Elogios ao Nós, Programadores

"Assim como o Uncle Bob, passei boa parte da minha carreira atuando como consultor, professor e participando de congressos de computação. Isso é importante porque tive a oportunidade de conhecer e jantar com muitos dos personagens deste livro. Ou seja, este livro retrata meus amigos de profissão, e posso afirmar que ele narra fielmente os fatos. Na verdade, está escrito e foi pesquisado de forma incrível — os fatos retratam tudo o que ocorreu."

— Do Posfácio de Tom Gilb

"Não consigo pensar em nenhum outro livro que passe uma visão tão ampla sobre os primórdios da programação."

— Mark Seeman

"A obra Nós, Programadores *é uma viagem fascinante pela história dos computadores e da programação. Podemos conferir os vislumbres maravilhosos da vida de alguns dos nomes proeminentes da área. Além disso, o livro relata, de forma prazerosa, a trajetória de Uncle Bob como programador."*

— Jon Kern, coautor do Manifesto Ágil

"Em Nós, Programadores, *Bob entrelaça de forma genial uma história extremamente envolvente sobre os programadores, proporcionando um rico contexto histórico, relatos humanos e revelações surpreendentes sobre os pioneiros da nossa indústria — tudo isso equilibrado com a quantidade adequada de detalhes técnicos. Ainda que represente apenas uma pequena parte dessa vasta história, ele consegue intercalar o relato com observações e críticas pertinentes. Nesta edição, também conhecemos toda a sua história pessoal, bem como suas reflexões sobre o futuro. Uma leitura leve e divertida."*

— Jeff Langr

Nós, Programadores

Da Ada à Inteligência Artificial: Uma Crônica da Programação

Robert C. Martin

ALTA BOOKS
GRUPO EDITORIAL
Rio de Janeiro, 2026

Nós, programadores

ISBN: 978-85-508-2735-3

Authorized translation from the English language edition, entitled We, Programmers: A Chronicle of Coders from Ada to AI, 1st Edition, ISBN 9780135344262 by Robert Martin, published by Pearson Education, Inc. publishing as Addison-Wesley Professional, Copyright © 2025 Pearson Education, Inc., the owner of all rights to publish and sell the same. PORTUGUESE language edition published by Starlin Alta Consulting and Books do Brasil, Copyright © 2026.

Impresso no Brasil — 1ª Edição, 2026 — Edição revisada conforme o Acordo Ortográfico da Língua Portuguesa de 2009.

Dados Internacionais de Catalogação na Publicação (CIP)

M386n

1.ed. Martin, Robert C.
Nós, programadores : da Ada à inteligência artificial : uma crônica da programação / Robert C. Martin. – 1.ed. – Rio de Janeiro : Alta Books, 2026.
480 p.; il.; 15,7 x 23 cm.

Título original: We, Programmers: A Chronicle of Coders from Ada to AI.
ISBN 978-85-508-2735-3

1. Programação. 2. História da computação. 3. Desenvolvimento de software. 4. Tecnologia da informação. I. Título.

CDD 005.1

Índice para catálogo sistemático:
1. Programação : História da computação 005.1

Diretor Editorial: Anderson Vieira.
Vendas Governamentais: Cristiane Mutüs.
Gerência Marketing: Viviane Paiva.

Produtora Editorial: Isabella Gibara.
Tradução: Cibelle Ravaglia.
Revisão Gramatical: Denise Himpel.
Diagramação: Joyce Matos.
Revisão Técnica: Jorge André dos Santos
Analista de Sistemas e Mestre em Ciências Computacionais.

ALTA BOOKS
GRUPO EDITORIAL

Rua Viúva Cláudio, 291 — Bairro Industrial do Jacaré
CEP: 20.970-031 — Rio de Janeiro (RJ)
Tels.: (21) 3278-8069 / 3278-8419
www.altabooks.com.br — altabooks@altabooks.com.br
Ouvidoria: ouvidoria@altabooks.com.br

Editora
afiliada à:

ASSOCIADO

Para Timothy Michael Conrad

Sumário

Preâmbulo


```
vim .
```

Cinco caracteres simples iniciam meu editor de texto favorito. Mas não se trata de um editor qualquer — é o NeoVim. A versão atual do NeoVim vem repleta de atalhos personalizados, suporte a LSP [Language Server Protocol], realce de sintaxe, diagnósticos de erros integrados e muito mais. Com tanta possibilidade de customização, o NeoVim inicia em poucos milissegundos, possibilitando que a edição de arquivos aconteça num instante. Mesmo com milhares de arquivos, os LSPs indicam rapidamente o estado do projeto, e os erros são encaminhados para menus de correção ágil, facilitando a navegação. Com apenas alguns comandos, consigo compilar, inicializar ou executar testes. Meu computador chega a gerar código a partir de instruções simples em inglês, graças à IA! Além disso, essa mesma IA consegue programar junto comigo enquanto digito, gerando um montão de código (de qualidade duvidosa) num piscar de olhos.

> Tudo isso é simplesmente *impressionante*. A experiência com o NeoVim é fantástica, tranquila e extremamente rápida. Contudo, para alguns, usar o NeoVim é considerado algo *arcaico* — até

> mesmo um sacrilégio. "Ludita!"[1] acusam alguns desenvolvedores, criticando a escolha de gastar tempo configurando meu editor quando há *ambientes* completos prontos para uso. O IntelliJ oferece funcionalidades que minha mente, moldada pelo NeoVim, nem consegue conceber!

