Exercícios - Linguagens de Programação

Para responder estes exercícios, utilize como refer6encia o livro texto da disciplina:

SEBESTA, Robert W. **Conceitos de linguagens de programação.** 4. ed. Porto Alegre: Bookman, 2002. 624 p. ISBN 9788573076080. (*pode ser qualquer edição mais recente*)

1. Explique, com um exemplo prático de código (em qualquer linguagem que você conheça), uma situação em que a sintaxe esteja correta, mas a semântica resulte em erro ou comportamento inesperado.

2. Compare a importância da sintaxe em linguagens como Python e C. Como a diferença na ênfase (indentação em Python vs. chaves em C) influencia a legibilidade e a propensão a erros?

3. Pegue um pequeno trecho de código que você mesmo escreveu em alguma tarefa (5 a 20 linhas). Identifique e descreva detalhadamente três diferenças entre erros/ambiguidade sintática e erros/ambiguidade semântica nesse trecho. Anexe o código original e um print/trecho do editor mostrando o local do erro (se houver).

4. Defina, com suas próprias palavras, o que significa abstração em linguagens de programação e dê um exemplo de recurso de linguagem que permite maior abstração.

5. Diferencie abstração de dados de abstração de controle. Dê um exemplo de cada, citando linguagens que você já usou.

6. Em relação à amarração (binding), explique o impacto de uma linguagem que adota amarração estática versus amarração dinâmica de tipos. Quais vantagens e riscos você percebe?

7. Pesquise uma linguagem que permita tipagem dinâmica e outra que use tipagem estática. Explique, com base em sua experiência, em qual delas é mais fácil cometer erros semânticos.

8. Compare duas versões de um mesmo programa: uma em linguagem tipada estaticamente e outra em linguagem tipada dinamicamente. Discuta como a presença/ausência de checagens estáticas afeta descoberta de bugs, refatoração e produtividade; inclua exemplos concretos de bugs detectados/prevenidos.

9. Linguagens como C permitem conversão implícita de tipos (coerção), enquanto outras (como Rust) exigem conversão explícita (casting). Discuta as consequências dessa diferença em termos de segurança e eficiência.

10. Proponha uma situação em que a tipagem fraca seja útil e outra em que ela possa causar problemas sérios no desenvolvimento de software.

11. Analise o papel dos tipos primitivos em linguagens de programação. Você considera essencial ter uma grande variedade de tipos básicos (como em C) ou poucos tipos genéricos (como em Python)? Justifique.

12. Projete um tipo abstrato de dados (por exemplo, uma fila ou mapa) e escreva duas implementações distintas (p.ex. array circular vs lista ligada). Para cada implementação:

a) Forneça a mesma interface/assinatura pública.
b) Mostre como diferentes decisões de binding (linkagem estática vs despacho dinâmico/virtual) afetam desempenho e extensibilidade.
c) Meça tempos/uso de memória em exemplos reais (incluir código de benchmark e saída) e discuta onde a abstração vazou (leakage) e por quê.

13. Imagine que uma nova linguagem de programação fosse criada com sintaxe idêntica à do Python, mas com semântica igual à do C. Que problemas você acha que surgiriam para os programadores?

14. Compare a forma como listas/arrays são tratados em pelo menos duas linguagens diferentes que você conhece. Como a diferença na amarração de tipos e na abstração de dados influencia a programação nesses casos?

15. Reflita: em sua opinião, o que é mais prejudicial ao iniciante em programação – erros de sintaxe ou erros semânticos? Justifique com base em exemplos concretos (preferencialmente de sua própria experiência).

Observações:

- ✓ O trabalho é individual.
- ✓ Em caso de detecção de plágio ou de uso irresponsável de IA, o trabalho receberá conceito zero.