Capítulo 1

Introdução

O domínio de uma linguagem de programação requer a compreensão dos mecanismos da linguagem e de como ela é implementada. Há grande diferença, por exemplo, entre quem apenas aprende que se deve evitar conversão entre valores de tipos diferentes e quem sabe o porquê dessa recomendação, entendendo como os valores são representados e as transformações que sofrem ao serem convertidos.

É possível que alguém aprenda uma linguagem de programação e até elabore programas de razoável complexidade sem conhecer os fundamentos da computação. Esse será, porém, um aprendizado limitado e superficial, que normalmente desestimula o aluno, fazendo-o relegar o aprofundamento de seus conhecimentos e o estudo das matérias correlatas necessárias à sua formação profissional (projetos de programas complexos, análise de algoritmos, verificação e validação, etc.), tornando-o um repetidor e adaptador de soluções prontas. O melhor é mostrar ao aluno desde o início que uma linguagem de programação assenta-se em fundamentos que devem ser compreendidos para que se possa elaborar os programas com propriedade e, principalmente, entender as limitações e o comportamento dos programas desenvolvidos. Os seguintes tópicos descrevem os fundamentos que considero essenciais e mostram sua utilidade para os capítulos deste livro:

Arquitetura de computadores, abordando os conceitos de bit, byte, palavra, barramentos e comunicação entre memória e processador, modos de endereçamento, componentes básicos do processador e ciclo de instrução. Útil ao Capítulo 5 (Identificadores e Variáveis) para a compreensão da relação entre variável e memória, e dos conceitos de alocação de espaço em memória, classe de armazenamento e ligação de identificadores.

Sistemas de numeração, incluindo os modos de representação binária dos valores inteiros e reais. Destes, deve ser dada especial atenção à representação na forma de sinal, expoente e mantissa, adotada pelo padrão IEC 60559 [1] que normatiza as operações de ponto flutuante. Útil aos Capítulos 3 (Tipos de Dados), 4 (Literais e Constantes), 6 (Operadores e Expressões) e 14 (Identificação e Tratamento de Erros) para a compreensão dos operadores de deslocamento e demais operações binárias; dos limites de representação, maior e menor valor representável em um dado tipo; da existência de valores especiais como infinito, zeros negativos e valores não numéricos; e dos erros de representação e arredondamento.

Lógica proposicional, abordando os operadores de conjunção, disjunção e negação, e os conceitos de valor-verdade de proposições simples e compostas, tabela-verdade, equivalência e simplificação de expressões. Útil aos Capítulos 6 (Operadores e Expressões), 7 (Estruturas Condicionais) e 8 (Estruturas de Repetição) para a compreensão dos operadores lógicos e relacionais, dos mecanismos de

ordem e sequenciamento na avaliação de expressões; e do uso de expressões lógicas no controle de desvios e repetições.

Algoritmos, incluindo a noção de algoritmo (que não se confunde com a de receita, e tampouco pode ser expressa em termos vagos como “uma sequência lógica de comandos para resolver um problema”) e as construções fundamentais de sequência, decisão e repetição. Útil aos Capítulos 2 (Compilação e Visão Preliminar dos Programas C), 7 (Estruturas Condicionais), 8 (Estruturas de Repetição) e 9 (Funções e Procedimentos) para a compreensão das estruturas básicas de controle e dos mecanismos de atribuição, entrada e saída e modularização.

Descrição sintática, com o estudo da notação BNF (*Backus-Naur Form*, em inglês) para a especificação de construções sequenciais, alternativas, optativas e repetitivas. Útil aos Capítulos 6 (Operadores e Expressões), 7 (Estruturas Condicionais) e 8 (Estruturas de Repetição) para a compreensão da sintaxe dos operadores e estruturas de controle.

Está claro que o efeito desses fundamentos vai além dos capítulos relacionados, pois os demais pressupõem o domínio dos que foram citados. Por exemplo, não se pode entender estruturas, uniões e ponteiros sem que se tenha entendido tipos de dados e variáveis.

