EDUCATION WEEK CONTENT PARTNER
EDUCATION WEEK PDK PHI DELTA KAPPA International

A Programação de Computadores Regressa à Escola

Por Yasmin B. Kafai e Quinn Burke, *Phi Delta Kappan*
5 de setembro de 2013
Traduzido por Leonel Morgado, Universidade Aberta, a 23 de abril de 2019

Aprender a programar inicia os alunos na resolução de problemas, na conceção de aplicações e na criação de ligações online.

Testemunhamos um notável regresso às escolas da programação de computadores. Nos anos 1980, muitas escolas tinham atividades laboratoriais de programação em Basic, Logo ou Pascal, nas quais os alunos participavam uma vez por semana, como iniciação à área. Contudo, em meados da década de 1990, a maioria das escolas tinha abandonado a programação. Em grande medida, tal declínio adveio de falta de integração com os conteúdos programáticos e escassez de docentes qualificados. Contudo, também se levantara a questão da finalidade. Com a disseminação durante os anos 1990 de pacotes multimédia prontos a usar, através de discos CD-ROM, quem é que quereria a labuta de lidar com erros de sintaxe e problemas de depuração ao criar estas aplicações? Esta questão, por si só, parecia negar a necessidade de aprender a programar na escola, tendo-se acumulado com o entusiasmo gerado pela Internet. As escolas começaram a ensinar aos alunos a melhor forma de surfar a Web em vez de como mergulhar nelça e compreender como efetivamente funciona. Em geral, as escolas esqueceram a programação, algumas mesmo considerando-a completamente desnecessária e outras rotulando-a de algo demasiado difícil de ensinar e de aprender.

Mas isto está a mudar. Nos últimos cinco anos, testemunhei um renovado interesse em retomar a aprendizagem e ensino da programação em todos os níveis do pré-escolar ao secundário. Só que são as culturas da juventude, de base digital (e não as escolas) que lideram esta ressurgência (Kafai & Peppler, 2011). Aparentemente, há acesso a computadores em toda a parte, especialmente fora da escola, estando as crianças e os jovens a ser inovadores com a tecnologia — frequentemente, com dispositivos móveis — para criarem os seus próprios videojogos, os seus próprios projetos de arte interativa, até mesmo as suas próprias roupas programáveis, com têxteis eletrónicos. Além disso, os mesmos computadores que usam para estas criações ligam-nos a redes mais amplas, com outros jovens que partilham os mesmos interesses e têm idêntico empenho em ligarem-se uns aos outros através do ato de criação.

As escolas bem poderiam aprender algo com estas comunidades informais de produção criativa e participação em rede. Afinal de contas, apesar deste crescimento das comunidades juvenis interligadas, muito poucos jovens estão a usar os seus dispositivos inteligentes (computadores portáteis, tablets, telemóveis, robôs) para algo mais do que consumo massificado ou reprodução de conteúdo comercializado. Estes nativos digitais podem ser capazes de manipular tecnicamente os dispositivos mais recentes, mas –

decididamente – a sua capacidade para os empregar de forma crítica, criativa e seletiva é bem menor.

Qual é então o papel da programação na facultação de um emprego mais produtivo da tecnologia? E qual é o papel das escolas na introdução de programação a uma gama mais ampla de jovens, particularmente face às tentativas falhadas no passado, por parte das escolas, de ensinar a produzir código? Como irão as escolas abordar os desafios da diversidade e da equidade, tão prevalentes nas culturas informáticas? Face a estas questões que a educação enfrenta, no contexto da viabilidade económica deste país, temos primeiro de compreender o que é o pensamento computacional, como podemos ensiná-lo e porque é que a participação computacional conjunta em comunidades online e em escolas tradicionais proporciona novas oportunidades para envolver os alunos.

O que é o Pensamento Computacional?

