Sociedade Brasileira de Educação Matemática

Desenvolvimento do Pensamento Computacional: estrutura lógica de repetição com o uso da programação em blocos

Paulo Roberto Anastácio[1]

Rudolph dos Santos Gomes Pereira[2]

Willian Damin[3]

Resumo: Este artigo investiga o desenvolvimento do pensamento computacional por meio da utilização do Scratch como recurso didático no ensino e na aprendizagem de programação, com ênfase na compreensão e aplicação de estruturas lógicas de repetição. A pesquisa, de abordagem qualitativa, fundamenta-se na análise de atividades práticas realizadas por alunos universitários da disciplina de Programação. Os resultados evidenciam que os participantes foram capazes de identificar padrões de repetições e elaborar soluções mais eficientes por meio do uso adequado de laços de repetição, contribuindo para a clareza, a organização e funcionalidade dos algoritmos desenvolvidos. Além disso, foram observadas as habilidades relacionadas à abstração e à decomposição de problemas, consideradas pilares do pensamento computacional. Conclui-se que a integração do Scratch em atividades de ensino de programação favorece o desenvolvimento do pensamento computacional e contribui para a aprendizagem dos alunos, possibilitando diferentes níveis de apropriação conceitual dos conteúdos abordados.

Palavras-chave: Pensamento Computacional. Programação em Blocos. Scratch. Estrutura de Repetição. Ensino de Programação.

Developing Computational Thinking: Logical Repetition Structures through Block-Based Programming

Abstract: This article investigates the development of computational thinking through the use of Scratch as a teaching resource in programming education and learning, with an emphasis on the understanding and application of logical repetition structures. The qualitative research is based on the analysis of practical activities carried out by university students in the Programming course. The results show that participants were able to identify repetition patterns and develop more efficient solutions through the appropriate use of repetition loops, contributing to the clarity, organization, and functionality of the developed algorithms. Furthermore, skills related to abstraction and problem decomposition, considered pillars of computational thinking, were observed. It is concluded that the integration of Scratch into programming teaching activities favors the development of computational thinking and contributes to student learning, enabling different levels of conceptual appropriation of the content covered.

Keywords: Computational Thinking. Block-Based Programming. Scratch. Repetition Structure. Programming Education.

Desarrollo del Pensamiento Computacional: estructura lógica de repetición mediante el uso de la programación por bloques

Resumen: Este artículo investiga el desarrollo del pensamiento computacional mediante el uso de Scratch como recurso didáctico en la enseñanza y el aprendizaje de la programación, con énfasis en la comprensión y aplicación de estructuras de repetición lógica. La investigación cualitativa se basa

[1] Mestrado em Ensino. Universidade Estadual do Norte do Paraná/UENP, Bandeirantes, PR, Brasil. E-mail: paulo.anastacio@uenp.edu.br. Orcid: https://orcid.org/0009-0004-0326-9731.

[2] Doutorado em Educação. Universidade Estadual do Norte do Paraná/UENP, Cornélio Procópio, PR, Brasil. E-mail: rudolphsantos@uenp.edu.br. Orcid: https://orcid.org/0000-0003-0504-7329.

[3] Doutorado em Ensino de Ciência e Tecnologia. Universidade Estadual do Norte do Paraná/UENP, Cornélio Procópio, PR, Brasil. E-mail: wdamin@uenp.edu.br. Orcid: https://orcid.org/0000-0002-6795-9772.

eISSN 2317-904X DOI: 10.37001/emr.v31i91.4551 Recebido • 13/04/2025 | Aprovado • 16/06/2026 | Publicado • 30/06/2026

en el análisis de actividades prácticas realizadas por estudiantes universitarios en el curso de Programación. Los resultados muestran que los participantes fueron capaces de identificar patrones de repetición y desarrollar soluciones más eficientes mediante el uso adecuado de bucles de repetición, lo que contribuyó a la claridad, organización y funcionalidad de los algoritmos desarrollados. Además, se observaron habilidades relacionadas con la abstracción y la descomposición de problemas, consideradas pilares del pensamiento computacional. Se concluye que la integración de Scratch en las actividades de enseñanza de la programación favorece el desarrollo del pensamiento computacional y contribuye al aprendizaje de los estudiantes, posibilitando diferentes niveles de apropiación conceptual del contenido abordado.

Palabras clave: Pensamiento Computacional. Programación por Bloques. Scratch. Estructura de Repetición. Enseñanza de Programación.

1 Introdução

O avanço contínuo das tecnologias transformou a forma como as informações são processadas e utilizadas em diversos setores da sociedade, incluindo o campo educacional. No ensino de programação, esse progresso impulsionou o desenvolvimento de ferramentas pedagógicas que visam tornar o aprendizado mais acessível e significativo, sobretudo para iniciantes. Entre essas ferramentas destaca-se o Scratch, uma linguagem de programação visual baseada em blocos, projetada para facilitar o ensino de lógica computacional de forma intuitiva e criativa (RODRIGUES, 2015).

Estudos demonstram que o uso do Scratch em ambientes educacionais contribui para o desenvolvimento do raciocínio lógico, da criatividade e da capacidade de resolução de problemas, ao mesmo tempo em que proporciona aos alunos uma experiência prática e lúdica no processo de aprendizagem (SOBREIRA *et al*., 2013; MALONEY, 2010). A ferramenta permite que os alunos criem programas sem a necessidade de memorizar comandos complexos de linguagens tradicionais, promovendo a aprendizagem por meio da experimentação e da construção ativa de conhecimento.