Estou contando tudo isso porque é algo surpreendente. Não me surpreende o fato de eu utilizar o que alguns chamariam de tecnologias antiquadas, nem a eterna briga entre engenheiros de software sobre preferências — isso faz parte do cotidiano. O que verdadeiramente me espanta é como o total poder de edição se tornou rotineiro para nós, engenheiros — um simples detalhe em meio à rotina repleta de linhas de código, reuniões e mensagens no Slack. Autocompletar, realce de sintaxe e documentação (por vezes) confiável são funções que já esperamos. Lembre-se: antes dos editores modernos de texto, os programadores passaram décadas sem uma linguagem de alto nível, muito tempo sem realce de sintaxe e, praticamente, 70 anos sem LSPs para oferecer ferramentas de autocompletar e refatoração em quase qualquer linguagem. A edição de texto é, de fato, uma das maravilhas criadas pela humanidade.

Além de ter um pé no passado com meu editor de texto, adoro explorar histórias de outros tempos. Penso nos "programadores de verdade", que sincronizavam seu código com a memória de tambor [drum memory] a fim de garantir a velocidade ideal de leitura. Eu daria qualquer coisa para ver um desses mestres, verdadeiros gênios do seu ofício, trabalhando. Talvez seja pura nostalgia, mas as aventuras do passado parecem mais grandiosas, suas descobertas mais significativas e o trabalho, mais inspirador. Ler *Nós, Programadores*, foi como viajar no tempo e caminhar ao lado dos responsáveis por cada avanço transformador na história da computação.

1. O termo "Ludita" se refere a um movimento de trabalhadores ingleses do século XIX que protestavam contra as máquinas da Revolução Industrial, muitas vezes as destruindo por temerem o desemprego. Atualmente, a palavra é usada de forma pejorativa para descrever alguém que se opõe a novas tecnologias ou ao progresso, preferindo métodos considerados ultrapassados. [N. da T.]

Consegui imaginar os jantares de Charles Babbage, que tanto inspirava quanto amedrontava os convidados com sua Máquina Diferencial — aquele monstruoso aparelho mecânico em um barulho metálico e incessante enquanto realizava algo que devia parecer magia. Ver a Máquina Diferencial em ação deve ter sido tão encantador quanto a nossa primeira experiência com um LLM ou com um autocompletar do Copilot, onde os convidados provavelmente exclamam "As máquinas realmente conseguem pensar!" Senti a pressão de equipes trabalhando dia e noite e a necessidade urgente de melhorar o processamento durante os cálculos críticos da Segunda Guerra Mundial, que possibilitaram a criação da bomba atômica. E pensar que, naquela época, não havia cadeiras Herman Miller nem mesas ergonômicas — na verdade, sequer dispunham de monitores ou teclados! Mesmo assim, eles realizaram o inimaginável e mudaram o rumo da história. *Nós, Programadores* é uma das narrativas mais envolventes da trajetória da computação.

Seria surpreendente encontrar um programador que nunca tivesse ouvido falar do Uncle Bob ou que não conhecesse seu trabalho. Ele é, sem sombra de dúvida, uma das figuras mais prolíficas do nosso meio. Durante muitos anos, eu o conhecia apenas pelo nome, pela imagem do perfil no Twitter e por suas contribuições eminentes para o código limpo e metodologia ágil. Para mim, ele era a própria encarnação da teoria de programação mais complexa e abstrata que se possa imaginar. Tudo mudou quando, um dia, começamos a interagir pelo Twitter, o que se desdobrou em e-mails, ligações e até um podcast. Por meio dessas trocas, minha visão se transformou. Robert C. Martin é muito mais do que meus estudos universitários me fizeram acreditar. Ele é um profissional pragmático, disposto a fazer concessões quando necessário. Durante e após nosso podcast, um dos comentários mais recorrentes foi: "Ele ri e sorri muito!" Isso atesta seu caráter e revela uma vida bem vivida. Ele é um verdadeiro engenheiro de software e uma fonte de aprendizado para todos nós.

Estou pessoalmente exausto das incontáveis discussões sobre espaçamentos, editores de texto e dos debates entre OOP e FP que fervilham no X em 280 caracteres ou menos. O que realmente me

fascina é conhecer os criadores das tecnologias que, ironicamente, embasam os argumentos que tanto nos influenciaram. *Nós, Programadores* entrega algo muitíssimo significativo: uma conexão com o passado e a esperança de um futuro melhor, tendo como mediador perfeito, talvez, o Uncle Bob.

— *ThePrimeagen*

Prefácio

Estou prestes a contar a história de como tudo começou. Trata-se de uma narrativa cheia de reviravoltas, que entrelaça as vidas e os desafios de pessoas extraordinárias, a época singular em que viveram e as máquinas incríveis que dominavam com maestria.

Mas, antes de nos aventurarmos por este emaranhado de caminhos sinuosos e distintos, convém oferecer um pequeno vislumbre do que está por vir — um aperitivo para despertar sua curiosidade.

Diz o ditado que a necessidade é a mãe da invenção, mas nada gera necessidade como a guerra. O verdadeiro impulso do nosso setor surgiu das convulsões bélicas, especialmente durante a Segunda Guerra Mundial.

Na década de 1940, a tecnologia bélica estava muito à frente de nossa capacidade computacional. Era impensável que legiões de pessoas, utilizando calculadoras de mesa, conseguissem acompanhar as exigências de cálculo de todos os setores do conflito.

O desafio residia na enorme quantidade de somas, subtrações, multiplicações e divisões necessárias para aproximar a trajetória de um projétil disparado de um canhão até seu alvo. Nada disso podia ser resolvido com fórmulas simples como $d=rt$ ou $s=\frac{1}{2}at^2$;

a tarefa exigia dividir o tempo e a distância em milhares de pequenos intervalos e simular, passo a passo, o percurso do projétil — uma simulação de cálculos quase que da escola primária em escala colossal.