Em 2006, a professora Jeannette Wing, da universidade Carnegie Mellon, definiu o pensamento computacional como sendo todos os "aspetos da conceção de sistemas, resolução de problemas e compreensão de comportamentos humanos" (2006, p. 6). Wing afirmou que a compreensão computacional do mundo proporciona uma visão específica para a compreensão de problemas e para os solucionar. O pensamento computacional (embora por vezes estritamente associado à informática) pode, na verdade, ser entendido melhor como sendo a extensão dos princípios da informática a outras áreas, para ajudar a decompor os elementos de qualquer problema, determinar as suas inter-relações e as relações das partes com o todo, para depois conceber algoritmos que permitam uma solução automatizada. O pensamento computacional não está limitado à matemática nem às ciências, aplica-se também às humanidades, em áreas como o jornalismo ou a literatura.

Pensar como um informático tem o potencial de articular e fazer avançar melhor outras disciplinas académicas. Mas como está presente o pensamento computacional no quotidiano, como é relevante? Wing apresenta vários exemplos. Atente-se à tarefa de arrumar e organizar peças de Lego. Se uma criança organizar as peças pelas regras "todos os blocos retangulares grossos na mesma caixa", "todas as peças finas noutra" e assim por diante, está a fazer aquilo a que os informáticos chamam *hashing* (indexação-dispersão). Evidentemente, a maioria das crianças (e dos adultos) arruma montes de peças de Lego simplesmente deitando-as para um balde grande. Mas imaginemos que uma criança queria construir um projeto grande com peças de Lego e precisava de construí-lo escolhendo peças específicas de uma certa sequência. Andar sempre à procura num monte grande de peças de Lego seria muito mais demorado do que procurar em peças organizadas por tipo, forma ou até pela cor. Definir estas categorias permitiria reduzir o tempo de procura e deixaria o construtor livre para se concentrar no que ele ou ela pretende realmente concretizar: construir e não procurar. Isso seria particularmente útil ao construir as estruturas mais ambiciosas e de maior rigor.

A definição de "pensamento computacional" proposta por Wing originou reações muito variadas de Informáticos e de educadores relativamente ao que deve ser considerado como literacia digital. Qual é o contributo do pensamento computacional para o raciocínio e a Comunicação, num mundo cada vez mais digital? Até que ponto é que as escolas encorajam a resolução sistemática de problemas de forma interdisciplinar, decompondo processos e problemas para determinar as suas relações, antes de os voltar

a juntar? Estas perguntas não são necessariamente uma novidade para as escolas (Grover & Pea, 2013). Embora já haja computadores nas escolas há 30 anos, o pensamento computacional não integrou o currículo. Ensinar processamento de texto e a criação de apresentações no PowerPoint não envolve os alunos nas análises mais profundas que são necessárias para o pensamento crítico e criativo (Collins & Halverson, 2009). A maior parte dos jovens não tem qualquer conceção (ou tem-na muito incipiente) da informática, sob a perspetiva das ciências da computação, como disciplina ou como algo que possam aplicar às suas vidas quotidianas. Em suma, os alunos têm de saber não apenas mais sobre informática, mas sobre o que significa, em última análise, pensar de forma mais sistemática, para resolver com mais eficiência todos os tipos de problemas.

Ensinar o pensamento computacional

Então com que se pareceria o pensamento computacional nas escolas? Como o poderíamos ensinar? A definição de pensamento computacional como conceção de sistemas, resolução de problemas e compreensão de comportamentos humanos, admita-se, tem aqui um âmbito muito amplo. Vários grupos profissionais, como a Associação de Professores de Informática norte-americana (*Computer Science Teachers Association*) e organizações não governamentais como a Shodor desenvolveram padrões académicos e atividades formativas para tornar o pensamento computacional mais acessível ao ensino pré-escolar, básico e secundário. A programação tem, invariavelmente, desempenhado um papel em todas as propostas de currículos. E contudo, embora a programação seja preeminente, nenhuma linguagem de programação é consensualmente considerada a melhor entre os proponentes. Seja a linguagem Java/JavaScript, Python, C/C++, HTML ou linguagens introdutórias como o Sratch ou o Alice, o ensino dos conceitos subjacentes à linguagem que os proporciona (e não a linguagem em si) é o que é relevante.

