Capítulo X – Avaliação de Linguagens

"A linguagem não serve somente para expressar pensamentos, mas para tornar possível pensamentos que não poderiam existir sem ela."

Bertrand Russell

Linguagens de programação são ferramentas fundamentais para o profissional de computação. Programadores adquirem maior habilidade para resolver problemas e para aprender novas LPs quando têm um bom conhecimento a respeito dos conceitos de linguagens de programação.

Mesmo aqueles profissionais que não atuam diretamente como programadores necessitam ter conhecimento a respeito desses conceitos. Afinal, esse conhecimento é necessário em todas as etapas do processo de desenvolvimento de software. Ele é importante para analisar a viabilidade do desenvolvimento de uma aplicação, para estimar o tempo necessário e o custo da implementação, para definir a técnica de projeto de software a ser utilizada e mesmo para aumentar a efetividade na comunicação entre programadores e projetistas de software.

Conhecimento sobre os conceitos de LPs é requisito essencial para realizar uma das tarefas mais fundamentais no processo de desenvolvimento de software: a seleção da linguagem mais apropriada para a implementação de uma aplicação. Nessa direção, esse capítulo objetiva apresentar alguns critérios para serem usados na avaliação de LPs com vistas à seleção de linguagens. Esses critérios são utilizados posteriormente para fazer uma comparação entre C, C++ e JAVA.

Definir critérios para avaliação de LPs não é uma tarefa fácil. Existem inúmeras possibilidades. Para se ter uma idéia, cada um dos conceitos apresentados nesse livro tem potencial para se tornar um critério de avaliação. Outra dificuldade se relaciona com o fato desses critérios poderem estar em diferentes níveis de granularidade. Critérios mais gerais podem ser aplicados em um maior expectro de situações, enquanto os mais específicos permitem uma comparação mais precisa e detalhada. Por fim, existe a questão da relevância do critério. Critérios são considerados mais ou menos importantes de acordo com a perspectiva de quem os analisa. Certamente, outros profissionais de computação podem perfeitamente escolher um conjunto diferente de critérios (e que, eventualmente, contenham critérios de maior relevância) ao apresentado aqui.

Dificuldade semelhante surge na realização de comparações entre linguagens. Independentemente do conjunto de critérios selecionado, é certo que existirão controvérsias e argumentos contrários a respeito das avalia-

ções realizadas nesse capítulo com relação a uma ou outra LP em um ou mais critérios.

Por conta disso, o leitor deve considerar os critérios e a comparação apresentados nesse capítulo apenas como uma visão do autor. Eles não podem e não devem ser encarados como a única expressão da realidade, a qual pode ser sempre analisada segundo diferentes perspectivas. Inclusive, recomenda-se ao próprio leitor, quando necessário, definir o seu próprio conjunto de critérios, de acordo com o que considera mais relevante, e fazer sua própria comparação entre LPs.

10.1 Critérios para Avaliação

A avaliação de linguagens de programação pode ser necessária em várias situações. Ao se desenvolver um sistema computacional, é necessário avaliar as LPs disponíveis para escolher aquela que pode trazer maiores facilidades e benefícios para a elaboração e operação do sistema. Avaliar LPs também é importante para uma instituição no momento de definir ou padronizar as linguagens usadas na implementação de seus projetos. A avaliação de LPs pode ainda ser necessária em comparações técnico-científicas.

Os critérios escolhidos para avaliação foram separados em dois grupos. No primeiro grupo foram colocados os critérios mais gerais, os quais propiciam uma comparação abrangente das linguagens mas não abordam especificamente os mecanismos e conceitos existentes na LP. O segundo grupo de critérios contém os critérios mais específicos, os quais enfocam fundamentalmente os mecanismos e conceitos oferecidos pelas LPs para possibilitar a implementação de uma ou outra característica em um sistema computacional.

