Parte I

Conceitos Fundamentais

Na Parte I, são apresentados os conceitos fundamentais que 'formam' uma linguagem de programação.

Nos Capítulos de 2 a 7 são expostos os conceitos de valores, tipos, variáveis, expressões, comandos e abstrações. Esses conceitos são mostrados em uma seqüência lógica, que acreditamos ser a mais didática para o entendimento de cada um dos "ingredientes" que compõem as linguagens de programação.

2

Linguagens de Programação: Noções Preliminares

"Vivere tota vita discendum est et, quod magis fortasse miraberis, tota vita discendum est mori."

—Sêneca

As linguagens de programação são amplamente conhecidas e utilizadas por profissionais da área de computação. Na maioria das vezes, tais profissionais são apenas usuários de linguagens e ambientes de programação.

Para que linguagens de programação possam ser utilizadas, elas precisam ter sido projetadas e implementadas. 'Alguém' precisou descobrir do que os profissionais de computação precisam e o que eles esperam de uma linguagem de programação. Isso inclui estudos sobre o que é essencial nas linguagens para que os problemas possam ser resolvidos, uma tarefa que também é realizada pelos projetistas de linguagens. Ainda mais do que isso: o que se deseja de uma linguagem de programação precisa ser passível de implementação. Todas essas considerações devem deixar claro que o

uso de uma linguagem não dota o usuário do conhecimento necessário para projetar e implementar uma linguagem de programação.

O projeto de uma linguagem de programação passa por uma questão essencial: o que é uma linguagem de programação?

Para responder a essa questão, podemos iniciar pela caracterização do que é um programa. Segundo [7], um programa é uma entidade que se manifesta de duas formas:

1. como um *documento*, ele especifica uma seqüência de operações a serem executadas;

2. durante a sua *execução*, ele efetivamente leva a cabo as operações especificadas.

Um programa, portanto, é uma *máquina abstrata* – já que manipula e produz entidades abstratas, que são os *dados*. Ele é ao mesmo tempo a *descrição* de uma máquina (o *documento*) e a própria máquina (quando está em *execução*).

Sendo uma máquina abstrata, o programa precisa ser *implementado* em um mecanismo físico para que sua execução abstrata possa ser *fisicamente simulada*. O mecanismo físico que simula a execução de programas é o computador.

Uma linguagem de programação determina os *recursos* disponíveis e sua forma de utilização para construir máquinas abstratas específicas, de tal forma que elas possam ser simuladas adequadamente em computadores.

Uma linguagem de programação é um conjunto de recursos que podem ser compostos para constituir programas específicos, mais um conjunto de regras de composição que garantem que todos os programas podem ser implementados em computadores com qualidade apropriada.

Para projetar uma linguagem de programação, alguns itens gerais precisam ser observados:

- *requisitos*: qual o universo de problemas que queremos resolver com a dada linguagem?

- *expressividade*: qual a forma mais natural de representar os elementos da linguagem que provêem os requisitos desejados?

- *paradigma*: qual a forma mais adequada para representar os problemas a serem resolvidos (álgebra, lógica, composições de funções, seqüências de operações, etc.), e qual o paradigma de programação mais apropriado para se resolver problemas dos domínios de aplicação desejados?

- *implementação*: os requisitos, juntamente com sua forma de representação, são passíveis de implementação?

- *eficiência*: os requisitos são implementados em um patamar aceitável de eficiência?

2.1 SINTAXE E SEMÂNTICA

Na sua essência, uma linguagem de programação deve atender aos requisitos esperados de forma eficiente. Para tanto, a linguagem deve ser representada de uma forma única, e seus elementos devem também ter significados únicos. Devemos, portanto, definir a linguagem de forma:

- *sintática*: como é escrito cada um dos elementos da linguagem; e

- *semântica*: o que significa cada um dos elementos da linguagem.

