Introdução aos Computadores , à Internet e à Web

Objetivos

Neste capítulo, você aprenderá:

- Conceitos básicos do computador.
- Os diferentes tipos de linguagens de programação.
- A história da linguagem de programação C.
- A finalidade da Standard Library de C.
- Os elementos de um ambiente de desenvolvimento de um típico programa escrito em C.
- Como C oferece um alicerce para o estudo complementar das linguagens de programação em geral, em especial, da C++, Java e C#.
- A história da Internet e da World Wide Web.

1.1 Introdução

- O conteúdo do livro enfatiza a obtenção de clareza nos programas por meio de técnicas comprovadas da programação estruturada.
- Você aprenderá programação do modo correto desde o início.
- O livro traz exemplos de centenas de programas completos que funcionam são apresentados aqui, e mostram--se as saídas produzidas quando esses programas são executados em um computador.
- Chamamos isso de ‘técnica de código vivo’. Você pode fazer o download de todos esses programas-exemplo em nosso site, em <www.deitel.com/books/chtp6/>.
- É o software (ou seja, os comandos que você escreve para instruir o computador sobre como executar ações e tomar decisões) que controla os computadores (frequentemente chamados de hardware).

1.1 Introdução (Cont.)

- Este texto introduz a programação em C, que foi padronizada em 1989 como ANSI X3.159-1989 nos Estados Unidos, por meio do American National Standards Institute (ANSI), e depois no mundo, por meio dos esforços da International Standards Organization (ISO).
- Chamamos isso de C Padrão.
- Também apresentamos a C99 (ISO/IEC 9899:1999), a versão mais recente do padrão C.
- Um novo padrão C, que foi chamado informalmente de C1X, está em desenvolvimento e provavelmente será publicado por volta de 2012.
- O Apêndice E, opcional, apresenta a biblioteca C de programação de jogos Allegro.

1.1 Introdução (Cont.)

- O apêndice mostra como usar o Allegro para criar um jogo simples.
- Mostramos como exibir gráficos e animar objetos de forma suave, e explicamos recursos adicionais como som, entrada pelo teclado e saída em arquivo texto.
- O apêndice inclui links da Web e recursos que apontam para mais de 1.000 jogos Allegro e tutoriais sobre técnicas avançadas do Allegro.
- Os custos com computação diminuíram tremendamente
- devido ao rápido desenvolvimento de tecnologias de hardware e software.

1.1 Introdução (Cont.)

- Os grandes computadores eram chamados de mainframes, e as versões atuais são muito utilizadas no comércio, no governo e na indústria.
- A tecnologia do chip de silício tornou a computação tão econômica que mais de um bilhão de computadores de uso geral são utilizados no mundo inteiro.
- Outros bilhões de computadores de uso especial são usados em dispositivos eletrônicos inteligentes, como sistemas de navegação de automóveis, dispositivos economizadores de energia e controladores de jogos.

1.1 Introdução (Cont.)

- C++ é uma linguagem de programação orientada a objetos baseada em C, é de tal interesse hoje em dia que incluímos uma introdução detalhada a C++ e à programação orientada a objetos nos Capítulos 15 a 24.
- Para que você se mantenha atualizado com os desenvolvimentos em C e C++ pela Deitel & Associates, registre-se gratuitamente e receba nosso boletim por e-mail, Deitel® Buzz Online, em:
  - `www.deitel.com/newsletter/subscribe.html`
- Verifique nossa lista crescente de Resource Centers relacionados a C em:
  - `www.deitel.com/ResourceCenters.html`

1.1 Introdução (Cont.)

- Alguns dos ‘Resource Centers’ que serão valiosos enquanto você lê a parte de C deste livro são C, Code Search Engines and Code Sites, Computer Game Programming e Programming Projects. A cada semana, anunciamos nossos Resource Centers mais recentes no boletim.
- Erratas e atualizações para este livro são postadas em:Errata and updates for this book are posted at
  - `www.deitel.com/books/chtp6/`

1.2 Computadores: Hardware e Software

- Um computador é um dispositivo capaz de realizar cálculos e tomar decisões lógicas em velocidades bilhões de vezes mais rápidas que os seres humanos.
- Por exemplo, muitos dos computadores pessoais de hoje podem executar bilhões de adições por segundo.
- Os supercomputadores mais rápidos de hoje podem executar milhares de trilhões (quatrilhões) de instruções por segundo!
- Para se ter uma ideia, um computador que executa um quatrilhão de instruções por segundo pode realizar mais de 100.000 cálculos por segundo para cada pessoa no planeta!

1.2 Computadores: Hardware e Software (Cont.)

- Computadores processam dados sob o controle de conjuntos de instruções chamados programas de computador.
- Esses programas guiam o computador por meio de conjuntos ordenados de ações especificadas por pessoas a quem chamamos de programadores de computador.
- Um computador é composto de vários dispositivos (por exemplo, teclado, tela, mouse, disco rígido, memória, DVDs e unidades de processamento) que são denominados hardware.
- Os programas executados em um computador são chamados de software.
- Os custos de hardware têm diminuído drasticamente nos últimos anos, a ponto de os computadores pessoais terem se tornado um artigo comum.

