Scientia cum Industria

PLATINUM OPEN ACCESS

eISSN 2318-5279

Práticas Educacionais em Ciência, Engenharia e Matemática

Equações polinomiais do 2° grau e pensamento computacional: explorando um diálogo possível

Emanuel Orlandi
Laurete Zanol Sauer
Carine G. Webber *

Programa de Pós-Graduação em Ensino de Ciências e Matemática
Universidade de Caxias do Sul, Caxias do Sul, RS, Brasil
*Autor correspondente: cgwebber@ucs.br

Recebido: 19 de Outubro de 2023
Revisado: 11 de Dezembro de 2023
Aceito: 18 de Dezembro de 2023
Publicado: 29 de Dezembro de 2023

Resumo: Este trabalho apresenta uma proposta para o ensino de equações polinomiais do 2° grau com o desenvolvimento do pensamento computacional, conforme exigências dos documentos normativos educacionais nacionais. Diante disso, a linguagem de programação escolhida foi o Scratch e, por conta disso, apresenta-se a importância de pensadores como Seymour Papert para os avanços no ramo do pensamento computacional e na definição que atualmente é conhecida. Além disso, mostra-se a proposta de ensino detalhada e as considerações levantadas a partir da sua análise.

Keywords: equações polinomiais do 2° grau, pensamento computacional, linguagem de programação *Scratch*.

Educational Practices in Science, Engineering and Mathematics

Quadratic equations and computational thinking: exploring a possible dialogue

Abstract: This work presents a proposal for teaching 2nd grade polynomial equations with the development of computational thinking, in accordance with the requirements of national educational normative documents. Therefore, the programming language chosen was Scratch and, because of this, the importance of thinkers like Seymour Papert for advances in the field of computational thinking and in the definition that is currently known is presented. Furthermore, the detailed teaching proposal and the considerations raised from its analysis are shown.

Keywords: 2nd degree polynomial equations, computational thinking, Scratch programming language.

Introdução

Desde que foi introduzida em 2018, a Base Nacional Comum Curricular (BNCC) tem causado mudanças de grande relevância no contexto da educação no Brasil. Esse documento estabelece os conhecimentos, competências e habilidades que os alunos devem adquirir em cada fase da Educação Básica no país. Em conformidade com essa diretriz, a incorporação de tecnologias na sala de aula tem se tornado uma prática cada vez mais frequente, visando enriquecer os processos de ensino e aprendizagem, tornando-os mais dinâmicos e interativos. Essas alterações têm como meta impactar positivamente a qualidade da educação oferecida no Brasil, visando proporcionar uma formação mais abrangente e alinhada com as necessidades do mundo contemporâneo. Nesse contexto, a BNCC foi concebida com o propósito de ser "um documento completo e contemporâneo, que corresponde às demandas do estudante desta época, preparando-o para o futuro" [1].

Uma das exigências da BNCC quanto às habilidades relacionadas ao 9° ano do Ensino Fundamental é a aprendizagem de equações polinomiais do 2° grau. Além

disso, ela sugere o desenvolvimento do pensamento computacional pelos estudantes ao longo da sua formação básica. Diante disso, a proposta desse trabalho une as duas exigências contidas no documento. A BNCC contempla, ainda, o uso de Tecnologias Digitais da Informação e Comunicação (TDICs) nas aulas.

Tais recomendações do documento se devem ao fato de que os estudantes vivem em uma realidade em que o acesso à informação é quase que instantâneo. Nesta perspectiva, para o educador e pesquisador Marc Prensky [2], esses jovens estão acostumados a recorrer primeiramente a fontes digitais e à Web antes de procurarem em livros ou na mídia impressa. Por conta desses comportamentos e atitudes, e por entender a tecnologia digital como uma linguagem, Prensky [2] os descreve como nativos digitais, uma vez que "falam" a linguagem digital desde que nasceram. Pensando nisso, a proposta de ensino foi organizada de modo que os estudantes possam desenvolver o pensamento computacional a partir do Scratch após o estudo dos conceitos relacionados às equações do 2° grau.

Fundamentação Teórica

Construcionismo de Papert

Seymour Papert foi um dos teóricos mais relevantes para a educação no século XX. Nascido na África do Sul, tornou-se um matemático e posteriormente professor em uma das mais renomadas universidades dos Estados Unidos, o Massachusetts Institute of Technology (MIT). Em 1980, publicou a sua primeira versão do livro "A máquina das crianças", no qual o conceito do construcionismo foi apresentado. No prefácio à edição revisada brasileira desta obra, Papert [3] já deixa claro o seu entendimento a respeito da importância que o computador tem no contexto educacional, por seu potencial no desenvolvimento da autonomia intelectual do estudante.

