Scientific Society Journal
ISSN: 2595-8402
Journal DOI: 10.61411/rsc31879

REVISTA SOCIEDADE CIENTÍFICA, VOLUME 7, NÚMERO 1, ANO 2024

ARTIGO ORIGINAL

Desenvolvimento do pensamento computacional: o uso do Scratch na introdução à programação

Ana Maria Di Grado Hessel[1]; Erick Quintino Dos Santos[2]; Estefânia Portomeo Cançado Lemos[3]; Daniel Couto Gatti[4]; José Rodolfo Dias Dos Santos[5]; Rommel Gabriel Gonçalves Ramos[6]

Como Citar:
Hessel; Ana Maria di Grado, DOS SANTOS; Erick Quintino, LEMOS; Estefânia Portomeo Cançado, GATTI; Daniel Couto, DOS SANTOS; José Rodolfo Dias; RAMOS; Rommel Gabriel Gonçalves. Desenvolvimento do pensamento computacional: o uso do scratch na introdução à programação. Revista Sociedade Científica, vol.7, n. 1, p.892-923, 2024.
https://doi.org/10.61411/rsc202427417

DOI: 10.61411/rsc202427417

Área do conhecimento: Interdisciplinar

Sub-área: Matemática; Computação

Palavras-chaves: Pensamento computacional; scratch; algoritmização; raciocínio lógico; programação.

Publicado: 19 de fevereiro de 2024

Resumo

O presente trabalho aborda o desenvolvimento do pensamento computacional através do uso do Scratch na introdução à programação. Inicialmente, são apresentados os conceitos de pensamento computacional e Scratch, bem como a importância do desenvolvimento do pensamento computacional para iniciar na programação. Este trabalho também reflete as principais dificuldades de egressos na programação e as principais causas da desistência desses iniciantes. Por fim, são destacadas as vantagens do uso do Scratch no desenvolvimento do pensamento computacional, tais como o estímulo à criatividade, o desenvolvimento de habilidades cognitivas e a promoção da colaboração. Com base nas informações apresentadas, é possível perceber a relevância do uso do Scratch como ferramenta de ensino para a introdução à programação e desenvolvimento do pensamento computacional.

1. Introdução

O pensamento pode ser definido como um processo cognitivo que envolve a percepção, a memória, a linguagem, a criatividade e o raciocínio, entre outros fatores. De acordo com Rosenthal (2017), o pensamento é fundamental para a tomada de decisões e resolução de problemas. Além disso, segundo Gazzaniga (2018), o

[1]PUCSP ✉
[2]UEMG-MG ✉
[3]UNEP/UEMG-MG ✉
[4]PUCSP ✉
[5]UEMG-MG ✉
[6]PUCSP/UEMG-MG ✉

Scientific Society Journal
ISSN: 2595-8402
Journal DOI: 10.61411/rsc31879

pensamento é responsável pela nossa capacidade de aprender, comunicar e interagir com o mundo.

A palavra algoritmização vem de algoritmos, que, de acordo com Sedgewick e Wayne (2011, p. 6), "um algoritmo é uma sequência bem definida de instruções computacionais que, quando executadas, realizam uma tarefa específica". Essa definição se alinha com a ideia de que um algoritmo é uma sequência lógica de instruções que leva à solução de um problema. Já Knuth (1997, p. 1) define algoritmo como "um procedimento passo a passo para resolver um problema matemático bem definido".

Assim sendo, fica evidente que um algoritmo é uma sequência de passos que leva à solução de um problema específico. Por fim, Cormen et al. (2009, p. 5) definem algoritmo como "uma sequência finita de instruções bem definidas e não ambíguas, cada uma das quais pode ser executada mecanicamente em um período de tempo finito e com uma quantidade finita de esforço". Essa definição destaca a importância de que cada instrução do algoritmo seja clara e sem ambiguidade, além de ressaltar que a execução do algoritmo deve ser possível em um período de tempo finito;

Dessa forma, podemos concluir que, Segundo Brackmann (2019, p. 45), "algoritmização é o processo de transformar uma solução para um problema em um algoritmo que possa ser implementado em uma linguagem de programação".

A algoritmização, levada ao âmbito tecnológico, para a área da computação, leva-nos a visualizar a base da programação. Para se tornar programador, entender algoritmos é uma etapa crucial e a mais importante, pois, assim, é levantado o conceito do pensamento computacional que não é apenas concentrado na computação, mas leva uma pessoa a compreender como, de fato, um computador age. Ele tem sido destacado como uma habilidade importante para a formação de profissionais em áreas relacionadas à tecnologia da informação e comunicação (Rosen et al., 2013).

Scientific Society Journal
ISSN: 2595-8402
Journal DOI: 10.61411/rsc31879

2. Referencial teórico

2.1 Conceitos de pensamento computacional

Aprofundando no conceito de pensamento computacional, podemos ver que a primeira vez que o termo foi introduzido na comunidade de tecnologia foi na década de 80, com Seymour Papert. Segundo Papert, o pensamento computacional é "uma maneira de pensar que envolve a criação de imagens mentais de como os modelos são formados e manipulados e uma maneira de pensar sobre como o pensamento pode ser estruturado e organizado" (Papert, 1993, p. 18).

Wing (2006) diz que o PC é um processo mental que envolve o uso de conceitos e estratégias fundamentais da ciência da computação para resolver problemas, projetar sistemas e entender o comportamento humano e natural.

Segundo Barr e Stephenson (2011), "o pensamento computacional é um conjunto de habilidades mentais que permite resolver problemas de maneira mais eficiente e eficaz", e essa abordagem requer "pensamento lógico, algorítmico e crítico" (p. 34).

Brackmann (2017, p. 29) define o PC da seguinte maneira: “é 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”.

O pensamento computacional segundo Raabe, Brackmann e Campos (2018) é dividido em 4 conceitos principais, são eles:

- Decomposição
- Abstração
- Reconhecimento de Padrões
- Algoritmização

Scientific Society Journal
ISSN: 2595-8402
Journal DOI: 10.61411/rsc31879

2.1.1 Decomposição

A decomposição é descrita por autores como Wing (2006) como uma forma de decomposição "dividir um problema complexo em partes menores, mais gerenciáveis e compreensíveis", e é uma habilidade fundamental do pensamento computacional que pode ser aplicada em diversas áreas, incluindo a programação (Resnick et al., 2009).

Segundo Almeida e Mendes (2018), um exemplo de decomposição seria o processo de criação de um jogo no Scratch. Antes de começar a programar, é necessário decompor o jogo em partes menores, como a criação dos personagens, a definição das regras, a implementação dos controles e a criação dos cenários.

Cada uma dessas partes pode ser decomposta ainda mais em tarefas menores e mais simples, até que o problema seja resolvido por completo. Dessa forma, a decomposição auxilia na organização e resolução de problemas complexos de forma mais eficiente.

2.1.2 Abstração

A abstração é definida como a capacidade de identificar as características e propriedades essenciais de um objeto, abstraindo-se das suas particularidades irrelevantes (Wing, 2006).

Um exemplo de abstração pode ser o processo de criação de uma função para somar dois números em um programa de computador. O programador precisa identificar quais são as informações essenciais para realizar a operação de soma e abstrair-se das particularidades de cada número em si.