Nesse contexto, Zaharija (2013) destaca que o Scratch favorece o desenvolvimento de habilidades cognitivas e sociais ao estimular uma forma de pensar criativa, sistemática e colaborativa. Sua interface amigável e o sistema de blocos de comandos — organizados por categorias como movimento, controle e repetição — possibilitam a construção de algoritmos arrastando fragmentos de código coloridos e estruturados, sem a exigência de digitação ou memorização de sintaxe. Isso torna o processo de programação mais acessível e menos intimidador para alunos em fase inicial de contato com conceitos computacionais.

Apesar dos benefícios, o ensino de programação ainda enfrenta obstáculos significativos, como altos índices de reprovação e evasão nos cursos da área de Tecnologia da Informação (RAMOS *et al*., 2015). Essas dificuldades estão frequentemente relacionadas

à abstração dos conteúdos, à limitação de tempo em sala de aula e à baixa familiaridade dos alunos com o pensamento lógico-formal exigido pela disciplina (CALDEIRA; VILELA, 2016). Além disso, alunos e professores relatam dificuldades tanto na assimilação quanto na mediação dos conteúdos, indicando a necessidade de metodologias e recursos que tornem o processo de ensino e de aprendizagem mais eficaz (ROSA; GIRAFFA, 2011).

Diante desse cenário, o objetivo deste artigo é apresentar a integração da ferramenta Scratch como recurso didático no processo de ensino e de aprendizagem de programação, com ênfase na compreensão e aplicação de estruturas lógicas de repetição. A proposta busca demonstrar como essa abordagem pode contribuir para superar as dificuldades frequentemente encontradas no ensino introdutório de programação, promovendo o desenvolvimento do pensamento computacional por meio de uma linguagem visual acessível, interativa e significativa.

2 Pensamento Computacional

O termo Pensamento Computacional (PC) foi introduzido por Jeannette Wing (2006) e definido como uma habilidade cognitiva que permite formular problemas e desenvolver soluções de forma que um computador – ou um ser humano auxiliado por ferramentas computacionais – possa efetivamente executá-los. Trata-se de um processo mental que envolve a aplicação de conceitos fundamentais da ciência da computação para resolver problemas de maneira sistemática, eficiente e lógica, independentemente da necessidade de codificação ou de conhecimento prévio de linguagens de programação.

De acordo com Brackmann (2017), o Pensamento Computacional pode ser compreendido a partir de quatro pilares fundamentais que sustentam o desenvolvimento dessa habilidade cognitiva no contexto educacional. Esses pilares servem como base para a criação de estratégias didáticas que incentivem o raciocínio lógico, a resolução de problemas e a autonomia dos alunos. São eles:

1. Decomposição: Consiste na habilidade de dividir um problema complexo em partes menores e mais manejáveis, facilitando sua compreensão e resolução. Essa prática permite que o aluno organize suas ideias, identifique etapas e distribua tarefas de forma estruturada;

2. Reconhecimento de Padrões: Refere-se à capacidade de identificar similaridades, ou repetições em problemas distintos. Ao reconhecer padrões, o aluno pode prever comportamentos, reaproveitar soluções e construir estratégias mais rápidas e eficazes,

tornando o processo de resolução mais automatizado e lógico;

3. Abstração: Trata-se da habilidade de filtrar informações relevantes e ignorar os detalhes que não interferem diretamente na solução do problema. A abstração permite focar nos elementos essenciais, facilitando a modelagem de situações reais em representações computacionais simplificadas;

4. Algoritmos: Diz respeito à formulação de instruções ou passos organizados de forma lógica para resolver um problema. A construção de algoritmos desenvolve o pensamento sequencial e fortalece a compreensão de como estruturar soluções eficientes, replicáveis e testáveis. Esses pilares, quando trabalhados de forma integrada, contribuem para a formação de alunos mais autônomos, críticos e criativos.

A programação, por sua vez, constitui uma das formas mais diretas de desenvolver o pensamento computacional. Ao programar, o aluno é desafiado a organizar sequências lógicas, estruturar comandos, prever comportamentos e corrigir erros, o que implica um raciocínio estruturado e iterativo. Para Costella, Licks e Teixeira (2016), programar exige que o aluno pense de forma lógica e objetiva, subdividindo problemas complexos em partes menores para facilitar a compreensão e a resolução.

A utilização de linguagens visuais como o Scratch tem se mostrado uma alternativa eficaz para o ensino introdutório de programação, sobretudo por eliminar a barreira da sintaxe textual, tornando o processo mais acessível a iniciantes (MALONEY *et al*., 2010). A abordagem baseada em blocos coloridos e encaixáveis permite ao aluno compreender conceitos fundamentais como variáveis, condições, eventos e, especialmente, estruturas de repetição, que são essenciais para a construção de algoritmos eficientes.

Dessa forma, integrar a programação ao processo educacional não significa apenas ensinar códigos, mas sim desenvolver habilidades cognitivas superiores que incluem a capacidade de análise, planejamento e inovação, competências fundamentais para a formação de sujeitos críticos e criativos na era digital.