Durante séculos, todo esse processamento era feito por exércitos de pessoas munidas de caneta e papel. Só no século passado, começaram a surgir as máquinas de cálculo para auxiliar nessa empreitada. Organizar os cálculos — e as equipes que os executavam — era uma tarefa hercúlea[1], que frequentemente levava semanas ou até meses para ser concluída.

No século XIX, já se sonhava com máquinas capazes de realizar tais feitos, e alguns protótipos, ainda que rudimentares, chegaram a ser construídos. Mas eles eram meros brinquedos e bizarrices, destinados a ostentar nos jantares da elite, pois poucos os consideravam ferramentas úteis — sobretudo pelo alto custo envolvido.

No entanto, a Segunda Guerra Mundial transformou esse cenário. A necessidade era urgente, o custo, irrelevante. Foi então que aqueles primeiros sonhos se concretizaram e imensos mecanismos de cálculo passaram a ser criados.

Os pioneiros que programavam e operavam aquelas máquinas foram os desbravadores do nosso campo. No começo, trabalharam sob condições extremamente rudimentares: as instruções de programação eram, literalmente, perfuradas — um furo de cada vez — em longas fitas de papel, que a máquina lia e executava. Era um estilo de programação extenuante, repleto de detalhes minuciosos e implacável em sua rigidez. Além disso, a execução desses programas podia se arrastar por semanas, exigindo monitoramento constante e intervenções frequentes. Por exemplo, para executar um loop, era preciso reposicionar manualmente a

1 Para ver isso na prática, recomendo a celebração do Dia do Pi de 2024, onde mais de 100 dígitos foram calculados manualmente por uma equipe muito bem coordenada de algumas centenas de pessoas ao longo de uma semana. No momento em que eu escrevia este livro, "O maior cálculo manual em um século! [Dia do Pi 2024]", publicado no YouTube por Stand-up Maths, em 13 de março de 2024.

fita para cada iteração e verificar, passo a passo, se o loop deveria ser interrompido ou não.

Com o passar do tempo, as máquinas eletromecânicas deram lugar aos computadores eletrônicos a válvulas, que armazenavam dados em ondas sonoras que, por sua vez, percorriam longos tubos de mercúrio. A fita de papel foi substituída por cartões perfurados e, por fim, pelos programas armazenados em memória. Essas novas tecnologias, impulsionadas pelos primeiros pioneiros, abriram caminho para inovações.

Os primeiros compiladores, surgidos no início na década 1950, não eram mais que montadores com palavras-chave especiais que carregavam e acionavam subrotinas pré-escritas — originadas, muitas vezes, de fitas de papel ou magnéticas. Posteriormente, os compiladores passaram a ser testados com expressões e tipos de dados, embora continuassem rudimentares e lentos. No final da década de 1950, o FORTRAN de John Backus e o COBOL de Grace Hopper inauguraram uma nova era. O código binário, que antes era escrito à mão, passou a ser gerado por programas capazes de ler e interpretar textos abstratos.

No início da década de 1960, o ALGOL de Dijkstra elevou ainda mais o nível de abstração, e, alguns anos depois, o SIMULA 67, de Dahl e Nygaard, levou essa evolução a patamares superiores.

A partir daí, surgiram tanto a programação estruturada quanto a orientada a objetos.

Nesse ínterim, John Kemeny e sua equipe aproximaram a computação do cidadão comum ao lançar o BASIC e o time-sharing, em 1964. O BASIC era uma linguagem que quase qualquer pessoa conseguia entender e usar, enquanto o time-sharing permitia que diversos usuários compartilhassem, de maneira simultânea e conveniente, um único computador de alto custo.

E então vieram Ken Thompson e Dennis Ritchie, que, entre o final da década de 1960 e o início da década de 1970, abriram as portas para uma revolução no desenvolvimento de software com a criação

da linguagem C e do sistema Unix. A partir desse ponto, a corrida tecnológica estava lançada.