10.1.1 Critérios Gerais

A avaliação dos seguintes critérios gerais deve ser suficiente para definir a LP a ser adotada em um sistema ou por uma instituição em boa parte das situações nas quais isso é necessário.

a. Aplicabilidade: avalia se a LP oferece todos os mecanismos necessários para o desenvolvimento de aplicações em geral ou de uma aplicação específica.
b. Confiabilidade: avalia se a LP promove o projeto e implementação de sistemas computacionais confiáveis através do uso de conceitos que maximizem a detecção automática de erros e não estimulem a ocorrência de erros.

c. Facilidade de Aprendizado: avalia se a LP oferece uma quantidade regular de conceitos e é simples o bastante para ser facilmente aprendida por um programador.
d. Eficiência: avalia o quanto a LP demanda de recursos de memória e processamento durante a execução dos programas.
e. Portabilidade: avalia a facilidade para se migrar os códigos fonte dos programas de uma plataforma para outra.
f. Suporte ao Método de Projeto: avalia se a LP suporta o método de projeto a ser usado na aplicação específica ou adotado pela instituição.
g. Evolutibilidade: avalia se a LP oferece conceitos adequados para estimular a criação de programas legíveis e facilmente atualizáveis.
h. Reusabilidade: avalia os meios e facilidades oferecidas pela LP para permitir a reutilização de código.
i. Integração com outros softwares: avalia quais mecanismos a LP oferece (ou necessita) para construção de programas que incorporem (ou sejam incorporados) por programas implementados em outras LPs.
j. Custo: avalia o custo financeiro necessário para o uso da linguagem, para a aquisição das ferramentas de desenvolvimento (tais como, editores de programas, compiladores, depuradores) na plataforma na qual será desenvolvida a aplicação e para uso da aplicação (tal como, por exemplo, se existe custo para executar a aplicação utilizando uma máquina virtual).

10.1.2 Critérios Específicos

Esse grupo de critérios é mais apropriado para realizar comparações técnico-científicas entre LPs, embora alguns deles, em algumas situações, também possam ser importantes na definição da LP adotada em um sistema ou por uma instituição. Para não tornar o processo de comparação muito extenso, optou-se por selecionar apenas critérios considerados mais relevantes. Os critérios específicos são listados a seguir.

a. Escopo: avalia se a LP requer a definição (ou declaração) explícita das entidades de programação, associando-lhes um escopo de visibilidade determinado pelo organização textual do programa.
b. Expressões e Comandos: avalia se a LP oferece uma boa variedade de expressões e/ou comandos para a construção de programas estruturados.
c. Tipos Primitivos e Compostos: avalia se a LP oferece uma ampla variedade de tipos primitivos e compostos, permitindo representar qualquer categoria importante de dado. Complementarmente, esse critério deve levar em conta a forma de tratamento do tipo *String* na LP.

d. Gerenciamento de Memória: avalia se a LP oferece um mecanismo próprio para gerenciamento de memória ou se o deixa sob a responsabilidade do programador.
e. Persistência de Dados: avalia o suporte oferecido pela LP para a realização de operações relacionadas com a persistência de dados.
f. Passagem de Parâmetros: avalia os modos e mecanismos disponíveis na LP para realização da passagem de parâmetros.
g. Encapsulamento e Proteção: avalia se a LP oferece mecanismos para encapsulamento e proteção dos dados.
h. Sistema de Tipos: avalia se a LP impõe (ou não) uma disciplina rigorosa na realização de operações sobre os dados, coibindo a execução de operações sobre dados de tipos para os quais não foi planejada e impedindo a manipulação de dados de um determinado tipo como se fossem de outro.
i. Verificação de Tipos: avalia se a verificação de tipos das operações é feita estática ou dinamicamente.
j. Polimorfismo: avalia os tipos de polimorfismos oferecidos pela LP. Em caso de existência de polimorfismo de inclusão, é importante avaliar se a LP oferece herança simples ou múltipla.
k. Exceções: avalia se a LP oferece mecanismos específicos para tratamento de exceções ou se o controle de erros é deixado sob responsabilidade exclusiva dos programadores.
l. Concorrência: avalia os recursos oferecidos pela LP para a construção de programas concorrentes.

10.2 Uma Comparação entre C, C++ e JAVA

Nessa seção é feita uma comparação das LPs C, C++ e JAVA com base nos critérios gerais e específicos apresentados na seção anterior. Para cada critério é indicado se a LP o atende, atende apenas parcialmente ou não atende. Para cada critério são apresentadas as razões que levaram a essa classificação.