3 Encaminhamento metodológico

A pesquisa é de natureza qualitativa, com foco na compreensão e interpretação dos fenômenos observados durante o processo de ensino e de aprendizagem de programação. O estudo foi desenvolvido com seis alunos do primeiro ano do curso de Ciência da Computação de uma universidade pública do estado do Paraná.

Foi proposto um curso introdutório com atividades práticas, cuja estrutura curricular

contemplou os principais conceitos de lógica de programação. As estruturas de decisão e de repetição foram especialmente selecionadas por serem apontadas por Pereira *et al*. (2012) como fundamentais para o ensino de programação no nível superior.

A análise das produções dos alunos foi orientada com base nos pilares do Pensamento Computacional (decomposição, reconhecimento de padrões, abstração e algoritmos) com o objetivo de verificar se os participantes foram capazes de concluir todas as etapas das atividades propostas (RAABE *et al*., 2018). O curso teve uma carga horária total de 20 horas, distribuídas em dez encontros presenciais, com duração de duas horas cada, organizados em três módulos: I) lógica de programação; II) estrutura de decisão e; III) estrutura de repetição. Apresentamos aqui, a discussão dos resultados encontrados na atividade avaliativa referente ao módulo de estrutura de repetição.

Buscando avaliar a capacidade dos alunos de utilizarem estruturas de repetição, essa atividade foi desenvolvida com a intenção de que eles consigam criar uma estrutura lógica para selecionar o maior e o menor número dentre os números inseridos pelo usuário. Assim, é necessário que o aluno efetue cálculos, além de utilizar estruturas lógicas, estruturas de condição e estruturas de repetição.

Quadro 1 – Atividade avaliativa do módulo de estrutura de repetição

Desenvolva um algoritmo no qual o usuário digite inúmeros números inteiros e ao digitar -1, o programa exiba na tela qual foi o maior e o menor número digitado.

Fonte: os autores.

Para a análise dos resultados obtidos a partir das atividades práticas, adotou-se como base teórica os autores que compõem o referencial teórico desse artigo. A avaliação das produções dos alunos considerou, portanto, a presença e a aplicação dos elementos essenciais desse modelo: decomposição, reconhecimento de padrões, abstração e construção de algoritmos. Cada atividade foi examinada à luz desses pilares, com o intuito de identificar indícios de desenvolvimento do raciocínio lógico, da organização de ideias e da capacidade dos alunos de estruturar soluções de forma sequencial, eficiente e coerente. Essa abordagem permitiu não apenas verificar a assimilação de conteúdos relacionados à lógica de programação, como também analisar o avanço cognitivo dos participantes na formulação de estratégias para resolução de problemas computacionais.

Quadro 2 – Descrição do Pensamento Computacional

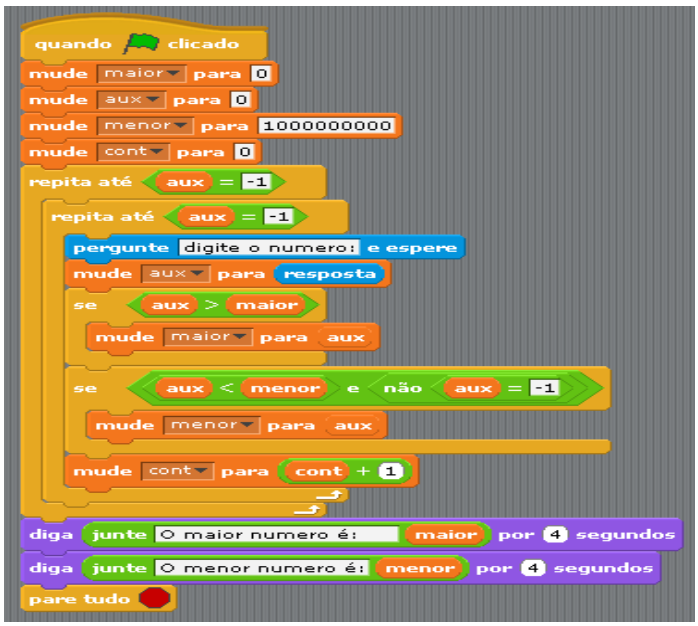
Decomposição - O que precisa ser feito para selecionar os números inseridos?
Nessa primeira parte espera-se que aluno divida o problema em partes menores; por exemplo, distinguir os números menores dos maiores e pensar como ele fará a seleção dos numerais.

Padrões - Quais são as semelhanças encontradas na atividade?
Momento em que os alunos farão os agrupamentos, no caso da inserção de números, quais os padrões que o aluno irá criar para armazenar tais informações, e como elas serão tratadas dentro do algoritmo, e qual o padrão utilizado para que o programa finalize a repetição e exiba os resultados.
Abstração - Quais as diferenças entre os algoritmos digitados?
Na terceira etapa o aluno distingue os algoritmos por meio de comparações com os números inseridos anteriormente e para que isso ocorra é necessário que ele desenvolva uma estrutura que envolva estruturas lógicas e de repetição.
Algoritmo - Como estruturar um código a fim de resolver essa atividade?
Nessa etapa o aluno deve organizar e estruturar o código responsável por efetuar os cálculos e comparações dos valores inseridos no programa e para que isso ocorra, o mesmo necessita utilizar e organizar estruturas lógicas por meio de algoritmos que seguem uma ordem lógica.