A sintaxe determina a forma de manipular programas em uma linguagem. Ela é construída visando facilitar essa manipulação, tendo em vista o uso previsto para a linguagem (caracterizado pelos itens de projeto apresentados na seção anterior: *requisitos, expressividade, paradigma, implementação* e *eficiência*).

A semântica determina a interpretação pretendida para cada elemento sintático da linguagem. Para as linguagens de programação, em geral a semântica é caracterizada sob três aspectos (cf. por exemplo [43] ou [71] para uma explicação bem mais técnica):

1. *semântica axiomática*: descrita tipicamente através de um conjunto de axiomas equacionais que relacionam diferentes expressões sintáticas na linguagem. Assim, da mesma forma como ocorre com um dicionário para termos em por-

tuguês, o significado de uma expressão é determinado através de "sinônimos" [25, 31];

2. *semântica operacional*: descrita pelas operações efetuadas pelas expressões e seus resultados. A semântica operacional caracteriza o significado de cada expressão em um programa pelo que aquela expressão faz. O significado de um programa é o comportamento deste quando executado em uma máquina [47]; e

3. *semântica denotacional*: descrita pelos conjuntos de dados associados a cada expressão. Considerando o programa como uma máquina de transformação de dados, o significado de cada expressão pode ser caracterizado em termos dos dados que ela transforma. O significado de um programa pode ser visto como uma função matemática que mapeia as entradas do programa para suas respectivas saídas [51].

2.2 COMPILADORES E INTERPRETADORES

Uma vez definidas a sintaxe e a semântica da linguagem, precisamos de uma forma eficiente para a implementação de cada um dos elementos da linguagem. Essa implementação é feita por *processadores*: compiladores e interpretadores, ou ainda um misto entre estas formas de processadores.

- *Compiladores*: os programas são transformados, por inteiro, em um conjunto correspondente de instruções que podem ser fisicamente efetuadas pelo computador. A execução do programa é feita diretamente através dessas instruções de computador; o programa inteiro constitui uma unidade a ser transferida "em um único lote" para o computador.

- *Interpretadores*: cada expressão do programa é executada diretamente, ou seja, transformada nas instruções correspondentes e acionada no computador sem ter-se necessariamente que transformar o programa inteiro em uma unidade de execução. As expressões do programa são executadas 'sob demanda'.

A construção de processadores é um tema complexo que merece um livro específico para seu desenvolvimento. Ela exige o conhecimento profundo e completo da linguagem de programação considerada, dos pontos de vista sintático e semântico, bem como dos computadores nos quais os programas escritos naquela linguagem serão executados.

2.3 METODOLOGIAS DE PROGRAMAÇÃO

Formas mais eficientes de desenvolvimento requerem novas formas de se resolver problemas. Do ponto de vista de linguagens de programação, essas novas soluções podem requerer desde novos elementos básicos nas linguagens até novos paradigmas de programação.

Temos hoje diferentes paradigmas de linguagens de programação amplamente utilizados que advêm de formas diferentes de solucionar problemas. Tais paradigmas estão relacionados não apenas com os domínios de aplicação desejados, mas também com os métodos para a solução de problemas. Linguagens de programação orientadas a objetos podem ser vistas como ferramentas para construir soluções de problemas orientadas *por* objetos, linguagens imperativas podem ser vistas como ferramentas para construir soluções de problemas orientadas *por* dados, e linguagens declarativas podem ser vistas como ferramentas para construir soluções de problemas orientadas *por* relações entre declarações. Nesse caso, os elementos a serem representados são respectivamente objetos, dados e relações. Cada linguagem deve prover mecanismos que permitam a representação e a manipulação naturais dos elementos básicos pelos quais se orientam os problemas tratados.

