# Aprenda a programar em dez anos

***Teach Yourself Programming in Ten Years[1] (2014)***
***http://norvig.com/21-days.html***


## Peter Norvig

***Tradução e adaptação para a língua portuguesa: Prof. Augusto Manzano[2]***


**Por que todo mundo está com tanta pressa?**

Entre em qualquer livraria, e você verá títulos de livros como: *Teach Yourself Java in 24 Hours* (*Aprenda Java em 24 Horas*) assim como diversas variações oferecendo lições de C, SQL, Ruby, Algoritmos e muitos outros em dias ou horas. Realizei no sítio da Amazon pesquisa com os termos [title: teach, yourself, hours, since: 2000] e obtive uma lista com 512 livros[3]. Os primeiros dez por cento da lista eram livros relacionados à computação, os demais eram relacionados a outros assuntos. Troquei "dias" por "horas" e obtive resultados similares[4].

A conclusão é que ou as pessoas estão com muita pressa em aprender computação ou as atividades relacionadas ao estudo da programação de computadores são extremamente fáceis de aprender se comparado a qualquer outro assunto relacionado a computadores.

Analisando o que um título como *Teach Yourself C++ in 24 Hours* (*Aprenda C++ em 24 Horas*)[5] pode significar:

- **Aprenda** (*Teach Yourself*): Em vinte e quatro horas você não terá tempo suficiente de escrever programas que sejam significativos e não conseguirá aprender com seu sucesso ou fracasso. Você não terá tempo para trabalhar com um programador experiente e entender o que é conviver dentro do ambiente de desenvolvimento. Em resumo, você não terá tempo para aprender muito. Logo este livro aborda tão somente um entendimento superficial do assunto, como disse Alexander Pope "Aprender pouco é uma coisa perigosa".
- **C++**: Em vinte quatro horas você deve ser capaz de aprender a sintaxe básica do C++ (isso se você já tiver algum conhecimento de uma linguagem de programação similar), mas não vai aprender muito sobre como utilizar essa sintaxe. Em resumo, se você era, vamos dizer um programador BASIC[6], você aprenderá a escrever programas no estilo BASIC usando a sintaxe C++ mas não aprenderá em que C++ é bom (ou ruim). Então, qual o ponto? Alan Perlis disse em certa ocasião: "Uma linguagem que não afeta a maneira de como você pensa sobre programação, não vela a pena ser aprendida". Se você necessita aprender um pouco de C++ (*JavaScript* ou *Processing*) para interagir com alguma ferramenta existente para resolver uma tarefa específica, de fato, você não estará aprendendo a programar, estará aprendendo apenas como resolver essa tarefa.
- **em 24 Horas** (*in 24 Hours*): Não é tempo suficiente para aprender algo complexo, como indicado a seguir.

---

[1] NT (nota tradutor): A data original da publicação do artigo é incerta. Há uma referência no sítio do autor http://norvig.com/21-days.html indicando os anos de 2001 e 2014. Algumas pesquisas em sítios Internet apontam 2001 como possível ano de publicação da primeira versão do artigo, como por exemplo https://news.ycombinator.com/item?id=3439772. Possivelmente esta seja a data próxima de publicação, uma vez que Peter Norvig, usa em uma de suas pesquisas indicada no artigo do ano de 2000 como ano de referência para este artigo. No entanto, o artigo passou por revisões do autor e sofreu uma pequena atualização do texto, existindo uma segunda edição datada de 2014. A tradução hora apresentada é baseada na segunda versão do texto original, mantendo alusão a detalhes indicados na primeira versão do artigo.

[2] NT: O texto original deste artigo encontra-se publicado no sítio do autor, possuindo diversos *links* para acesso de informações complementares. Desta forma, as mesmas informações indicadas no texto original estão grafadas neste texto via notas de rodapé com a referência LE de *link externo*.

[3] NT: Pesquisa similar realizada em janeiro de 2018 retorna aproximadamente 960 referências.

