Capítulo 1

Computadores, pessoas e programação

"Especialização é para insetos."

— R. A. Heinlein

Neste capítulo apresentamos algumas das coisas que achamos que tornam a programação importante, interessante e divertida. Também apresentamos algumas ideias e ideiais fundamentais. Esperamos derrubar alguns mitos populares sobre a programação e os programadores. Este é um capítulo para passar os olhos e depois retornar quando você estiver lutando com algum problema de programação e se perguntando se vale a pena.

1.1 Introdução
1.2 Software
1.3 Pessoas
1.4 Ciência da Computação
1.5 Computadores estão em toda parte
1.5.1 Com telas e sem telas
1.5.2 Navegação
1.5.3 Telecomunicações
1.5.4 Medicina
1.5.5 Informação
1.5.6 Uma visão vertical
1.5.7 E então?
1.6 Ideais para programadores

1.1 Introdução

Como quase todo o aprendizado, aprender a programar é um problema do tipo ovo e galinha: queremos começar, mas também queremos saber por que aquilo que estamos prestes a aprender é importante. Queremos aprender uma habilidade prática, mas também nos certificar de que não é apenas uma moda passageira. Queremos saber que não iremos desperdiçar nosso tempo, mas não queremos ser aborrecidos por propagandas enganosas e cheias de moral. Por ora, apenas leia o que lhe pareça interessante neste capítulo e volte mais tarde quando você sentir a necessidade de refrescar a memória sobre por que os detalhes técnicos são importantes fora da sala de aula.

Este capítulo reflete nossa opinião sobre aquilo que julgamos ser interessante e importante na programação. Ele explica a motivação de nos mantermos neste campo por décadas. Este é um capítulo para ter uma ideia sobre os objetivos fundamentais e uma ideia a respeito de que tipo de pessoa um programador deve ser. Um livro técnico para iniciantes inevitavelmente contém muito mais material básico. Neste capítulo, passamos por cima dos detalhes técnicos e consideramos o panorama geral. Por que a programação é uma atividade que vale a pena? Qual é o papel da programação em nossa civilização? Em que áreas o programador pode orgulhar-se de sua contribuição? Onde a programação se enquadra no grande mundo do desenvolvimento, distribuição e manutenção de software? Onde a programação se enquadra quando as pessoas falam sobre “Ciência da Computação”, “Engenharia de Software”, “Tecnologia da Informação”, etc.? O que faz um programador? Que habilidades um bom programador possui?

Para um estudante, a razão mais urgente para entender uma ideia, uma técnica ou um capítulo pode ser a de passar em um exame com uma nota boa – mas devem haver mais razões para aprender além dessa! Para alguém que trabalhe na indústria de software, a razão mais urgente para entender uma ideia, uma técnica ou um capítulo pode ser a de encontrar algo que possa ajudá-lo em um projeto em andamento – e que não irá aborrecer o chefe que tem em mãos o próximo contracheque, as promoções e as demissões – mas tem que haver mais

razões para aprender além dessas! Trabalhamos melhor quando sabemos que o nosso trabalho de alguma maneira torna o mundo melhor para as pessoas viverem. Para tarefas que temos que realizar durante alguns anos (as "coisas" que constituem as profissões e carreiras), ideais e ideias mais abstratas são cruciais.

Nossa civilização depende de software. Melhorar o software e encontrar novos usos para o software são duas das maneiras em que os indivíduos podem ajudar a melhorar a vida de muitos. A programação tem um papel essencial nisso.

1.2 Software

O bom software é invisível. Você não pode vê-lo, senti-lo, pesá-lo ou bater nele. O software é uma coleção de programas em execução em alguns computadores. Às vezes, podemos ver o computador. Muitas vezes, podemos ver somente algo que contém o computador, como um telefone, uma câmera, uma máquina de fazer pão, um carro ou uma turbina eólica. Podemos ver o que o software faz. Podemos ficar chateados ou incomodados se ele não faz aquilo que deveria fazer. Podemos ficar chateados ou incomodados se o que ele deveria fazer não atende às nossas necessidades.

Quantos computadores existem no mundo? Não sabemos; no mínimo, bilhões. Devem existir mais computadores do que pessoas no mundo. Uma estimativa de 2004 da ITU (International Telecommunication Union, uma agência das Nações Unidas) indica 772 milhões de PCs e a maioria dos computadores não são PCs.

Quantos computadores você usa (mais ou menos diretamente) a cada dia? Existem mais de 30 computadores no meu carro, dois no meu telefone celular, um no meu aparelho de MP3 e um na minha câmera. Ainda há o meu *notebook* (no qual foi escrita a página que você está lendo) e o meu computador de mesa. O controlador do ar condicionado que mantém o calor e a umidade sob controle é um computador simples. Existe um controlando o elevador do departamento de Ciência da Computação. Se você usa uma televisão moderna, existe pelo menos um computador em algum lugar dentro dela. Uma curta navegação na Web leva você a fazer contato com dezenas – possivelmente centenas – de servidores através de um sistema de telecomunicações formado por milhares de computadores – comutadores de telefonia, roteadores e assim por diante.

Não, eu não dirijo com 30 *notebooks* no assento traseiro do meu carro! A questão é que a maioria dos computadores não se parece com a imagem popular de um computador (com uma tela, um teclado, um mouse, etc.); são pequenas "peças" embutidas no tipo de equipamento que usamos. Portanto, esse carro não tem nada que se pareça com um computador, nem mesmo uma tela para exibir mapas e instruções de percurso (embora tais dispositivos sejam populares em outros carros). Entretanto, o motor contém diversos computadores que fazem coisas tais como controlar a injeção de combustível e monitorar a temperatura. O piloto automático envolve pelo menos um computador, o rádio e o sistema de segurança possuem mais alguns e suspeitamos que os controles de abrir/fechar janelas sejam controlados por computadores. Os modelos novos têm até computadores que monitoram continuamente a pressão dos pneus.

De quantos computadores você depende para executar as suas atividades diárias? Você come; se vive em uma cidade moderna, conseguir a sua comida é um grande esforço, que envolve pequenos milagres de planejamento, transporte e armazenamento. O gerenciamento das redes de distribuição é sem dúvida computadorizado, assim como são os sistemas de comunicação que conectam tudo isso. As fazendas modernas são altamente computadorizadas; junto ao galpão você encontra computadores usados para monitorar o rebanho (idade, saúde, produção de leite, etc.); equipamentos de fazendas são cada vez mais computadorizados, e a quantidade de formulários exigidos pelos vários serviços do governo podem fazer qualquer fazendeiro honesto chorar. Se alguma coisa vai mal, você pode ler tudo a respeito dela no jornal; com certeza, os artigos do jornal foram escritos em computadores, colocados nas páginas por computadores e impressos por equipamento computadorizado (se você ainda lê a "edição árvore morta") – muitas vezes após terem sido transmitidos eletronicamente para o local de impressão. Livros são produzidos da mesma maneira. Se você se desloca para outra cidade, os fluxos de tráfego são monitorados por computadores na tentativa (geralmente vã) de evitar congestionamentos. Você prefere usar o trem? O trem também vai ser computadorizado; alguns até operam sem condutor, e os subsistemas do trem, como os anúncios, os vagões e os tíquetes envolvem muitos computadores. A indústria de entretenimento atual (música, filmes, televisão, shows) está entre os maiores usuários de computadores. Mesmo os filmes que não são desenhos animados usam pesadamente a animação (computadorizada); a música e a fotografia também tendem a ser digitais (isto é, usar computadores) tanto na gravação quanto na distribuição. Se você adoecer, todos os exames que seu médico solicitar irão envolver computadores; os registros médicos são muitas vezes computadorizados e a maior parte do equipamento médico que você vai encontrar se for para um hospital para ser tratado contém computadores. A menos que você esteja em uma cabana na floresta sem acesso a equipamentos movidos a energia elétrica (incluindo lâmpadas) você usa energia. O óleo é descoberto, extraído, processado e distribuído através de sistemas que usam computadores em cada etapa ao longo do caminho, desde a perfuradora na terra até a sua bomba local de gasolina (petróleo). Se você compra gasolina com um cartão de crédito, você novamente põe em uso todo um conjunto de computadores. É a mesma história para carvão, gás, energia solar e eólica.