Então quem pode dizer que ensinar a programar com estas linguagens terá mais êxito do que o que foi testemunhado nos anos 1980, com a breve surtida do Logo e do Pascal pelas escolas?

A resposta, defendemos, é que as crianças já têm vindo a usar código para criar e partilhar. Durante a última década, uma diversidade de sítios Web gerados por jovens emergiram, empenhados em fazer e partilhar online multimédia programável, seja ela composta por videojogos, projetos de arte interativa ou histórias digitais. De natureza inerentemente faça-você-mesmo (DIY, *do-it-yourself*) sítios Web como o Newgrounds, o Planet Kodu, o Scratch Online e o Looking Glass (entre outros) encorajam os jovens a programar, não tanto como uma disciplina que se estuda, mas como oportunidades para criar e partilhar online. Nesta cultura faça-você-mesmo de iniciativa individual combinada com contributos do grupo e colaboração, vemos três tendências-chave na forma como os jovens aprendem agora a programar computadores:

1.ª: Olhar mais as aplicações e menos para o código

Em vez de fazer exercícios de escrita de código para aprender os algoritmos e as estruturas de dados, as crianças agora aprendem a programar criando aplicações específicas, que podem ser até videojogos ou histórias interativas. São cativadas pelo

potencial de criar algo real, tangível, que pode ser partilhado com outras pessoas. Isto transforma a aprendizagem da programação (pelo menos, inicialmente) que deixa de ser o estudo de uma disciplina abstrata para passar a ser uma forma de fazer e estar digitalmente no mundo.

2.ª: Olhar mais as comunidades e menos para as ferramentas

Felizmente, a última década testemunhou o desenvolvimento de muitas e admiráveis linguagens de programação introdutórias, que tornaram a produção de código um processo mais intuitivo e pessoal. O Scratch (http://scratch.mit.edu) e o Alice (http://alice.org) são os dois exemplos principais. Mas os programadores estão a constatar que as ferramentas não bastam. Cada ferramenta requer um público e a oportunidade de juntar criadores que partilhem os mesmos interesses, através da Internet. Consequentemente, as ferramentas como o Scratch e o Alice têm agora comunidades online vastas, com milhões de jovens utilizadores. A última versão do Scratch (na passada primavera foi lançada a versão 2.0) até funciona completamente online, para que as crianças possam programar e partilhar a partir de um único sítio Web, realçando tacitamente o facto de que a comunidade de práticas se tornou, com efeito, na ferramenta-chave para aprender a codificar.

3.ª: Olhar mais para a criação por remistura e menos para a criação a partir do zero

A programação deixou de ser uma atividade individual, na qual o código-fonte é ocultado e protegido ciosamente. No espírito do movimento do código-fonte aberto, há uma tendência acelerada para cada um partilhar o seu código e encorajar os participantes a experimentar as criações uns dos outros, só pelo facto de as poderem ajustar e expandir. Com a ideia de que tal abertura amplia o potencial de inovação, os jovens utilizadores abraçam a experimentação e a partilha mais livremente, desafiando o paradigma tradicional, do topo para a base, característico da informática e das escolas em geral.

Falando genericamente, vemos estas três tendências como uma mudança social, uma deslocação de uma visão predominantemente individualística da tecnologia para uma com um foco maior nas dimensões sociológicas e culturais subjacentes à aprendizagem da programação, conceptualizando de forma nova o pensamento computacional como participação computacional.