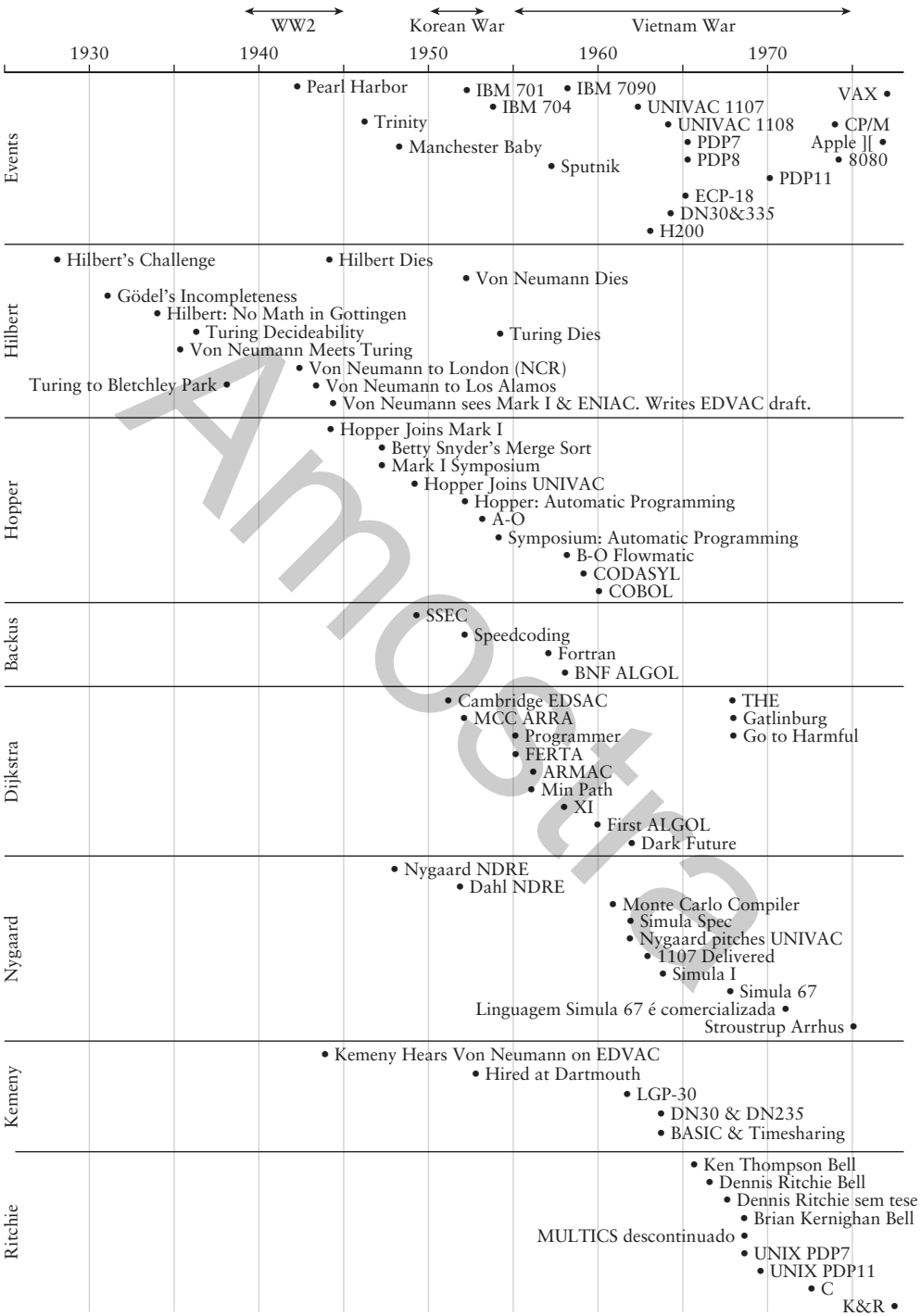
A revolução dos mainframes na década de 1960 foi sucedida pela ascensão dos minicomputadores na década de 1970 e, posteriormente, pela revolução dos microcomputadores na década de 1980. Nesta mesma década, o advento do computador pessoal transformou o setor, seguido rapidamente pela revolução orientada a objetos, pela era da internet e, depois, pela revolução ágil. O software começou, enfim, a fazer parte de *tudo*.

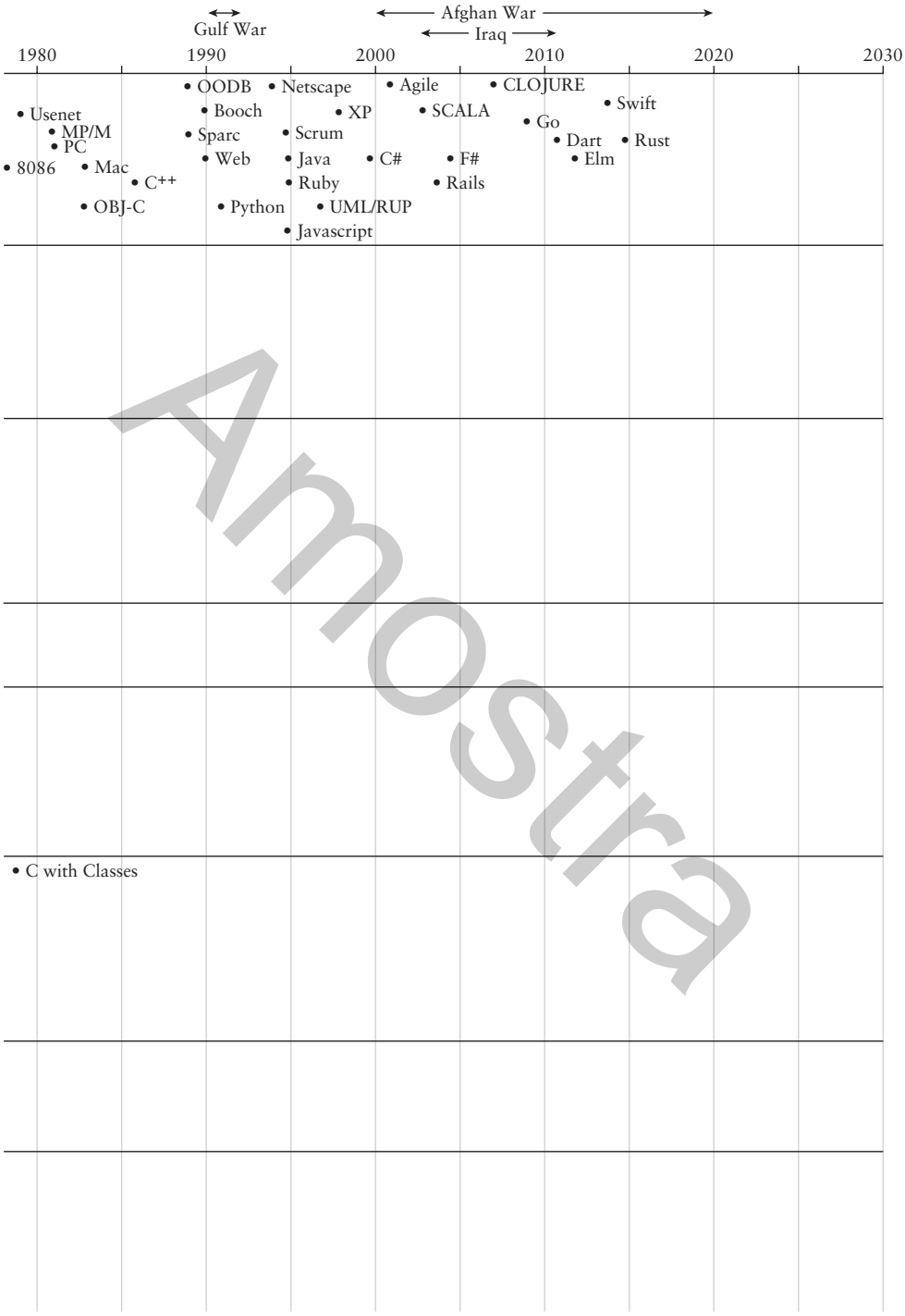
Os atentados de 11 de setembro e o estouro da bolha das dotcom atrasaram um pouco o movimento, mas logo surgiram a revolução Ruby/Rails e a era dos dispositivos móveis. A internet se espalhou por *todos os lugares*; as redes sociais floresceram e, depois, entraram em declínio enquanto a inteligência artificial emergia com o potencial de transformar (ou prejudicar) tudo.