Assim, a função pode ser utilizada diversas vezes, com diferentes números, sem precisar ser reescrita para cada caso específico. Essa capacidade de abstração é fundamental para o desenvolvimento de programas de computador e é uma habilidade importante a ser desenvolvida no ensino de programação (Brennan & Resnick, 2012).

Scientific Society Journal
ISSN: 2595-8402
Journal DOI: 10.61411/rsc31879

2.1.3 Reconhecimento de padrões

O reconhecimento de padrões é um dos pilares do pensamento computacional, que envolve identificar regularidades em dados e informações. Um exemplo de aplicação desse pilar é a análise de imagens. Segundo Raabe, Brackmann e Campos (2018), "A análise de imagens pode ser entendida como um processo que consiste em extrair informações a partir de imagens, sejam elas digitais ou analógicas, por meio da detecção, identificação e/ou quantificação de características específicas da imagem".

Um exemplo prático de reconhecimento de padrões em análise de imagens é a detecção de objetos em uma cena. De acordo com Lam et al (2018), "O reconhecimento de objetos é uma tarefa fundamental em muitas aplicações de visão computacional, incluindo vigilância, rastreamento, navegação autônoma, reconhecimento de face e classificação de imagens". Por meio do uso de algoritmos de processamento de imagem, é possível detectar objetos em uma imagem e identificá-los de acordo com padrões previamente estabelecidos, como por exemplo, forma, cor e textura.

2.1.4 Algoritmização

Algoritmização é o processo de transformar uma tarefa em uma sequência de passos bem definidos e precisos, a fim de serem executados por um computador ou outro dispositivo automatizado. Segundo Dromey (1996), algoritmo é uma "sequência finita de passos precisamente definidos para realizar uma tarefa específica".

Conforme Santos e Machado (2019), a algoritmização é fundamental para a programação, já que é a partir dela que se constrói um código estruturado e organizado, capaz de ser compreendido e executado pelo computador.

Scientific Society Journal
ISSN: 2595-8402
Journal DOI: 10.61411/rsc31879

Nesse sentido, a algoritmização envolve a habilidade de pensar de forma estruturada e sequencial, e de ser capaz de decompor problemas complexos em tarefas menores e mais simples.

Segundo Aho, Hopcroft e Ullman (1983 p. 3), a algoritmização é uma "arte" que requer habilidade e criatividade para encontrar a solução mais eficiente e elegante para um problema. Eles afirmam que "a habilidade em algoritmização é fundamental para o sucesso em ciência da computação".

Além disso, Gries e Schneider (1993, p.2) enfatizam que a algoritmização não é apenas sobre a criação de algoritmos eficientes, mas também sobre a capacidade de comunicar ideias claramente e de forma concisa. Eles afirmam que "a algoritmização é a arte de apresentar soluções de forma clara e concisa para que possam ser facilmente entendidas e executadas".

2.2 A algoritmização e o pensamento lógico

Algoritmização e pensamento lógico são dois conceitos fundamentais para o desenvolvimento do pensamento computacional. Segundo Ribeiro e Lopes (2017), a algoritmização envolve a capacidade de organizar informações e transformá-las em um conjunto de passos bem definidos que podem ser seguidos para resolver um determinado problema. Já o pensamento lógico envolve a habilidade de raciocinar de forma sistemática e coerente, utilizando o raciocínio dedutivo para inferir informações a partir de premissas.

De acordo com Valente e Almeida (2016), a algoritmização é uma ferramenta importante para o desenvolvimento do pensamento lógico, uma vez que exige que o aluno organize as informações de forma lógica e coerente, seguindo uma sequência de passos que possibilite a resolução do problema. Por sua vez, o pensamento lógico é fundamental para a algoritmização, uma vez que é necessário um raciocínio dedutivo para a construção de algoritmos precisos e eficientes.

Scientific Society Journal
ISSN: 2595-8402
Journal DOI: 10.61411/rsc31879

Um exemplo prático de algoritmização no ensino do pensamento lógico é o uso de problemas matemáticos para ensinar a criar algoritmos. Conforme destacado por Oliveira e Gomes (2017 p.4), "a resolução de problemas matemáticos pode ser utilizada para ensinar algoritmos, já que envolve a formulação de um problema e sua solução, que é o objetivo dos algoritmos". Por meio do processo de algoritmização, os estudantes aprendem a decompor um problema complexo em tarefas menores e sequenciais, identificar as informações relevantes e definir uma solução clara e precisa para o problema proposto.

2.3 Ferramentas de programação para desenvolvimento do pensamento computacional

Existem diversas ferramentas de programação que podem ser utilizadas no desenvolvimento do pensamento computacional, e cada uma delas apresenta suas próprias características e funcionalidades.

O Scratch é uma dessas ferramentas, e é especialmente indicado para iniciantes por utilizar a programação em blocos, o que torna mais fácil a visualização da lógica de programação (GROVER e PEA, 2013).

Outra ferramenta bastante utilizada é o Code.org, que oferece um conjunto de atividades e recursos para o ensino da programação para crianças, jovens e adultos (ALMSTRUM et al., 2014).

Além dessas ferramentas, existe o App Inventor, que permite que os usuários criem aplicativos para Android a partir de blocos de programação visual (KIRKPATRICK et al., 2014). Outra ferramenta é o Greenfoot, que utiliza a linguagem de programação Java e é especialmente indicado para o ensino de programação orientada a objetos (KÖLLING, 2003).

Scientific Society Journal
ISSN: 2595-8402
Journal DOI: 10.61411/rsc31879

Essas ferramentas são importantes para o desenvolvimento do pensamento computacional porque permitem que os usuários construam algoritmos e resolvam problemas de maneira criativa e colaborativa, o que contribui para o aprendizado da programação e o desenvolvimento de habilidades cognitivas importantes (WING, 2006).

2.3.1 Scratch como ferramenta para programação

O Scratch é uma ferramenta de programação visual que pode ser utilizada por alunos de todas as idades e níveis de habilidade, permitindo a criação de histórias interativas, jogos e animações. Segundo Resnick (2007), o uso do Scratch permite que os alunos possam aprender programação e ao mesmo tempo desenvolver habilidades de resolução de problemas, raciocínio lógico e criatividade e, também, auxilia no desenvolvimento do pensamento computacional.

> O Scratch foi desenvolvido por um grupo de pesquisadores do Lifelong Kindergarten Group no MIT Media Lab, liderado por Mitchel Resnick, que tinha como objetivo criar uma ferramenta acessível e divertida para ensinar programação para crianças (RESNICK et al., 2009).

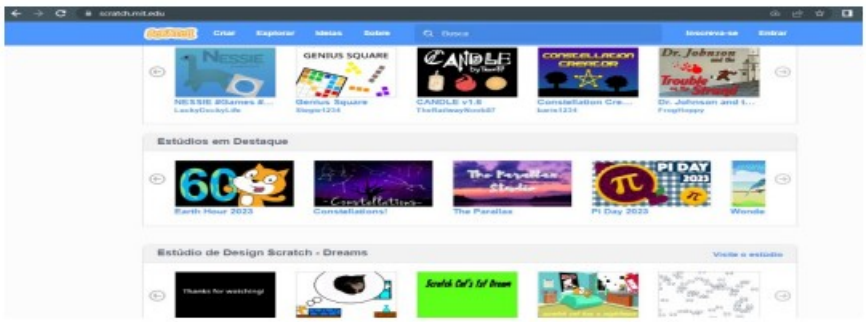

Figura 01: Tela inicial do Scratch. Fonte: https://scratch.mit.edu/. Printscreen feito pelo pesquisador.