1.2 Computadores: Hardware e Software (Cont.)

- Neste livro, você aprenderá métodos comprovados para o desenvolvimento de software que podem reduzir os custos do desenvolvimento de software — programação estruturada (nos capítulos de C) e programação orientada a objetos (nos capítulos de C++).

1.3 Organização dos Computadores

- Todo computador pode ser dividido em seis unidades lógicas ou seções:
- Unidade de entrada. Essa é a seção ‘receptora’ do computador. Ela obtém informações (dados e programas) de vários **dispositivos de entrada** e as coloca à disposição das outras unidades, de forma que possam ser processadas. A maioria das informações, hoje, é inserida nos computadores por meio de dispositivos como teclados e mouses. As informações também podem ser inseridas de muitas outras maneiras, incluindo comandos de voz, varredura de imagens e códigos de barras, leitura de dispositivos de armazenamento secundários (como disco rígidos, unidades de CD, unidades de DVD e unidades USB — também chamadas de ‘pen-drives’) e recebimento de informações pela Internet (por exemplo, quando você baixa vídeos do YouTube™, e-books da Amazon e outros meios similares).

1.3 Organização dos Computadores (cont.)

- Unidade de saída. Essa é a seção de ‘expedição’ do computador. Ela pega as informações que foram processadas pelo computador e as coloca em vários dispositivos de saída para torná-las disponíveis ao uso fora dele. A maior parte das informações que saem de computadores hoje em dia é exibida em telas, impressa, tocada em players de áudio (como os populares iPods® da Apple®) ou usada para controlar outros dispositivos. Os computadores também podem enviar suas informações para redes como a Internet.

1.3 Organização dos Computadores (cont.)

- Unidade de memória. Essa é a seção de ‘armazenamento’ de acesso rápido do computador, que tem uma capacidade relativamente baixa. Retém informações obtidas por meio da unidade de entrada, de forma que elas possam estar imediatamente disponíveis para processamento quando forem necessárias. A unidade de memória retém também informações processadas até que elas possam ser colocadas em dispositivos de saída pela unidade de saída. As informações na unidade de memória são voláteis: elas normalmente se perdem quando o computador é desligado. A unidade de memória é frequentemente chamada de memória ou memória primária..

1.3 Organização dos Computadores (cont.)

- Unidade lógica e aritmética (ULA, ou ALU, Arithmetic and Logic Unit). Essa é a seção de 'processamento' do computador. Ela é responsável por executar cálculos como adição, subtração, multiplicação e divisão. Também contém os mecanismos de decisãocque permitem ao computador, por exemplo, comparar dois itens na unidade de memória e determinar se eles são ou não iguais. Nos sistemas atuais, a ULA está próxima a UC, dentro da CPU.

1.3 Organização dos Computadores (cont.)

- Unidade central de processamento (CPU, Central Processing Unit). Essa é a seção 'administrativa' do computador. Ela coordena e supervisiona o funcionamento das outras seções. A CPU diz à unidade de entrada quando as informações devem ser lidas para a unidade de memória, diz à ALU quando as informações da unidade de memória devem ser utilizadas em cálculos e diz à unidade de saída quando enviar as informações da unidade de memória para certos dispositivos de saída. Muitos dos computadores de hoje possuem várias CPUs e, portanto, podem realizar muitas operações simultaneamente — esses computadores são chamados de multiprocessadores. Um processador multi-core executa o multiprocessamento em um único chip integrado — por exemplo, um processador dual-core tem duas CPUs e um processador quad-core tem quatro CPUs.

1.3 Organização dos Computadores (cont.)

- Unidade de armazenamento secundário. Essa é a seção de 'depósito' de grande capacidade e de longo prazo do computador. Os programas ou dados que não estão sendo ativamente utilizados pelas outras unidades são normalmente colocados em dispositivos de armazenamento secundário (por exemplo, os discos rígidos) até que sejam novamente necessários, possivelmente horas, dias, meses ou até anos mais tarde. Portanto, as informações nos dispositivos de armazenamento secundário são consideradas persistentes — elas são preservadas mesmo quando o computador é desligado. As informações no armazenamento secundário levam muito mais tempo para serem acessadas do que as informações na memória primária, mas o custo por unidade de armazenamento secundário é muito menor que o custo por unidade de memória primária. Alguns exemplos de dispositivos de armazenamento secundário são os CDs, os DVDs e as unidades flash (algumas vezes chamadas de cartões de memória), que podem guardar centenas de milhões ou bilhões de caracteres.

1.4 Computação pessoal, distribuída e cliente/servidor

- Em 1977, a Apple Computer popularizou a computação pessoal.
- Em 1981, a IBM, o maior vendedor de computadores do mundo, introduziu o IBM Personal Computer (PC).
- Isso rapidamente disponibilizou a computação pessoal nas organizações comerciais, industriais e do governo, onde os mainframes da IBM eram muito utilizados.
- Esses computadores eram unidades 'isoladas'; as pessoas transportavam discos para lá e para cá a fim de compartilhar informações (isso era frequentemente chamado de 'sneakernet').
- Essas máquinas poderiam ser conectadas para formar redes de computadores, às vezes por linhas telefônicas e às vezes em redes locais (LANs, local area networks) dentro de uma organização.