E isso nos traz ao presente e levanta reflexões sobre o futuro. Tudo isso, e muito mais, será explorado nas páginas deste livro. Então, se você está preparado, aperte os cintos — a viagem promete ser intensa.

Linha do Tempo

Neste livro, as pessoas e os acontecimentos narrados estão organizados nesta linha do tempo. Conforme você avança nas histórias e se depara com os eventos, pode situá-los no contexto ao encontrá-los aqui. Por exemplo, pode ser interessante saber que o FORTRAN e o Sputnik ocorreram simultaneamente, ou que Ken Thompson ingressou na Bell Labs antes que Dijkstra afirmasse que as instruções GOTO eram prejudiciais.


WW2
Korean War
Vietnam War
1930
1940
1950
1960
1970
Events
• Pearl Harbor
• IBM 701
• IBM 7090
VAX •
• IBM 704
• UNIVAC 1107
• Trinity
• UNIVAC 1108
• CP/M
• Manchester Baby
• PDP7
Apple][•
• PDP8
• 8080
• Sputnik
• PDP11
• ECP-18
• DN30&335
• H200
Hilbert
• Hilbert's Challenge
• Hilbert Dies
• Von Neumann Dies
• Gödel's Incompleteness
• Hilbert: No Math in Gottingen
• Turing Decideability
• Turing Dies
• Von Neumann Meets Turing
• Von Neumann to London (NCR)
Turing to Bletchley Park •
• Von Neumann to Los Alamos
• Von Neumann sees Mark I & ENIAC. Writes EDVAC draft.
Hopper
• Hopper Joins Mark I
• Betty Snyder's Merge Sort
• Mark I Symposium
• Hopper Joins UNIVAC
• Hopper: Automatic Programming
• A-O
• Symposium: Automatic Programming
• B-O Flowmatic
• CODASYL
• COBOL
Backus
• SSEC
• Speedcoding
• Fortran
• BNF ALGOL
Dijkstra
• Cambridge EDSAC
• THE
• MCC ARRA
• Gatlinburg
• Programmer
• Go to Harmful
• FERTA
• ARMAC
• Min Path
• XI
• First ALGOL
• Dark Future
Nygaard
• Nygaard NDRE
• Dahl NDRE
• Monte Carlo Compiler
• Simula Spec
• Nygaard pitches UNIVAC
• 1107 Delivered
• Simula I
• Simula 67
Linguagem Simula 67 é comercializada •
Stroustrup Arrhus •
Kemeny
• Kemeny Hears Von Neumann on EDVAC
• Hired at Dartmouth
• LGP-30
• DN30 & DN235
• BASIC & Timesharing
Ritchie
• Ken Thompson Bell
• Dennis Ritchie Bell
• Dennis Ritchie sem tese
• Brian Kernighan Bell
MULTICS descontinuado •
• UNIX PDP7
• UNIX PDP11
• C
K&R •


Gulf War
Afghan War
Iraq
1980
1990
2000
2010
2020
2030
• Usenet
• MP/M
• PC
• 8086
• Mac
• C++
• OBJ-C
• OODB
• Booch
• Sparc
• Web
• Python
• Netscape
• Scrum
• Java
• Ruby
• Javascript
• XP
• C#
• UML/RUP
• Agile
• SCALA
• F#
• Rails
• CLOJURE
• Go
• Swift
• Dart
• Rust
• Elm
• C with Classes

Sobre este livro

Antes de começarmos, preciso esclarecer alguns pontos sobre o livro e sobre o autor que acredito ser importante aos leitores.

- Conto com 60 anos de programação na data de hoje, ainda que, para ser justo, os anos da minha adolescência, dos 12 aos 18, talvez devessem ser vistos de outra forma. Mesmo assim, desde 1964 até hoje, participei em grande parte da "era da computação". Vivi e presenciei muitos dos eventos marcantes, e até mesmo basilares, desse campo. Assim, o que você está prestes a ler foi escrito por alguém que integra um grupo pequeno, e cada vez menor, dos primeiros desbravadores na área. E, embora esse grupo não possa se gabar de ter sido o primeiro, *podemos* dizer que pegamos o bastão que nos foi passado.
- Esta obra abrange dois séculos. Muitos podem achar que os nomes e ideias apresentados nas histórias são pouco familiares — quase perdidos na poeira do tempo. Por isso, ao final deste livro, você encontrará um *Glossário de Termos*, contendo descrições dos principais equipamentos mencionados, e uma *lista de Personagens Secundários* — um elenco de pessoas tacitamente citadas nos textos que se seguem.

- O *Glossário de Termos* traz definições dos hardwares mencionados. Se encontrar um computador ou dispositivo que não reconheça, dê uma olhada nele.
- A *lista de Personagens Secundários* reúne os nomes daqueles que, direta ou indiretamente, influenciaram a indústria da programação. Embora extensa, ela ainda fica aquém, representando apenas uma fração dos pioneiros. Muitos dos nomes que aqui aparecem se perderam na neblina do tempo e nos recantos dos mecanismos de busca. Leia a lista e surpreenda-se com as personalidades que encontrará. Lembre-se: essa relação é apenas a ponta do iceberg. Vale ainda notar que a maioria desses profissionais partiu recentemente.