Chegar à participação computacional

Já a questão de quem é que está realmente a participar computacionalmente, é outra história. As três tendências acima descritas estão a ocorrer fundamentalmente fora das escolas formais. Nas escolas, o ensino da informática continua, resoluto, do topo para a base, centrado em conferir princípios abstratos antes de poder ocorrer qualquer aplicação direta. Em disciplinas ao nível do ensino secundário que sejam dedicadas a temas como Princípio das Ciências da Computação, é compreensível que se inicie pela abstração, dada a diversidade de programa a cobrir. Mas a quase total ausência de

trabalhos com participação computacional nas várias disciplinas anteriores, orientadas para a iniciação tecnológica, faz com que poucos alunos sequer ponderem a possibilidade se estudar os princípios das ciências da computação, muito menos que venham a encontrá-las no seu ensino formal.

Os números comprovam-no. Só 2.100 de entre 42.000 escolas secundárias dos Estados Unidos da América escolhem uma disciplina vocacional (AP, *Advanced Placement*) de informática (College Board, 2012). O número de disciplinas de introdução à informática diminuiu 17% desde 2005. Esta queda é, literalmente, indesculpável, dado que a direção-geral norte-americana de estatísticas do trabalho (Bureau of Labor Statistics, 2012) refere sistematicamente as profissões ligadas à informática como aquelas que apresentam maior crescimento no país, antecipando-se que sejam necessários 4 milhões de novos profissionais até 2020. A disparidade de género e étnico-racial na informática é também um obstáculo significativo. As mulheres representam 56% de todos os alunos que realizam provas de disciplinas vocacionais, mas eram só 21% dos alunos que realizaram o exame das disciplinas vocacionais de informática em 2011. E só 29 dos alunos que efetuaram esse exame eram negros (menos de 1% do total).

Nos anos 1990 e até no início dos anos 2000, estas desigualdades de longa data eram atribuídas, em geral, à infoexclusão. Muito do esforço dos anos 1990 se centrou em proporcionar acesso aos computadores, não nos currículos nem na pedagogia, com iniciativas como o Net Day, dedicadas inteiramente a colocar computadores nas escolas e a ligá-los à Internet. Embora tais iniciativas tenham contribuído para atacar a questão do acesso, permaneceu aquilo que o académico da área do multimédia, Henry Jenkins (2006) chamou a "falha de participação", relativamente ao uso dos meios digitais por parte das crianças, de formas criativas e refletidas.

Desmontando a noção de que o mero acesso poderia resolver o problema da equidade, os educadores e tecnólogos Mark Warschauer e Tina Matuchniak (2010) compararam duas escolas, que tinham o mesmo número de computadores mas estavam situadas em áreas muito diferentes, em termos socioeconómicos. Embora os alunos tivessem o mesmo acesso a computadores na duas escolas, aquilo que era ensino e aquilo que os alunos aprendiam na escola era muito diferente. Os alunos na área com estatuto socioeconómico mais elevado aprenderam a trabalhar de formas criativas e colaborativas usando os computadores, por vezes até programando. Os alunos da comunidade com menos rendimentos eram orientados para o processamento de texto e para a aprendizagem sobre a mera utilização técnico-funcional da máquina.

A participação na computação não é apenas ter acesso, mas também ter currículos e pedagogia de qualidade. Tal qualidade pode ocorrer quando a programação de computadores permite que as crianças produzam colaborem e adaptem conteúdo que seja significativo a nível pessoal. O foco no pensamento computacional iria resolver a falta de currículos cativantes nas disciplinas de tecnologia do ensino pré-escolar, básico e secundário, além de ensinar às crianças conceitos e competências para resolver problemas de forma algorítmica. Por outro lado, a participação computacional centra-se nas práticas e perspetivas pedagógicas necessárias para contribuir, de forma significativa, para redes sociais mais vastas, incluindo o ensino formal (mas não só). É aqui, na rede mais ampla dos pensadores criativos e críticos, que os educadores podem firmar novas normas académicas e sociais sobre aquilo que significa utilizar a tecnologia com significado.