1.4 Computação pessoal, distribuída e cliente/servidor (Cont.)

- Isso levou ao fenômeno da computação distribuída.
- As informações são facilmente compartilhadas entre as redes, onde computadores chamados de servidores (servidores de arquivos, servidores de banco de dados, servidores Web etc.) oferecem capacidades que podem ser utilizadas pelos computadores clientes distribuídos pela rede; daí o termo computação cliente/servidor.
- C é bastante usada para escrever os softwares para sistemas operacionais, redes de computadores e aplicações cliente/servidor distribuídas.

1.5 A Internet e a World Wide Web

- Com a introdução da World Wide Web — que permite que os usuários de computadores localizem e vejam documentos baseados em multimídia sobre quase todos os assuntos pela Internet —, a Internet explodiu e tornou-se o principal mecanismo de comunicação do mundo.
- As aplicações de hoje podem ser escritas para que se comuniquem com computadores espalhados pelo mundo inteiro.

1.6 Linguagens de máquina, simbólicas e de alto nível

- Os programadores escrevem instruções em várias linguagens de programação; algumas claramente compreensíveis por meio do computador e outras que exigem passos de tradução intermediários.
- As linguagens de computador podem ser divididas em três tipos gerais:
  - Linguagens de máquina
  - Linguagens simbólicas
  - Linguagens de alto nível
- Um computador pode entender com clareza somente sua própria linguagem de máquina.
- A linguagem de máquina é a 'linguagem natural' de um computador em particular, e é definida pelo projeto de hardware daquela máquina.

1.6 Linguagens de máquina, simbólicas e de alto nível (Cont.)

- A linguagem de máquina normalmente é chamada de código objeto.
- As linguagens de máquina consistem geralmente em sequências de números (em última instância, reduzidos a ls e 0s) que instruem os computadores a executar suas operações mais elementares, uma de cada vez.
- As linguagens de máquina são dependentes da máquina, isto é, uma linguagem de máquina em particular só pode ser usada em um tipo de computador.

1.6 Linguagens de máquina, simbólicas e de alto nível (Cont.)

- As linguagens de máquina são incômodas para as pessoas, como pode ser visto na seguinte seção de um dos primeiros programas de linguagem de máquina que soma horas extras ao salário básico a pagar e armazena o resultado em pagamento bruto:
  - +1300042774
    +1400593419
    +1200274027
- Em vez de usar as sequências de números que os computadores podiam entender com clareza, os programadores começaram a usar abreviações semelhantes às das palavras inglesas para representar as operações elementares do computador.
- Essas abreviações formaram a base das linguagens simbólicas (ou *assembly*).

1.6 Linguagens de máquina, simbólicas e de alto nível (Cont.)

- Programas tradutores, chamados de assemblers (montadores), foram desenvolvidos para converter os programas, à velocidade do computador, em linguagem simbólica na linguagem de máquina.
- A seção de um programa em linguagem simbólica mostrada a seguir também soma horas extras ao salário básico a pagar e armazena o resultado em pagamento bruto:

```
load    salario
add     horaExtra
store   valorTotal
```

- Embora tal código seja mais claro para as pessoas, ele será incompreensível para os computadores até que seja traduzido para a linguagem de máquina.

1.6 Linguagens de máquina, simbólicas e de alto nível (Cont.)

- O uso de computadores aumentou rapidamente com o advento das linguagens simbólicas, mas ainda eram exigidas muitas instruções para realizar até as tarefas mais simples.
- Para acelerar o processo de programação, foram desenvolvidas as linguagens de alto nível, por meio das quais uma única instrução realizaria tarefas significativas.
- Programas tradutores, chamados compiladores, convertem os programas em linguagem de alto nível na linguagem de máquina.
- As linguagens de alto nível permitem que os programadores escrevam instruções que se parecem muito com o inglês comum e contêm notações matemáticas comumente usadas.

1.6 Linguagens de máquina, simbólicas e de alto nível (Cont.)

- Um programa de folha de pagamento escrito em uma linguagem de alto nível pode conter um comando como:
  - `valorTotal = salario + horaExtra;`
- C, C++, as linguagens .NET da Microsoft (como o Visual Basic, o Visual C++ e o Visual C#) e Java estão entre as mais poderosas e amplamente utilizadas linguagens de programação de alto nível.
- Foram desenvolvidos programas interpretadores que podem executar diretamente programas em linguagem de alto nível (sem o atraso da compilação), embora mais lentamente do que ocorre com os programas compilados.

1.7 A história de C

- C evoluiu de duas linguagens de programação anteriores, BCPL e B.
- A BCPL foi desenvolvida em 1967 por Martin Richards como uma linguagem para escrever software de sistemas operacionais e compiladores.
- Ken Thompson modelou muitas das características de sua linguagem B inspirado por suas correspondentes em BCPL, e utilizou B para criar as primeiras versões do sistema operacional UNIX no Bell Laboratories, em 1970.
- Tanto a BCPL como a B eram linguagens ‘sem tipo’, ou seja, sem definição de tipos de dados — todo item de dados ocupava uma ‘palavra’ na memória, e o trabalho de tratar um item de dados como um número inteiro ou um número real, por exemplo, era de responsabilidade do programador.