Os exemplos até agora são todos "operacionais"; estão diretamente ligados ao que você está fazendo. Fora disso, existe a importante e interessante área de design. As roupas que você veste, o aparelho telefônico que você usa, a máquina da fazer café que lhe propicia sua bebida favorita foram projetadas e fabricadas usando computadores. A qualidade superior das lentes fotográficas modernas e as formas requintadas dos equipamentos e utensílios modernos de uso diário se devem principalmente ao projeto e métodos de produção assistidos por computador. Os artesãos/projetistas/engenheiros que projetam nosso ambiente foram liberados de muitas restrições físicas consideradas fundamentais anteriormente. Se você ficar doente, os remédios indicados para a sua cura foram projetados usando computadores.

Finalmente, a pesquisa – a própria ciência – se apoia fortemente em computadores. Os telescópios que investigam os segredos de estrelas distantes não

poderiam ter sido projetados, construídos ou operados sem computadores, e as massas de dados que produzem não poderiam ser analisadas e compreendidas sem computadores. Um pesquisador de biologia de campo pode não usar muita computação (naturalmente, a menos que sejam usadas uma câmera, um gravador digital, um telefone, etc.), mas, ao retornar ao laboratório, os dados têm que ser armazenados, analisados, comparados com modelos computadorizados e comunicados a colegas cientistas. A química e a biologia modernas – incluindo pesquisas médicas – usam uma quantidade de computadores sequer sonhada poucos anos atrás e ainda inimaginável para algumas pessoas. O genoma humano foi sequenciado por computadores. Ou – sejamos precisos – o genoma humano foi sequenciado por humanos usando computadores. Em todos esses exemplos, vemos computadores como aquilo que nos permite fazer algo que seria difícil fazer sem eles.

Cada um desses computadores executa software. Sem software, seriam apenas amontoados de silício, metal e plástico: batentes de portas, âncoras de barcos e aquecedores de ambiente. Cada linha de software foi escrita por uma pessoa. Cada uma daquelas linhas que foi realmente executada era no mínimo razoável, se não correta. É surpreendente que tudo isso funcione! Estamos falando sobre bilhões de linhas de código (texto de programa) em centenas de linguagens de programação. Fazer tudo isso funcionar custou uma quantidade estonteante de esforços e envolveu um número inimaginável de habilidades. Queremos melhorar ainda mais para abranger cada dispositivo e cada equipamento de que precisamos. Apenas pense sobre algum serviço ou equipamento do qual você depende; o que você gostaria de melhorar? Se nada mais, queremos que nossos serviços e nossos equipamentos sejam menores (ou maiores), mais rápidos, mais confiáveis, com mais recursos, mais fáceis de usar, com maior capacidade, melhor aspecto e mais baratos. É provável que as melhorias que você imagina requeiram programação.

1.3 Pessoas

Computadores são construídos por pessoas para serem usados por pessoas. Um computador é uma ferramenta muito genérica; pode ser usado para uma variedade inimaginável de tarefas. Ele usa um programa para torná-lo útil para alguém. Em outras palavras, um computador é apenas uma peça de hardware até que alguém – algum programador – escreva um código para que ele faça algo útil. Muitas vezes esquecemos o software. E, mais frequentemente, esquecemos o programador.

Hollywood e fontes de desinformação similares de "cultura popular" têm atribuído a programadores imagens enormemente negativas. Por exemplo, todos vimos o *nerd* solitário, gordo e feio, sem convívio social, que é obsessivo por jogos de computador e que invade os computadores de outras pessoas. Ele (quase sempre um homem) deseja tanto destruir o mundo, quanto salvar o mundo. Obviamente, versões mais suaves de tais caricaturas existem no mundo real, mas nossa experiência mostra que não são mais frequentes entre desenvolvedores de software do que entre advogados, policiais, vendedores de carros, jornalistas, artistas ou políticos.

Pense a respeito de aplicações de computadores que você conhece em sua própria vida. Foram elas feitas por um solitário em uma sala escura? Claro que não; a criação de um software, um equipamento computadorizado ou um sistema bem sucedido envolve dezenas, centenas ou milhares de pessoas que desempenham um variado conjunto de papéis: por exemplo, programadores, projetistas (de programas), testadores, animadores, gerentes de grupos, psicólogos experimentais, projetistas de interfaces com os usuários, analistas, administradores de sistemas, pessoas para o relacionamento com clientes, engenheiros de som, gerentes de projetos, engenheiros de qualidade, estatísticos, engenheiros de interface de hardware, engenheiros de requisitos, responsáveis por segurança, matemáticos, pessoal de suporte a vendas, solucionadores de problemas, projetistas de redes, responsáveis por metodologias, gerentes de ferramentas de software, bibliotecários de software, etc. A variedade de papéis é imensa e se torna mais desconcertante devido aos nomes dos cargos, que variam de organização para organização: um "engenheiro" em uma organização pode ser um "programador" em outra organização e "desenvolvedor", "membro da equipe técnica" ou "arquiteto" em outras. Existem mesmo organizações que deixam seus empregados escolher os nomes de seus próprios cargos. Nem todos esses papéis envolvem diretamente programação. Entretanto, temos visto pessoalmente exemplos de pessoas que desempenham cada um dos papéis mencionados tendo como parte essencial de seu trabalho ler e escrever software. Além disso, um programador (desempenhando qualquer um desses papéis ou outros) pode interagir, em um curto espaço de tempo, com pessoas de uma grande variedade de áreas de aplicação, tais como biólogos, projetistas de máquinas, advogados, vendedores de carros, pesquisadores médicos, historiadores, geólogos, astronautas, engenheiros de aviação, administradores de madeireiras, cientistas de foguetes, construtores de pistas de boliche, jornalistas e animadores (sim, esta é uma relação elaborada a partir da experiência pessoal). Alguém também pode algumas vezes ser programador e desempenhar papéis de não programador em outras etapas de uma carreira profissional.

O mito de um programador como um ser isolado é apenas isto: um mito. Pessoas que gostam de trabalhar sozinhas escolhem áreas de trabalho onde isso é mais viável e geralmente reclamam do número de "interrupções" e de reuniões. Pessoas que preferem interagir com outras pessoas têm uma vida mais fácil, porque o desenvolvimento de software atualmente é uma atividade em equipe. As implicações são que habilidades sociais e de comunicação são essenciais e muito mais valorizadas do que indicam os estereótipos.

Em uma curta relação de habilidades altamente desejáveis para um programador (seja como for que você defina realisticamente um *programador*), você vai encontrar a habilidade de se comunicar bem – com pessoas com uma ampla variedade de experiências – informalmente, em reuniões, em correspondências e em apresentações formais. Estamos convencidos de que até você ter completado um ou dois projetos em equipe, você não terá ideia do que significa programação e se você realmente gosta dela. Entre as coisas que gostamos na programação estão as pessoas agradáveis e interessantes que encontramos e os diversos lugares que visitamos ao longo de nossas vidas profissionais.