Fonte: os autores.

A seguir foi representado, por meio da Figura 1, uma alternativa para a resolução da atividade avaliativa.

Figura 1 – Resolução da Atividade



Fonte: os autores.

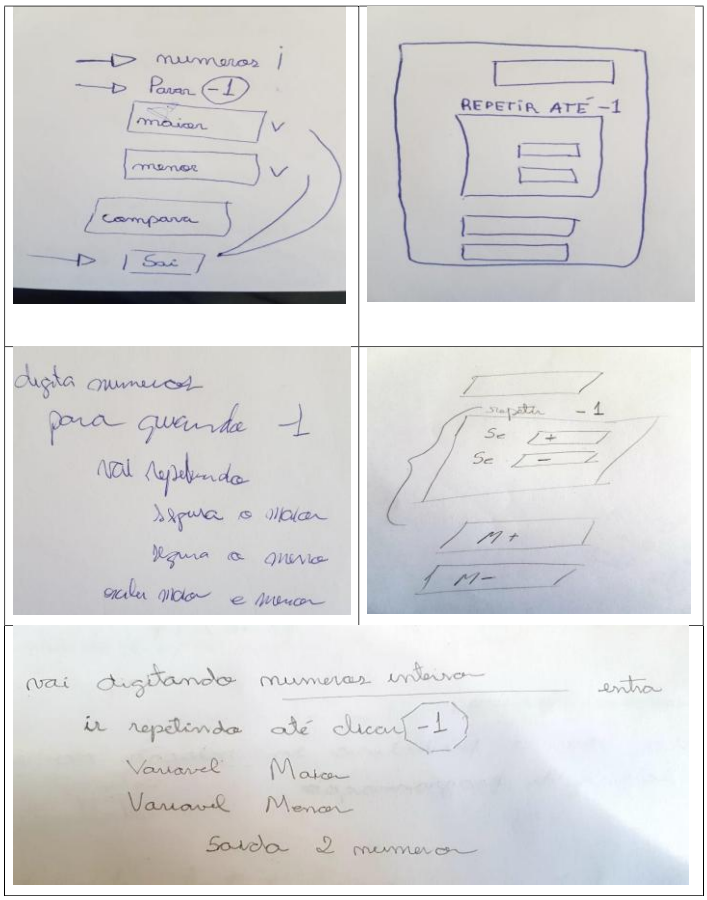
As atividades acima foram responsáveis por avaliar a utilização das estruturas lógicas necessárias para solucionar os problemas, com o intuito de observar a lógica que os alunos utilizaram para desenvolver os algoritmos e como organizam os dados.

4 Resultados e discussão

Nesta categoria analisamos os dados referentes às estruturas lógicas de repetição abordadas no Módulo III. A análise dos dados obtidos por meio da atividade avaliativa

evidenciou que alguns alunos fizeram uso de rascunhos para esboçar a decomposição do problema. Esses esquemas preliminares foram sistematizados e encontram-se representados no Quadro 3.

Quadro 3 – Esboço de A1, A2, A3, A4 e A5 da Atividade



Fonte: os autores.

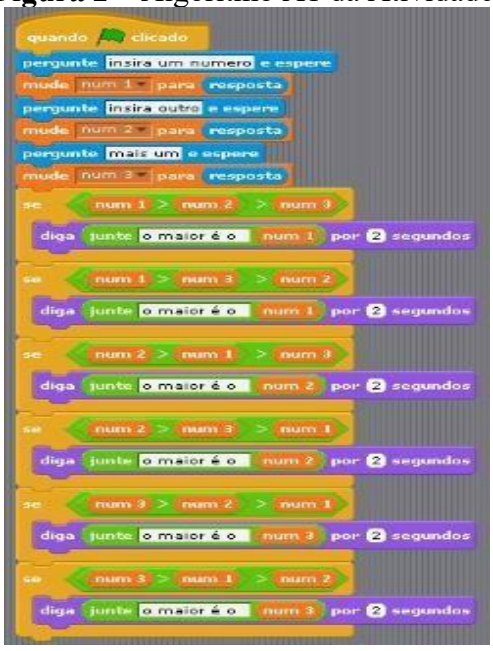
As imagens apresentadas no Quadro 3 indicam que os alunos, com exceção do aluno A6, demonstraram evidências iniciais de envolvimento com os processos que compõem o Pensamento Computacional (PC). Os esboços elaborados por eles revelam a construção de um plano mental preliminar, essencial para a transição entre a concepção da solução e sua implementação na linguagem de programação baseada em blocos, utilizando o Scratch (WING, 2016).

Esses registros gráficos, embora superficiais e pouco detalhados, desempenham um papel importante na organização e decomposição do problema, servindo como recurso de apoio à estruturação lógica da tarefa. Entre as técnicas utilizadas, destacam-se o uso de

retângulos para representar a decomposição do problema e de setas e linhas para indicar a sequência lógica das ações, evidenciando o desenvolvimento de padrões. Observa-se, ainda, a aplicação clara de estruturas de repetição, bem como a definição de parâmetros relevantes para a realização da abstração necessária à resolução da atividade. Essa ação demonstra o uso de um componente essencial do Pensamento Computacional, pois "a decomposição permite ao aprendiz identificar subproblemas e estruturar soluções mais eficientes" (BRENNAN; RESNICK, 2012, p. 9).