As próprias metodologias de desenvolvimento estão em constante evolução. Novos elementos são permanentemente incorporados às linguagens visando suprir as características emergentes dessas metodologias de desenvolvimento, e/ou novas formas de implementação de elementos das linguagens surgem junto com as mudanças nas metodologias. A verificação de tipos, por exemplo, que é realizada no momento da compilação nas linguagens tradicionais, é feita durante a execução em linguagens

mais modernas, em decorrência do requisito existente nas metodologias mais atuais de flexibilidade na representação de dados, objetos e relações.

2.4 CARACTERÍSTICAS DESEJÁVEIS PARA UMA LINGUAGEM DE PROGRAMAÇÃO

Informalmente, as características desejáveis para uma linguagem de programação são [54]:

1. *legibilidade*: as linguagens devem possuir única e exclusivamente elementos de fácil entendimento e não ambíguos. Para preservar a característica de legibilidade, as linguagens devem combinar:

   - *simplicidade e ortogonalidade*: a linguagem deve prover um número reduzido de elementos básicos, evitando multiplicidade de escrita e facilitando o entendimento dos programas descritos. A linguagem deve também conter um conjunto mínimo de primitivas que possam ser combinadas, de forma que componentes mais elaborados na linguagem sejam apenas combinações dos elementos básicos;
   - *instruções de controle que não comprometam a clareza dos programas*: as instruções de controle devem ser reduzidas ao conjunto estritamente essencial para o controle de fluxo de execução dos programas. Instruções de desvio (*go to*) explícito, como as existentes em linguagens mais antigas, são hoje preservadas em algumas linguagens apenas com propósitos bem específicos, dado que seu uso indiscriminado prejudica a legibilidade dos programas. Cada instrução da linguagem deve ter um propósito explícito para que seja utilizada adequadamente;
   - *facilidade para representação de tipos e estruturas de dados*: a linguagem deve prover facilidades para a representação de dados comumente necessários para a resolução de problemas. Tipos primitivos, tais como números inteiros e reais, bem como tipos compostos devem ter mecanismos de definição e manipulação intuitivos e de fácil assimilação;

- *sintaxe "limpa" e concisa*: cada instrução deve representar de forma única e intuitiva o seu significado;

2. *facilidade de escrita*: simplicidade e ortogonalidade são características desejáveis para a legibilidade que também facilitam a escrita de programas. Programas facilmente entendidos são, em geral, provenientes da facilidade de escrita provida pela linguagem. Além da simplicidade da linguagem, o suporte a abstrações deve ser provido na linguagem para facilitar a programação. Formas abstratas de representar transformações (por exemplo, de dados ou objetos) denotam modos facilitados de solucionar problemas, em que apenas os elementos essenciais são representados. Funções em linguagens imperativas, por exemplo, são formas abstratas de representar soluções, nas quais os parâmetros abstraem os dados reais a serem manipulados. Um bom projeto de linguagem deve, portanto, tratar da expressividade necessária para resolver problemas dos domínios requisitados de forma clara e natural;

3. *confiabilidade*: espera-se que soluções dadas aos problemas sejam confiáveis. Implementações correspondentes aos problemas especificados devem produzir os resultados esperados. É desejável, por exemplo, que os programadores não façam indesejavelmente operações com tipos conflitantes, de forma que resultados inexistentes sejam produzidos. Por isso, as linguagens atuais possuem, na sua maioria, mecanismos para a verificação de tipos. Apesar destes não serem de fácil implementação, o prejuízo causado por resultados equivocados possui um custo mais elevado que a verificação. Ainda neste sentido, a manipulação de exceções tem sido cada vez mais utilizada nas linguagens de programação atuais. Tais mecanismos permitem um tratamento uniforme de operações sem prejuízo à confiabilidade dos resultados; resultados são produzidos apenas para dados consistentes, e exceções são providas quando a consistência é violada;

4. *custo*: o custo associado a uma linguagem de programação vai desde o custo de desenvolvimento da mesma até sua amortização proveniente da aceitação no mercado e os custos relativos ao seu uso. Tem-se, dessa forma, o custo relativo ao treinamento de pessoas para o uso da linguagem, bem como o custo de

escrita de programas na linguagem, e os custos de compilação e de execução dos programas. Portanto, linguagens legíveis, de fácil escrita e confiáveis tendem a ter um custo mais baixo de treinamento e são mais aceitas no mercado, desde que dêem suporte a características desejáveis aos domínios de aplicação requisitados pelo mercado.