Uma consequência de tudo isso é que pessoas com uma grande variedade de habilidades, interesses e hábitos de trabalho são essenciais para a produção de um bom software.

Nossa qualidade de vida depende destas pessoas – algumas vezes, até mesmo a nossa própria vida. Ninguém pode desempenhar todos os papéis que mencionamos aqui; nenhuma pessoa sensível desejaria assumir todos os papéis. A questão é que você tem possibilidades de escolha mais amplas do que imagina; não que você tenha que fazer uma escolha em particular. Como um indivíduo, você vai "flutuar" em áreas de trabalho que combinam com suas habilidades, talentos e interesses. Falamos sobre "programadores" e "programação", mas obviamente a programação é apenas uma parte do quadro geral. As pessoas que projetam um navio ou um telefone celular não pensam em si mesmas como programadores. A programação é uma parte importante do desenvolvimento de software, mas é todo o desenvolvimento de software. De modo similar, para a maioria dos produtos, o desenvolvimento de software é uma parte importante do desenvolvimento do produto, mas nem tudo ali é desenvolvimento de produto.

Não supomos que você – nosso leitor – queira se tornar um programador profissional e passar o resto de sua vida escrevendo código. Mesmo os melhores programadores – especialmente os melhores programadores – não passam a maior parte de seu tempo escrevendo código. Entender problemas consome muito tempo e algumas vezes requer um esforço intelectual significativo. O desafio intelectual é o que muitos programadores têm em mente quando dizem que a programação é interessante. Muitos dos melhores programadores também possuem formação acadêmica em assuntos que não são geralmente considerados como parte de Ciência da Computação. Por exemplo, se você trabalha em software para pesquisa de genoma, você vai ser mais eficaz se souber algo de biologia molecular. Se trabalha em programas para analisar literatura medieval, você poderia ser bem melhor se lesse um pouco sobre tal literatura e talvez conhecesse um ou mais idiomas relevantes. Em particular, uma pessoa com uma atitude "tudo que me importa são computadores e programação" será incapaz de interagir com seus colegas não programadores. Tal pessoa não apenas vai perder as partes mais interessantes da interação humana (isto é, da vida), mas também vai ser um mau desenvolvedor de software.

Portanto, o que supomos? A programação é um conjunto de habilidades intelectualmente desafiadoras que fazem parte de muitas disciplinas técnicas importantes e interessantes. Além disso, a programação é uma parte essencial de nosso mundo, de modo que não conhecer os fundamentos de programação é como não saber os conceitos básicos de física, história, biologia ou literatura. Alguém totalmente ignorante em programação fica limitado a acreditar em mágica e é um perigo quando gente assim assume um cargo técnico. Se você lê Dilbert, pense no chefe de cabelos espetados como o tipo de gerente que você não quer encontrar ou (ainda pior) se tornar. Além disso, a programação pode ser divertida.

Mas o que supomos sobre a maneira em que você pode usar a programação? Talvez você use a programação como a principal ferramenta para seus estudos e trabalho, sem se tornar um programador profissional. Talvez você interaja profissionalmente e pessoalmente com outras pessoas como um projetista, escritor, administrador ou cientista de maneiras em que um conhecimento básico de pro-

gramação seja uma vantagem. Talvez seja programação em nível profissional a parte de seus estudos ou trabalho. Mesmo que você se torne um programador profissional é improvável que você não faça outra coisa além de programação.

Você pode se tornar um engenheiro focado em computadores ou um cientista de computação, mas mesmo assim você não vai "programar o tempo todo". A programação é uma maneira de apresentar ideias em código – uma maneira de auxiliar a resolver problemas. Não é nada – ou é um completo desperdício de tempo –, a menos que você tenha ideias que valham a pena apresentar e problemas que valham a pena resolver.

Se este é um livro sobre programação e se prometemos ajudar você a aprender como programar, por que enfatizamos assuntos que não são de programação e sobre o papel limitado da programação? Um bom programador compreende o papel do código e da técnica de programação em um projeto. Um bom programador (na maioria das vezes) é um bom membro de equipe e tenta arduamente entender como o código e a sua produção ajudam da melhor maneira todo o projeto. Por exemplo, imagine que eu tenha trabalhado em um novo aparelho de MP3 e tudo o que me importava era a beleza do código e a quantidade de recursos maravilhosos que eu poderia fornecer. Eu provavelmente iria insistir no maior e mais poderoso computador para executar o meu código. Poderia desdenhar a teoria de codificação de som por que isso "não é programação". Poderia ficar em meu laboratório em vez de sair para encontrar potenciais usuários, que indubitavelmente poderiam ter mau gosto para música e poderiam não apreciar os últimos avanços em programação de GUI (interface gráfica com o usuário). O resultado provável poderia ser desastroso para o projeto. Um computador maior poderia significar um aparelho de MP3 mais caro e muito provavelmente uma vida mais curta da bateria. A codificação é uma parte essencial do tratamento digital de música, de modo que falhar em prestar atenção aos avanços das técnicas de codificação poderia levar à necessidade de mais memória para cada música (a codificação pode fazer uma diferença de 100% para a mesma qualidade de saída). Um desprezo pelas preferências dos usuários – mesmo que pareçam estranhas e arcaicas para você – geralmente os leva a escolher algum outro produto. Uma parte essencial da boa programação é entender as necessidades dos usuários e as restrições que essas necessidades impõem à implementação (isto é, ao código). Para completar esta caricatura de um mau programador, precisamos apenas adicionar a tendência de entregar com atraso, devido a uma obsessão por detalhes e uma confiança excessiva na correção de código pouco testado. Encorajamos você a se tornar um bom programador, com uma visão ampla do que é necessário para produzir bom software. É nisso que se encontram tanto o seu valor para a sociedade quanto o segredo para a satisfação pessoal.

1.4 Ciência da computação

Mesmo na definição mais ampla, a programação é mais bem vista como uma parte de algo maior. Podemos vê-la como uma subdisciplina da Ciência da Computação, Engenharia de Computação, Engenharia de Software, Tecnologia de Informação ou qualquer outra disciplina relacionada a software. Vemos a pro-

gramação como uma tecnologia que capacita para as áreas de Computação e Informação de Ciência e Engenharia, bem como para Física, Biologia, Medicina, História, Literatura e outros campos acadêmicos e de pesquisa.

Reflita sobre ciência da computação. Um livro de registros de 1995 do governo norte-americano a define assim: "O estudo sistemático de sistemas de computação e computadores. O corpo de conhecimento resultante desta disciplina contém teorias para entender sistemas e métodos de computação; metodologia de projeto, algoritmos e ferramentas; métodos para teste de conceitos; métodos para análise e verificação, e representação e implementação de conhecimento". Como poderíamos esperar, a Wikipedia é menos formal: "Ciência de computador, ou Ciência da computação, é o estudo dos fundamentos teóricos de informação e computação e sua implementação e aplicação em sistemas de computação. A ciência da computação tem muitos subcampos; alguns enfatizam a computação de resultados específicos (como a computação gráfica), enquanto outros (como a teoria de complexidade computacional) se relacionam com propriedades de sistemas computacionais. Ainda outros focalizam os desafios de implementar computações. Por exemplo, a teoria de linguagens de programação estuda abordagens para descrever computações, enquanto que a programação de computadores aplica linguagens de programação específicas para resolver problemas computacionais específicos".