Agradecimentos

Mais uma vez, expresso minha gratidão à equipe da Pearson, que se dedicou tanto a publicar este livro: Julie Phifer, Harry Misthos, Julie Nahil, Menka Mehta e Sandra Schroeder. Também agradeço à equipe de produção que refinou o conteúdo: Maureen Forys, Audrey Doyle, Chris Cleveland e outros. Trabalhar com eles é sempre um prazer.

Agradeço a Andy Koenig e Brian Kernighan por me ajudar a estabelecer conexões.

Um agradecimento especial a Bill e John Ritchie por me proporcionarem tantas percepções valiosas sobre seu irmão, "Dear old DMR"[1].

Meus agradecimentos a Michael Paulson (também conhecido como ThePrimeagen) pelo Prefácio encantador.

Agradeço a Tom Gilb por sua hospitalidade, por suas ideias e por ter escrito um dos Posfácios mais divertidos que já li.

1. "DMR" são as iniciais de Dennis M. Ritchie, o gênio da computação por trás da criação da linguagem C e do sistema operacional UNIX. A expressão "Dear old DMR" é um termo de afeto e familiaridade usado por seus irmãos, que pode ser entendido como "nosso querido e velho DMR" ou, considerando seu falecimento, "nosso saudoso e querido DMR". [N. da T.]

Agradeço a Grady Booch, Martin Fowler, Tim Ottinger, Jeff Langr, Tracy Brown, John Kern, Mark Seeman e Heather Kanser por revisarem o manuscrito quando ele ainda estava bem bruto. A ajuda deles fez toda a diferença.

Como sempre, agradeço à minha maravilhosa e bela esposa, o amor da minha vida, e aos meus quatro filhos espetaculares e dez netos igualmente fantásticos. Vocês são tudo para mim. Escrever sobre software é apenas uma diversão.

Por último, não posso deixar de agradecer por ter uma vida perfeita — moro no paraíso.

Sobre o Autor

Robert C. Martin (Uncle Bob) é programador desde 1970. Ele fundou a Uncle Bob Consulting, LLC e, com seu filho Micah Martin, cofundou a Clean Coders, LLC. Martin publicou dezenas de artigos em diversas publicações especializadas e é palestrante frequente em congressos internacionais e feiras do setor. É autor e editor de inúmeros livros, entre os quais: *Designing Object-Oriented C++ Applications Using the Booch Method*; *Pattern Languages of Program Design 3*; *More C++ Gems*; *Extreme Programming in Practice*; *Agile Software Development: Principles, Patterns, and Practices*; *UML for Java Programmers*; *Código Limpo*; *O Codificador Limpo*; e *Functional Design: Principles, Patterns, and Practices*. Referência na área de desenvolvimento de software, Martin trabalhou durante três anos como editor-chefe do *The C++ Report* e foi o primeiro presidente da Agile Alliance.

I Preparando o cenário

Quem somos nós, programadores, e por que estamos aqui? E que máquinas são essas que tanto nos esforçamos para dominar?

1 Quem Somos Nós?

Nós, programadores, somos os que se comunicam com as máquinas e as fazem funcionar. Somos nós que damos vida a essas máquinas e, por consequência, às nossas economias e sociedades. Nada acontece neste mundo sem a gente. Nós comandamos o mundo!

As outras pessoas acreditam que são as donas do mundo, mas acabam nos repassando as regras — e somos nós que as transformamos nos comandos que as máquinas executam para governar tudo.

Mas nem sempre ocupamos essa posição dominante e indispensável. Nos primórdios da programação, os programadores eram invisíveis. Todos os olhos estavam voltados para os computadores e suas enormes promessas. Eram as máquinas — e aqueles que as criavam — que se destacavam e impressionavam, enquanto os programadores, que simplesmente faziam com que elas funcionassem, passavam despercebidos. Éramos apenas um ruído de fundo.

Sobre os primórdios da programação, Dijkstra disse:[1]

1 *The Humble Programmer* [O Programador Humilde], ACM Turing Lecture, 1972.

> "Como [cada computador] era uma máquina distinta, [o programador] sabia muito bem que seus programas teriam relevância apenas local e, além disso, como era evidente que aquela máquina teria uma vida útil limitada, ele sabia que muito pouco de seu trabalho teria valor duradouro".

Além do mais, o que fazíamos naquela época dificilmente poderia ser chamado de profissão, ou disciplina, ou mesmo um trabalho claramente delineado. Na prática, éramos verdadeiros duendes. De alguma forma, conseguimos fazer com que aqueles gigantes não confiáveis, rabugentos e terrivelmente caros, com velocidades lentas de processamento e memórias minúsculas, fizessem algo útil — às vezes. E fazíamos isso por meio dos métodos mais feios e perversos. Mais uma vez, recorrerei a Dijkstra:[2]

> "E naquela época, muitos programadores criativos extraíam uma imensa satisfação intelectual a partir dos truques astutos com os quais conseguiam ajustar o impossível às limitações de seus equipamentos."

Essa imagem dos programadores demorou a mudar. Na verdade, durante as décadas de 1960 e 1970, nossa imagem piorou: passamos da elite de jaleco branco que trabalhava isolada, para uma massa de nerds vista como mão de obra técnica, confinada em cubículos em vastos escritórios. Foi somente nos últimos anos que os programadores voltaram a ostentar status de profissionais de colarinho branco. E mesmo atualmente, nossa civilização ainda não entende por completo o quanto depende de nós, assim como nós não percebemos totalmente o poder que exercemos.