1.7 A história de C (Cont.)

- A linguagem C foi deduzida de B por Dennis Ritchie no Bell Laboratories, e originalmente implementada em um computador DEC PDP-11 em 1972.
- Inicialmente, C tornou-se conhecida como a linguagem de desenvolvimento do sistema operacional UNIX.
- Hoje, quase todos os sistemas operacionais são escritos em C e/ou C++. C está disponível para a maioria dos computadores.
- C é independente de hardware.
- Com um projeto cuidadoso, é possível escrever programas em C que sejam portáveis para a maioria dos computadores.

1.7 A história de C (Cont.)

- No final dos anos 1970, C evoluiu para o que agora é chamado de 'C tradicional'. A publicação do livro de Kernighan e Ritchie, *The C Programming Language*, pela Prentice Hall, em 1978, chamou muita atenção para a linguagem.
- A rápida expansão de C por vários tipos de computadores (às vezes chamados de plataformas de hardware) levou a muitas variações da linguagem que, embora semelhantes, eram frequentemente incompatíveis.
- Em 1989, o padrão foi aprovado; esse padrão foi atualizado em 1999.
- C99 é um padrão revisado para a linguagem de programação C, que aperfeiçoa e expande as capacidades da linguagem.

1.7 A história de C (Cont.)

- Nem todos os compiladores C populares admitem C99.
- Daqueles que admitem, a maioria implementa apenas um subconjunto dos novos recursos.
- Os capítulos 1 a 14 deste livro se baseiam no padrão internacional C (ANSI/ISO), amplamente adotado.
- O Apêndice G apresenta o C99 e oferece links para os principais compiladores e IDEs C99.

Dica de portabilidade 1.1

Como C é uma linguagem independente de hardware amplamente disponível, os aplicativos escritos em C podem ser executados com pouca ou nenhuma modificação em uma grande variedade de sistemas de computação diferentes.

1.8 A biblioteca-padrão de C

- Como veremos no Capítulo 5, os programas em C consistem em módulos ou peças chamadas funções.
- Você pode programar todas as funções de que precisa para formar um programa em C, mas a maioria dos programadores de C aproveita as ricas coleções de funções existentes na biblioteca-padrão de C.
- Visite o site indicado a seguir para obter a documentação completa da biblioteca-padrão de C, incluindo os recursos da C99:
  - www.dinkumware.com/manuals/default.aspx#Standard%20C%20Library
- Este livro encoraja uma abordagem de blocos de montagem (isto é, programação estruturada) para a criação de programas.

1.8 A biblioteca-padrão de C (Cont.)

- Evite reinventar a roda.
- Em vez disso, use as partes existentes — isso se chama reutilização de software, e é uma solução para o campo da programação orientada a objetos, conforme veremos em nossa explicação de C++.
- Ao programar em C, você normalmente utilizará os seguintes blocos de montagem:
  - Funções da biblioteca-padrão de C.
  - Funções que você mesmo criar.
  - Funções que outras pessoas criaram e tornaram disponíveis para você.

1.8 A biblioteca-padrão de C (Cont.)

- Se você utilizar funções existentes, poderá evitar a reinvenção da roda.
- No caso das funções em C ANSI ou C padrão, você sabe que elas foram cuidadosamente escritas, e sabe que, como está usando funções que estão disponíveis em praticamente todas as execuções de C padrão, seus programas terão uma chance maior de serem portáveis e livres de erros.

Dica de desempenho 1.1

Usar funções da biblioteca-padrão de C em vez de escrever suas próprias versões equivalentes pode melhorar o desempenho do programa, porque essas funções foram cuidadosamente escritas para serem executadas de modo eficaz.

Dica de desempenho 1.2

Usar funções e classes de bibliotecas-padrão, em vez de escrever suas próprias versões equivalentes, pode melhorar a portabilidade do programa, porque essas funções estão incluídas em praticamente todas as implementações-padrão de C.

1.9 C++

- C++ foi desenvolvida por Bjarne Stroustrup no Bell Laboratories.
- C++ tem suas raízes em C e apresenta várias características que melhoram a linguagem C, mas o mais importante é que fornece recursos para a programação orientada a objetos.
- Objetos são, essencialmente, componentes de software reutilizáveis que modelam itens do mundo real.
- O uso de uma abordagem de implementação e projeto modulares orientada a objetos pode tornar os grupos de desenvolvimento de software muito mais produtivos do que é possível, utilizando técnicas de programação que não suportam orientação a objetos.

1.9 C++ (Cont.)

- Nos capítulos 15 a 24 deste livro, apresentamos um tratamento resumido de C++, selecionado de nosso livro *C++: Como Programar, 7/e*.
- Ao estudar C++, verifique o conteúdo sobre C++ em nosso Resource Center on-line em: `www.deitel.com/cplusplus/`.

1.10 Java

- Microprocessadores vêm provocando um impacto profundo nos aparelhos eletrodomésticos e eletrônicos de consumo.
- Reconhecendo isso, em 1991, a Sun Microsystems desenvolveu uma linguagem baseada em C++ que passou a ser denominada Java.
- A popularidade da World Wide Web explodiu em 1993, e o pessoal da Sun viu o potencial imediato de usar a Java para criar páginas da Web com o chamado conteúdo dinâmico (por exemplo, interatividade, animações e coisas desse gênero).
- A Java atraiu a atenção da comunidade comercial devido ao interesse fenomenal na World Wide Web.