Nesse sentido, o educando pode se tornar menos dependente de adultos provedores de informação, o que vai ao encontro das ideias de Freire [4] em sua visão libertadora de ensino. Consoante a isso, Papert [3] afirma que existem áreas do conhecimento em que a transição de estilos orais para estilos letrados é muito difícil para os estudantes, como a Matemática por exemplo. Porém, a máquina (computador) tem o poder de fornecer um contexto e suavizar esse processo. Para compreender o construcionismo podemos fazer a seguinte analogia:

> [...] as crianças farão melhor descobrindo ('pescando') por si mesmas o conhecimento específico de que precisam; a educação organizada ou informal poderá ajudar mais se certificar-se de que elas estarão sendo apoiadas moral, psicológica, material e intelectualmente em seus esforços. O tipo de conhecimento que as crianças mais precisam e o que as ajudará a obter mais conhecimento [3].

Pensamento Computacional

A partir da perspectiva do construcionismo conceituado por Papert ainda na década de 1970, Pei, Weintrop e Wilensky [5] destacam que Papert e seus colegas já imaginavam um sistema de educação que integraria o pensamento computacional na vida cotidiana. Mas antes disso se concretizar, a tecnologia da época não fornecia a experiência educacional presente nas escolas atualmente. Portanto, de acordo com Pei, Weintrop e Wilensky [5], colocando essa ideia em prática, Papert e seus colegas investigaram como os ambientes matemáticos construcionistas poderiam impulsionar essa integração e permitir que alunos de todos os níveis e diferentes interesses se envolvessem profundamente em atividades que são matemática e computacionalmente ricas.

Neste sentido, Wilensky apud Pei, Weintrop e Wilensky [5] destaca que as novas tecnologias expandem o conteúdo matemático para além dos limites estabelecidos pelo currículo escolar. Com isso, o estudante passa a tornar concretos os conceitos matemáticos abstratos e explorar áreas da matemática anteriormente inacessíveis, criando uma "nova matemática". Diante deste ponto de vista, e por entender a importância do computador e do construcionismo no meio educacional e no desenvolvimento do pensamento matemático, Papert (1980) apud Lodi & Martini [6] foi o primeiro a utilizar a expressão "pensamento computacional".

De acordo com Lodi & Martini [6], ao contrário do uso de Wing em 2006, a expressão utilizada por Papert em Mindstorms não é de forma alguma uma tentativa de definição. O que é central em Mindstorms não é "pensar" – é "construir", por meios computacionais, versões concretas de conceitos matemáticos abstratos (como o ensino de equações do 2° grau, por exemplo). O apelo do computador é que ele fornece uma referência concreta para os conceitos abstratos a serem compreendidos. Conforme Pei, Weintrop e Wilensky [6] a definição do pensamento computacional foi popularizada pelo artigo de Jeannette Wing intitulado Computational Thinking, onde ela argumentou: "à leitura, escrita e aritmética, devemos adicionar o pensamento computacional à toda habilidade analítica de cada criança".

O pensamento computacional é uma habilidade que permite ao estudante pensar em um problema e de que forma se pode resolver este problema, otimizando os processos. Além disso, de acordo com Kurshan apud Brackmann [7], essa habilidade pode ser definida como:

> [...] uma distinta capacidade criativa, crítica e estratégica humana de saber utilizar os fundamentos da Computação, nas mais diversas áreas do conhecimento, com a finalidade de identificar e resolver problemas, de maneira individual ou colaborativa, através de passos claros, de tal forma que uma pessoa ou uma máquina possam executá-los eficazmente.

Mas afinal, como desenvolver o pensamento computacional? Uma das possibilidades é utilizar a linguagem Scratch, a qual foi criada por uma equipe pertencente ao *Media Lab* do (MIT) em *Boston*, a *Lifelong Kindergarten Group*. A plataforma permite criações utilizando programação por blocos. As opções de uso são variadas, incluindo animações, jogos, apresentações e outras aplicações.