O algoritmo desenvolvido pelo aluno A1, representado na Figura 2, evidencia que o participante enfrentou dificuldades na resolução da atividade. Ao analisar o código observa-se que ele não conseguiu implementar uma estrutura de repetição, o que comprometeu a organização lógica do programa e a criação de padrões necessários para o processamento adequado dos dados. Além disso, não foram identificadas estratégias de abstração que permitissem a eliminação de informações irrelevantes, o que indica limitações na capacidade de sintetizar e estruturar a solução de forma eficiente (WING, 2008, p. 371).

Figura 2 – Algoritmo A1 da Atividade



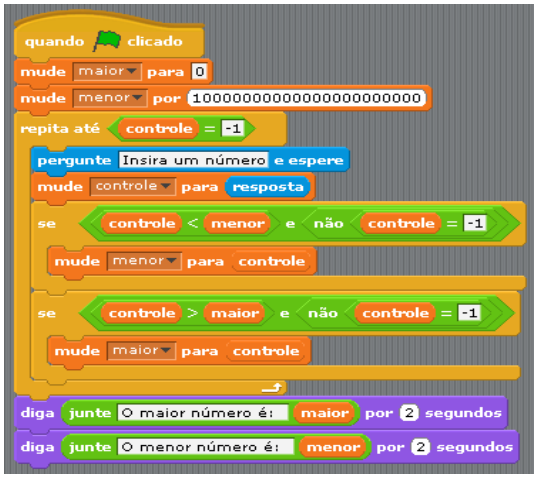
Fonte: os autores.

O código desenvolvido por A1 revela uma tentativa de resolução da atividade por meio do uso de estruturas condicionais. No entanto, a utilização isolada dessas estruturas não foi suficiente para atender à complexidade da proposta, que exigia, também, o emprego de estruturas de repetição. Outra fragilidade observada refere-se à ausência de uma ordem lógica clara na organização das informações, evidenciando dificuldades já na etapa inicial

de decomposição do problema. Apesar dessas limitações na implementação do algoritmo, o esboço elaborado pelo aluno demonstra que ele foi capaz de construir um plano mental preliminar. Nesse código é possível identificar a intenção de empregar estruturas de repetição, o reconhecimento de padrões e até mesmo uma sequência lógica de ações. No entanto, o aluno não conseguiu transpor esse plano mental para o ambiente de programação de forma funcional, o que indica dificuldades no uso integrado das habilidades que compõem o Pensamento Computacional.

O código apresentado na Figura 3, desenvolvido pelo aluno A2, também apresentou algumas fragilidades. No entanto, diferentemente do aluno A1 o A2 conseguiu estruturar grande parte do algoritmo proposto. A principal dificuldade observada esteve relacionada à lógica de programação, especialmente na organização das estruturas de repetição e condição que não foram articuladas de forma eficaz para atender aos objetivos da atividade. O algoritmo apresentou falhas no tratamento das informações inseridas pelo usuário, comprometendo os padrões estabelecidos para a solução do problema. Como consequência, as variáveis foram preenchidas de forma incorreta, resultando na exibição de valores imprecisos ou indevidos. É frequente encontrar aluno com dificuldade na identificação de informações relevantes, que percebe a relação entre conceitos matemáticos, construção de modelos matemáticos e de organização de procedimento coerentes (SUPARMAN; TURMUDI, 2024). Esses aspectos evidenciam que embora A2 tenha demonstrado compreensão parcial da tarefa, ainda apresenta limitações no domínio das estruturas lógicas necessárias à resolução adequada do problema. Dificuldades como essas "levam a erros nos processos de resolução de problemas, especialmente em tópicos matemáticos que exigem habilidades de pensamento de ordem superior" (SITEPU *et al*., 2026, p. 236).

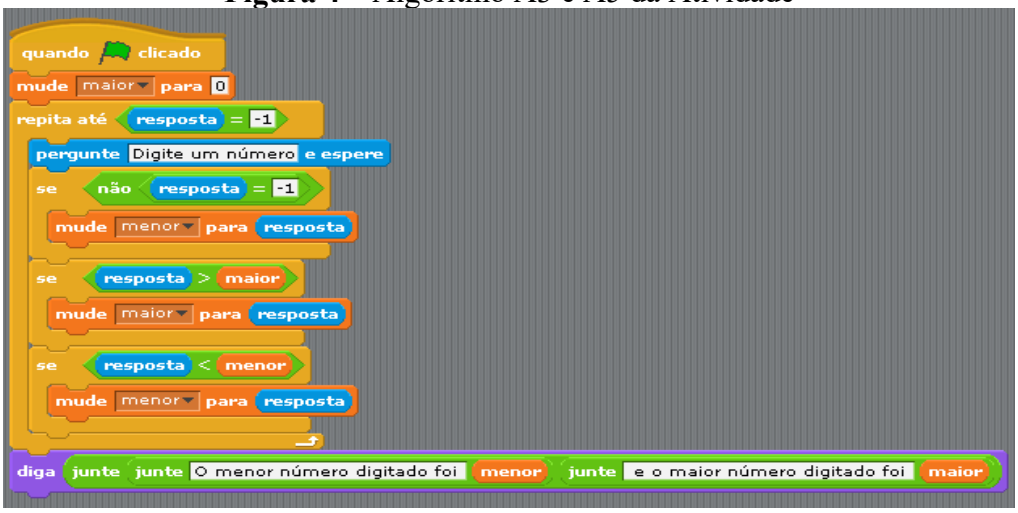
Figura 3 – Algoritmo A2 da Atividade



Fonte: os autores.

