47. [Sebesta, 2000] Como as instruções de declaração de tipo para variáveis simples afetam a legibilidade de uma linguagem, considerando que algumas linguagens não as exigem?

A influência das declarações de tipo explícitas na legibilidade de uma linguagem de programação constitui um tema central no debate entre tipagem estática (Java, C++, C#) e tipagem dinâmica (Python, JavaScript, Ruby). É importante ressaltar que a legibilidade não é uma métrica absoluta: ela varia conforme o contexto (tamanho do projeto, experiência da equipe e domínio de aplicação).

Em linguagens de tipagem estática, o programador declara o tipo da variável antes de seu uso. Essa prática traz benefícios significativos para a legibilidade, como autodocumentação, contratos claros e a detecção precoce de erros. Contudo, a verbosidade excessiva pode introduzir "ruído visual", especialmente em operações simples. No trecho em C++ apresentado a seguir, a assinatura da função deixa claro quais tipos de valores devem ser fornecidos, aumentando a segurança e a clareza estrutural.

```
// Tipos explícitos: 'preco' é double e 'quantidade' é int
double calcularTotal(double preco, int quantidade) {
    double total = preco * quantidade;
    return total;
}
```

Em linguagens de tipagem dinâmica, os tipos são inferidos em tempo de execução, dispensando declarações explícitas no código-fonte. Essa abordagem prioriza a concisão e a flexibilidade. Entretanto, essa liberdade pode comprometer a legibilidade em cenários complexos. No trecho em Python apresentado a seguir, embora visualmente mais limpo, o código não revela, apenas pela assinatura, como os parâmetros devem ser fornecidos, exigindo que o leitor analise o corpo da função para compreender seu comportamento.

```
# Tipos implícitos: foco na operação, não na estrutura
def calcular_total(preco, quantidade):
    total = preco * quantidade
    return total
```

Linguagens contemporâneas (Kotlin, Swift, TypeScript, C# e Java) buscam conciliar os benefícios de ambas as abordagens por meio da inferência de tipo. O compilador deduz o tipo de variáveis locais quando possível, mantendo a segurança da tipagem estática e exigindo declarações explícitas apenas onde são semanticamente relevantes, como em parâmetros e retornos de funções. No trecho em Java apresentado a seguir, a remoção da redundância na declaração de `total` simplifica a leitura sem sacrificar a clareza da interface pública da função. O consumidor do método ainda sabe exatamente quais tipos fornecer e esperar, enquanto o implementador beneficia-se de uma sintaxe mais enxuta.

```
// Parâmetros e retorno mantêm tipos explícitos; variável local usa inferência
public double calcularTotal(double preco, int quantidade) {
    var total = preco * quantidade; // 'total' é inferido como double
    return total;
}
```