Critérios Gerais:

a. Aplicabilidade: C e C++ atendem complementamente esse critério, mas JAVA só o atende parcialmente. C, C++ e JAVA são linguagens de propósito geral, contudo JAVA não oferece recursos para controlar diretamente o hardware. Para fazer isso programas em JAVA devem declarar métodos nativos e implementá-los em outras LPs.
b. Confiabilidade: JAVA atende esse critério, mas C++ e C não o atendem. C e C++ possuem inúmeras características estimuladoras de erros em programação (tais como, comando de desvio incondicional irrestrito e manipulação direta de endereços de memória através de operações com ponteiros).

c. Facilidade de Aprendizado: Nenhuma das três linguagens atende a esse requisito, embora certamente o aprendizado de C e JAVA seja muito mais fácil que o de C++. Embora seja uma linguagem simples, com poucos conceitos e bastante ortogonal, a exigência de uso massivo do conceito de ponteiros acaba aumentando a dificuldade de aprendizado de C. JAVA também é uma linguagem bem projetada, mas não é simples. Existem muitos conceitos (os quais nem sempre se combinam ortogonalmente) e muitas maneiras de se atingir uma determinada funcionalidade. C++ une as dificuldades de aprendizado de C e JAVA, além de possuir um número excessivamente alto de conceitos diferentes.
d. Eficiência: Dentre as LPs mais conhecidas, C é a LP mais eficiente computacionalmente. C++ se aproxima de C nesse aspecto, mas o uso dos mecanismos de orientação a objetos e um sistema de tipos mais rigoroso lhe impõem alguma perda de eficiência. JAVA é a menos eficiente das três LPs consideradas. Além de implementar vários mecanismos de verificação dinâmica de erros, o acionamento do coletor de lixo implica em perda substancial de desempenho de execução.
e. Portabilidade: Embora C e C++ sejam linguagens com versões padronizadas pela ANSI e pela ISO, é comum em implementações de compiladores oferecer características adicionais dependentes da plataforma na qual o compilador atua. O uso dessas características implica em redução da portabilidade. Adicionalmente, e com mais riscos de comprometimento da portabilidade, as definições de C e C++ deixam brechas para que certos conceitos sejam definidos conforme a plataforma de execução (tais como, o intervalo de valores dos tipos inteiros). JAVA, por outro lado, foi projetada de modo a evitar brechas que comprometessem a portabilidade. Contudo, JAVA ainda não foi padronizada por entidades como ANSI ou ISO. De fato, a detentora dos direitos de propriedade sobre essa LP foi quem estabeleceu um padrão, o qual deve ser obedecido por qualquer implementação de JAVA.
f. Suporte ao Método de Projeto: C suporta o método estruturado. JAVA suporta o método orientado a objetos. C++ suporta predominantemente o método orientado a objetos, mas como se trata de uma extensão de C, ela também suporta o método estruturado.
g. Evolutibilidade: C possui características que podem ser facilmente usadas para criar código ilegível e difícil de manter. C++ melhora esse aspecto em relação a C quando a orientação a objetos é utilizada, pois estimula o encapsulamento e a proteção de dados. JAVA só admite a programação orientada a objetos e ainda oferece estímulos para a construção de código bem documentado.
h. Reusabilidade: C oferece apenas o mecanismo de bibliotecas, compostas por funções, tipos, variáveis e constantes, para possibilitar o reuso

de código. C++ e JAVA oferecem bibliotecas de classes e ainda possuem o mecanismo de pacotes. Além disso, essas linguagens dispõem de meios para promover o encapsulamento e proteção de dados, o que estimula a construção de componentes reusáveis, e o polimorfismo universal, o que facilita a criação de código reusável e a implementação de `frameworks`.

i. Integração: C e C++ permitem a criação de programas que podem invocar código compilado por qualquer LP. JAVA possui uma interface específica, chamada JNI, que permite a integração de programas com código nativo em C e C++ apenas.

j. Custo: As versões padronizadas de C e C++ são linguagens de domínio público e, portanto, podem ser usadas gratuitamente. JAVA é de propriedade da SUN Mycrosystems, a qual tem liberado o uso gratuito dessa linguagem. Existem inúmeras ferramentas de apoio ao desenvolvimento de programas nessas linguagens nas diversas plataformas. Algumas dessas ferramentas são distribuídas gratuitamente. Não existe custo para uso das aplicações em qualquer dessas LPs.