[4] NT: Na primeira edição do artigo a referência correspondia a “Encontrei 248 entradas. As primeiras 78 eram livros sobre computadores, o septuagésimo nono era *Learn Bengali in 30 days*. Troquei “dias” por “horas” e obtive resultados incrivelmente similares: 253 livros, 77 de computadores, seguidos de *Teach Yourself Grammar and Style in 24 Hours* no septuagésimo oitavo lugar. Do total de 200, 96% eram livros de computadores”.

[5] NT: Na primeira versão do artigo o nome do livro indicado era *Learn Pascal in Three Days* (Aprenda Pascal em Três Dias).

[6] NT: No texto original, Peter Norvig refere-se a linguagem de programação como “Basic”. No entanto, o nome da linguagem é caracterizado por uma sigla e por esta razão deve ser escrita como BASIC - *Beginner's All-purpose Symbolic Instruction Code* (código de instrução simbólica para todos os iniciantes). Ao grafar BASIC como Basic pode dar a entender o uso da palavra como adjetivo *basic* (básico como basilar, essencial, fundamental, primordial, mais importante).

**Aprenda a Programar em Dez Anos.**

Bloom (1985), Bryan & Harter (1899), Hayes (1989) e Simmon & Chase (1973) indicam que se leva em média dez anos para desenvolver perícia em qualquer de uma variedade de áreas do conhecimento, incluindo jogar xadrez, compor músicas, operar telégrafos, pintar, tocar piano, nadar, jogar tênis e pesquisar neuropsicologia e topologia. Não há atalhos: mesmo Mozart, que foi um prodígio musical aos 4 anos de idade levou mais de 13 anos antes de compor alguma música com qualidade mundial. Os Beatles parecem ter disparado nas paradas de sucesso musical em primeiro lugar com a aparição no show do Ed Sullivan em 1964. Mas eles estavam tocando em pequenos clubes em Liverpool e Hamburgo desde 1957, e mesmo que tenham conseguido alguma aparição em massa, o primeiro grande sucesso Sgt. Peppers veio somente em 1967.

Malcolm Gladwell[7] popularizou a ideia de que a experiência necessária vem após 10.000 horas e não com 10 anos. Henri Cartier-Bresson (1908-2004) partiu de outra métrica: "Suas primeiras 10.000 fotografias são seu pior trabalho." (Ele não previu que com câmeras digitais algumas pessoas podem chegar a essa marca em apenas uma semana). A verdadeira experiência para ser obtida pode levar uma vida inteira. Samuel Johnson (1709-1784) afirma que “Excelência em qualquer setor pode ser alcançada com o trabalho de uma vida toda; não é possível comprá-la por menos.”. Chaucer (1340-1400) complementa que: “a vida é muito curta, leva-se muito tempo para aprender.” Hippocrates (460aC-375aC)[8] conhecido pela citação *ars longa, vita brevis* (arte longa, vida breve) tirada do trecho "*Ars longa, vita brevis, occasio praeceps, experimentum periculosum, iudicium difficile*" (Arte longa, vida breve, ocasião repentina, experiência perigosa, difícil julgamento), que pode ser considerada “A vida é curta, [o] ofício é longo, a oportunidade é fugaz, a experiência é traiçoeira e o julgamento é difícil”. É óbvio que nenhum número pode expressar a resposta final: não parece razoável supor que todas as habilidades (como: programar, jogar xadrez, jogar damas e reproduzir música) poderiam exigir exatamente a mesma quantidade de tempo para serem dominadas, nem que todas as pessoas tenham exatamente a mesma quantidade de tempo. Como menciona o Prof. K. Anders Ericsson[9] “Na maioria dos domínios é notável o quanto de tempo, até mesmo para os indivíduos mais talentosos, é necessário para atingir os mais altos níveis de desempenho. O número de 10.000 horas apenas dá-lhes uma sensação de que estamos falando em anos de 10 a 20 horas por semana relacionados a aqueles que são apontados como indivíduos talentosos naturalmente e que necessitam chegar ao mais alto nível”.