A programação é uma ferramenta; é uma ferramenta fundamental para expressar soluções para problemas práticos e fundamentais de modo que eles possam ser testados, melhorados a partir de experimentos e usados. A programação é o encontro de ideias e teorias com a realidade. É aqui que a ciência da computação pode se tornar uma disciplina experimental, em vez de pura teoria, e impactar o mundo. Neste contexto, como em muitos outros, é essencial que a programação seja uma expressão de práticas testadas com sucesso, assim como de teorias. Ela não deve degenerar em mera especulação: ser apenas um código, escrito de qualquer maneira ultrapassada, que atenda a uma necessidade imediata.

1.5 Os computadores estão em toda parte

Ninguém sabe tudo o que existe para saber sobre computadores ou software. Esta seção apenas fornece alguns exemplos. Talvez você encontre algo de que goste. Ao menos, você poderá comprovar que a variedade de usos de computadores – e, portanto, da programação – é bem maior do que qualquer indivíduo possa compreender.

Muitas pessoas imaginam o computador como uma pequena caixa cinza conectada a uma tela e um teclado. Tais computadores tendem a se esconder sob mesas e ser bons para jogos, bate-papos abrir e-mails e músicas. Outros computadores, denominados *notebooks*, são usados em aviões por homens de negócios entediados para abrir planilhas eletrônicas, jogar e ver vídeos. Essa caricatura é apenas a ponta do iceberg. A maioria dos computadores funciona longe de nossa vista e faz parte de sistemas que mantêm a nossa civilização funcionando. Alguns ocupam salas inteiras; outros são menores do que uma pequena moeda. Muitos dos computadores mais interessantes não interagem diretamente com humanos através de um teclado, mouse ou equipamento similar.

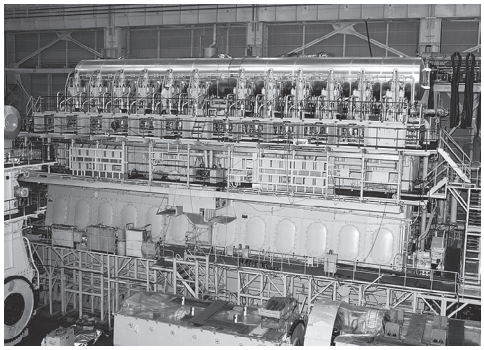
1.5.1 Com telas e sem telas

A noção de um computador como uma grande caixa quadrada com uma tela e um teclado é bem comum e muitas vezes difícil de desfazer. Entretanto, considere estes dois computadores:

Estes dois "brinquedos" (que por acaso são relógios) são basicamente computadores. De fato, podemos dizer que são essencialmente o mesmo modelo de computador com diferentes sistemas de E/S (entrada/saída). O da esquerda controla uma pequena tela (similar às telas de computadores convencionais, porém, menor), e o segundo controla pequenos motores elétricos que movimentam ponteiros de relógios tradicionais e um disco para exibição dos dias do mês. Seus sistemas de entrada são quatro botões (vistos mais facilmente no relógio da direita) e um receptor de rádio, usado para sincronização com relógios "atômicos" de grande precisão. A maioria dos programas que controlam estes dois relógios é compartilhada entre eles.

1.5.2 Navegação

Estas duas fotos mostram um grande motor marítimo a diesel e o tipo de navio imenso que pode ser movido por ele:



Pense onde computadores e software desempenham papéis importantes nestes casos:

- *Projeto*: Obviamente, o navio e o motor foram projetados por meio de computadores. A lista de usos é quase infinita e inclui projetos de arquitetura e engenharia, cálculos em geral, visualização de espaços e peças e simulações do desempenho das peças.
- *Construção*: Um estaleiro moderno é altamente informatizado. A montagem de um navio é cuidadosamente planejada por meio de computadores e o trabalho é executado sob a orientação de computadores. As soldas são feitas por robôs. Em particular, um moderno petroleiro de casco duplo não pode ser construído sem os pequenos robôs de soldagem para fazer as soldas dentro do espaço entre os cascos, onde não há espaço para um ser humano. O corte de chapas de aço para navios foi uma das primeiras aplicações do mundo de CAD/CAM (*computer-aided design/computer-aided manufacture*).
- *O motor*: O motor tem injeção eletrônica de combustível e é controlado por algumas dezenas de computadores. Não é uma tarefa trivial quando se trata de um motor de 100.000 cavalos (como o da foto). Os computadores de controle do motor, por exemplo, ajustam continuamente a mistura de combustíveis para minimizar a poluição que seria produzida por um motor mal regulado. Muitas das bombas associadas ao motor (e a outras partes do navio) também são computadorizadas.
- *Gerenciamento*: Navios navegam para onde há carga para retirar e entregar. A programação de frotas de navios é um processo contínuo (computadorizado, é claro) para que as rotas se alterem com o tempo, com a oferta e a demanda e com o espaço e a capacidade de carga dos portos. Há mesmo sites onde você pode ver a posição dos principais navios mercantes a qualquer momento. O navio da foto transporta contêineres (é o maior do mundo: 397m de comprimento e 56m de largura), mas outros tipos de navios grandes e modernos são gerenciados de maneira semelhante.
- *Monitoração*: Um navio oceânico é bastante autônomo, isto é, a sua tripulação pode lidar com a maioria das contingências que podem surgir antes do próximo porto. Contudo, eles também fazem parte de uma rede que abrange o mundo todo. A tripulação tem acesso a informações sobre o tempo razoavelmente precisas (a partir e por meio de satélites computadorizados). As embarcações têm GPS (sistema de posicionamento global) e radar controlado e otimizado por computador. Se a tripulação precisa de um descanso, a maioria dos sistemas (incluindo o motor, radar, etc.) podem ser monitorados (via satélite) de uma sala de controle da empresa marítima. Se algo de anormal ocorrer, ou se a conexão "com a casa" cair, a tripulação será notificada.

Reflita sobre a consequência de uma falha em um dentre as centenas de computadores explicitamente ou implicitamente mencionados nessa breve descrição. O Capítulo 25 ("Programação de Sistemas Embarcados") examina o tema com um pouco mais de detalhes. Escrever código para um navio moderno é uma atividade habilidosa e interessante. E também útil. O custo de transporte

é realmente surpreendente. Você se beneficia quando compra algo que não é fabricado na sua região. O transporte marítimo geralmente é mais barato do que o transporte terrestre; atualmente, uma das razões para isso é o uso de computadores e informação.

1.5.3 Telecomunicações

Estas duas fotos mostram uma central telefônica e um telefone (que por acaso também é uma câmera, um aparelho de MP3, rádio FM e um navegador Web):

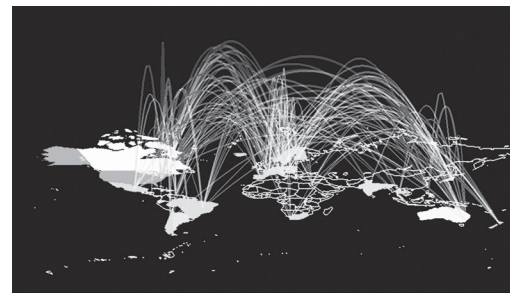
Pense em onde computadores e software desempenham funções importantes neste caso. Você pega um telefone, faz uma ligação, a pessoa para a qual você discou responde e você fala. Ou talvez você fale com uma secretária eletrônica, ou envie uma foto da câmera de seu telefone, ou você envie uma mensagem de texto (clica em "enviar" e deixa o telefone fazer a discagem). Obviamente, o telefone é um computador. Isso é especialmente evidente se o telefone (como a maioria dos celulares) tiver uma tela e oferecer recursos além dos tradicionais "antigos serviços de telefonia", como navegação na Web. Na verdade, tais telefones tendem a conter vários computadores: um para gerenciar a tela, um para falar com o sistema de telefonia, e talvez mais.