2.5 TIPOLOGIA DAS LINGUAGENS DE PROGRAMAÇÃO

Na Figura 2.1 apresentamos uma tipologia das linguagens de programação, em função das metodologias correntes de construção de programas. Para cada tipo de linguagem, apresentamos alguns exemplos representativos de linguagens ainda em uso.

As *linguagens assertivas* baseiam-se em expressões que modificam valores de entidades – dados ou objetos. Elas são subdivididas em *linguagens imperativas*, ou orientadas a dados, em que a construção de programas é dirigida pelas transformações que ocorrem nos dados; e *linguagens orientadas a objetos*, em que a construção de programas é dirigida pelas mudanças de estados de entidades abstratas denominadas *objetos*. Um objeto, intuitivamente, é um conjunto de repositórios de dados mais um conjunto de operadores capazes de transformar os valores dos dados nesses repositórios.

As *linguagens declarativas* baseiam-se em expressões que verificam ou induzem a que ocorram relações entre declarações. Elas são subdivididas em *linguagens funcionais*, em que essas relações são caracterizadas por mapeamentos entre estruturas simbólicas; *linguagens lógicas*, em que as relações são caracterizadas como expressões da lógica matemática; e *linguagens orientadas a satisfações de restrições*, em que as relações são caracterizadas como equações e inequações algébricas.

Cada uma dessas metodologias apresenta características que a torna mais adequada para certos tipos de problemas. Essas características determinam os recursos e propriedades sintáticas e semânticas desejáveis nas linguagens de programação correspondentes.

Para que esses recursos e propriedades sejam descritos, vários conceitos precisam ser introduzidos. Esses conceitos serão apresentados gradativamente nos próximos capítulos.

2.6 LEITURA RECOMENDADA

Dois livros interessantes para uma visão geral dos conceitos fundamentais de linguagens de programação são [7] e [64].

Alguns livros mais técnicos e que requerem uma maior maturidade matemática do leitor são [43] e [71].

Para mais explicações sobre cada forma de associar significado (semântica) a linguagens e programas, recomendamos [25, 31, 47, 51].

Um resumo sobre o histórico das linguagens de programação e a relação entre elas pode ser encontrado em [54]. Um estudo mais extenso sobre o assunto pode ser encontrado em [68] e [2].

Aos leitores também interessados em processadores de linguagens de programação recomendamos as leituras introdutórias [3] e [65].

2.7 EXERCÍCIOS

1. Acrescente à tipologia da Figura 2.1 mais linguagens de programação, que você conheça ou encontre na literatura. Você deveria ser capaz de encontrar pelo menos mais dez linguagens de cada tipo (excetuando talvez as linguagens orientadas a satisfações de restrições, que são mais recentes – mesmo para esse tipo de linguagens, você deveria ser capaz de encontrar pelo menos mais cinco linguagens além das apresentadas).

2. Alguns dos tipos apresentados na Figura 2.1 tiveram sua origem no final dos anos 50, como resultado do trabalho de diferentes pesquisadores. Outros são um pouco mais recentes, mas de maneira geral cada tipo de linguagem já conta com um histórico representativo. As linguagens de programação freqüentemente evoluem, umas a partir das outras, e sendo assim construa uma "árvore