1.10 Java (Cont.)

- A Java agora é usada para desenvolver aplicativos empresariais de grande porte, para aumentar a funcionalidade de servidores Web (os computadores que oferecem o conteúdo que vemos em nossos navegadores da Web), para oferecer aplicativos a aparelhos eletrônicos de consumo (tais como telefones celulares, pagers e assistentes pessoais digitais) e muitas outras finalidades.

1.11 Fortran, COBOL, Pascal e Ada

- Centenas de linguagens de alto nível foram desenvolvidas, mas só algumas obtiveram ampla aceitação.
- A FORTRAN (FORmula TRANslator) foi desenvolvida pela IBM Corporation, em meados da década de 1950, para ser usada no desenvolvimento de aplicativos científicos e de engenharia que exigem cálculos matemáticos complexos.
- A FORTRAN ainda é amplamente usada, especialmente em aplicativos de engenharia.
- A COBOL (COmmon Business Oriented Language) foi desenvolvida no final da década de 1950 por fabricantes de computadores, pelo governo dos EUA e por usuários de computadores industriais.

1.11 Fortran, COBOL, Pascal e Ada (Cont.)

- A COBOL é utilizada principalmente em aplicativos comerciais que exigem manipulação precisa e eficiente de grandes quantidades de dados.
- Muitos softwares para aplicações comerciais ainda são programados em COBOL.
- Durante a década de 1960, o esforço para desenvolvimento de softwares de grande porte encontrou sérias dificuldades.
- As pessoas se deram conta de que o desenvolvimento de software era uma atividade mais complexa do que tinham imaginado.
- Durante essa década, a pesquisa resultou na evolução da programação estruturada — um enfoque disciplinado para a escrita de programas mais claros e mais fáceis de testar, depurar e modificar do que os grandes programas produzidos com técnicas anteriores.

1.11 Fortran, COBOL, Pascal e Ada (Cont.)

- Um dos resultados mais tangíveis dessa pesquisa foi o desenvolvimento da linguagem de programação Pascal, pelo professor Niklaus Wirth, em 1971.
- Em homenagem ao matemático e filósofo do século XVII, Blaise Pascal, ela foi projetada para o ensino de programação estruturada e logo se tornou a linguagem de programação preferida na maioria das faculdades.
- Pascal não tinha muitos dos recursos necessários nas aplicações comerciais, industriais e do governo, de modo que não foi muito bem aceita fora dos círculos acadêmicos.

1.11 Fortran, COBOL, Pascal e Ada (Cont.)

- A linguagem Ada foi desenvolvida com o patrocínio do Departamento de Defesa (DoD) dos Estados Unidos durante a década de 1970 e início dos anos 1980.
- Centenas de linguagens separadas eram usadas para produzir os maciços sistemas de software de comando e controle do DoD.
- O departamento queria uma linguagem que se ajustasse à maior parte de suas necessidades.

1.11 Fortran, COBOL, Pascal e Ada (Cont.)

- A linguagem Ada recebeu esse nome em homenagem à Lady Ada Lovelace, filha do poeta Lord Byron.
- O primeiro programa de computador do mundo, datado do início do século XIX (para o Analytical Engine Mechanical Computing Device, projetado por Charles Babbage), foi escrito por ela.
- Um recurso importante da Ada é chamado de multitarefa; ela permite que os programadores especifiquem que muitas atividades devem acontecer em paralelo.

1.12 BASIC, Visual Basic, Visual C++, C# e .NET

- A linguagem de programação BASIC (Beginner's All-purpose Symbolic Instruction Code) foi desenvolvida em meados da década de 1960 no Dartmouth College, como um meio para escrever programas simples.
- A principal finalidade do BASIC era familiarizar os iniciantes com as técnicas de programação.
- A linguagem Visual Basic da Microsoft, introduzida no início da década de 1990 para simplificar o desenvolvimento de aplicações para Microsoft Windows, tornou-se uma das linguagens de programação mais populares do mundo.

1.13 BASIC, Visual Basic, Visual C++, C# e .NET (Cont.)

- As ferramentas de desenvolvimento mais recentes da Microsoft fazem parte de sua estratégia corporativa de integrar a Internet e a Web em aplicações de computador.
- Essa estratégia é implementada na plataforma .NET da Microsoft, que oferece as capacidades que os responsáveis pelo desenvolvimento precisam para criar e executar aplicações de computador que possam funcionar em computadores que estão distribuídos pela Internet.
- As três principais linguagens de programação da Microsoft são Visual Basic (baseada originalmente na linguagem de programação BASIC), Visual C++ (baseada no C++) e Visual C# (uma nova linguagem baseada em C++ e Java, desenvolvida exclusivamente para a plataforma .NET).
- Visual C++ também pode ser usada para compilar e executar programas em C.