O Scratch foi lançado em 2007 e desde então tem sido utilizado em todo o mundo para ensinar programação a crianças e jovens. De acordo com Maloney et al. (2010), o Scratch se tornou popular entre educadores por sua interface intuitiva e pela

Scientific Society Journal
ISSN: 2595-8402
Journal DOI: 10.61411/rsc31879

facilidade de uso, que permite aos alunos experimentar, criar e compartilhar seus próprios projetos de forma colaborativa.

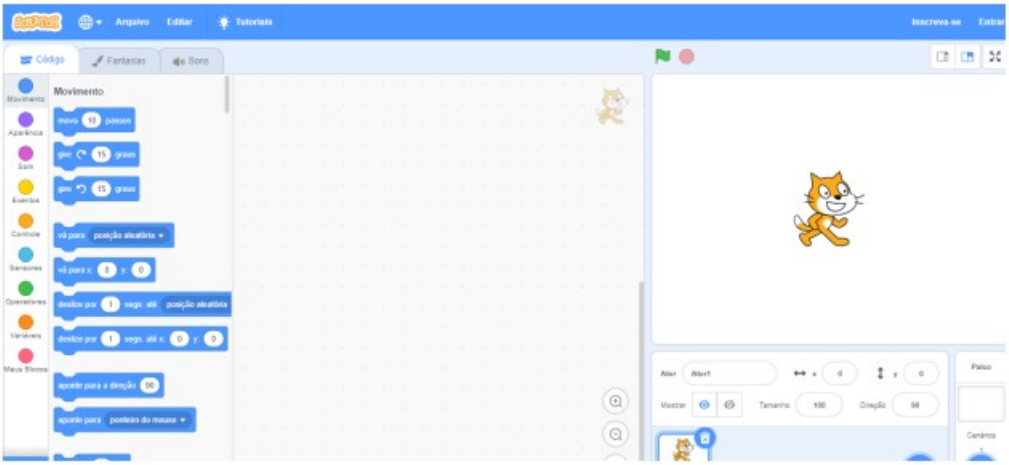

Figura 02: Tela da área de construção de projetos do Scratch. Fonte: https://scratch.mit.edu/projects/editor/?tutorial=getStarted.

2.3.2 Projeto prático no scratch

Existem, no site do Scratch (https://scratch.mit.edu/), vários exemplos de projetos finalizados e postados por usuários. Um desses exemplos é o projeto “Haxball Offline”, um simulador do jogo “Haxball” (https://www.haxball.com/) mas que, diferente do original, funciona de forma desconectada da internet para dois jogadores, por meio do uso do teclado do computador. O criador utilizou blocos de controladores, movimentos e sentidos para finalizar o projeto.

Scientific Society Journal
ISSN: 2595-8402
Journal DOI: 10.61411/rsc31879

Figura 03 - Projeto no Scratch. Fonte: https://scratch.mit.edu/projects/3005452/. Printscreen feito pelo pesquisador.

Esse projeto, feito pelo usuário "FRKN", simula uma partida do jogo online "Haxball", de forma offline para dois jogadores. O objetivo do jogo é mover seu jogador, caracterizado como os círculos vermelho e azul e levar a bola, que seria o círculo menor e branco, até o gol do adversário, com opção de chutar.

Figura 04 - Projeto do Scratch em ação. Fonte: https://scratch.mit.edu/projects/3005452/. Printscreen feito pelo pesquisador.

Para construir esse projeto, o usuário combina comandos de movimentação e mudanças de ação quando alguma tecla é pressionada. Ao passar da bola na delimitação de onde é o gol, o placar se altera, sem limite de tempo ou gols. Quando a tecla "Restart" é pressionada, no meio do placar, o jogo se reinicia.

Scientific Society Journal
ISSN: 2595-8402
Journal DOI: 10.61411/rsc31879

Figura 05 - Blocos de comando do projeto.Fonte: https://scratch.mit.edu/projects/3005452/editor/. Printscreen feito pelo pesquisador.

Quando se analisa a construção do projeto, é possível notar que existem oito "fantasias", figuras que simulam objetos ou personagens. São eles: os jogadores, a bola, as bordas do campo, as traves e os botões "Restart" e "How to play". Há, também, um cenário simulando o campo de futebol.

Na parte da montagem do código, foi usado o bloco inicial de clique. Em seguida, existe uma ordem de blocos que envolve os fatores do jogo em ação. Em primeiro lugar, os jogadores e a bola tomam suas devidas posições, feito pelo bloco de movimento X e Y. O cenário é feito e o jogo é iniciado ao apertar o botão de início. Em seguida, os blocos de controle definem, junto com blocos operadores, de movimento e aparência, as regras do jogo para um perfeito funcionamento.

2.4 Utilizando o Scratch

Segundo Resnick (2009, p. 14), o aprendizado do Scratch envolve uma abordagem hands-on, em que os estudantes aprendem a partir da experimentação e da criação de seus próprios projetos. Ele enfatiza a importância da liberdade criativa, em

Scientific Society Journal
ISSN: 2595-8402
Journal DOI: 10.61411/rsc31879

que os estudantes são encorajados a explorar e testar ideias, ao invés de seguir um conjunto rígido de instruções.

Resnick (2009, p.14) sugere que os estudantes comecem com projetos simples e gradualmente avancem para projetos mais complexos, permitindo que eles adquiram habilidades e confiança ao longo do caminho.

Além disso, Resnick (2009, p. 14) enfatiza a importância do trabalho colaborativo, incentivando os estudantes a compartilhar seus projetos com seus colegas e a aprender uns com os outros.

Ele destaca que o Scratch é uma ferramenta poderosa para promover a aprendizagem social e a construção de comunidades de aprendizagem em torno da programação.

O uso da ferramenta Scratch é bastante simples, não havendo grandes dificuldades para aprender o que cada bloco e comando constitui e faz. A seguir, veja um pequeno roteiro de como usar a ferramenta:

1. Primeiramente, acesse o site oficial do Scratch (https://scratch.mit.edu/) ou baixe o programa em seu computador.
2. Crie uma conta no site ou faça login com uma conta existente.
3. Na página inicial do Scratch, clique em "Criar" para começar um novo projeto.
4. Arraste e solte blocos de programação da barra lateral para a área de programação central.
5. Use os blocos para criar comandos que movam personagens, reproduzam sons, façam interações entre objetos, entre outras ações.
6. Personalize os personagens e objetos do projeto, alterando suas cores, formas, tamanhos e posições.
7. Teste o projeto clicando no botão "Ver Projeto" para ver como ele é executado.
8. Compartilhe o projeto com outros usuários do Scratch, publique-o na galeria ou exporte-o para outros dispositivos.