A parte do telefone que gerencia a tela, navega na Internet, etc., é provavelmente a mais familiar aos usuários de computadores: ela simplesmente executa uma interface gráfica com o usuário para "todas as coisas comuns". O que é desconhecido e em grande parte inimaginando pelos usuários é o enorme sistema com o qual o pequeno telefone conversa enquanto faz o seu trabalho. Eu disco um número do Texas, mas você está de férias em Nova York, e ainda assim em poucos segundos o seu telefone toca e eu ouço o seu "Olá!" com o barulho do trânsito da cidade ao fundo. Muitos telefones podem executar esse truque para praticamente quaisquer dois locais na Terra, e nós simplesmente consideramos isso normal. Como o meu telefone encontra o seu? Como o som é transmitido?

Como é codificado em pacotes de dados? A resposta poderia preencher muitos livros, muito mais grossos do que este, mas envolve uma combinação de hardware e software presente em centenas de computadores espalhados pela área geográfica em questão. Se você tiver azar, alguns satélites de telecomunicações (eles próprios sistemas computadorizados) também estarão envolvidos – digo "azar" porque não podemos compensar perfeitamente o desvio de 32 mil quilômetros para o espaço; a velocidade da luz (e, portanto, a velocidade da sua voz) é finita (cabos de fibra ótica são muito melhores: mais curtos, mais rápidos e transportam muito mais dados). A maior parte disso funciona incrivelmente bem; os backbones dos sistemas de telecomunicações são 99,9999% confiáveis (por exemplo, 20 minutos de inatividade em 20 anos – que é 20 / 20×365×24×60). O problema tende a ser a comunicação entre o telefone celular e a central telefônica mais próxima.

Existe software para conectar os telefones, separar as nossas palavras em pacotes de dados que são enviados através de fios e conexões por rádio para serem roteados, para a correção de todos os tipos de falhas, para o acompanhamento contínuo da qualidade e confiabilidade dos serviços e, claro, para o faturamento. Mesmo manter o controle de todas as partes físicas do sistema requer grandes quantidades de software inteligente: o que fala com o quê? Que peças serão usadas em um novo sistema? Quando você precisa fazer alguma manutenção preventiva?

Possivelmente, o *backbone* do sistema de telecomunicações do mundo, que consiste em sistemas semi-independentes, mas interligados, é o maior e mais complicado artefato construído pelo homem. Para tornar as coisas um pouco mais reais: lembre-se, esta não é apenas a antiga e aborrecida telefonia com alguns novos zunidos e assobios. As várias infraestruturas fundiram-se. É nelas que também funcionam a Internet, o nosso sistema bancário e de comércio e é também nela que os nossos programas de televisão são transportados para as estações de difusão. Assim, podemos acrescentar outro par de fotos para ilustrar telecomunicações:

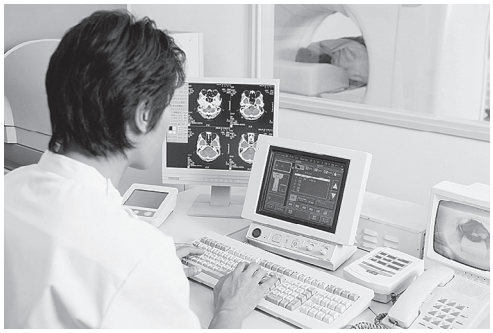


A sala é o pregão da bolsa norte-americana em Nova York, Wall Street, e o mapa é uma representação de partes do *backbone* da Internet (um mapa completo seria confuso demais para ser útil).

Como é natural, também gostamos de fotografia digital e de usar computadores para desenhar mapas especializados que nos ajudem a visualizar informações.

1.5.4 Medicina

Estas duas fotos mostram um *tomógrafo* computadorizado e um ambiente de cirurgia assistida por computador (também chamada de "cirurgia auxiliada por robô" ou cirurgia "robótica"):



Pense onde computadores e software desempenham papéis importantes neste caso. Os *tomógrafos* são basicamente computadores; os pulsos que eles enviam são controlados por um computador, e as leituras não passam de rabiscos até que sejam tratadas por algoritmos bastante sofisticados para convertê-las em algo que reconheçamos como uma imagem (tridimensional) de uma parte relevante do corpo humano. Para fazer a cirurgia computadorizada, temos que ir muitos passos adiante. Uma grande variedade de técnicas de imagem é utilizada para permitir que o cirurgião veja o campo cirúrgico dentro do paciente com ampliação significativa ou com melhor iluminação do que seria possível de outra maneira. Com o auxílio de um computador, um cirurgião pode usar ferramentas que são muito finas para uma mão humana segurar ou chegar a um local que a mão humana não poderia alcançar sem cortes desnecessários. A utilização de cirurgia minimamente invasiva (laparoscopia) é um exemplo simples do que tem minimizado a dor e o tempo de recuperação de milhões de pessoas. O computador também pode ajudar a estabilizar a "mão" do cirurgião para permitir um trabalho mais preciso do que seria possível. Finalmente, um sistema "robótico" pode ser operado remotamente, tornando assim possível a um médico ajudar alguém remotamente (pela Internet). Os computadores e programas envolvidos são desafiadores, complexos e interessantes. A interface com o usuário, controle de equipamentos e os desafios de geração de imagens, por si só, manterão milhares de pesquisadores, engenheiros e programadores ocupados por décadas.

Ouvimos falar de uma discussão entre um grande grupo de médicos sobre qual nova ferramenta tinha fornecido a melhor ajuda possível para eles em seu trabalho: O *tomógrafo* computadorizado? O equipamento de ressonância magnética? As máquinas automatizadas de análise de sangue? As máquinas de ultrassonografia de alta resolução? PDAs? Depois de alguma discussão, um "vencedor" surpreendente surgiu desta "competição": o acesso instantâneo aos registros dos pacientes. Conhecer o histórico médico de um paciente (doenças anteriores, medicamentos ingeridos anteriormente, alergias, problemas hereditários, esta-

do geral de saúde, medicamentos em uso, etc.) simplifica o problema do diagnóstico e minimiza a chance de erros.

1.5.5 Informação

Estas duas fotos mostram um PC comum (ou melhor, dois) e parte de uma central de servidores:

Vínhamos nos concentrando nos equipamentos eletrônicos pelo motivo de sempre: não se pode ver, sentir ou ouvir software. Não podemos apresentar-lhe uma fotografia maravilhosa de um programa, e por isso mostramos um equipamento que executa um. No entanto, muito software lida diretamente com "informação". Portanto, vamos considerar o "uso comum" dos "computadores comuns" executando "software comum".