Ensinar pensamento computacional com Scratch leva a colocar algoritmos e programação no centro da reflexão, em uma abordagem tradicional ou orientada para a "computação criativa" [8]. Ainda conforme Tchounikine [8], a abordagem do pensamento computacional pode ser dividida nas seguintes categorias:

- apreender um problema e sua solução em diferentes níveis (abstração);
- pensar nas tarefas a serem executadas como uma série de etapas (algoritmos);
- entender que para resolver um problema complexo ele deve ser decomposto em vários problemas simples (decomposição);
- entender que um novo problema provavelmente está relacionado a outros problemas já resolvidos pelo aluno (reconhecimento de padrões); e
- perceber que a solução para um problema pode ser usada para resolver toda uma gama de problemas semelhantes (generalização).
-

Na mesma perspectiva de Tchounikine [8], Brackmann [7] categoriza o ensino do pensamento computacional da maneira ilustrada na figura 1. Tem-se como pilares do pensamento computacional: decomposição, reconhecimento de padrões, abstração e algoritmos.

.

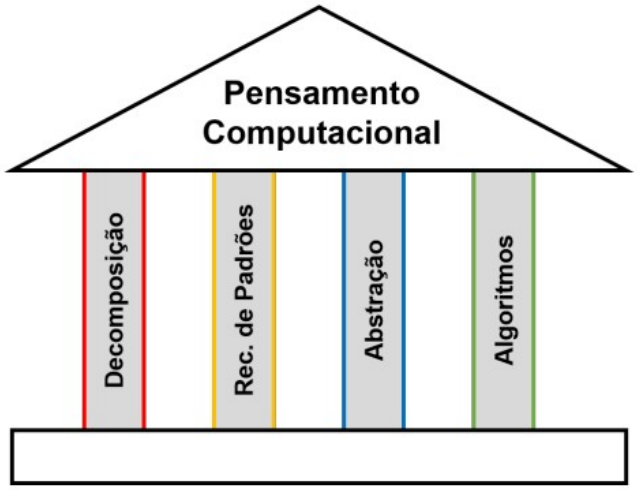

Figura 1. Os quatro pilares do pensamento computacional.

Diante das categorias citadas anteriormente, percebe-se que o pensamento computacional pode ser uma estratégia aliada ao ensino de matemática, a fim de auxiliar na compreensão e resolução de problemas complexos.

Metodologia e Desenvolvimento

A metodologia utilizada é a apresentação de uma proposta de ação didática envolvendo o ensino de equações polinomiais do 2° grau e do pensamento computacional, sendo o desenvolvimento organizado da seguinte maneira:

DADOS DE IDENTIFICAÇÃO
Componente curricular: Matemática
Turma: 9° ano – Ensino Fundamental
Tema: Equações polinomiais do 2° grau
Resultados de aprendizagem esperados

- Identificar os principais blocos programáveis do Scratch.
- Construir algoritmos no caderno como esboço para serem desenvolvidos no Scratch.
- Analisar e compreender rotinas já construídas e a lógica utilizada na sua construção.
- Desenvolver uma rotina no Scratch que forneça ao usuário a solução de uma equação do 2° grau conforme os coeficientes informados.

A realização da proposta se dará por meio de oito encontros, totalizando 700 minutos.

1° encontro (2h.a.)
Visto que para a aplicação desta proposta de ensino os estudantes já devem ter estudado todos os conceitos relacionados as equações do 2° grau, iniciaremos este encontro com uma sondagem dos conhecimentos prévios relacionados ao assunto e que serão necessários para o desenvolvimento da proposta. A partir desta sondagem serão observadas as eventuais lacunas de aprendizado que possam dificultar o desenvolvimento da proposta. Diante disso, poderão ser retomados alguns conceitos.

https://pdf.ac/Zrik8

2° encontro (2h.a.)

Após a sondagem inicial dos conhecimentos prévios, será realizada uma breve discussão sobre o conceito de pensamento computacional e sobre como surgiu o Scratch. O texto a seguir será fornecido ao estudante e deve ser colado em seu caderno.

O que é pensamento computacional e como surgiu o *Scratch*?

O pensamento computacional é uma habilidade fundamental no mundo atual, que envolve a capacidade de resolver problemas de forma lógica e estruturada, de maneira semelhante à forma como os computadores processam informações. Ele consiste em decompor problemas complexos em partes menores, identificar padrões, desenvolver algoritmos e testar soluções. O pensamento computacional não se limita apenas à programação, mas também é aplicável em diversas áreas, como matemática, ciência, engenharia e até mesmo na vida cotidiana. Ao adquirir essa habilidade, você estará mais preparado para enfrentar desafios e tomar decisões informadas em um mundo cada vez mais digital e tecnológico.