Os algoritmos desenvolvidos pelos alunos A3, A4, A5 e A6 atendem plenamente aos objetivos propostos na atividade avaliativa. Todos conseguiram resolver o problema de forma correta, empregando estratégias alinhadas aos pilares do Pensamento Computacional. Observou-se que embora os quatro alunos tenham apresentado soluções funcionais e conceitualmente coerentes, utilizaram abordagens distintas na estruturação do código. Essa diversidade na escolha das estruturas lógicas evidencia tanto a autonomia quanto a compreensão dos conceitos de programação, permitindo diferentes caminhos para alcançar a mesma solução. Isso corrobora com Silva *et al*. (2020) que evidenciam que a resolução de problemas computacionais possui natureza heurística, permitindo que diferentes alunos mobilizem processos de exploração, planejamento, análise e verificação para alcançarem soluções válidas para um mesmo problema.

Figura 4 – Algoritmo A3 e A5 da Atividade



Fonte: os autores.

Os alunos A3 e A5 desenvolveram algoritmos utilizando uma estrutura de repetição que incorporava três estruturas condicionais, demonstrando domínio na combinação de diferentes recursos da lógica de programação. No caso do aluno A4, cujo código está representado na Figura 5, observou-se uma abordagem diferenciada na resolução do problema.

Figura 5 – Algoritmo A4 da Atividade

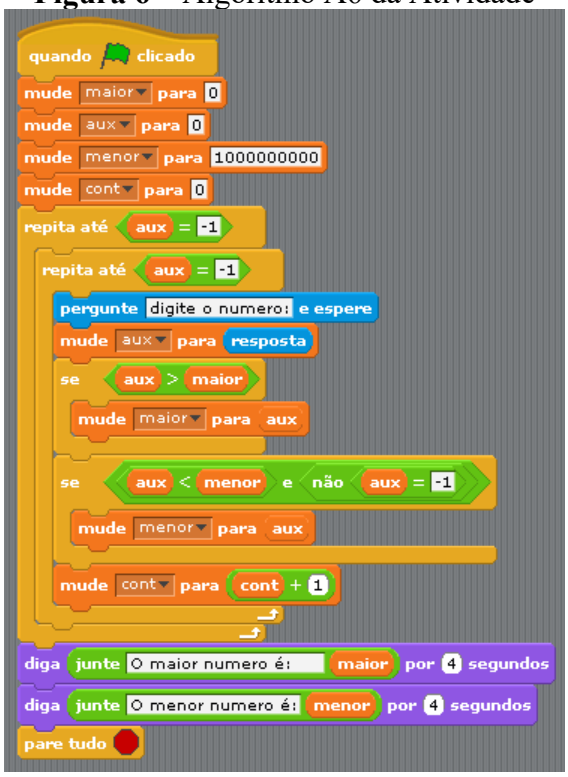
Fonte: os autores.

O aluno A4 utilizou duas estruturas de repetição combinadas com uma estrutura condicional, além de empregar uma **lista**, recurso responsável por armazenar dados de forma organizada. Essa escolha técnica o diferenciou dos demais colegas, uma vez que o uso de listas não havia sido previamente abordado durante o curso. Tal iniciativa evidencia a autonomia do aluno e sua disposição em explorar de maneira proativa as funcionalidades e ferramentas disponibilizadas pelo ambiente de programação Scratch, a fim de encontrar soluções mais eficientes para o problema proposto.

O algoritmo desenvolvido pelo aluno A6, apresentado na Figura 6, usou duas estruturas de repetição combinadas com uma estrutura condicional, além de empregar uma **variável auxiliar**, cuja função foi armazenar temporariamente determinadas informações ao longo da execução do código. Essa estratégia demonstra compreensão quanto ao controle de fluxo e ao armazenamento intermediário de dados, aspectos essenciais para o desenvolvimento de algoritmos mais robustos e eficientes. A solução proposta revela organização lógica e aplicação dos princípios do Pensamento Computacional; sobretudo no que diz respeito à abstração e ao planejamento sequencial.

Pode-se dizer que tanto o aluno A4 como o A6, a partir da decisão de investigarem os recursos adicionais do Scratch, demonstraram um comportamento exploratório que transcende a simples execução da tarefa proposta e, segundo Silva (2020), as etapas do PC favorecem a construção autônoma de estratégias para a resolução de problemas.

Figura 6 – Algoritmo A6 da Atividade



Fonte: os autores.

Levando em consideração esse cenário, a ferramenta proporcionou liberdade de escolha aos alunos, possibilitando que eles usufruíssem de diferentes caminhos lógicos para solucionar a problemática. Diante dessas variedades de caminhos utilizados pelos alunos para solucionaram a atividade, a ferramenta ofereceu a liberdade de escolha e experimentação de comandos, podendo realizar inúmeras tentativas que poderão dar certo ou errado, refletindo sobre o melhor caminho para solucionar o problema, estimulando e desenvolvendo, assim, o Pensamento Computacional, pois, conforme Silva (2020), a diversidade de estratégia sugere não apenas a apropriação dos conceitos envolvidos, mas também o exercício de competências relacionadas à reflexão, depuração e tomada de decisão durante a construção das soluções.