Uma "central de servidores" é um conjunto de computadores que prestam serviços Web. Ao usar o Google (um mecanismo de busca), encontramos as seguintes informações fornecidas pela Wikipedia (um dicionário da Web). Em 2004, estimava-se que a central de servidores do Google tinha as seguintes especificações:

- 719 *racks*
- 63.272 máquinas
- 126.544 UCPs
- 253 THz de potência de processamento
- 126.544 GB de RAM
- 5.062 TB de espaço em disco rígido

Um GB é um gigabyte, ou seja, cerca de 1.000.000.000 de caracteres. Um TB, um terabyte, é cerca de 1.000 GB, ou seja, aproximadamente 1.000.000.000.000 de caracteres. Hoje, as "centrais" são muito maiores. Trata-se de um caso bastante extremo, mas toda grande empresa executa programas na Web para interagir com seus usuários/clientes. Exemplos: Amazon (livros e outras vendas), Amadeus (passagens aéreas e aluguel de automóveis) e eBay (leilões on-line). Milhões de pequenas empresas, organizações e indivíduos também estão presentes na Web. A maioria não executa o seu próprio software, mas muitas o fazem, e grande parte dele não é simples.

O outro, e mais tradicional, grande uso de computação tem a ver com contabilidade, processamento de pedidos, folha de pagamento, armazenamento de registros, cobrança, gerenciamento de estoques, registros pessoais, registros de estudantes, registros de pacientes, etc. – os registros que praticamente toda organização (comercial, não comercial, governamental, privada) mantém. Existem registros que são a espinha dorsal de suas respectivas organizações. Em termos de computação, processar tais registros parece simples: na maioria das vezes alguma informação (registro) é apenas armazenada, recuperada e muito pouco é feito com ela. Exemplos:

- O meu voo para Chicago às 12:30 está no horário?
- Gilbert Sullivan teve sarampo?
- A máquina de café que Juan Valdez encomendou já foi enviada?
- Que tipo de cadeira da cozinha que Jack Sprat comprou em 1996 (mais ou menos)?
- Quantas ligações telefônicas foram feitas do código de área 212, em agosto de 2006?
- Qual foi o número de cafeteiras vendidas em janeiro e qual o valor total?

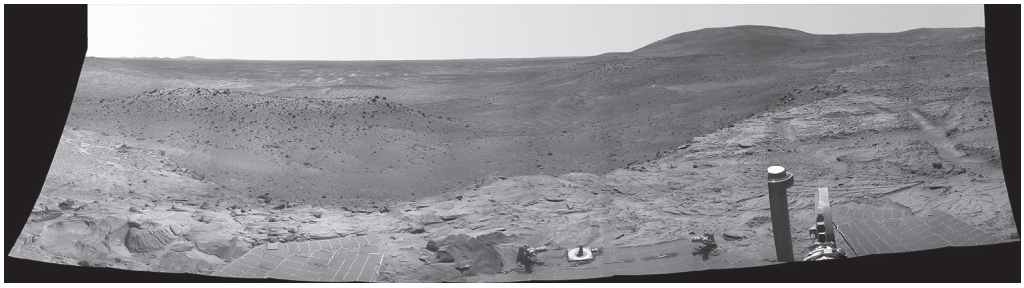
A grande escala dos bancos de dados envolvidos torna estes sistemas altamente complexos. A isso, acrescentemos a necessidade de resposta rápida (muitas vezes em menos de dois segundos para consultas individuais) e correta (pelo menos na maioria das vezes). Atualmente, não é incomum que as pessoas falem de terabytes de dados (um byte é a quantidade de memória necessária para armazenar um caractere comum). Isso é o "processamento de dados" tradicional, que está se fundindo com "a Web", pois a maioria dos acessos às bases de dados agora é realizada a partir de interfaces com a Web.

Esse tipo de uso do computador é muitas vezes chamado de *processamento de informações*. Concentra-se em dados – muitas vezes grandes quantidades de dados. Isso leva a uma série de desafios na organização e transmissão de dados, e dá origem a muitos trabalhos interessantes sobre a forma de apresentar grandes quantidades de dados de maneira compreensível: a "interface com o usuário" é um aspecto muito importante do tratamento de dados. Por exemplo, pense em analisar uma obra de literatura mais antiga (por exemplo, *Os contos da Cantuária*, de Chaucer, ou *Dom Quixote*, de Cervantes) para descobrir o que o autor realmente escreveu, comparando dezenas de versões. Precisamos pesquisar nos textos com diversos critérios fornecidos pela pessoa que faz a análise e mostrar os resultados de uma forma que auxilie a descoberta de pontos importantes. Ao pensar em análise de textos, vêm à minha mente as publicações: hoje, cada artigo, livro, folheto, jornal, etc., é produzido em um computador. Projetar software para suportar bem isso é ainda, para a maioria das pessoas, um problema que carece de uma solução realmente boa.

1.5.6 Uma visão vertical

Às vezes, se afirma que um paleontólogo pode reconstruir um dinossauro completo e descrever o seu estilo de vida e ambiente natural a partir do estudo de um único osso pequeno. Isso pode ser um exagero, mas há algo de verdadeiro

na ideia de olhar um artefato simples e pensar nas suas implicações. Observe esta foto que mostra a paisagem de Marte feita por uma câmera em uma das sondas de Marte da NASA:



Se você quer fazer "ciência espacial", uma maneira é tornar-se um bom programador. Os programas espaciais empregam muitos projetistas de software, especialmente aqueles que também conhecem algo de Física, Matemática, Engenharia Elétrica, Engenharia Mecânica, Engenharia Médica, etc.: áreas que embasam os programas espaciais tripulados e não tripulados. Colocar duas destas sondas para vagar em Marte por mais de quatro anos (a sua vida útil estimada foi de três meses) é uma das maiores conquistas tecnológicas de nossa civilização.

A foto foi enviada à Terra através de um canal de comunicação com um retardo de transmissão de 25 minutos em cada sentido; o processo emprega muita programação inteligente e matemática avançada para assegurar que a imagem seja transmitida usando o mínimo de bits possível sem que nenhum deles se perca. Na Terra, a foto é então processada usando algoritmos para restaurar a cor e minimizar a distorção causada pela óptica e pelos sensores eletrônicos.

Os programas de controle das sondas são, obviamente, programas : as sondas movem-se de forma autônoma por 24 horas de cada vez e seguem as instruções enviadas da Terra no dia anterior. A transmissão é gerenciada por programas.

Os sistemas operacionais utilizados pelos diversos computadores embarcados nas sondas, a transmissão e a reconstrução das fotos são programas, assim como as aplicações utilizadas para escrever este capítulo. Os computadores que executam esses programas são concebidos e produzidos utilizando programas CAD/CAM (*computer-aided design* e *computer-aided manufacture*). Os chips usados nos computadores são produzidos em linhas de montagem computadorizadas construídas com ferramentas de precisão, e essas ferramentas também usam computadores (e software) na sua concepção e fabricação.

O controle de qualidade destes longos processos de construção envolve muito cálculo sério. Todo o código foi escrito por seres humanos em uma linguagem de programação de alto nível e traduzido para código de máquina por um compilador, que é um programa. Muitos destes programas interagem com usuários usando GUIs e trocam dados utilizando fluxos de entrada/saída. Finalmente, muita programação é usada para processamento de imagens (incluindo o processamento das fotos das sondas), animação e edição de fotos (existem versões das fotos das sondas circulando pela Web nas quais aparecem "marcianos").

1.5.7 E então?

O que todas estas aplicações e sistemas de software "extravagantes e complicados" têm a ver com a aprendizagem de programação e o uso de C++? A ligação é simplesmente que muitos programadores chegam a trabalhar em projetos como esses. Esse é o tipo de coisas que uma boa programação pode ajudar a alcançar. Além disso, cada exemplo usado neste capítulo envolve C++ e, pelo menos, algumas das técnicas que descrevemos neste livro. Sim, existem programas C++ em aparelhos de MP3, em navios, em turbinas eólicas, em Marte e no projeto do genoma humano. Para mais aplicações usando C++, consulte www.research.att.com/~bs/applications.html.

