas a procedimentos que mudam o estado do programa.

Já o paradigma declarativo expressa a logica do resultado, o que o programa deve atingir dentro de seu domínio. Formalmente, além das características já citadas, deve existir uma clara correspondência do programa com a lógica matemática, não podem existir efeitos colaterais (interferências de outras partes do programa). Submodelo: O paradigma funcional estabelece que os programas são construídos a partir da aplicação e composição de funções. Sua base é o cálculo lambda.

3. (1,5) Ortogonalidade é uma característica importante para a análise de linguagens de programação, podendo inclusive influenciar outras características ou mesmo critérios. Como você explicaria o significado de tal característica? Cite e explique pelo menos dois elementos que poderiam influenciá-la de maneira positiva e outros dois que poderiam influenciá-la de maneira negativa em uma linguagem de programação qualquer. Ao invés de citar elementos, se desejar, apresente dois exemplos positivos e dois exemplos negativos de ortogonalidade em uma linguagem de programação qualquer.

Ortogonalidade é a capacidade usar construções primitivas em combinações arbitrárias e obter resultados consistentes, sem efeitos colaterais. Ou seja, cada construção não afeta as outras.

Influências positivas: A simplicidade de construções pode melhorar a ortogonalidade. Assim o programador pode construir uma solução para um problema complexo após aprender apenas o pequeno conjunto de construções da linguagem e como combiná-las como for necessário. Regras claras e rígidas sobre a combinação eliminam ambiguidades.

Influências negativas: Sobrecarga de operadores, por exemplo, quando '+' pode ser utilizado tanto para a soma de números quanto para concatenar listas. Exceções também prejudicam a ortogonalidade, por exemplo, um array não pode ser retornado por uma função em C.

4. (1,5) Indique se as seguintes expressões são válidas ou não em PROLOG. Caso não sejam, explique por quê. Caso sejam válidas, quando houver variáveis, indique quais associações são formadas.

I. `circulo(X, Y, R) = circulo(X1, Y1, R1).`
Válida. Associações: R1 = R e X1 = X e Y1 = Y.

II. `voo(O, Gramado) = voo('pelotas', Destino).`
Válida. Associações: Gramado = Destino e O = pelotas.

III. `palavra(carro)= carro.`
Inválida. O termo carro é diferente do fato de carro ser uma palavra, não é possível fazer uma associação. O correto seria palavra(carro) = palavra(X). para fazer a associação X = carro.

5. (1,0) Explique os tipos de formas funcionais composição, construção e aplica-se a todos.

Forma funcional, ou função de ordem superior, recebe outras funções como parâmetros e/ou tem como resultado uma função. Tipos:

- **Composição: a ordem superior(h) tem como parâmetro duas funções e uma será aplicada dentro da outra.**
    - **h ≡ f ° g significa que h (x) ≡ f (g (x))**
- **Aplica-se a todos: a ordem superior(α) recebe uma função(h). Se a ordem superior for aplicada em uma lista de argumentos, aplica-se a função a todos da lista.**
    - **Sendo h (x) ≡ x + 2, α(h, (0, 1, 2)) resulta em (2, 3, 4)**

6. (1,0) Explique dois exemplos de problemas que podem ocorrer na utilização de ponteiros.

Dois ponteiros apontarem para a mesma variável, a utilização de um para acessar a variável e modificá-la causa efeito colateral no outro.

Apontar para uma variável que já foi desalocada porque estava em uma função que não está mais na pilha, pois o fluxo de execução já deixou o seu escopo.

7. (1,0)Considere o trecho abaixo, indique onde as seguintes variáveis seriam armazenadas(heap, espaço livre, pilha, segmento de dados, segmento de código):

- variável ‘a’ (linha 1): **segmento de dados com o resultado do malloc no heap(monte)**
- variável ‘b’ (linha 8): **pilha.**
- variável ‘i’ (linha 3): **pilha.**
- variável ‘j’ (linha 4): **segmento de dados.**

8. (1,0) Analise o trecho de código seguinte, escrito em uma pseudolinguagem similar à linguagem C:

- O que seria impresso na tela caso a linguagem usasse escopo dinâmico:
  0 6 0 7
- O que seria impresso na tela caso a linguagem usasse escopo estático:
  0 10 0 11

CORREÇÃO:

Q1:1,5

Q2:1,0 - Explicar mais.

Q3:1,5

Q4:1,5

Q5:1,0

Q6:1,0

Q7:1,0

Q8:0,75 - 0,6,4,7