Por anos, fomos considerados o mal necessário. Executivos e gerentes de produto sonhavam com o dia em que os programadores não seriam mais necessários. E havia razão para essa esperança, pois as máquinas continuavam a evoluir em processamento e capacidade — não apenas um pouco, mas em uma proporção simplesmente inimaginável.

2 Ibid.

E, no entanto, o sonho de uma sociedade sem programadores jamais se concretizou. Na verdade, a necessidade de programadores nunca diminuiu; ela só cresceu para acompanhar o avanço das máquinas. Em vez de nos tornarmos menos essenciais e menos capacitados, os programadores passaram a ser ainda mais imprescindíveis, exigindo habilidades cada vez maiores. Na prática, os programadores se especializaram, como os médicos. Hoje em dia, você tem que contratar o *tipo certo* de programador.

Apesar dos melhores esforços para eliminar nossa presença, a demanda por programadores só aumentou e se diversificou. Agora, muitos acreditam que a inteligência artificial será a solução para tudo. Mas, acredite, o resultado será o mesmo: com o aumento do processamento das máquinas, a necessidade e o valor dos programadores aumentará ainda mais.

Essa trajetória, da invisibilidade ao protagonismo, pode ser percebida nos filmes populares da época. Robby, o robô de *Planeta Proibido* (1956), era o mordomo britânico de postura imperturbável. A máquina era o personagem. Quase não se percebia que seu código fora escrito por um cientista maluco.

O robô de *Perdidos no Espaço* (1965) é semelhante. Supostamente programado pelo Dr. Smith, o robô era ele mesmo um personagem — afinal, o personagem era a máquina.

HAL 9000, de *2001: Uma Odisseia no Espaço* (1968), era o protagonista. O Dr. Chandra, o programador, foi mencionado apenas brevemente: quando a máquina cantou "Daisy Bell" ao ser desconectada.

Esse padrão se repete indefinidamente. Em *Colossus 1980* (1970), a máquina é o protagonista principal, e os programadores são vítimas desamparadas que, por fim, se tornam escravizados.

Em *Short Circuit: O Incrível Robô* (1986), a máquina assume o protagonismo, enquanto conta com suporte de um programador ingênuo, azarado e atrapalhado.

Em *Jogos de Guerra* (1983), vislumbramos o início de uma transição. O computador, Joshua (ou WOPR), é o personagem, mas o programador está presente ajudando a resolver a situação — embora desempenhe um papel secundário em relação a um adolescente, o verdadeiro herói da história.

A verdadeira mudança chegou com *Jurassic Park* (1993). Os computadores eram importantes, mas não eram personagens. Dennis Nedry, o programador-chefe, era o personagem — e o vilão da trama.

Como os tempos mudaram. Em agosto de 2014, ministrei uma palestra aos programadores da Mojang, em Estocolmo. Mojang é a empresa que nos trouxe o *Minecraft*. Após a palestra, fomos todos tomar uma cerveja e nos acomodamos num agradável beer garden, cercado por uma cerca viva. Um garoto que estava na rua, com cerca de 12 anos, correu até a cerca e chamou um dos programadores: *"Você é o Jeb?"* Jens Bergensten, de longos cabelos ruivos e de óculos, assentiu com serenidade e lhe deu seu autógrafo.

Os programadores se tornaram os heróis dos jovens que crescem *desejando* ser como nós.

Por que Estamos Aqui?

Por que estamos aqui? Por que, nós, programadores, existimos?

Talvez essa pergunta seja muito existencial. Aff... vamos mudar: por que somos necessários? Por que as pessoas nos pagam para fazer o que fazemos? Por que elas não fazem o que fazemos?

Talvez você pense que é porque somos inteligentes. Somos, é claro, mas esse não é o motivo. Talvez você acredite que seja porque somos especialistas em tecnologia. E, de fato, somos; mas, novamente, esse também não é o motivo. O motivo pode surpreendê-lo. Provavelmente não é o que você espera. Na realidade, ele se encontra em um traço de nossa personalidade que, fará com que você estremeça, ao encarar a seguinte verdade: adoramos detalhes. Temos fascínio pelos detalhes. Nadamos

contra a corrente em rios de detalhes. Atravessamos pântanos e brejos repletos de detalhes. E adoramos isso. Vivemos para isso. Trabalhamos com afinco e alegria. Somos… gerentes de detalhes.

Mas isso ainda não responde ao porquê de sermos necessários. A resposta é simples: a sociedade já não funciona sem telefones.

Telefones. Por que os chamamos assim? Na verdade, eles não são telefones! Eles nada têm a ver com os telefones de antigamente. Alexander Graham Bell jamais apontaria para um iPhone e diria que o aparelho é descendente direto da invenção dele e de Watson.

Eles não são telefones; são supercomputadores de bolso. São portais e porta de entrada para informações, fofocas, entretenimento e… tudo. E não conseguimos imaginar a vida sem eles. Sem nosso tempo diante da tela, nos encolheríamos de tal forma que a depressão se tornaria inevitável.

Claro, estou fazendo uma brincadeira com a situação; mas a graça está justamente no fato de tudo isso ser inegavelmente verdadeiro. Se, de repente, nossos celulares parassem de funcionar, nossa civilização chegaria ao fim num instante.

Mas o que isso tem a ver conosco? Por que precisamos gerenciar todos os detalhes dos telefones? Por que nem todo mundo consegue fazer isso sozinho?

Tenho certeza de que conhece alguém que teve uma grande *ideia de um app* sensacional e quis que você o programasse. Essa pessoa sabia que faria uma fortuna com a ideia — estaria disposta, inclusive, a dividir essa fortuna com você numa proporção de 20 a 80, se você *simplesmente escrevesse o código*.