Scientific Society Journal
ISSN: 2595-8402
Journal DOI: 10.61411/rsc31879

Os blocos de comando do Scratch oferecem uma grande variedade de possibilidades para construção de projetos. As opções de blocos são de: ações, aparências, sons, eventos, controles, sensores, operadores, variáveis e blocos que podem ser criados pelo usuário.

Figura 06 - Blocos de comando do Scratch. Fonte: https://scratch.mit.edu/projects/editor/. Printscreen feito pelo pesquisador.

A importância do Scratch no desenvolvimento do pensamento computacional se deve ao fato de que ele permite que os usuários aprendam a lógica da programação de maneira acessível e lúdica. Ao criar projetos interativos e divertidos, os usuários desenvolvem habilidades de pensamento computacional, como resolução de problemas, decomposição de tarefas complexas em tarefas menores e algoritmização. Como destaca Resnick (2007,p 14.), um dos criadores do Scratch:

> "O Scratch permite que as pessoas aprendam a pensar criativamente, raciocinar sistematicamente e trabalhar colaborativamente – habilidades essenciais para a vida no século XXI" (Resnick, 2007, p. 14).

Scientific Society Journal
ISSN: 2595-8402
Journal DOI: 10.61411/rsc31879

Além disso, o Scratch ajuda a democratizar o acesso à programação, permitindo que pessoas de todas as idades e níveis de habilidade criem projetos funcionais sem a necessidade de conhecimentos avançados em programação.

Isso pode ter um impacto significativo na formação de novos programadores e no incentivo à diversidade no campo da tecnologia, como destacado por Kafai e Burke (2019): "O Scratch é uma ferramenta para democratizar a programação, tornando-a acessível a um público mais amplo, especialmente para crianças e jovens que, de outra forma, podem não ter acesso a oportunidades de aprendizado formal em programação".

2.5 Conceitos básicos de programação com o Scratch

Segundo Lira e Carvalho, em seu livro "Introdução à Programação de Computadores", publicado em 2012, os conceitos básicos de programação são: variáveis, estruturas de decisão, estruturas de repetição, operadores aritméticos e lógicos, funções e procedimentos. Além disso, é importante entender a lógica de programação, que inclui a capacidade de decompor um problema em partes menores e resolver cada uma delas individualmente.

2.5.1 Variáveis

No Scratch, os blocos que ensinam sobre variáveis são encontrados na categoria "Variáveis". Alguns dos blocos dessa categoria incluem:

•"Definir _ como": usado para criar uma nova variável e atribuir um valor a ela.

•"Mudar _ por _": usado para alterar o valor de uma variável existente, adicionando ou subtraindo um número específico.

•"_": usado para se referir ao valor atual de uma variável em um determinado momento no código.

Segundo Barbosa (2016), uma maneira de aprender variáveis com o Scratch é através da criação de um jogo simples que envolva o uso desses conceitos. Um exemplo é criar um jogo de perguntas e respostas em que a resposta correta é armazenada em

Scientific Society Journal
ISSN: 2595-8402
Journal DOI: 10.61411/rsc31879

uma variável. À medida que o jogador avança no jogo, as perguntas ficam mais difíceis e a variável é atualizada com as novas respostas corretas.

2.5.2 Estruturas de decisão e repetição

No Scratch, os blocos responsáveis pelas estruturas de decisão são os "if" (se) e "if-else" (se não). Com eles, é possível programar uma série de comandos que serão executados somente se uma determinada condição for atendida. Por exemplo, em um jogo, é possível programar uma ação somente se o personagem estiver em determinada posição na tela ou se ele tiver atingido um certo número de pontos.

Os principais blocos do Scratch que ensinam sobre estruturas de repetição são o "repeat" e o "repeat until". O bloco "repeat until" permite que um conjunto de comandos seja repetido até que uma determinada condição seja atendida, enquanto o bloco "repeat" permite que um conjunto de comandos seja repetido um número específico de vezes.

Além disso, o bloco "if" pode ser utilizado em conjunto com a estrutura de repetição para criar condicionais dentro do loop (Scratch, 2023). Os blocos "if" e "if-else" podem ser encontrados na categoria "controle" do Scratch. É importante entender que a estrutura de decisão é uma das bases da programação e é fundamental para a criação de algoritmos mais complexos.

Com esses blocos, é possível criar programas que tomam decisões e agem de maneira diferente de acordo com a situação apresentada (Scratch, 2023). Uma maneira de aprender estruturas de decisão com o Scratch é por meio da criação de jogos simples, como propõe o tutorial "Criando um jogo de perguntas e respostas" do site do Scratch.

Nele, os usuários aprendem a utilizar os blocos de "se", "senão" e "repetir até" para criar o fluxo de perguntas e respostas do jogo (Scratch, 2023). Além disso, é possível utilizar outras ferramentas de apoio, como o "Scratch Cards" da Raspberry Pi Foundation, que traz atividades para ensinar programação com Scratch, incluindo estruturas de decisão (Raspberry Pi Foundation, 2023).

Scientific Society Journal
ISSN: 2595-8402
Journal DOI: 10.61411/rsc31879

2.5.3 Operadores aritméticos e lógicos

Os blocos do Scratch que ensinam sobre operadores aritméticos e lógicos são (Scratch, 2023):

• Blocos aritméticos: os blocos "Somar", "Subtrair", "Multiplicar" e "Dividir" são exemplos de operadores aritméticos que podem ser encontrados na categoria "Operadores" do Scratch.

• Blocos lógicos: os blocos "Igual a", "Maior que", "Menor que", "E" e "Ou" são exemplos de operadores lógicos que podem ser encontrados na categoria "Operadores" do Scratch.

Segundo o próprio Scratch (2023), para aprender sobre operações aritméticas e lógicas no Scratch, é recomendável começar com atividades simples que explorem cada um desses conceitos. Por exemplo, criar um programa que some dois números e exiba o resultado na tela. Para isso, é possível utilizar o bloco "soma", que se encontra na categoria "Operadores".

Já para a aprendizagem sobre operadores lógicos, pode-se criar um programa que verifique se um número é par ou ímpar. Para isso, é possível utilizar o bloco "resto", que retorna o resto da divisão entre dois números, e o bloco "igual a", que compara se o resto é igual a zero. Caso seja igual, o número é par, caso contrário, é ímpar.

Além disso, é possível explorar outros blocos como "multiplicação", "divisão", "subtração" e "maior que", "menor que", "diferente de", entre outros, para desenvolver habilidades de operações aritméticas e lógicas no Scratch. Ao utilizar esses blocos, é possível desenvolver programas que realizam cálculos matemáticos e operações lógicas, permitindo que os usuários compreendam o funcionamento desses operadores na programação (Scratch, 2023).

2.5.4 Funções e procedimentos

Scientific Society Journal
ISSN: 2595-8402
Journal DOI: 10.61411/rsc31879

Para Bezerra (2020), para aprender funções e procedimentos com o Scratch, é recomendado seguir alguns passos básicos. O primeiro é entender o conceito de função, que é um conjunto de comandos que executam uma tarefa específica dentro do programa. Já o procedimento é um bloco de comandos que pode ser executado várias vezes em diferentes partes do programa.