1.14 Tendência-chave em software: tecnologia de objetos

- A linguagem de programação C++, desenvolvida na AT&T por Bjarne Stroustrup no início da década de 1980, é baseada em duas linguagens: C e Simula 67, uma linguagem de programação de simulação desenvolvida no Norwegian Computing Center e lançada em 1967.
- C++ absorveu as características da C e acrescentou as capacidades da Simula para a criação e a manipulação de objetos.
- A tecnologia de objeto é um esquema de empacotamento que ajuda a criar unidades de software significativas.
- Existem objetos de data, de hora, de contracheque, de fatura, de áudio, de vídeo, de arquivo, de registro, e assim por diante.

1.14 Tendência-chave em software: tecnologia de objetos (Cont.)

- Na verdade, quase todo substantivo pode ser representado de forma razoável como um objeto.
- Vivemos em um mundo de objetos.
- Existem carros, aviões, pessoas, animais, prédios, semáforos, elevadores etc.
- Antes do surgimento das linguagens orientadas a objetos, as linguagens de programação procedural (como Fortran, COBOL, Pascal, BASIC e C) focalizavam as ações (verbos) em vez das coisas ou objetos (substantivos).
- Os programadores, vivendo em um mundo de objetos, programavam usando verbos, principalmente.
- Isso tornava a escrita de programas complicada.

1.14 Tendência-chave em software: tecnologia de objetos (Cont.)

- Agora, com a disponibilidade de linguagens orientadas a objetos bastante populares, como C++, Java e C#, os programadores continuam a viver em um mundo orientado a objetos e podem programar de uma maneira orientada a objetos.
- Esse é um processo mais natural do que a programação procedural, e resultou em ganhos de produtividade importantes.
- Um problema significativo da programação procedural é que as unidades de programa não espelham as entidades reais de maneira eficaz, de modo que essas unidades não são particularmente reutilizáveis.

1.14 Tendência-chave em software: tecnologia de objetos (Cont.)

- Não é incomum que os programadores ‘comecem do nada’ a cada novo projeto e tenham que escrever um software semelhante a partir ‘do zero’. Isso desperdiça tempo e dinheiro, pois as pessoas ‘reinventam a roda’ repetidamente. Com a técnica de programação orientada a objeto, as entidades de software criadas (chamadas classes), se projetadas corretamente, tendem a ser reutilizáveis em projetos futuros.
- O uso de bibliotecas de componentes reutilizáveis
- pode reduzir bastante o esforço exigido para implementar certos tipos de sistemas.

Observação sobre engenharia de software 1.1

Grandes bibliotecas de classes com componentes de software reutilizáveis estão disponíveis na Internet. Muitas dessas bibliotecas são gratuitas.

1.14 Tendência-chave em software: tecnologia de objetos (Cont.)

- Algumas organizações relatam que o principal benefício da programação orientada a objetos não é a reutilização de software, mas sim que o software que elas produzem é mais inteligível, mais bem organizado e mais fácil de manter, modificar e depurar.
- Isso pode ser significativo, pois talvez até 80 por cento dos custos do software estejam associados não com os esforços originais para desenvolver o software, mas com a evolução e a manutenção contínuas desse software por todo o seu tempo de vida..
- Quaisquer que sejam os benefícios percebidos, é evidente que a programação orientada a objetos será a metodologia-chave de programação por muitas décadas.

1.15 Ambiente de desenvolvimento de programa típico em C

- Os sistemas de C++ geralmente consistem em várias partes: um ambiente de desenvolvimento de programas, a linguagem e a biblioteca-padrão de C.
- Normalmente, os programas em C percorrem seis passos até que possam ser executados (Figura 1.1).
- São os seguintes: editar, pré-processar, compilar, 'ligar', carregar e executar.
- Nós nos concentramos aqui em um sistema de C sob Linux típico, embora este seja um livro genérico sobre C.