**Então, você quer ser um programador**

Aqui está minha receita para o sucesso na área de programação:

- Interesse-se por programação, faça porque é divertido. Tenha certeza que isso continuará a ser divertido o suficiente para que você dedique dez anos ou dez mil horas de sua vida nisso.
- Programe. A melhor forma de praticar aprendizagem é aprender fazendo[10]. De uma maneira mais técnica, “o nível máximo de desempenho para os indivíduos em um determinado domínio não é alcançado de forma automática em função da experiência estendida, mas sim do nível de desempenho que pode ser aumentado, mesmo, por indivíduos altamente experientes como resultado deliberado de seu próprio esforço em melhorar.” (p. 366)[11] e “o aprendizado mais efetivo requer uma tarefa definida com o nível de dificuldade adequado ao indivíduo em particular, sendo que há uma maneira de ocorrer o retorno sobre a experiência com a oportunidade de efetuar repetições e proceder as correções de erros.” (p. 20-21) do livro *Cognition in Practice: Mind, Mathematics, and Culture in Everyday Life* (Cognição na Prática: Mente, Matemática e Cultura na Vida Cotidiana)[12], uma referência deste ponto de vista.
- Converse com outros programadores; leia outros programas. Isso pode ser mais importante do que qualquer livro ou curso de treinamento.
- Caso deseje, invista quatro anos em uma universidade (ou mais em uma pós-graduação). Isso lhe credenciará e dará acesso a alguns empregos que requerem alguma formação e um grande entendimento do campo de trabalho, mas se você não gosta da escola, você pode (com alguma dedicação) obter experiência semelhante no trabalho ou por conta própria[13]. De qualquer forma, a aprendizagem obtida somente a partir de livros não

[7] LE: Autor do livro *Outliers: The Story of Success* publicado pela *Little, Brown and Company* no ano de 2008 com 309 páginas disponível em https://www.amazon.com/Outliers-Story-Success-Malcolm-Gladwell/dp/0316017922. Em 27 fev. 2017.

[8] NT: No texto original a data é referenciada como (c. 400BC) tendo necessitado de pesquisa mais aprofundada. Visite o sítio https://global.britannica.com/biography/Hippocrates. Em 27 fev. 2017.

[9] LE: Autor de diversos livros que podem ser obtidos em https://www.amazon.com/K.-Anders-Ericsson/e/B000APB8AQ/ref=dp_byline_cont_book_1. Em 27 fev. 2017.

[10] LE: Visite o sítio http://www.engines4ed.org/hyperbook/nodes/NODE-120-pg.html. Em 27 fev. 2017.

[11] LE: Visite o sítio http://www.umassd.edu/. Em 27 fev. 2017.

[12] LE: Visite o sítio https://www.amazon.com/exec/obidos/ASIN/0521357349. Em 27 fev. 2017.

[13] NT: No entanto, é necessário considerar que a cada dia as organizações estão exigindo profissionais com qualificação técnica.

será suficiente. “Educação em ciências da computação não faz de ninguém um gênio em programação tanto quanto estudar pincéis e pigmentos não faz um bom pintor.” segundo Eric Raymond, autor do livro *The New Hacker's Dictionary* (O Novo Dicionário do Hacker)[14]. Um dos melhores programadores que contratei tinha só o ensino médio; ele produziu um grande software, tem seu próprio grupo de discussão e fez dinheiro suficiente em mercado de ações para comprar sua própria *nightclub*[15].

- Trabalhe em projetos com outros programadores. Seja o melhor programador em alguns projetos, seja o pior em outros. Quando você é o melhor você testa suas habilidades para liderar um projeto e para inspirar outros com a sua visão. Quando você é o pior aprende o que os mestres fazem e o que não gostam de fazer (porque eles fazem você fazer por eles).
- Trabalhe em projetos após outros programadores. Entenda um programa escrito por outra pessoa. Veja o que é preciso para entender e corrigir quando os programadores originais não estão por perto. Pense em como projetar seus programas para tornar mais fácil para aqueles que irão mantê-los depois de você.
- Aprenda ao menos meia dúzia de linguagens de programação. Inclua uma linguagem que enfatize abstração de classes (como Java ou C++), uma que enfatize abstração funcional (como Lisp, ML ou Haskell), uma que suporta abstração sintática (como Lisp), uma que suporta especificações declarativas (como Prolog ou C++ com templates) e uma que enfatize paralelismo (como Clojure ou Go[16])[17].
- Lembre-se, existe um “computador” na “ciência da computação”. Saiba quanto tempo leva para seu computador executar uma instrução, buscar uma palavra na memória (com ou sem cache), ler palavras consecutivas da memória secundária[18] e procurar por uma nova posição em disco (veja a tabela 1).
- Envolva-se no esforço de padronização de uma linguagem. Pode ser o comitê ANSI C++, ou na padronização do estilo de programação local, se é usada identação com 2 ou 4 espaços. De qualquer maneira, você aprende sobre o que outras pessoas gostam em uma linguagem, o quanto elas se sentem sobre isso e talvez até sobre o motivo de como elas se sentem assim.
- Tenha o bom senso de sair do processo de padronização o quanto antes.