Em relação à utilização do Scratch para aprender esses conceitos, é possível utilizar os blocos específicos para funções e procedimentos na aba "Métodos" na biblioteca de blocos. Também é interessante criar projetos simples que utilizam essas estruturas, como por exemplo um jogo que utiliza uma função para contar pontos ou um programa que executa um procedimento de movimento de um personagem (Scratch, 2023).

Para se aprofundar no assunto, é recomendado estudar as características e possibilidades das funções e procedimentos no Scratch, assim como explorar as diferentes opções de parâmetros e argumentos. Fonte: (BEZERRA, et al., 2020).

3. Metodologia

3.1 Tipo de pesquisa

A pesquisa qualitativa é um tipo de investigação que se concentra na compreensão de fenômenos sociais, culturais e humanos complexos, muitas vezes através da análise de dados não numéricos, como observações, entrevistas, diários, narrativas e documentos (DENZIN; LINCOLN, 2011). Segundo Minayo (2010, p.21), a pesquisa qualitativa é uma "técnica de investigação que se preocupa com a qualidade das informações, isto é, com a compreensão dos fenômenos sociais, culturais e psicológicos que não podem ser quantificados e medidos".

Este trabalho se propõe como uma pesquisa qualitativa, visando levantar e coletar dados sobre pensamento computacional, Scratch e iniciantes na programação e computação para entender as dificuldades dos estudantes novos com entrevista, análise

Scientific Society Journal
ISSN: 2595-8402
Journal DOI: 10.61411/rsc31879

de documentos e artigos relacionados ao tema e, também, revisão bibliográfica de autores antigos e recentes, relacionando os espectros do tema.

Para a entrevista, foi usado um formulário feito pelo Google Forms e enviado aos estudantes de Sistemas de Informação na Universidade do Estado de Minas Gerais pela plataforma do WhatsApp e pessoas com interesse em iniciar na programação.

Houve, também, análise de artigos encontrados sobre os temas "pensamento computacional" e "uso do Scratch em salas de aula", levantando as hipóteses sobre o auxílio da aplicação aos novos estudantes de programação. Os principais artigos analisados são "Computational Thinking" de Jeanette Wing (2006), "Scratch: programming for all" de Mitchell Resnick (2009) e "Pensamento Computacional: uma habilidade essencial no século XXI" de Caroline Hayashi Carvalho Brackmann (2019).

Com isso, o trabalho de conclusão segue uma linha de relacionar o quão útil o Scratch pode ser para novos estudantes de programação, visando diminuir as dificuldades encontradas por esses alunos no que leva à desistência por parte de muitos dos cursos de computação. As limitações são o desconhecimento por parte dos alunos da aplicação Scratch, onde trabalhará para explicar, detalhadamente, o uso e a utilidade da plataforma de programação em blocos.

3.2 Participantes da pesquisa

Os participantes da pesquisa realizada são, em sua maioria, estudantes egressos do curso de Sistemas de Informação da Universidade do Estado de Minas Gerais. Também foram entrevistados pessoas interessadas na área de programação, como quem já trabalha ou tem interesse em ingressar em cursos relacionados.

4. Resultados

4.1 Análise dos dados coletados

Scientific Society Journal
ISSN: 2595-8402
Journal DOI: 10.61411/rsc31879

Tivemos um total de vinte e seis respostas para a entrevista, com a porcentagem das respostas de cada questão.

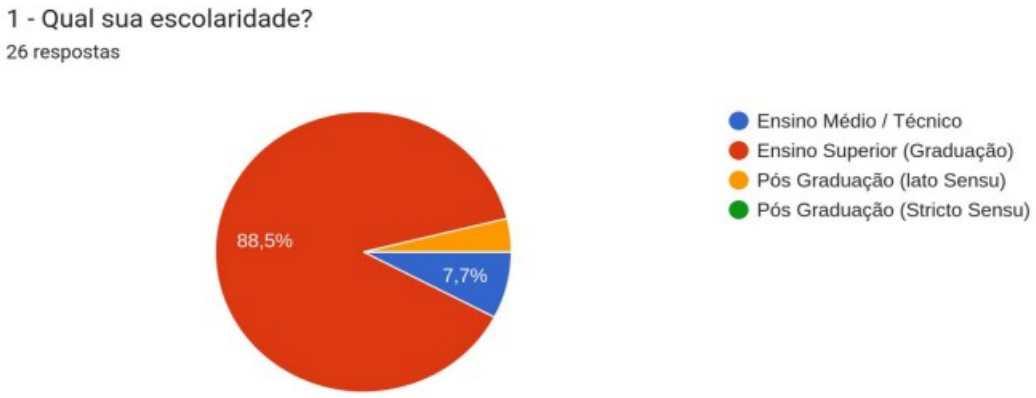

Figura 07 – Escolaridade. Fonte: Dados coletados em entrevista. Gráfico automático do Google Forms.

Figura 08 – Trabalha com Programação/Computação. Fonte: Dados coletados em entrevista. Gráfico automático do Google Forms.

Scientific Society Journal
ISSN: 2595-8402
Journal DOI: 10.61411/rsc31879

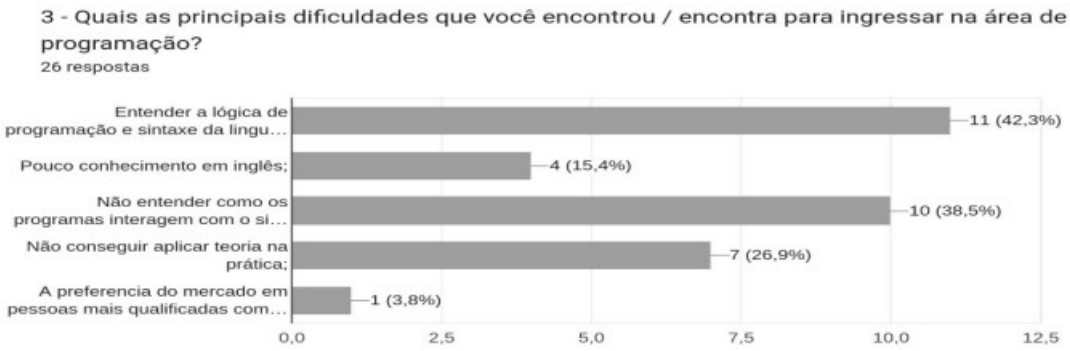

Figura 09 – Questionário. Fonte: Dados coletados em entrevista. Gráfico automático do Google Forms.

Nesta questão, nota-se que a compreensão da lógica de programação, ou seja, a algoritmização, juntamente com a sintaxe das linguagens, são as maiores dificuldades encontradas por estudantes de programação. Destaca-se, também, a não compreensão de como os programas interagem com o sistema operacional e softwares.