A tabela 10.1 sumariza as avaliações dos critérios gerais nessas LPs.

Critérios Gerais	**C**	**C++**	**JAVA**
Aplicabilidade	Sim	Sim	Parcial
Confiabilidade	Não	Não	Sim
Facilidade de Aprendizado	Não	Não	Não
Eficiência	Sim	Sim	Parcial
Portabilidade	Não	Não	Sim
Suporte ao Método de Projeto	Estruturado	Estruturado ou Orientado a Objetos	Orientado a Objetos
Evolutibilidade	Não	Parcial	Sim
Reusabilidade	Parcial	Sim	Sim
Integração	Sim	Sim	Parcial
Custo	Dependente da Ferramenta de Desenvolvimento	Dependente da Ferramenta de Desenvolvimento	Dependente da Ferramenta de Desenvolvimento

Tabela 10. 1 - Avaliação de Critérios Gerais

Critérios Específicos:

a. Escopo: Todas as três LPs requerem a definição explícita de entidades. Essa definição associa às entidades um escopo de visibilidade. As regras de escopo são praticamente as mesmas nas três LPs, havendo apenas pequenas diferenças, tais como, JAVA não permite a definição aninhada de entidades com mesmo nome; C não permite a definição de variáveis no bloco de definição do comando *for*; JAVA e C++ possibilitam tornar visível, ou não, os campos de uma estrutura.

b. Expressões e Comandos: Todas as três LPs oferecem uma ampla variedade de expressões e comandos. Novamente, a maioria das expressões e comandos apresenta significativa similaridade nas três LPs. As principais diferenças são o retorno booleano das condições dos comandos de seleção e repetição em JAVA, enquanto em C e C++ eles retornam inteiros; a substituição do comando *goto* em JAVA pelos escapes rotulados e tratamento de exceções; e a existência de precedência entre operandos em JAVA.
c. Tipos Primitivos e Compostos: C não oferece o tipo booleano. C++ chega a definir esse tipo, contudo ele equivale a um tipo inteiro. Seu efeito, portanto, só aumenta a legibilidade da LP. Expressões booleanas continuam produzindo e utilizando valores inteiros. C e C++ utilizam o tipo ponteiro para implementar tipos recursivos e para manipular valores do tipo função. JAVA não trata métodos como valores de primeira classe e não oferece uniões e tipos enumerados. Tais ausências são compensadas por propriedades da orientação a objetos nessa LP. Nenhuma dessas LPs oferece conjuntos potência.
d. Gerenciamento de Memória: C e C++ deixam a responsabilidade para o programador. JAVA usa um coletor de lixo.
e. Persistência de Dados: C e C++ oferecem funções (e classes) de biblioteca para realização de operações sobre arquivos e deixam toda a responsabilidade pela execução da persistência de dados com o programador. É possível fazer interface com gerenciadores de bancos de dados, mas não existe uma definição na linguagem sobre como isso deve ser feito. Além de dispode de uma biblioteca de classes para realização de operações sobre arquivos, JAVA possui o recurso de serialização, o que facilita sobremaneira a realização de persistência. JAVA também define o JDBC, uma interface padrão para o trabalho com gerenciadores de bancos de dados.
f. Passagem de Parâmetros: C usa apenas o mecanismo de passagem por valor, o que provoca o uso de ponteiros como parâmetros em várias oportunidades, e permite a criação de funções com lista de parâmetros variável. JAVA usa apenas passagem por valor para tipos primitivos e por cópia de referência para objetos. Dessas LPs, C++ é a que oferece um maior leque de opções, tendo funções com lista de parâmetros variável, parâmetros `default`, passagem por valor e por referência.
g. Encapsulamento e Proteção: C oferece apenas encapsulamento de dados, isto é, não é possível estabelecer em uma única entidade sintática dados e as operações sobre eles. C também não possibilita a proteção dos dados. C++ e JAVA oferecem recursos tanto para encapsulamento quanto para proteção.
h. Sistemas de Tipo: O sistema de tipos de C é facilmente violável. Inúmeras características, tais como, uniões livres, coerções e aritmética de

ponteiros, possibilitam tratar valores definidos de um tipo como sendo de outro. C++ apresenta um sistema de tipos mais rigoroso quando são usadas as características de orientação a objetos. Contudo, todas as fragilidades do sistema de tipos de C também estão presentes. JAVA possui um sistema de tipos bastante rigoroso.