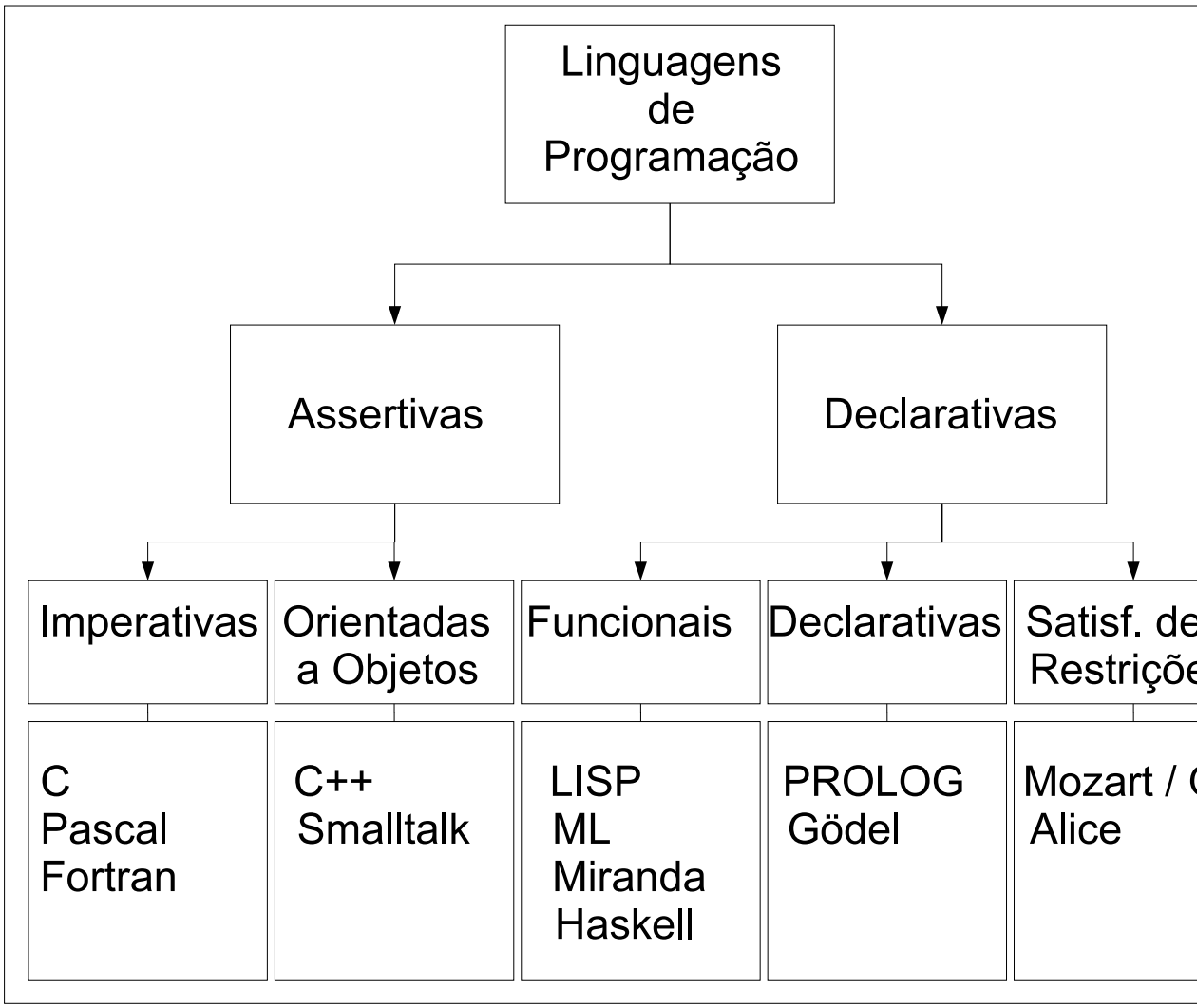

Figura 2.1 Tipologia de Linguagens de Programação

genealógica" das linguagens que você apresentou no exercício anterior, indicando também para cada linguagem o ano de criação e país (ou países, quando uma linguagem resultou de projetos de pesquisas multinacionais) de origem.