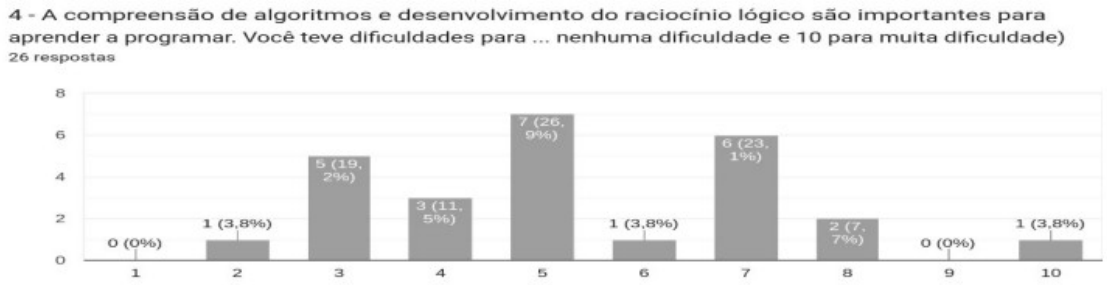

Figura 10 - QuestionárioFonte: Dados coletados em entrevista. Gráfico automático do Google Forms.

Scientific Society Journal
ISSN: 2595-8402
Journal DOI: 10.61411/rsc31879

Figura 11 – Questionário. Fonte: Dados coletados em entrevista. Gráfico automático do Google Forms.

Nota-se que, em sua maioria, os entrevistados tiveram algum tipo de dificuldade no começo do ensino de programação. 38,5% dos entrevistados sentiram-se perdidos em alguns conceitos, mas entenderam a base da programação e 34,6% tiveram mais dificuldades desde os estudos iniciais.

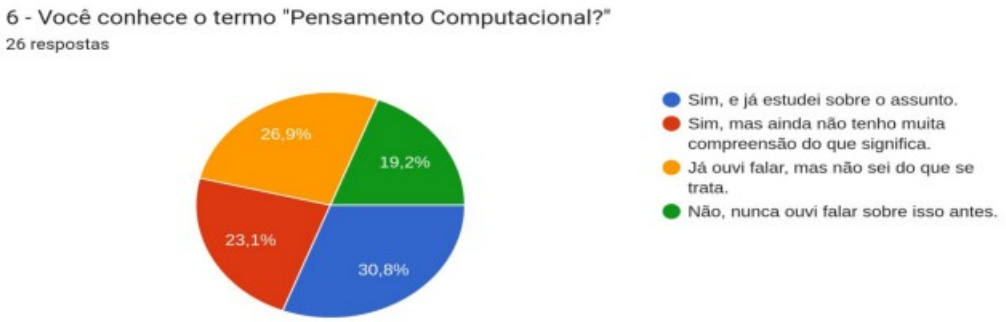

Figura 12 – Questionário. Fonte: Dados coletados em entrevista. Gráfico automático do Google Forms.

Scientific Society Journal
ISSN: 2595-8402
Journal DOI: 10.61411/rsc31879

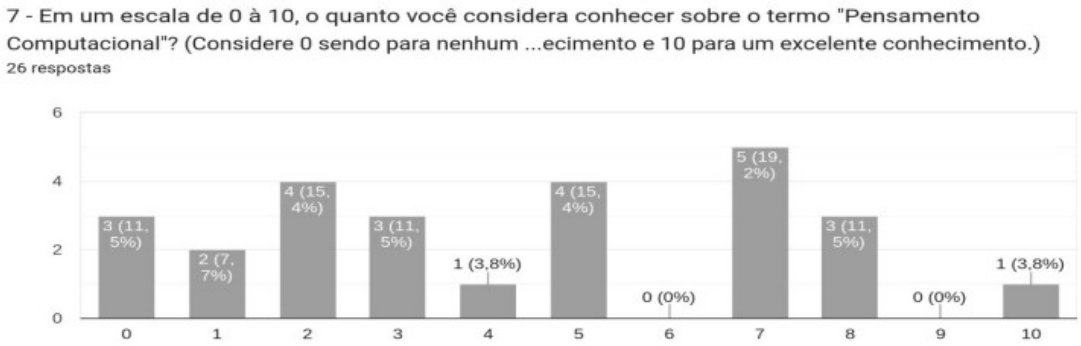

Figura 13 – Questionário. Fonte: Dados coletados em entrevista. Gráfico automático do Google Forms.

Figura 14 – Questionário. Fonte: Dados coletados em entrevista. Gráfico automático do Google Forms.

Nesta questão fica explícito que todos os entrevistados consideram que o pensamento computacional tem, pelo menos, alguma importância nos dias atuais. Mais da metade (53,8%) consideram o pensamento computacional como sendo muito importante na atualidade.

Scientific Society Journal
ISSN: 2595-8402
Journal DOI: 10.61411/rsc31879

Figura 15 – Questionário. Fonte: Dados coletados em entrevista. Gráfico automático do Google Forms.

Figura 16 – Questionário. Fonte: Dados coletados em entrevista. Gráfico automático do Google Forms.

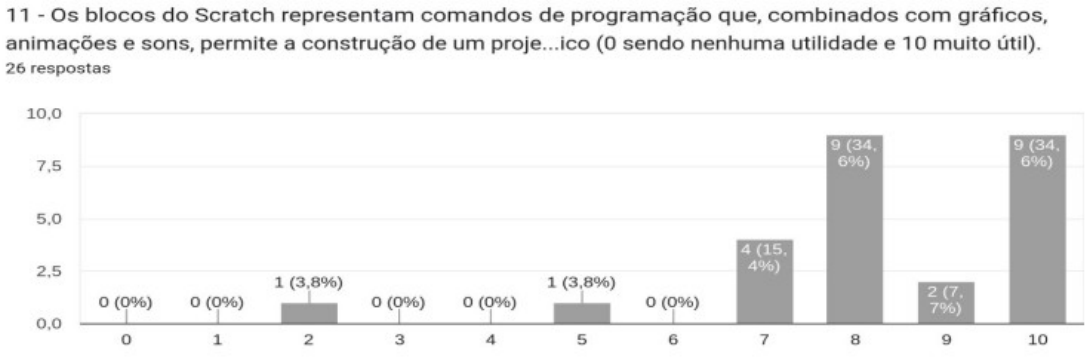

Figura 17 – Questionário. Fonte: Dados coletados em entrevista. Gráfico automático do Google Forms.

Scientific Society Journal
ISSN: 2595-8402
Journal DOI: 10.61411/rsc31879

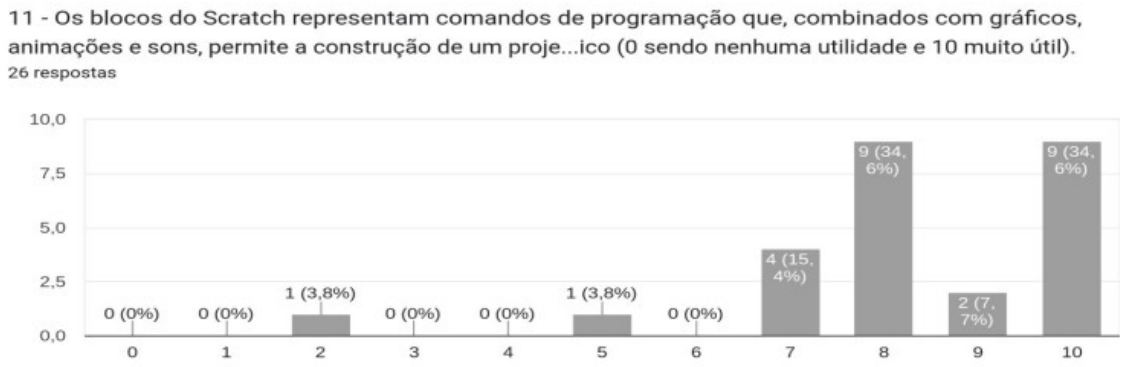

Figura 18 – Questionário. Fonte: Dados coletados em entrevista. Gráfico automático do Google Forms.