Com isso em mente, é questionável o quão longe você pode ir apenas lendo livros. Antes do nascimento do meu primeiro filho eu li todos os livros *How To* (Como Fazer) sobre o assunto e mesmo assim me sentia como um novato. Trinta meses depois, quando nasceu meu segundo filho, voltei aos mesmos livros para uma atualização? Não. Em vez disso resolvi confiar na minha experiência pessoal com o primeiro filho, que acabou por ser mais útil e reconfortante do que milhares de páginas escritas por especialistas.

Fred Brooks, em seu artigo *No Silver Bullets: Essence and Accidents of Software Engineering* (Sem Balas de Prata: Essência e Acidentes da Engenharia de Sofware)[19] identificou um plano de três partes para encontrar grandes projetistas de software[20]:

1. Sistematicamente identifique os melhores projetistas o quanto antes.
2. Tenha um orientador de carreira, que seja responsável pelo desenvolvimento de um minucioso plano de carreira.
3. Promova oportunidades para desenvolvedores iniciantes interagirem uns aos outros.

Isso pressupõe que algumas pessoas possuem as qualidades necessárias para serem grandes projetistas; o trabalho é encorajá-las adequadamente. Alan Perlis[21] explica que “Todos podem ser ensinados a esculpir: Michelangelo teria que ser ensinado a não esculpir, assim é com os grandes programadores”.

---

[14] NT: Visite o sítio https://www.amazon.com/New-Hackers-Dictionary-3rd/dp/0262680920. Em 27 fev. 2017.

[15] NT: Casa noturna, clube noturno ou boate.

[16] NT: Na primeira versão do artigo Peter Norvig menciona nesta categoria, além das linguagens Clojure ou Go a linguagem Sisal.

[17] NT: Além das linguagens indicadas na primeira versão do artigo Peter Norvig menciona na lista as linguagens Icon e Scheme que dão suporte à programação com co-rotinas.

[18] NT: Memória secundária se refere a qualquer mecanismo de gravação de dados, podendo ser um disco rígido, um disquete (sobrevivente), uma mídia óptico-magnética como CDs e DVDs, unidades *flash* como *pendrives*, entre outras possibilidades de armazenamento estático de dados.

[19] NT: Norvig refere-se ao artigo indicado simplesmente como *No Silver Bullets*. Mas ao fazer o acesso ao *link* indicado encontra-se o artigo denominado como *No Silver Bullet – Essence and Accidents of Software Engineering*. Por este motivo optou-se na tradução em manter o nome indicado na fonte consultada.

[20] NT: Apesar das restrições pessoais que o tradutor tem em relação ao serviço Wikipédia devido sua falta de controle nos temas publicados, mantém sua indicação devido a referência ter sido feita por Peter Norvig e acrescenta a indicação do sítio http://worrydream.com/refs/Brooks-NoSilverBullet.pdf onde pode-se ter acesso ao artigo completo de Fred Brooks. Em 27 fev. 2017.

[21] LE: Visite o sítio http://pu.inf.uni-tuebingen.de/users/klaeren/epigrams.html. Em 27 fev. 2017.