É isso mesmo. Basta "só" escrever o código. Sem mistério.

Por que *elas* não escrevem o código? Afinal, a ideia é *delas*. Por que elas *simplesmente* não escrevem o código?

A resposta óbvia é: elas não sabem. Mas isso não é verdade. Elas sabem. Elas conseguem descrever com clareza o comportamento

que desejam — ainda que de forma um tanto abstrata. Então, por que não conseguem transformar essas ideias abstratas em realidade?

Suponha que Jimmy, nosso empreendedor entusiasmado, esteja absolutamente convencido de que ganhará rios de dinheiro se ele conseguir desenhar uma única linha vermelha[3] na tela. Só isso. Tipo, todo mundo quer uma linha vermelha, não é mesmo? Então, como Jimmy deve proceder?

Jimmy encara seu celular e tudo o que vê é um prisma retangular com algumas pequenas protuberâncias, recuos e cavidades. O que ele deveria fazer com aquilo? Claro que ele precisa de um programador, porque nós sabemos lidar com esse tipo de coisa.

Mas espere. Você já espirrou enquanto segurava o celular? Já reparou nas minúsculas gotículas que ampliam os detalhes da tela? Basta observar por um instante o interior dessas pequenas gotículas de secreção para perceber que existem *pontos* ali. Pontos *coloridos*. Na verdade, são pontos vermelhos, verdes e azuis dispostos como se estivessem organizados em uma grade retangular.

Assim, ao espirrar, Jimmy enxerga esses pontos e, por um breve instante, se dá conta de algo: sua linha vermelha será formada a partir de pontos vermelhos!

Se ele se permitir pensar um pouco mais, perceberá que essas três cores podem ser combinadas. E, indo além, compreenderá que deve haver um modo de controlar o brilho de cada ponto para criar uma variedade de cores. Pode ser que ele nem saiba sobre o sistema RGB, mas todo mundo aprendeu na escola que misturar certas cores gera outras. O conceito, afinal, não é complicado.

Caso pense mais um pouco, ele perceberá que a grade retangular indica que esses pontos têm coordenadas. Talvez ele não se lembre

3. Lauris Beinerts. "The Expert (Short Comedy Sketch)". Publicado no YouTube em 23 de março de 2014. (Estava disponível no momento em que eu escrevia este livro.)

dos conceitos de álgebra do primeiro ano ou de coordenadas cartesianas, mas, novamente, até uma criança entende a ideia de uma grade retangular. Afinal, quando você disca os números em seu celular, os botões estão dispostos dessa forma!

Tá bom, eu sei, hoje em dia ninguém disca as teclas de um telefone e muitos nem sabem qual é o significado da palavra *discagem*. Mas vamos deixar isso de lado. Em apenas alguns instantes preciosos de reflexão, Jimmy conclui que pode desenhar sua linha vermelha ativando os pontos vermelhos que se alinham, de forma… linear.

"Como é que se faz isso?" — Jimmy se pergunta. Será que ele se recorda da velha fórmula linear: $y=mx+b$? Provavelmente não. Mas, após mais alguns instantes de reflexão, ele perceberá que os pontos que compõem sua linha vermelha obedecem a uma proporção fixa entre o vertical e o horizontal. Talvez ele nem perceba que está redescobrindo os fundamentos do cálculo; mas tudo bem. A ideia de inclinação de uma reta não é um conceito tão difícil de compreender.

Se continuar refletindo, Jimmy perceberá que a única razão pela qual consegue enxergar esses pontos é porque o muco na tela atua como o microscópio de Leeuwenhoek. Os pontos são *minúsculos*! Isso significa que ele precisará desenhar *muitos* deles, e suas coordenadas terão, todas, que obedecer à sua fórmula do coeficiente angular. Como ele vai fazer *isso*?

Além do mais, se ele desenhar apenas uma linha de pontos, essa linha será muito fina. Talvez nem seja visível! Por isso, será necessário desenhar a linha com certa *espessura*.

Nesse ponto, os momentos de reflexão de Jimmy já consumiram quase uma hora, e ele se dá conta de que deixou de pensar nos milhões que sua linha vermelha poderia render e ficou preso em analisar *os pontos*. Talvez ele também perceba que apenas refletiu superficialmente sobre o problema que tem diante de si. Afinal, ele não sabe como mandar o celular acender esses pontos, muito menos como transmitir coordenadas para ele, ou dizer para repetir a ação até que todos os pontos sejam ativados!

E, o mais importante, *ele simplesmente não quer mais pensar nisso*! Ele quer voltar a se concentrar em como sua linha vermelha vai lhe render uma fortuna e em como posicionará os anúncios dela na plataforma X, e assim por diante…

Dane-se os pontos. Jimmy vai deixar que algum programador se preocupe com essa parte. Ele não *quer* se aprofundar nesse nível de detalhe. *Mas nós queremos!* Essa é a nossa marca registrada — nosso defeito de personalidade e nosso superpoder. Adoramos todos esses detalhes. Ficamos encantados em descobrir como juntar inúmeros pontinhos na tela para formar uma linha vermelha. A linha propriamente dita não nos interessa muito.

O que nos fascina é o desafio de combinar todos esses detalhes minúsculos em uma linha vermelha.

Então, por que somos necessários? Porque a sociedade precisa de pessoas que se importam com os detalhes. Esses profissionais — nós — liberam a sociedade para pensar em outras coisas, como o Desafio do Balde de Gelo, *Angry Birds* ou até mesmo jogar paciência enquanto espera no consultório do dentista.

Enquanto a esmagadora maioria das pessoas evitar os detalhes, sempre haverá a necessidade daqueles entre nós que mergulham de cabeça neles. É isso que somos. Somos os gerentes de detalhes de todo o mundo.