Este livro não contém material sobre os fundamentos necessários ao aprendizado de uma linguagem de programação, mas o leitor interessado, já iniciado na área, terá conhecimento deles e saberá onde adquirir o material para revisão. Os neófitos encontram esse material nos Capítulos 2 a 6 do livro do prof. Pinheiro [2], e também nas referências mantidas atualizadas no sítio deste livro (www.facp.pro.br/livroc) [3].

1.1 PADRÃO DA LINGUAGEM

O padrão da linguagem C especifica a semântica das suas operações e estruturas, além do funcionamento de várias funções auxiliares, definindo o comportamento que os programas em C devem exibir. É, portanto, um padrão para os implementadores da linguagem – aqueles que desenvolvem compiladores – e não para os programadores, que a utilizam na elaboração de programas aplicativos. No entanto, os programadores de aplicativos devem conhecer o padrão da linguagem que utilizam. Só assim podem elaborar programas que funcionem do mesmo modo em todas as plataformas para as quais sejam compilados com compiladores aderentes ao padrão; e só assim podem adotar de modo consciente soluções que se desviam do padrão: há muitos programadores confiantes que certas construções utilizadas são próprias da linguagem, desconhecendo que são meras extensões do compilador e que podem não funcionar em outras circunstâncias (com outros compiladores ou em outras plataformas).

As seguintes definições são adotadas neste livro:

Plataforma. É o conjunto constituído pelo processador e pelos softwares que caracterizam o ambiente de programação e execução, incluindo o sistema operacional e o compilador. É comum designar uma plataforma por um de seus componentes definidores, geralmente o sistema operacional. Assim, plataforma Linux designa o ambiente computacional que utiliza o sistema operacional Linux, e plataforma

Windows o ambiente que utiliza o sistema operacional Windows, subentendendo-se que os demais componentes, embora possam variar, estão presentes e são compatíveis entre si.

Arquitetura. Refere-se ao processador e aos softwares de controle usados para caracterizar o ambiente computacional, definindo aspectos do processamento relativos ao tamanho da palavra, representação binária de inteiros, tratamento de sinais de interrupção, etc. Neste livro, as restrições relativas à arquitetura geralmente estendem-se às plataformas. Por exemplo, quando uma característica é dita dependente da arquitetura, também será dependente da plataforma.

Implementação. Refere-se ao compilador e aos demais softwares auxiliares (montadores, ligadores, etc.), com suas opções de controle e bibliotecas, usados para produzir código adequado a uma *arquitetura-alvo* (ou plataforma-alvo). Chama-se arquitetura-alvo aquela na qual será executado o programa objeto da compilação.

Nem todos os aspectos da linguagem são estritamente padronizados. O padrão não define os comportamentos para os quais nem todas as arquiteturas oferecem o mesmo nível de suporte, ou quando eles são implementados de diferentes modos em diferentes arquiteturas. Esses casos são caracterizados como:

Comportamento não especificado. Comportamento não especificado pelo padrão, caracterizado pela adoção de valores ou ações para as quais existem alternativas que podem ser livremente escolhidas pela implementação. Esse tipo de situação não pode ser ignorada pela implementação, nem pode haver uma ação fora das alternativas especificadas pelo padrão. Por exemplo, a ordem de avaliação dos argumentos de uma função – eles devem ser avaliados antes da transferência do controle, mas cada implementação é livre para escolher a estratégia de avaliação mais adequada.

Comportamento dependente de implementação. Comportamento não especificado pelo padrão que pode ser implementado de modo variado por diferentes implementações. Por exemplo, a propagação do bit de mais alta ordem quando um inteiro sinalizado tem seus bits deslocados para a direita. Desse modo, a operação deslocamento binário para a direita é definida apenas para inteiros não sinalizados e para inteiros sinalizados positivos. Quando o inteiro é sinalizado negativo, cada implementação da linguagem é livre para adotar a solução mais conveniente.