A partir da análise dos algoritmos desenvolvidos pelos alunos elaboramos o Quadro 4, que apresenta os principais aspectos observados durante a execução das atividades. O quadro contempla critérios como a utilização de estruturas de repetição e condição, o emprego de recursos adicionais (como listas e variáveis auxiliares), bem como o nível de aplicação dos pilares do Pensamento Computacional. Essa organização permite visualizar, de forma objetiva, as estratégias adotadas pelos participantes, bem como suas respectivas dificuldades e avanços no processo de resolução do problema proposto.

Quadro 4 – Desempenho dos Alunos

Aluno	Estrutura de Repetição	Uso de Condição	Recurso Adicional	Aplicação do PC	Observações
A1	Não utilizou	Sim	Não	Parcial (esboço mental não concretizado)	Não solucionou a atividade
A2	Parcial	Sim	Não	Parcial (falhas em padrões e lógica)	Estrutura incompleta e imprecisa
A3	Sim	Sim	Não	Completa (aplicou todos os pilares)	Código funcional e bem estruturado
A4	Sim	Sim	Lista (não trabalhada em aula)	Completa e inovadora	Solução autônoma e criativa
A5	Sim	Sim	Não	Completa (semelhante a A3)	Código coerente e funcional
A6	Sim	Sim	Variável auxiliar	Completa (boa abstração e organização)	Estratégia bem estruturada

Fonte: os autores.

A análise dos resultados obtidos ao longo da atividade avaliativa revelou diferentes níveis de compreensão e aplicação dos conceitos de programação entre os alunos participantes. De modo geral, observou-se que a maioria dos alunos foi capaz de empregar estruturas de repetição e condição de maneira funcional, demonstrando domínio parcial ou completo dos pilares do Pensamento Computacional.

Os alunos A3, A4, A5 e A6 destacaram-se por apresentar soluções corretas, bem estruturadas e alinhadas aos objetivos da atividade, evidenciando habilidades como abstração, decomposição e reconhecimento de padrões. Dentre eles, o aluno A4 mostrou um desempenho particularmente significativo ao incorporar recursos não explorados em aula, como o uso de listas, demonstrando iniciativa e autonomia.

Por outro lado, os alunos A1 e A2 enfrentaram maiores dificuldades; especialmente na implementação de estruturas lógicas e na organização sequencial dos algoritmos. Esses resultados indicam que, embora nem todos os alunos tenham alcançado um nível avançado de desempenho, a proposta metodológica contribuiu para o desenvolvimento progressivo do

pensamento computacional, oferecendo subsídios para intervenções pedagógicas mais direcionadas em futuras atividades.

5 Considerações finais

Este artigo teve como objetivo analisar o desenvolvimento do Pensamento Computacional por meio da utilização da ferramenta Scratch, com ênfase na compreensão e aplicação de estruturas lógicas de repetição. A atividade avaliativa proposta possibilitou observar, de forma prática, como os alunos lidam com os desafios relacionados à lógica de programação e à estruturação algorítmica.

Os resultados obtidos evidenciam que a programação em blocos, quando associada a uma abordagem pedagógica, favorece o desenvolvimento de habilidades cognitivas relacionadas ao Pensamento Computacional, como a decomposição, a abstração, o reconhecimento de padrões e a construção de algoritmos. A análise dos algoritmos desenvolvidos pelos alunos permitiu identificar diferentes níveis de apropriação dos conceitos, destacando-se a importância do acompanhamento docente e da oferta de estratégias diversificadas para promover a aprendizagem.

Concluímos que o Scratch pode ser uma ferramenta eficaz no processo de ensino e de aprendizagem de programação; especialmente por proporcionar um ambiente visual e interativo que reduz as barreiras enfrentadas por alunos iniciantes. Além disso, o uso das estruturas de repetição, como foco didático, contribuiu para a consolidação do raciocínio lógico e da autonomia na resolução de problemas computacionais.

Referências

BRACKMANN, Christian Puhlmann. **Desenvolvimento do pensamento computacional através de atividades desplugadas na educação básica**. 2017. 226f. Tese (Doutorado em Informática da Educação) – Universidade Federal do Rio Grande do Sul, Porto Alegre, 2017.

BRENNAN, Karen; RESNICK, Mitchel. **New frameworks for studying and assessing the development of computational thinking**. Proceedings of AERA, Vancouver, p. 1–25, 2012. Disponível em: https://scratched.gse.harvard.edu/ct/files/AERA2012.pdf. Acesso em: 13 Nov 2025.

CALDEIRA, Jefta; VILELA, Ana Paula. **Um Mapeamento Sistemático para auxiliar na escolha de plataformas EAD para o ensino-aprendizagem de Algoritmos e Programação de Computadores.** In: Brazilian Symposium on Computers in Education (Simpósio Brasileiro de Informática na Educação-SBIE). 2016. p. 52.