Mas afinal, como desenvolver o pensamento computacional? Uma das possibilidades é utilizar a linguagem Scratch, a qual foi criada por uma equipe pertencente ao *Media Lab* do *Massachusetts Institute of Technology* (MIT) em *Boston*, a *Lifelong Kindergarten Group*. A plataforma permite criações utilizando programação por blocos. As opções de uso são variadas, incluindo animações, jogos, apresentações e outras aplicações.

Bom, mas não estamos aqui para estudar toda a história do pensamento computacional e da criação do Scratch não é mesmo?

Explorando o *Scratch*

Neste momento começaremos o processo de construção da rotina que fornecerá ao usuário as raízes/soluções de uma equação do 2° grau. Para que isso ocorra, é preciso que os estudantes estejam ambientados com o Scratch. Neste sentido, utilizaremos a estratégia *Think-Pair-Share* para exploração dos blocos programáveis da plataforma.

A Etapa 1 (Pense) da TPS será para uma análise individual e rápida do visual do Scratch, mais focada na área destacada na Figura 2 (2 minutos); na Etapa 2 (Discuta com um par) os estudantes formarão duplas para discutirem as suas observações e elencar os principais pontos; já na Etapa 3 (Compartilhe com o grande grupo) os pares apresentarão as suas perspectivas iniciais sobre o ambiente do Scratch e escreverão uma característica no *Mentimeter* que será projetado no quadro. (link: https://www.menti.com/al2fuvocziz3)

Construída a nuvem de palavras (gerada automaticamente pelo *Mentimeter* após a inserção das palavras) com as características elencadas pelos estudantes, discutiremos as categorias dos blocos programáveis e as suas funcionalidades. A partir dessa análise, os estudantes receberão um resumo com as características de cada categoria.

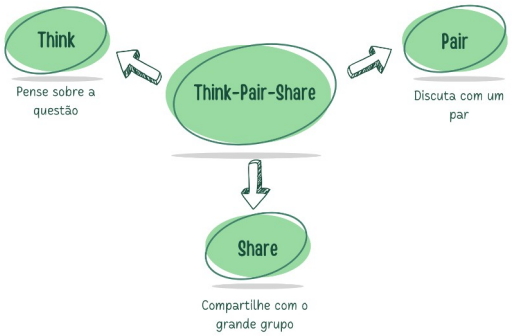

Figura 2. Esquema de etapas da TPS adaptado de [9]

3° encontro (1h.a.)

Exemplos de rotinas

Para os estudantes começarem a entender como funciona o processo de construção de um algoritmo iremos construir, conjuntamente, uma rotina que fornece ao usuário a informação se um número é par ou ímpar.

Para isso, precisamos de um esboço. Portanto, revisaremos qual é o conceito de paridade. Se um número ao ser dividido por 2 resulta em resto 0, é par! Se o resto for 1, é ímpar! Feito isso, já podemos começar a estruturar uma rotina no quadro.

- Diga um número inteiro.
- Se a divisão deste número por 2 resultar em resto 0, então o número é par.
- Senão o número é ímpar.

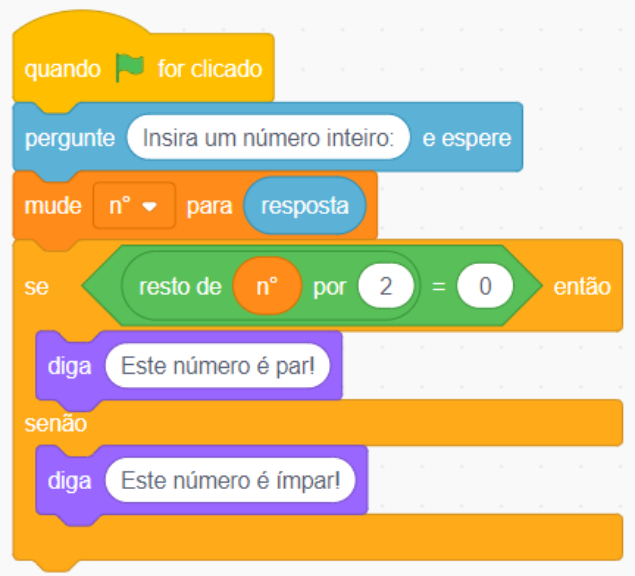

Figura 3. Visão da construção da rotina

Para construção desta rotina é preciso começar com um bloco de eventos. A partir disso, iniciamos perguntando ao usuário um número inteiro e armazenamos o valor inserido na resposta como uma variável n°. Em seguida, analisamos o resto da divisão da variável n° (resposta) por 2 e retornamos ao usuário: se o resto da divisão por 2 for igual a 0, diga "Este número é par!", senão (quando o resto da divisão por 2 resultar em 1), diga "Este número é ímpar!".