Perlis está dizendo que os grandes têm alguma qualidade interna que transcende sua formação. Mas, de onde vem esta qualidade? É inata? Ou eles desenvolvem-na através da diligência? Como Auguste Gusteau (chef de cozinha, no filme de ficção Ratatouille) coloca, "qualquer um pode cozinhar, mas só o destemido pode ser grande." Penso nisso mais como vontade de dedicar uma grande parte de sua vida à prática deliberativa. Mas talvez seja uma forma de resumir isso. Ou, como o crítico de Gusteau, Anton Ego, diz: "Nem todos podem se tornar um grande artista, mas um grande artista pode vir de qualquer lugar".

Então, vá em frente, compre aquele livro de Java/Ruby/Javascript/PHP; você provavelmente terá alguns deles para uso. Mas isso não mudará sua vida ou sua experiência como programador em 24 horas ou 21 dias. Que tal trabalhar duro para melhorar continuamente ao longo de 24 meses? Bem, agora você está começando a chegar em algum lugar...

**Bibliografia**


Bloom, B. *Developing Talent in Young People*. New York: Ballantine Books, 1985. 572 p. ISBN 978-03-4531-509-0.

Brooks, F. P. J. *No Silver Bullets:* Essence and Accidents of Software Engineering. IEEE Computer, Washington, v. 20, n. 4, p. 10-19, 1987. DOI 10.1109/MC.1987.1663532.

Bryan, W. L. & Harter, N. *Studies on the telegraphic language*: The acquisition of a hierarchy of habits. Psychology Review, v. 6(4), p. 345-375, jul 1899.

Hayes, J. R. *The Complete Problem Solver*. New Jersey: Lawrence Erlbaum Associates, 1989. 376 p. ISBN 978-08-0580-309-9.

Chase, W. G. & Simon, H. A. *Perception in Chess*. Department of Psychology Carnegie-Mellon University, Pittsburgh, v. 4(1), p. 55-81, jul 1973.

Lave, J. *Cognition in Practice*: Mind, Mathematics, and Culture in Everyday Life. Cambridge: Cambridge University Press, 1988. 232 p. ISBN 978-05-2135-734-0.


**Apêndice: Escolha de Linguagens**

Várias pessoas me perguntam sobre quais linguagens de programação devem aprender primeiro. Não existe uma resposta, mas considere estes pontos:

- *Use seus amigos*. Quando me perguntam “que sistema operacional devo usar, Windows, Unix ou Mac?” Minha resposta normalmente é: “use o que seus amigos usam”. A vantagem que você obtém de aprender com seus amigos irá compensar qualquer diferença intrínseca entre o sistema operacional ou entre linguagens de programação. Considere seus futuros amigos: a comunidade de programadores de que você fará parte se continuar. A linguagem de programação escolhida tem uma grande comunidade crescente ou é um pequeno grupo prestes a morrer? Existem livros, sítios e fóruns *on-line* para obter respostas? Você gosta das pessoas nesses fóruns?
- *Mantenha a simplicidade*. Linguagens de programação como C++ e Java são voltadas ao uso profissional por grandes equipes de desenvolvedores experientes que estão preocupados com a eficiência do tempo de execução de seus códigos. Como resultado, essas linguagens têm partes complicadas projetadas para essas circunstâncias. Você está preocupado em aprender a programar. Você não precisa dessa complicação. Você quer uma linguagem que foi projetada para ser fácil de aprender e lembrar por um programador novato.
- *Interaja*. De que forma você aprenderia a tocar piano: de modo normal e interativo, no qual você escuta uma nota logo que pressiona uma tecla ou em modo *batch* (por lote) em que você só ouve as notas depois de terminar a música inteira? Obviamente, o modo interativo torna a aprendizagem mais fácil para o piano, e também para a programação. Insista em uma linguagem com um modo interativo e use-a.

A partir desses critérios, minhas recomendações para uma primeira linguagem seria Python ou Scheme. Outra opção é a linguagem Javascript, não porque seja perfeitamente bem projetada para iniciantes, mas porque há muitos tutoriais *on-line* para ela, como os encontrados na Khan Academy. Suas prioridades como programador podem mudar,há outras boas escolhas. Se sua idade é de um único dígito, você pode optar e preferir Alice, Squeak ou Blockly (alunos mais velhos também podem desfrutar destas linguagens). O importante é que você escolha uma e comece.