Após conhecer brevemente a ferramenta Scratch, metade dos entrevistados concordam que a aplicação talvez possa ser útil no ensino da programação, a depender dos objetivos de uso e perfil do aluno. Já 34,6% consideram a ferramenta como ótima no auxílio inicial.

Figura 19 - QuestionárioFonte: Dados coletados em entrevista. Gráfico automático do Google Forms.

Nessa questão, é perguntado se o Scratch pode auxiliar em habilidades essenciais para programação. Metade dos entrevistados acreditam que a ferramenta é ótima para o desenvolvimento das habilidades, enquanto 30,8% responderam que talvez possa ajudar, mas não sendo a principal finalidade.

Scientific Society Journal
ISSN: 2595-8402
Journal DOI: 10.61411/rsc31879

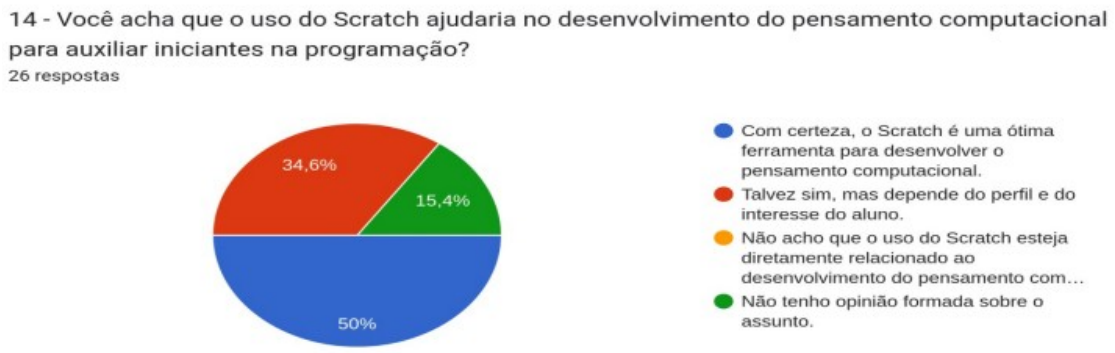

Figura 20- Questionário. Fonte: Dados coletados em entrevista. Gráfico automático do Google Forms.

Por fim, a metade dos entrevistados consideram o Scratch uma ótima ferramenta para o desenvolvimento do pensamento computacional, 34,6% responderam que depende do perfil e interesse do aluno e 15,4% não tem opinião formada sobre o assunto.

Em sua maioria, os entrevistados julgaram o Scratch como uma aplicação positiva para o desenvolvimento do pensamento computacional, ou seja, a melhora no raciocínio lógico e algoritmização para que a introdução em programação tenha menos dificuldades e desistências.

6. Conclusão

6.1 síntese dos principais resultados encontrados

A conclusão a que se pode chegar, ao final deste trabalho é que o pensamento computacional surge como uma simples, porém com trabalhoso desenvolvimento abordagem para a melhora na introdução dos egressos em programação. O Scratch foi inventado e é visto como uma das principais ferramentas para o desenvolvimento do pensamento computacional, já que suas funcionalidades são baseadas em blocos de montagem que necessitam de uma lógica para funcionar.

Por fim, é importante ressaltar que o Scratch surge como uma opção para o desenvolvimento do pensamento computacional e de suas principais características,

Scientific Society Journal
ISSN: 2595-8402
Journal DOI: 10.61411/rsc31879

anteriormente citadas neste trabalho. Assim sendo, o aprendizado do Scratch em si é um modo de desenvolver o pensamento computacional, o raciocínio lógico e a algoritmização das coisas. Essas habilidades bem desenvolvidas significam uma melhor capacitação dos novos programadores e um caminho mais fácil no aprendizado de novas linguagens de programação.

Na educação, o pensamento computacional pode ser utilizado para desenvolver habilidades de resolução de problemas, raciocínio lógico, criatividade e colaboração. Segundo Giannakos, Jaccheri e Krogstie (2015), o uso do pensamento computacional no ensino pode ajudar os estudantes a se prepararem melhor para as habilidades exigidas pelo mercado de trabalho. Além disso, as habilidades desenvolvidas pelo pensamento computacional podem ser aplicadas em outras disciplinas e áreas de estudo, permitindo que os alunos tenham uma formação mais completa. Com isso, é possível afirmar que este TC introduz uma possibilidade de capacitação para, sobretudo, egressos na programação.

Alguns estudos destacam que o Scratch oferece um ambiente de programação acessível e amigável, permitindo que alunos sem conhecimentos prévios em programação desenvolvam suas habilidades de resolução de problemas, raciocínio lógico e criatividade (Kelleher & Pausch, 2005; Resnick et al., 2009).Além disso, o Scratch tem sido apontado como uma ferramenta que pode contribuir para o desenvolvimento de uma geração de jovens criadores de tecnologia, que utilizam suas habilidades de programação para construir suas próprias ferramentas e soluções tecnológicas.

Segundo Papert (1993), o Scratch pode ser visto como um ambiente para a criação de novos mundos e não apenas para a resolução de problemas. Ele argumenta que a aprendizagem da programação pode ter um impacto significativo na formação de habilidades importantes, tais como a capacidade de pensar de forma crítica e a criatividade, que podem ser aplicadas em diversas áreas da vida.

Scientific Society Journal
ISSN: 2595-8402
Journal DOI: 10.61411/rsc31879

As principais limitações deste trabalho foram a falta de estudos longitudinais que permitam analisar a eficácia da ferramenta em longo prazo. Segundo Resnick et al. (2014), "mais pesquisas são necessárias para determinar a eficácia do Scratch na promoção do pensamento computacional em longo prazo". Além disso, segundo Grover e Pea (2013), "ainda não existe uma definição clara e consensual sobre o que é pensamento computacional, o que torna difícil a sua avaliação", apesar de muitos autores relatarem as características principais do pensamento computacional.

Este trabalho pode contribuir para futuros estudos ao explorar mais profundamente como a utilização do Scratch pode impactar na aprendizagem de programação e no desenvolvimento do pensamento computacional em diferentes grupos de alunos e em diferentes contextos educacionais. Além disso, estudos futuros podem se concentrar em explorar as melhores práticas para a integração do Scratch no currículo de educação em informática, incluindo como avaliar e medir os resultados da aprendizagem com o uso do Scratch.

Outra possibilidade de estudo futuro seria explorar o uso do Scratch em outras áreas, além da educação em informática, como a matemática, ciências e artes. O Scratch pode ser utilizado para a criação de projetos interdisciplinares, que envolvam diferentes áreas do conhecimento, e futuras pesquisas podem investigar como essa abordagem pode contribuir para a aprendizagem dos alunos e para o desenvolvimento de habilidades como criatividade, pensamento crítico e colaboração.