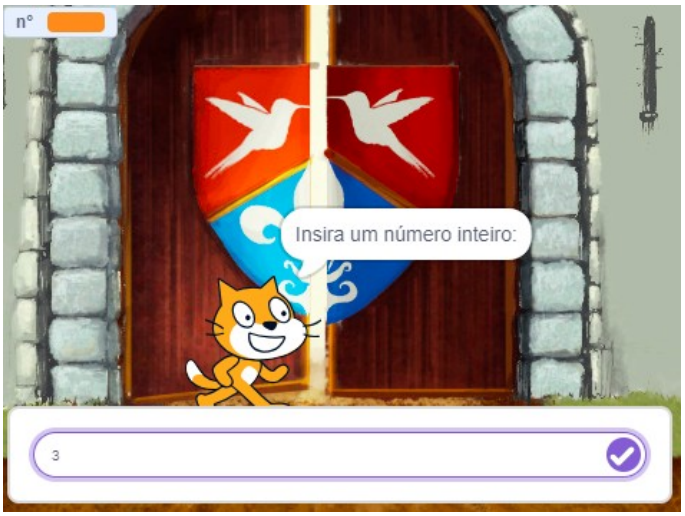

Figura 4. Visão inicial do usuário ao utilizar a rotina.

Figura 5. Visão final do usuário ao utilizar a rotina.

Começaremos o desenvolvimento do pensamento computacional a partir da análise e criação de rotinas simples, até chegar no produto final do projeto. Essa rotina inicial será muito importante para a discussão a respeito do armazenamento de valores em variáveis como a n° e dos blocos condicionais como apresentada no exemplo da paridade.

4° encontro (2h.a.)

Os estudantes irão analisar a rotina apresentada na Figura 6 e com auxílio do professor, reconstruí-la. Essa rotina fornece o resultado de uma equação do 1° grau ao usuário dados os seus coeficientes. Após a análise da rotina, os estudantes poderão ter ideias de melhoria na programação em blocos ou no cenário. (Link: https://scratch.mit.edu/projects/897404025/editor.)

Figura 6. Rotina da equação do 1° grau.

A partir dessa rotina aprofundaremos a discussão a respeito da criação de variáveis e o seu uso para resolução dos cálculos algébricos. Além disso, os estudantes deverão começar a analisar uma rotina que calcula o discriminante de uma equação do 2° grau conforme os coeficientes inseridos e retorna ao usuário o número de raízes. Link da rotina autoral: https://scratch.mit.edu/projects/898578659/.

Antes de iniciarmos a análise e construção da rotina, é preciso relembrar a Lei Geral da Equação do 2° grau:

$ax^2 + bx + c = 0$

Agora você deve imaginar que para construir uma rotina que forneça ao usuário o valor do discriminante é preciso que ele informe os seus coeficientes. Em seguida, é preciso realizar o cálculo do do discriminante, também conhecido como delta. E, por fim, analisar o resultado e retornar ao usuário quantas raízes possui a sua equação. Portanto, podemos esboçar a rotina da seguinte maneira:

- 1° passo: solicitar ao usuário que informe os coeficientes a, b e c.
- 2° passo: armazenar a resposta de cada coeficiente em uma variável com o respectivo nome.
- 3° passo: criar uma nova variável que calcule o valor do discriminante a partir dos coeficientes informados e armazenados nas variáveis.
- 4° passo: a partir da análise dos três casos possíveis de discriminante, retornar ao usuário o número de raízes da sua equação.

A partir dessa discussão os estudantes irão analisar a rotina modelo (link anterior) e construir a sua própria rotina seguindo os passos estabelecidos anteriormente no esboço.

5° encontro (2h.a.)
Continuação da construção da rotina do cálculo do discriminante e auxílio aos estudantes.

6° encontro (1h.a.)
Este encontro será destinado para a discussão da proposta de trabalho a ser realizada pelos estudantes e os critérios de avaliação. A avaliação será feita pelos pares e pelo professor, conforme Quadro 1.