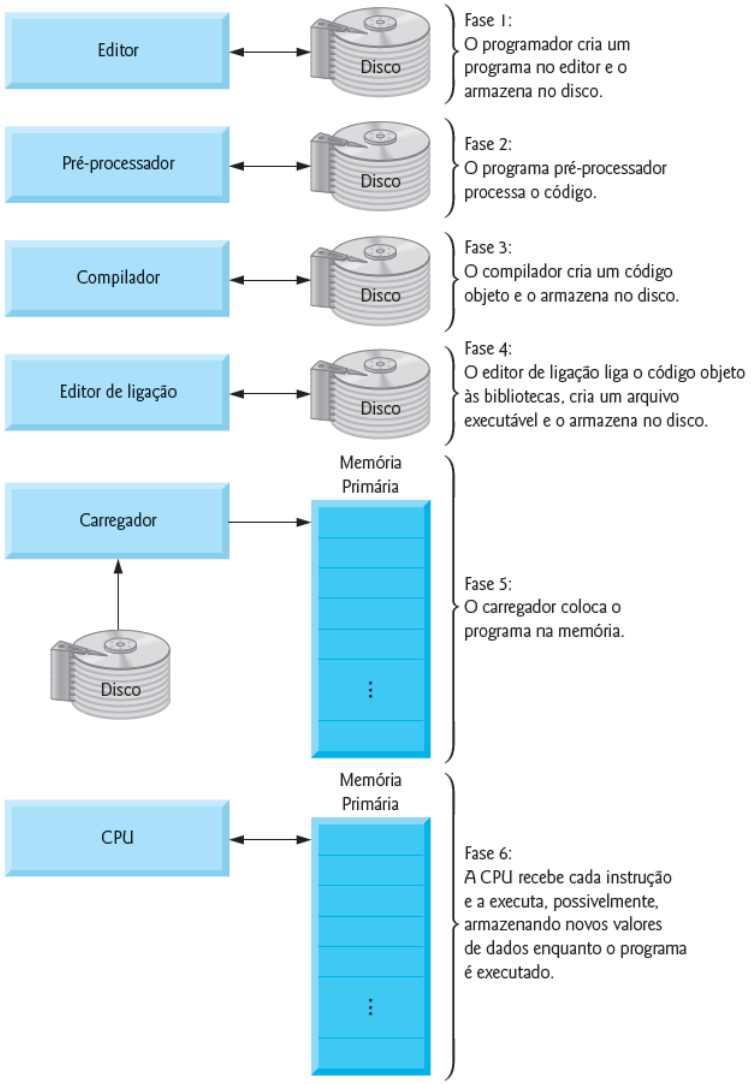

Figura 1.1 ■ Ambiente típico de desenvolvimento em C.

1.15 Ambiente de desenvolvimento de programa típico em C (Cont.)

- [Nota: os programas expostos neste livro executarão com pouca ou nenhuma modificação na maioria dos sistemas C atuais, inclusive em sistemas baseados no Microsoft Windows.] Se você não estiver usando um sistema Linux, consulte os manuais de seu sistema ou pergunte a seu instrutor como realizar essas tarefas em seu ambiente.
- Verifique nosso Resource Center de C em `www.deitel.com/C` para localizar tutoriais referentes a compiladores e ambientes de desenvolvimento em C populares para iniciantes.
- Isso é realizado por um programa editor.

1.15 Ambiente de desenvolvimento de programa típico em C (Cont.)

- Dois editores bastante usados em sistemas Linux são *vi* e *emacs*.
- Pacotes de software para os ambientes de desenvolvimento de programa integrados C/C++, como Eclipse e Microsoft Visual Studio, possuem editores bastante integrados ao ambiente de programação.
- Você digita um programa em C utilizando o editor e, se necessário, faz correções. O programa, então, é 'guardado' em um dispositivo de armazenamento secundário como, por exemplo, um disco rígido.
- Os nomes dos arquivos de programas em C deverão terminar com a extensão .c.

1.15 Ambiente de desenvolvimento de programa típico em C (Cont.)

- Na Fase 2, você dá o comando para compilar o programa.
- O compilador traduz o programa em C para um código em linguagem de máquina (também chamado de código objeto).
- Em um sistema C, um programa pré-processador é executado automaticamente, antes de começar a fase de tradução pelo compilador.
- O pré-processador C obedece a comandos especiais, chamados de diretivas do pré-processador, que indicam que certas manipulações devem ser feitas no programa antes da compilação.

1.15 Ambiente de desenvolvimento de programa típico em C (Cont.)

- Essas manipulações normalmente consistem em incluir outros arquivos de texto no arquivo a ser compilado e executar substituições de texto variadas.
- As diretivas mais comuns do pré-processador serão discutidas nos capítulos iniciais; uma discussão detalhada de todas as características do pré-processador aparecerá no Capítulo 13.
- Na Fase 3, o compilador traduz o programa C para um código em linguagem de máquina.

1.15 Ambiente de desenvolvimento de programa típico em C (Cont.)

- A próxima fase é chamada de ligação (link).
- Os programas em C normalmente contêm referências a funções definidas em outro lugar, como nas bibliotecas-padrão ou nas bibliotecas privadas de grupos de programadores que trabalham juntos em um projeto particular.
- O código objeto produzido pelo compilador C normalmente contém 'buracos' por causa dessas partes que faltam.
- Um editor de ligação (ou linker) liga o código objeto com o código das funções que estão faltando para produzir uma imagem executável (sem pedaços faltando).
- Em um sistema Linux típico, o comando para compilar e 'ligar' um programa é chamado de cc (ou gcc).

1.15 Ambiente de desenvolvimento de programa típico em C (Cont.)

- Para compilar e 'ligar' um programa chamado bem-vindo.c, digite:
  - `cc welcome.c`
- no prompt do Linux e pressione a tecla *Enter* (ou *Return*).
- [Nota: os comandos do Linux diferenciam maiúsculas de minúsculas; não se esqueça de digitar o c minúsculo e cuide para que as letras no nome do arquivo estejam corretas.]
- Se o programa compilar e 'ligar' corretamente, é produzido um arquivo chamado a.out.
- Essa é a imagem executável de nosso programa bem-vindo.c.

1.15 Ambiente de desenvolvimento de programa típico em C (Cont.)

- A próxima fase é chamada de carga (ou *loading*).
- Antes de um programa ser executado ele deve ser colocado na memória.
- Isso é feito pelo carregador (ou *loader*), que pega a imagem executável do disco e a transfere para a memória.
- Componentes adicionais de bibliotecas compartilhadas que oferecem suporte ao programa do usuário também são carregados.
- Finalmente, o computador, sob o controle de sua CPU, executa o programa, uma instrução por vez.