i. Verificação de Tipos: As verificações de tipos feitas por C são todas estáticas. C++ e JAVA fazem quase toda a verificação estaticamente, mas também podem fazer alguma verificação dinâmica. Isso ocorre, por exemplo, ao se usar o polimorfismo de inclusão e, somente no caso de JAVA, ao se tentar realizar um acesso à posição inexistente em um vetor.
j. Polimorfismo: C somente possui coerção e sobrecarga embutidas na LP. C++ possui todos os tipos de polimorfismo embutidos na LP e permite ao programador fazer sobrecarga de funções e operadores, além de criar código com polimorfismo paramétrico e por inclusão. JAVA possui todos os tipos de polimorfismos embutidos, mas só oferece ao programador a possibilidade de sobrecarregar operadores e usar polimorfismo por inclusão.
k. Exceções: C não oferece um mecanismo específico para lidar com exceções. C++ oferece um mecanismo, mas ele não precisa necessariamente ser usado ou respeitado. JAVA oferece um mecanismo de exceções rigoroso para lidar com exceções.
l. Concorrência: C e C++ não oferecem recursos específicos para lidar com programas concorrentes. É necessário utilizar bibliotecas específicas da plataforma na qual o sistema será executado. JAVA oferece recursos para criação de `threads` e sincronização de métodos de um objeto.

A tabela 10.2 sumariza as avaliações dos critérios específicos nessas LPs.

Critérios Específicos	**C**	**C++**	**JAVA**
Escopo	Sim	Sim	Sim
Expressões e Comandos	Sim	Sim	Sim
Tipos Primitivos e Compostos	Sim	Sim	Sim
Gerenciamento de Memória	Programador	Programador	Sistema
Persistência de Dados	Biblioteca de Funções	Biblioteca de Classes e Funções	Biblioteca de Classes , Serialização e JDBC
Passagem de Parâmetros	Lista variável e passagem por valor	Lista variável, parâmetros default, passagem por valor e por referência	Passagem por valor e por cópia de referência

Encapsulamento e Proteção	Parcial	Sim	Sim
Sistema de Tipos	Não	Parcial	Sim
Verificação de Tipos	Estática	Estática e Dinâmica	Estática e Dinâmica
Polimorfismo	Coerção e Sobrecarga	Coerção, Sobrecarga, Paramétrico e Inclusão	Coerção, Sobrecarga e Inclusão
Exceções	Não	Parcial	Sim
Concorrência	Não	Não	Sim

Tabela 10. 2 - Avaliação de Critérios Específicos

10.3 Considerações Finais

Existem várias outros conjuntos de critérios que podem ser usados para comparar LPs. Existem também várias outras comparações interessantes que podem ser encontradas na literatura ou na própria Internet. David A. Wheeler [Wheeler, 1997] utiliza o documento de especificação de ADA para realizar uma ampla comparação entre ADA 95, C, C++ e JAVA. Patricia K. Lawlis [Lawlis, 1997] faz uma comparação de diversas LPs baseada em um conjunto de critérios considerados importantes para a escolha de linguagens em organizações. Lutz Prechelt [Prechelt, 2000] faz uma comparação empírica entre C, C++, JAVA e outras linguagens baseadas em implementações de um mesmo conjunto de requisitos.

10.4 Exercícios

1. Escolha uma outra LP de seu conhecimento e a inclua na comparação feita na seção 10.2 desse capítulo.

2. Especifique os requisitos de um sistema que você precise implementar. Selecione algumas LPs disponíveis para serem usadas na implementação. Use os critérios especificados na seção 7.1 para escolher a LP a ser utilizada. Justifique a sua escolha.

3. Defina o seu próprio conjunto de critérios de avaliação e o utilize para avaliar as LPs do exercício 3 e escolher a LP a ser utilizada pelo mesmo sistema especificado no exercício 3. Justifique a sua escolha.