1.6 Ideais para programadores

O que queremos de nossos programas? O que queremos em geral, que não seja uma característica específica de um determinado programa? Queremos *correção* e como parte disso, *confiabilidade*. Se o programa não faz o que deveria fazer e não podemos confiar nele, este programa é, na melhor das hipóteses, bastante incômodo e, na pior, um perigo. Queremos que um programa seja *bem projetado* para que atenda bem a uma necessidade real; não interessa que um programa seja correto, se o que faz é irrelevante para nós, ou se faz algo corretamente de uma forma que nos incomoda. Também queremos que seja *acessível*; eu poderia preferir um Rolls-Royce ou um jato executivo à minha forma habitual de transporte, mas a menos que eu seja zilionário, o custo pesará em minha escolha.

Esses são os aspectos de software (dispositivos, sistemas), que podem ser apreciados de fora, por não programadores. Eles devem ser os ideais dos programadores e devemos tê-los na mente em todos os momentos, especialmente nas fases iniciais de desenvolvimento, se queremos produzir software bem-sucedido. Além disso, temos de nos preocupar com os ideais relacionados ao próprio código: o nosso código deve ser *manutenível*, isto é, sua estrutura deve ser tal que alguém que não escreveu o código possa compreendê-lo e fazer alterações. Um programa bem sucedido "vive" por um longo tempo (muitas vezes por décadas) e será alterado muitas vezes. Por exemplo, ele será portado para um novo hardware, ganhará novos recursos, será modificado para usar novos recursos de E/S (telas, vídeo, som), para interagir com novas linguagens naturais, etc. Só um programa mal sucedido nunca será modificado. Para ser manutenível, um programa deve ser simples em relação aos seus requisitos, assim como o código deve representar diretamente as ideias expressas. A complexidade – a inimiga da simplicidade e da facilidade de manutenção – pode ser intrínseca a um problema (nesse caso, só temos de lidar com ela), mas também pode surgir a partir da expressão pobre de ideias no código. Temos de tentar evitar isso a partir de um bom estilo de codificação – o estilo faz a diferença! Não parece muito difícil, mas é. Por quê? A programação é basicamente simples: basta dizer à máquina o que ela deve fazer. Então, por que a programação pode ser desafiadora? Computadores são fundamental-

mente simples; eles podem apenas executar algumas operações, como a adição de dois números e escolher a próxima instrução a ser executada com base numa comparação de dois números. O problema é que nós não queremos que os computadores façam coisas simples. Queremos que "a máquina" faça coisas que são tão difíceis que precisamos de sua ajuda. Mas computadores são minuciosos, implacáveis, bestas burras. Além disso, o mundo é mais complexo do que gostaríamos de acreditar, de modo que realmente não sabemos as implicações do que pedimos. Só queremos um programa para "fazer algo parecido com isto" e não queremos ser incomodados com detalhes técnicos. Também tendemos a supor "o senso comum". Infelizmente, o senso comum não é tão comum entre os seres humanos e é totalmente ausente em computadores (embora alguns programas realmente bem concebidos possam imitá-lo em casos específicos e bem compreendidos).

Essa linha de pensamento conduz à ideia de que "a programação é compreensão": quando você consegue programar uma tarefa, você a compreende. Por outro lado, quando entender completamente uma tarefa, você pode escrever um programa para fazê-la. Em outras palavras, podemos ver a programação como parte de um esforço para entender completamente um tópico. Um programa é uma representação precisa da nossa compreensão de um tema. Ao programar, você consome bastante tempo tentando entender a tarefa que você está tentando automatizar.

Podemos descrever o processo de desenvolvimento de um programa em quatro fases:

- *Análise*: Qual é o problema? O que o usuário quer? O que significa a necessidade do usuário? De que recursos o usuário pode dispor? Que tipo de confiabilidade precisamos?
- *Projeto*: Como vamos resolver o problema? Qual deve ser a estrutura geral do sistema? De que partes ele consiste? Como as partes se comunicam entre si? Como o sistema se comunica com seus usuários?
- *Programação*: Expresse a solução para o problema (o projeto) em código. Escreva o código de uma forma que satisfaça todas as restrições (tempo, espaço, custo, confiabilidade e assim por diante). Certifique-se de que o código é correto e manutenível.
- *Teste*: Verifique se o sistema funciona corretamente em todas as circunstâncias exigidas, através do teste sistemático.

A programação seguida do teste é muitas vezes chamada de *implementação*. Obviamente, essa simples divisão do desenvolvimento de software em quatro partes é uma simplificação. Livros grossos foram escritos sobre cada um desses quatro temas, e há ainda mais livros sobre como eles se relacionam entre si. Uma coisa importante a salientar é que esses estágios de desenvolvimento não são independentes e não ocorrem estritamente em sequência. Normalmente, iniciamos com a análise, mas os resultados dos testes podem ajudar a melhorar a programação; problemas em conseguir um programa que funcione podem indicar um problema com o projeto; trabalhar no projeto pode sugerir aspectos do problema que até o momento tinham sido negligenciados na análise. Na verdade, usar o sistema geralmente mostra os pontos fracos da análise.

O conceito crucial aqui é realimentação (*feedback*). Aprendemos com a experiência e modificamos o nosso comportamento com base no que aprendemos. Isso é essencial para o desenvolvimento eficaz de software. Para qualquer grande projeto, não sabemos tudo o que há para saber sobre o problema e sua solução antes de começarmos. Podemos testar ideias e obter realimentação por meio da programação, mas nos estágios iniciais de desenvolvimento é mais fácil (e mais rápido) obter realimentação ao escrever as ideias do projeto, experimentar as ideias do projeto e usar os cenários com amigos. A melhor ferramenta de projeto que conhecemos é um quadro negro (ou branco, se você preferir odores químicos em vez do pó de giz). Nunca projete sozinho se você puder evitar! Não inicie a codificação antes de ter testado suas ideias, explicando-as a alguém. Discuta modelos e técnicas de programação com amigos, colegas, potenciais usuários e assim por diante, antes de mergulhar no teclado. É incrível o quanto você pode aprender simplesmente ao tentar articular uma ideia. Afinal, um programa nada mais é do que uma expressão (em código) de algumas ideias.

Da mesma forma, quando você ficar emperrado na implementação de um programa, olhe por cima do teclado. Pense sobre o problema em si, e não na sua solução incompleta. Fale com alguém: explique o que você quer fazer e por que isso não funciona. É surpreendente como muitas vezes você encontra a solução apenas explicando com cuidado o problema para alguém. Não depure (tente localizar erros de programa) sozinho se você não precisa!

O foco deste livro é implementação e, especialmente, programação. Não ensinamos "resolução de problemas", ainda que mostremos muitos exemplos de problemas e suas soluções. Grande parte da solução de problemas é reconhecer um problema conhecido e aplicar uma técnica de solução conhecida. Só quando a maioria dos subproblemas for resolvida assim você poderá ceder ao emocionante e criativo desafio de buscar novas perspectivas de soluções. Então, vamos nos concentrar em mostrar a forma de expressar ideias com clareza no código.