Comportamento indefinido. Comportamento devido ao uso de construções ou dados errados, para o qual o padrão não determina uma ação específica. Nesses casos, as implementações da linguagem podem escolher ignorar a situação de erro (com resultados imprevisíveis), adotar uma linha de ação bem documentada que, entretanto, será própria da implementação, ou simplesmente interromper a compilação ou execução. Por exemplo, o tratamento do estouro de valores inteiros não é definido pelo padrão; portanto, uma implementação pode adotar a redução do valor à faixa de representação do tipo inteiro utilizado, enquanto outra pode simplesmente interromper a execução.

As características da linguagem para as quais o comportamento não é completamente padronizado devem ser evitadas: um programa que funciona em uma dada

implementação pode deixar de funcionar se compilado para outra arquitetura, ou com outro compilador, ou mesmo com outra versão do mesmo compilador. Além disso, se alguma característica é descrita pelo padrão como indefinida, sua implementação não precisa nem mesmo ser coerente: um compilador pode adotar um curso de ação diferente para diferentes instâncias da mesma situação.

O padrão original da linguagem C, desenvolvido pelo Instituto Americano de Padronização (ANSI, do inglês *American National Standards Institute*), foi publicado em 1989 e adotado em 1990 pela Organização Internacional de Padrões (ISO¹), juntamente com a Comissão Internacional Eletrotécnica (IEC, do inglês *International Electrotechnical Commission*). Esse padrão é conhecido pela sigla ISO/IEC 9899:1990. Ele foi corrigido em 1994, aumentado em 1995 e corrigido novamente em 1996. Uma nova versão foi publicada em 1999, sendo conhecida como ISO/IEC 9899:1999. Correções técnicas foram publicadas em 2001, 2004 e 2007, e reunidas, com adições, em uma terceira versão publicada em 2011. Este livro adota a versão ISO/IEC 9899:1999, com as correções publicadas até 2007, e incorpora as adições pertinentes da versão de 2011 [4].

1.2 PREPARANDO O AMBIENTE DE PROGRAMAÇÃO

É essencial dispor de um bom ambiente de programação. Programar se aprende programando – mesmo para aqueles que já sabem uma linguagem. A primeira providência é instalar um compilador e, preferencialmente, um ambiente de programação. A documentação dos ambientes e compiladores citados a seguir é abundante e facilmente encontrada. O sítio do livro (www.facp.pro.br/livroc) [3] contém orientações para a obtenção e instalação de alguns desses programas.

Todos os programas deste livro foram compilados e testados com o compilador gcc (*GNU Compiler Collection*, em inglês) [2], utilizando a biblioteca glibc (*GNU C Library*, em inglês) [5] como implementação das funções definidas no padrão. Tanto o compilador gcc quanto a biblioteca glibc são parte da maioria das distribuições Linux e, para aquelas que não os incorporam, são facilmente instalados por meio dos procedimentos padronizados para a instalação de novos pacotes.

Existem versões do gcc para ambientes Windows, as mais conhecidas são: MinGW (*Minimalist GNU for Windows*, em inglês) e CygWin (*GNU + Cygnus + Windows*). Os compiladores distribuídos exclusivamente para o ambiente Windows, como o MSVC++ (*Microsoft Visual C++*, em inglês), também podem ser usados: este livro indica os comportamentos não padronizados, de sorte que o aluno saberá identificar as diferenças em relação ao padrão adotado. Alguns ambientes de programação, como o CodeBlocks, funcionam com diversos compiladores.

¹ O nome inglês da organização é *International Organization for Standardization*. A sigla, embora próxima do nome inglês, foi concebida a partir do grego *isos* (igual) para ser independente de idioma.

PRÁTICA 1.1 Instale um ambiente de programação em sua máquina e verifique se ele está operacional.

1. Instale um compilador C ou verifique se já está instalado (alguns ambientes de programação já incorporam o compilador C).
2. Instale um ambiente de programação e siga sua documentação para verificar se ele está operacional.

O ideal é utilizar ambientes simples que permitam a digitação e compilação de programas sem a necessidade de criação de projetos – o foco deve ser a elaboração de programas, sem preocupação com configurações de projetos e demais funcionalidades presentes em ambientes complexos, principalmente para quem está iniciando. Consulte o sítio do livro para sugestões de ambientes de programação. ■