COSTELLA, Leonardo.; LICKS, Gabriel Paludo.; TEIXEIRA, Adriano Canabarro. **Uma solução livre e de baixo custo para prática e aprendizagem de programação e robótica.** In: Brazilian Symposium on Computers in Education (Simpósio Brasileiro de Informática na Educação-SBIE). 2016. p. 846.

ROSA, Maurício Machado; GIRAFFA, Lucia Maria Martins**. O ensino de programação de computadores e EAD: uma parceria possível**. In: 17º Congresso Internacional da ABED, 2011, Brasil. 2011.

MALONEY, John; RESNICK, Mitchel; RUSK, Natalie; SILVERMAN, Brian; EASTMOND, Evelyn. **The scratch programming language and environment.** ACM Transactions on Computing Education, vol. 10, n. 4, p. 1-15, 2010.

PEREIRA, Paulo Sérgio; MEDEIROS, Max; MENEZES, Jônatas Webber Mendes. **Análise do Scratch como ferramenta de auxílio ao ensino de programação de computadores**. In: **CONGRESSO BRASILEIRO DE EDUCAÇÃO EM ENGENHARIA (COBENGE)**, 40., 2012, Belém, PA. Anais [...]. Belém: ABENGE, 2012.

RAABE, André Luís Alice; BRACKMANN, Christian Puhlmann; CAMPOS, Flávio Rodrigues. **Currículo de referência em tecnologia e computação: da educação infantil ao ensino fundamental**. São Paulo: CIEB, 2018. E-book em PDF. Disponível em:http://curriculo.cieb.net.br/assets/docs/Curriculo_de_Referencia_em_Tecnologia_e_Computacao.pdf. Acesso em: 20 nov 2018.

RAMOS, Vinicius; FREITAS, Mateus Felipe; GALIMBERTI, Maurício Floriano; MARIANI, Antonio Carlos; WAZLAWICK, Raul Sidnei. **A comparação da realidade mundial do ensino de programação para iniciantes com a realidade nacional: revisão sistemática da literatura em eventos brasileiros**. In: **SIMPÓSIO BRASILEIRO DE INFORMÁTICA NA EDUCAÇÃO (SBIE)**, 26., 2015, Maceió. Anais [...]. Porto Alegre: Sociedade Brasileira de Computação, 2015. p. 318–327.

RODRIGUES, Carla Lopes.; ZEM-LOPES, Aparecida Maria.; MARQUES, Leonardo; ISOTANI, Seiji. Pensamento Computacional: transformando ideias em jogos digitais usando o *Scratch*. In: WORKSHOP DE INFORMÁTICA NA ESCOLA, XXI, 2015, Maceió. **Anais do XXI WIE 2015**: Ciência, Inovação e Empreendedorismo: Integração Latino-Americana pela Educação, Maceió: Sociedade Brasileira de Computação, 2015, UFAL, 2015, p. 62-71.

SILVA, Ricardo Scucuglia Rodrigues; GADANIDIS, George; HUGHES, Janette; NAMUKASA, Immaculate Kizito. Pensamento Computacional como uma iniciativa heurística: soluções de alunos sobre problemas de programação. **Pro-Posições**, Campinas, SP, v. 31, p. e20180034, 2020. Disponível em: https://periodicos.sbu.unicamp.br/ojs/index.php/proposic/article/view/8664287. Acesso em: 9 jun. 2026.

SITEPU, Israil, *et al*. Revealing Students'Computational Thinking Error Patterns in Solving Two-Variable Linear Inequality Systems, **Journal of Mathematics Education**, Indonesia, v. 12, n. 1, p. 235-252, 2026. Disponível: https://jurnalnasional.ump.ac.id/index.php/alphamath/article/view/30386/9760. Acesso em: 03 Jun. 2026.

SOBREIRA, Elaine Silva Rocha; TAKINAMI, Olga Kikue; SANTOS, Verônica Gomes dos. **Programando, criando e inovando com o Scratch: em busca da formação do cidadão do século XXI**. Jornada de Atualização em Informática na Educação, Porto Alegre, v. 1, n. 1, p. 126–152, 2013.

SUPARMAN, Juandi Dadang; TURMUDI, Wahyudin. Computational thinking in mathematics instruction integrated STEAM education: Global trend and students achievement in the last two decades. Beta: **Jurnal Tadris Matematika**, Mataram, v. 17, n. 2, p. 101–134, 2024. Disponível em: https://jurnalbeta.ac.id/index.php/betaJTM/article/view/643. Acesso em: 07 Jun. 2025.

WING, Jeannette. Marie. Computational thinking. **Communications of the ACM**, New York, v. 49, n. 3, p. 33-35, 2006.

WING, Jeannette Marie. **Computational thinking and thinking about computing**. Philosophical Transactions of the Royal Society A, v. 366, n. 1881, p. 3717–3725, 2008.

WING, Jeannette Marie. Pensamento Computacional: Um conjunto de atitudes e habilidades que todos, não só cientistas da computação, ficaram ansiosos para aprender e usar. **Revista Brasileira de Ensino de Ciência e Tecnologia**, Ponta Grossa, v. 9, n. 2, p. 1-10, 2016.

ZAHARIJA, Goran; MLADENOVIĆ, Saša; BOLJAT, Ivica. **Introducing basic programming concepts to elementary school children**. Procedia - Social and Behavioral Sciences, Turkey, v. 106, p. 1576–1584, 2013.