**Apêndice: Livros e outros recursos**

Várias pessoas me perguntam em quais livros e páginas da web devem aprender. Repito que "a aprendizagem apenas de livros não é suficiente" mas posso recomendar o seguinte:

- **Scheme**: *Structure and Interpretation of Computer Programs* de *Abelson & Sussman* (Estrutura e Interpretação de Programas de Computadores)[22] é provavelmente o melhor livro de introdução a ciência da computação, pois ensina a programação como caminho para entender a ciência da computação. Você pode ver *online videos of lectures* (vídeos *on-line* de palestras)[23] sobre este livro, bem como o texto completo *on-line*. O livro é desafiador, talvez algumas pessoas precisem de outras abordagens para a aprendizagem.
- **Scheme**: *How to Design Programs* de *Felleisen et al.* (Como projetar programas)[24] é um dos melhores livros sobre como realmente projetar programas de uma forma elegante e funcional.
- **Python**: *Python Programming: Python Programming: An Intro to CS (Zelle)* (Programação Python: Uma Introdução à Ciência da Computação)[25] é uma boa introdução sobre a linguagem Python.
- **Python**: Vários tutorias online[26] estão disponíveis em Python.org[27].
- **Oz**: *Concepts, Techniques, and Models of Computer Programming* de *Van Roy & Haridi* (Conceitos, Técnicas e Modelos de Programação de Computadores)[28] é visto por alguns como o sucessor moderno de Abelson & Sussman. É um *tour* pelas grandes ideias de programação, abrangendo uma gama mais ampla do que Abelson & Sussman, sendo talvez mais fácil de ler e seguir. Ele usa a linguagem, Oz, que não é amplamente conhecida, mas serve como uma base para a aprendizagem de outras linguagens de programação.

**Notas**

T. Capey aponta que a página de pesquisa do livro *Complete Problem Solver* no sítio da Amazon possui os livros *Teach Yourself Bengali in 21 days* e *Teach Yourself Grammar and Style* sob a seção de livros *Customers Who Bought This Item Also Bought* (Consumidores que compram esse item também costumam comprar estes). Acredito que uma grande parte das pessoas que olham para esse livro estão vindo desta página. Obrigado a Ross Cohen por ajudar com Hipócrates.

| **Tabela1: Tempo médio de execução de várias operações num PC típico** | |
|---|---|
| **executar uma instrução simples** | **1 / 1.000.000.000 s. = 1 ns** |
| **busca a partir da memória cache L1** | **0,5 ns** |
| **branch misprediction (predictor de saltos)** | **5 ns** |
| **busca a partir da memória cache L2** | **7 ns** |
| **programação concorrente com exclusão mútua** | **25 ns** |
| **busca na memória principal** | **100 ns** |
| **envio de 2K bytes na rede de 1Gbps** | **20.000 ns** |
| **leitura sequencial de 1MB a partir da memória** | **250.000 ns** |
| **busca de nova posição em disco (seek)** | **8.000.000 ns** |
| **leitura sequencial de 1MB a partir do hard disk** | **20.000.000 ns** |
| **envio ida e volta de pacotes entre Eua e Europa** | **150 ms = 150.000.000 ns** |
| **s = segundo**<br>**ns = nanosegundo**<br>**ms = milissegundo** | |

[22] LE: Visite o sítio https://www.amazon.com/gp/product/0262011530. Em 27 fev. 2017.

[23] LE: Visite o sítio http://groups.csail.mit.edu/mac/classes/6.001/abelson-sussman-lectures/. Em 27 fev. 2017.

[24] LE: Visite o sítio https://www.amazon.com/gp/product/0262062186. Em 27 fev. 2017.

[25] LE: Visite o sítio https://www.amazon.com/gp/product/1887902996. Em 27 fev. 2017.

[26] LE: Visite o sítio https://wiki.python.org/moin/BeginnersGuide. Em 27 fev. 2017.

[27] LE: Visite o sítio https://www.python.org/. Em 27 fev. 2017.

[28] LE: Visite o sítio https://www.amazon.com/gp/product/0262220695. Em 27 fev. 2017.