Conclusão

A aprendizagem com tecnologia já deixou para trás a questão do acesso, avançando para a questão daquilo que cada um faz e daquilo que cada um partilha com os computadores. Passar da infoexclusão para a falha de participação passou a ser a força motriz em direção ao que chamamos participação computacional. Evidentemente, incorporar a participação computacional não será um passo pequeno para as escolas, onde o desempenho individual pesa muito mais do que as dinâmicas de grupo ou o esforço colaborativo, em termos de resultados académicos. A Web 2.0 ensinou-nos a importância da colaboração para facultar soluções mais criativas e baratas para os problemas. O acesso à participação e à colaboração em comunidades de programação são aspetos-chave para aprender conceitos e práticas fundamentais. Aprender a programar é também aprender a participar nos muitos públicos digitais – e vice-versa.

Embora só alguns de nós venham a ser informáticos profissionais, que virão a escrever o código e a conceber os sistemas que sustentarão grande parte da nossa vida quotidiana, da nossa aprendizagem e do nosso tempo livre, muitos se irão deparar com a necessidade de realizar algum tipo de programação, nalgum momento das suas vidas. Todos nós somos e continuaremos a ser utilizadores de tecnologias digitais, pelo que precisaremos por vezes de examinar de forma crítica e construtiva conceções e decisões que estiveram por trás da sua criação. Em termos da magnitude do que qualquer tipo de literacia confere ao indivíduo, Paulo Freire propôs que “ler a palavra é ler o mundo”. Consideramos que ler o código é certamente ler o mundo de hoje em termos da sua compreensão e em termos de ter a possibilidade de o refazer. A escolas, os seus líderes, os professores e os alunos desempenham um papel crucial na concretização desta oportunidade.

Notas finais

A rubrica **R&D** (dedicada à investigação e desenvolvimento) surge em cada fascículo da Kappan com o apoio da **Dean’s Alliance**, composta pelos diretores e presidentes das escolas e faculdades de ciências da educação das seguintes universidades: Universidade de Harvard, Universidade Estatal do Michigão, Universidade do Noroeste dos EUA, Universidade de Stanford, Faculdade de Professores da Universidade de Columbia, Universidade da Califórnia (Berkeley), Universidade da Califórnia (Los Angeles), Universidade do Michigão, Universidade da Pensilvânia e Universidade do Wisconsin.

Referências

- Bureau of Labor Statistics. (2012). Employment projections 2010-20. www.bls.gov/emp/
- CollegeBoard. (2012). AP course audit. https://apcourseaudit.epiconline.org/ledger/search.php
- Collins A., Halverson R. (2009). Rethinking education in the age of technology. Nova Iorque, NY, EUA: Teachers College Press.
- Grover S., Pea R. (2013). Computational thinking in K-12: A review of the state of the field. Educational Researcher, 42 (2), 59–69.

- Jenkins H., Clinton K., Purushotma R., Robison A.J., Weigel M. (2006). Confronting the challenges of participatory culture: Media education for the 21st century. Chicago, IL, EUA: MacArthur Foundation.
- Kafai Y.B., Peppler K.A. (2011). Youth, technology, and DIY: Developing participatory competencies in creative media production. Review of Research in Education, 35, 89–119.
- Warschauer M., Matuchniak T. (2010). New technology and digital worlds: Analyzing evidence of the equity in access, use and outcomes. Review of Research in Education, 34 (1), 179–225.
- Wing J.M. (2006). Computational thinking. Communications of the ACM, 49 (3), 33–35

Yasmin B. Kafai (kafai@upenn.edu) é professora de ciências da aprendizagem na Escola de Pós-Graduação em Educação da Universidade da Pensilvânia, em Filadélfia, estando também ligada ao Departamento de Ciências da Informação e da Computação. Quinn Burke (burkeqq@cofc.edu) é professor auxiliar de tecnologia educativa na Faculdade de Charleston e ex-professor do ensino secundário. Kafai e Burke são autores de um livro no prelo: Connected Code (MIT Press, 2014).