1.15 Ambiente de desenvolvimento de programa típico em C (Cont.)

- Para carregar e executar o programa em um sistema Linux, digite ./a.out no prompt e pressione *Enter*.
- Os programas nem sempre funcionam na primeira tentativa.
- Cada uma das fases precedentes pode falhar por causa de vários erros que abordaremos a seguir.
- Por exemplo, um programa, ao ser executado, poderia tentar dividir por zero (uma operação ilegal em computadores, da mesma maneira que é um valor não definido em aritmética).
- Isso faria o computador exibir uma mensagem de erro.

1.15 Ambiente de desenvolvimento de programa típico em C (Cont.)

- Você, então, retornaria à fase de edição, faria as correções necessárias e passaria novamente pelas fases restantes para determinar se as correções estariam funcionando de maneira apropriada.
- A maioria dos programas em C faz entrada e/ou saída de dados.
- Certas funções de C recebem sua entrada de stdin (fluxo de entrada padrão), o qual normalmente é o teclado, porém stdin pode ser conectado a outro fluxo.
- Os dados geralmente são enviados para saída em stdout (fluxo de saída padrão), que normalmente é a tela do computador, mas stdout pode ser conectado a outro fluxo.
- Quando dizemos que um programa imprime um resultado, normalmente queremos dizer que o resultado é exibido na tela.

1.15 Ambiente de desenvolvimento de programa típico em C (Cont.)

- Os dados podem ser enviados para a saída por meio de outros dispositivos, como discos e impressoras.
- Existe também um fluxo de erros padrão, chamado de stderr.
- O fluxo stderr (normalmente conectado à tela) é usado para exibir mensagens de erro.
- É comum que os usuários direcionem os dados de saída normais, isto é, stdout, para um dispositivo diferente da tela ao mesmo tempo que mantêm stderr direcionado para a tela, de maneira que o usuário possa ser imediatamente informado de erros.

Erro comum de programação 1.1
Erros como os de divisão por zero acontecem quando um programa está sendo executado; por isso, esses erros são chamados de 'erros durante a execução' (ou erros de 'runtime'). Dividir por zero é geralmente um erro fatal, isto é, um erro que causa o término imediato do programa sem que ele tenha executado seu trabalho com sucesso. Os erros não fatais permitem que os programas sejam executados até a conclusão, geralmente produzindo resultados incorretos.

1.16 Tendências de hardware

- A cada um ou dois anos, as capacidades dos computadores quase dobraram sem que houvesse qualquer aumento no preço.
- Isso normalmente é chamado de Lei de Moore, que recebeu esse nome em homenagem à pessoa que identificou e explicou essa tendência, Gordon Moore, cofundador da Intel, a empresa que fabrica a grande maioria dos processadores usados nos computadores pessoais atuais.

1.16 Tendências de hardware (Cont.)

- A Lei de Moore e tendências semelhantes são especialmente verdadeiras em relação à quantidade de memória para o armazenamento de programas que os computadores possuem, à quantidade de espaço para armazenamento secundário (tal como armazenamento em disco), que é utilizado para armazenar programas e dados em longo prazo, e às velocidades de seus processadores — as velocidades nas quais os computadores executam seus programas (isto é, fazem seu trabalho).
- O mesmo tem acontecido no campo das comunicações, qual os custos despencaram com a demanda por largura de banda nas comunicações, que atraiu uma enorme competição.

1.17 Notas sobre C e este livro

- Este livro é dirigido aos programadores novatos e, portanto, enfatizamos a clareza do programa.
- Nossa primeira ‘boa prática de programação’ é a seguinte:

Boa prática de programação 1.1
Escreva seus programas em C de uma maneira simples e direta. Às vezes, isso é chamado de KIS (‘Keep It Simple’ — ‘mantenha a simplicidade’). Não ‘force’ a linguagem tentando usos estranhos.

1.17 Notas sobre C e este livro (Cont.)

- Você ouviu dizer que C é uma linguagem portável, e que os programas escritos em C podem ser executados em muitos computadores diferentes.
- *Portabilidade é um objetivo difícil de ser atingido.*

Dica de portabilidade 1.2

Embora seja possível escrever programas portáveis em C, existem muitos problemas entre compiladores C diferentes e computadores diferentes, que podem tornar a portabilidade difícil de ser alcançada. Simplesmente escrever programas em C não garante portabilidade. Você, com frequência, precisará lidar diretamente com variações de computador.

1.17 Notas sobre C e este livro (Cont.)

- C é uma linguagem rica, e existem algumas sutilezas na linguagem e alguns assuntos avançados que não abordamos.
- Se você precisar de detalhes técnicos adicionais sobre C, sugerimos que leia o documento padrão C ou o livro de Kernighan e Ritchie: *The C Programming Language, Second Edition*.

Observação sobre engenharia de software 1.2

Leia os manuais para a versão de C que você está usando. Consulte-os com frequência, para se certificar de que está ciente da rica relação de recursos que C oferece e de que está usando esses recursos corretamente.

Observação sobre engenharia de software 1.3

Seu computador e seu compilador são bons professores. Se você não tiver certeza de como funciona um recurso de C, experimente compilar e executar o programa e veja o que acontece.