Grupo avaliado:	Assinale com "x" em cada avaliação		
Critérios	C5	C3	C1
1. O trabalho foi apresentado com clareza?			
2. O enredo criado para que o usuário forneça os coeficientes da equação é criativo?			
3. O cenário criado e o personagem são atrativos?			
4. Foi possível perceber que todos os integrantes do grupo colaboraram para a elaboração do trabalho?			
5. A rotina apresentada está correta?			
Legenda de conceitos: C5: Atingido com sucesso. C3: Atingido parcialmente. C1: Não atingido.	**Observações:**		

Quadro 1. Critérios da avaliação por pares.

Diante disso, os estudantes serão organizados em duplas para que iniciem a discussão a respeito de como desenvolverão o trabalho e começarem a esboçar a rotina em seus cadernos.

7° encontro (2h.a.)
Continuação das rotinas e auxílio aos estudantes.

8° encontro (2h.a.)
Ajustes finais nas rotinas e apresentações. Durante a apresentação dos trabalhos as demais duplas estarão avaliando os seus colegas, assim como o professor. Portanto, a avaliação somativa será uma média das avaliações pelos pares.

Um exemplo de um rotina possível pode ser acessada pelo *QR Code* apresentado a seguir.

https://scratch.mit.edu/projects/883730668/editor/.

Considerações Finais

O pensamento computacional é uma habilidade essencial para os estudantes, permitindo que eles enfrentem problemas de forma sistemática e otimizada, utilizando conceitos e práticas da computação em diversas áreas do conhecimento. A abordagem do pensamento computacional, aplicada por meio da linguagem Scratch, possibilita a divisão de problemas em etapas e a resolução de desafios complexos por meio de algoritmos. Ao incorporar princípios como abstração, decomposição, reconhecimento de padrões e generalização, os estudantes são capacitados a desenvolver uma abordagem criativa, crítica e estratégica para a solução de problemas, seja de maneira individual ou colaborativa. O ensino do pensamento computacional com o uso do Scratch oferece oportunidades para a interação prática com conceitos matemáticos complexos e promove a reflexão ativa sobre os processos de aprendizado.

Assim, essa habilidade mostra-se uma poderosa ferramenta no ensino de matemática, proporcionando aos alunos uma forma inovadora de explorar e compreender os desafios matemáticos. Ao desenvolver a capacidade de pensar computacionalmente, os estudantes adquirem uma valiosa habilidade para resolver problemas eficazmente e enfrentar os desafios do mundo contemporâneo. Assim sendo, pretende-se desenvolver tais habilidades com esta proposta de ensino.

Agradecimentos

Os autores agradecem os organizadores do XI SECIMSEG pelo espaço de discussão e reflexão voltados ao Ensino e à Educação e aos revisores pelas sugestões e recomendações para o aprimoramento na redação do artigo.

Referências

[1] Brasil, Ministério da Educação, Base Nacional Comum Curricular. Brasília, 2018.

[2] M. Prensky, Digital Natives Digital Immigrants. In: Prensky, M. On the Horizon. NCB University Press, v. 9 n. 5, October 2001.

[3] S. Papert, A máquina das crianças: repensando a escola na era da informática. Porto Alegre, RS: Artes Médicas, 2008.

[4] P. Freire, Pedagogia do Oprimido. Rio de Janeiro: Paz e Terra, 2018.

[5] C. (Yu) Pei, D. Weintrop, U. Wilensky, Cultivating Computational Thinking Practices and Mathematical Habits of Mind in Lattice Land. Mathematical Thinking And Learning, [S.L.], v. 20, n. 1, p. 75-89, 2 jan. 2018. Informa UK Limited.

[6] M. Lodi, S. Martini, Computational Thinking, Between Papert and Wing. Science & Education, [S.L.], v. 30, n. 4, p. 883-908, 28 abr. 2021. Springer Science and Business Media LLC. http://dx.doi.org/10.1007/s11191-021-00202-5.

[7] C. P. Brackmann, Desenvolvimento do pensamento computacional através de atividades desplugadas na educação básica. 2017. 226 f. Tese (Doutorado) - Curso de Pós Graduação em Informática na Educação, Centro de Estudos Interdisciplinares em Novas Tecnologias na Educação, Universidade Federal do Rio Grande do Sul, Porto Alegre, 2017.

[8] P. Tchounikine, Initier les élèves à la pensée informatique et à la programmation avec Scratch. [S. l.], v. 2, p. 1-36, 2017.

[9] G. Elmôr-Filho, L. Z. Sauer, N. N. Almeida, V. Villasboas, Uma Nova Sala de Aula é Possível: Aprendizagem Ativa na Educação em Engenharia. - 1.ed. – Rio de Janeiro: LTC, 2019.