A expressão direta de ideias em código é um ideal fundamental de programação. Isso é realmente bastante óbvio, mas até agora não temos bons exemplos. Vamos retornar a isso, repetidamente. Quando queremos um número inteiro em nosso código, o armazenamos em um **int**, que fornece o conjunto básico de operações com inteiros. Quando queremos uma sequência de caracteres, armazenamos em um **string**, que fornece as operações de manipulação de texto mais básicas. No nível mais básico, o ideal é que quando temos uma ideia, um conceito, uma entidade, algo que possamos pensar como "uma coisa", algo que podemos rabiscar em nosso quadro branco, algo a que possamos nos referir em nossas discussões, algo que o nosso livro-texto (um que não seja sobre Ciência da Computação) menciona, então queremos que este algo exista em nosso programa como uma entidade com um nome (um tipo) que propicie as operações que julgamos apropriadas para tal. Se quisermos trabalhar com Matemática, queremos um tipo **complex** para números complexos e um tipo **Matrix** para álgebra linear. Se quisermos trabalhar com gráficos, queremos um tipo **Shape**, um tipo **Circle**, um tipo **Color** e um **Dialog_box**. Quando lidamos com fluxos de dados, digamos, de um sensor de temperatura, queremos um tipo **istream** ("**i**" de *input*). Obviamente, cada um destes tipos deve fornecer as operações apropriadas e somente as operações apropriadas. Esses são apenas alguns poucos

exemplos tirados deste livro. Além disso, oferecemos ferramentas e técnicas para você construir seus próprios tipos para representar qualquer conceito que queira diretamente em seu programa.

A programação é em parte prática e em parte teoria. Se você for apenas prático, criará algo tosco, não escalonável, não manutenível. Se for apenas teórico, criará brinquedos inúteis.

Para uma visão diferente dos ideais de programação e algumas pessoas que fizeram contribuições importantes através de seus trabalhos com linguagens de programação, veja o Capítulo 22, "Ideais e História".

Revisão

As perguntas da revisão destinam-se a salientar as principais ideias explicadas neste capítulo. Uma maneira de encará-las é considerá-las como um complemento para os exercícios: os exercícios se concentram nos aspectos práticos da programação, enquanto as perguntas de revisão tentam ajudá-lo a articular as ideias e os conceitos. Nesse sentido, se assemelham a boas perguntas de entrevista.

1. O que é software?
2. Por que o software é importante?
3. Onde o software é importante?
4. O que poderia dar errado se algum software não funcionar? Dê alguns exemplos.
5. Onde o software desempenha um papel importante? Dê alguns exemplos.
6. Quais são alguns trabalhos relacionados ao desenvolvimento de software? Dê alguns.
7. Qual é a diferença entre Ciência da Computação e programação?
8. Onde é usado software na concepção, na construção e na utilização de um navio?
9. O que é uma central de servidores?
10. Que tipos de consultas que você faz *online*? Liste algumas.
11. Quais são alguns usos de software em ciência? Dê exemplos.
12. Quais são alguns usos de software em medicina? Dê exemplos.
13. Quais são alguns usos de software em entretenimento? Dê exemplos.
14. Que propriedades gerais podemos esperar de um bom software?
15. Qual é a aparência de um desenvolvedor de software?
16. Quais são as fases do desenvolvimento de software?
17. Por que o desenvolvimento de software pode ser difícil? Enumere algumas razões.
18. Quais são alguns usos do software que facilitam a sua vida?
19. Quais são alguns usos do software que tornam sua vida mais difícil?

Termos-chave

Estes termos apresentam o vocabulário básico de programação e de C++. Se você quer entender o que as pessoas falam sobre tópicos de programação e articular as suas próprias ideias, deve saber o que cada um deles significa.

análise	GUI	projeto
CAD/CAM	ideais	quadro negro
cliente	implementação	realimentação
comunicações	preço acessível	software
corretude	programação	teste
estereótipo	programador	usuário

Exercícios

1. Escolha uma atividade que você faz na maioria dos dias (como ir à aula, jantar ou assistir televisão). Faça uma lista de como computadores estão envolvidos direta ou indiretamente.
2. Escolha uma profissão, de preferência uma em que você tenha algum interesse ou algum conhecimento. Faça uma lista de atividades realizadas por pessoas desta profissão que envolvem computadores.
3. Troque sua lista de exercício 2 com um amigo que escolheu uma profissão diferente e melhore a lista dele. Quando vocês terminarem, comparem seus resultados. Lembre-se: não existe uma solução perfeita para uma questão em aberto; melhorias são sempre possíveis.
4. Baseado na sua própria experiência, descreva uma atividade que não teria sido possível sem os computadores.
5. Faça uma lista de programas aplicativos (software) que você usa diretamente. Liste apenas exemplos nos quais você obviamente interage com um programa (como quando seleciona uma nova música em um MP3) e não casos em que talvez haja um computador envolvido (como virar o volante de seu carro).
6. Faça uma lista de dez atividades que as pessoas executam e que não envolvem computadores de forma alguma, nem mesmo indiretamente. Isto pode ser mais difícil do que você imagina!
7. Identifique cinco tarefas para as quais computadores não são hoje utilizados, mas que na sua opinião serão utilizados em algum momento futuro. Escreva algumas frases para detalhar cada uma das tarefas que você escolheu.
8. Escreva uma explicação (pelo menos 100 palavras, mas menos de 500) sobre por que você gostaria de ser um programador. Se, por outro lado, você está convencido de que não gostaria de ser um programador, explique. Em qualquer caso, apresente argumentos bem fundamentados, lógicos.
9. Escreva uma explicação (pelo menos 100 palavras, mas menos de 500) sobre que outro cargo além de programador que você gostaria de desempenhar na indústria de computadores (independentemente de ser "programador" sua primeira escolha).

10. Você acha que algum dia os computadores evoluirão para seres conscientes, pensantes, capazes de competir com os seres humanos? Escreva um parágrafo curto (pelo menos 100 palavras) para embasar a sua posição.
11. Relacione algumas características que programadores de sucesso possuem em comum. Depois, relacione algumas características que popularmente se considera que os programadores possuem.
12. Identifique pelo menos cinco tipos de aplicações para programas de computador mencionados neste capítulo e escolha a que você achar mais interessante e de que você gostaria de participar algum dia. Escreva um parágrafo curto (menos de 100 palavras) explicando as razões da sua escolha.
13. Quanta memória seria necessária para armazenar (a) esta página de texto, (b) este capítulo, (c) toda a obra de Shakespeare? Suponha que um byte de memória armazena um caractere e tente uma precisão de cerca de 20%.
14. Quanta memória possui o seu computador? Memória principal? Disco?

Posfácio

Nossa civilização funciona com base em software. Software é uma área de diversidade insuperável e de oportunidades para trabalhos interessantes, socialmente úteis e rentáveis. Quando você for trabalhar com software, faça isto de uma maneira séria, com princípios: você quer ser parte da solução, e não aumentar os problemas.

Temos, obviamente, admiração pela diversidade de software que permeia a nossa civilização tecnológica. Nem todas as aplicações de software fazem o bem, é claro, mas isso é outra história. Aqui queremos enfatizar como o software é universal e como grande parte de nossa vida diária depende de software. Foi tudo escrito por pessoas como nós. Todos os cientistas, matemáticos, engenheiros, programadores, etc., que construíram o software mencionado brevemente aqui começaram como você está começando.

Agora, voltemos à tarefa mundana de aprender as habilidades técnicas necessárias para programar. Se você começar a se perguntar se vale a pena todo o seu trabalho (pessoas mais pensativas vão se perguntar sobre isso algum dia), volte e releia este capítulo, o Prefácio e partes do Capítulo 0 (“Notas para o leitor”). Se você começar a se perguntar se você pode lidar com tudo isso, lembre-se de que milhões têm conseguido tornar-se competentes programadores, projetistas, engenheiros de software, etc. Você também pode.