7. Declaração de direitos

Scientific Society Journal
ISSN: 2595-8402
Journal DOI: 10.61411/rsc31879

8. Referências

1. AHO, A. V.; HOPCROFT, J. E.; ULLMAN, J. D. Estruturas de Dados e Algoritmos. Rio de Janeiro: LTC, 1983.
2. MIT Scratch. Disponível em: https://scratch.mit.edu/. Acesso em: 03/04/2023.
3. SEDGEWICK, R. & WAYNE, K. Algorithms 4th Edition, 2021. Disponível em: <https://algs4.cs.princeton.edu/home/>. Acesso em: 03/04/2023.
4. ALMEIDA, M. B.; MENDES, L. L. Pensamento computacional na escola: conceitos e práticas. São Paulo: Blucher, 2018.
5. ALMSTRUM, V. L., & Collella, V. J. (2014). Seeking computational thinking in information technology education. Communications of the ACM, 57(9), 26–28. https://doi.org/10.1145/2641568.
6. ALMSTRUM, V., et al. Computing for everyone: Improving global computer science education with a focus on diversity. ACM Inroads, Nova York, v. 5, n. 2, p. 18-23, 2014.
7. BARR, V.; STEPHENSON, C. Bringing computational thinking to K-12: what is involved and what is the role of the computer science education community? ACM Inroads, v. 2, n. 1, p. 48-54, 2011.
8. RESNICK, Mitchel et al. Scratch: Programming for All. Communications of the ACM, v. 52, n. 11, p. 60-67, Nov. 2009. DOI: https://doi.org/10.1145/1592761.1592779.
9. RESNICK, M. All I Really Need to Know (About Creative Thinking) I Learned (By Studying How Children Learn) in Kindergarten. ACM SIGCHI Bulletin, v. 39, n. 3, p. 13-16, 2007.
10. BRACKMANN, C. H. C. Pensamento Computacional: uma habilidade essencial no século XXI. Revista Brasileira de Computação Aplicada, v. 11, n. 2, p. 50-60, 2019.

Scientific Society Journal
ISSN: 2595-8402
Journal DOI: 10.61411/rsc31879

11. BRACKMANN, C. Algoritmização no Ensino de Programação. 2016. Disponível em: http://www.inf.ufsm.br/~cbrackmann/alg-ensino-programacao.pdf. Acesso em: 23/03/2023.
12. BRACKMANN, C. et al. Computação desplugada: estratégias de ensino e recursos didáticos para o desenvolvimento do pensamento computacional. Revista de Informática Teórica e Aplicada, v. 25, n. 3, p. 31-44, 2018.
13. CORMEN, Thomas H. et al. Algoritmos: Teoria e Prática. 3° edição. Gen Ltc, 2012
14. Rosen, Y., Beck-Hill, D., & Lei, P. W. (2013). The impact of Scratch programming on seventh grade students' spatial reasoning skills. Computers & Education, 67, 193-204.
15. CORRÊA, E. F. F.. Jogos eletrônicos e o desenvolvimento do pensamento computacional: um estudo de caso. 2014. 79 f. Monografia (Graduação) - Curso de Ciência da Computação, Universidade de Brasília, 2017.
16. DENZIN, N. K.; LINCOLN, Y. S. O planejamento da pesquisa qualitativa: teorias e abordagens. 2. ed. Porto Alegre: Artmed, 2011.
17. DROMEY, R. G. How to Solve It by Computer. Prentice-Hall, Inc., 1996.
18. GAZZANIGA, M. S. O cérebro ético: as bases biológicas da moralidade. Artmed, 2018.
19. GIANNAKOS, M. N.; JACCHERI, L.; KROGSTIE, J. The Potential of Using Computational Thinking Skills to Enhance Learning. Education and Information Technologies, v. 20, n. 4, p. 679-693, 2015.
20. GRIES, D.; SCHNEIDER, F. B. A Logical Approach to Discrete Math. New York: Springer-Verlag, 1993.
21. GROVER, S., & Pea, R. (2013). Computational Thinking in K–12: A Review of the State of the Field. Educational Researcher, 42(1), 38–43. https://doi.org/10.3102/0013189x12463051

Scientific Society Journal
ISSN: 2595-8402
Journal DOI: 10.61411/rsc31879

22. GROVER, S., PEA, R. Computational thinking in K-12: A review of the state of the field. Educational Researcher, Thousand Oaks, v. 42, n. 1, p. 38-43, 2013.
23. ISTE. Computational Thinking: An Overview. Disponível em: https://www.iste.org/docs/ct-documents/ct-overview-and-features.pdf?sfvrsn=10e803e4_1. Acesso em: 23/03/2023.
24. KAFAI, Y. B. "Mindstorms as multimedia learning environments: Examining the potential of video editing software for supporting girls' science learning." Journal of the Learning Sciences, vol. 13, no. 3, pp. 403-446, 2014.
25. KAFAI, Y. B.; BURKE, Q. Co-Designing Creative Learning Futures: Intersections of Computer Science Education and the Learning Sciences. Educational Technology & Society, v. 22, n. 3, p. 1-4, 2019.
26. KAFAI, Y. & Burke, Q. (2014). Connected Gaming: What Making Video Games Can Teach Us about Learning and Literacy. The MIT Press.
27. KAFAL, Y.; BURKE, Q. Diversifying Access to Computer Science: The Role of the Scratch Programming Environment. In: ACM Transactions on Computing Education, v. 13, n. 1, p. 1-20, 2013.
28. KELLEHER, C., & Pausch, R. (2005). Lowering the barriers to programming: A taxonomy of programming environments and languages for novice programmers. ACM Computing Surveys (CSUR), 37(2), 83-137.
29. KELLEHER, C. & Pausch, R. (2005). Lowering the barriers to programming: a taxonomy of programming environments and languages for novice programmers. ACM Computing Surveys (CSUR), 37(2), 83-137.
30. KIRKPATRICK, D., et al. App Inventor for Android: A hands-on guide to building your own Android apps. O'Reilly Media, Inc., Sebastopol, 2014.
31. KÖLLING, M. The Greenfoot programming environment. Journal of Visual Languages & Computing, Amsterdam, v. 14, n. 4, p. 299-312, 2003.
32. KNUTH, D. E. Dancing links. arXiv.org, Ithaca, 2000. Disponível em: <https://arxiv.org/abs/cs/0011047>. Acesso em: 03/04/2023.

33. LAM, Eric; SHIRAI, Yuichi; YAO, Yongdong. "A review of object recognition approaches." ACM Computing Surveys (CSUR), v. 50, n. 4, 2018.
34. RAABE, A. L. A.; BRACKMANN, C. P.; CAMPOS, F. R. Currículo de referência em tecnologia e computação. CIEB, 2018. Disponível em:https://curriculo.cieb.net.br/assets/docs/Curriculo_de_Referencia_em_Tecnologia _e_Computacao.pdf>. Acesso em: 23/03/2023.
35. MALONEY, J. et al. The Scratch Programming Language and Environment. ACM Transactions on Computing Education (TOCE), v. 10, n. 4, p. 16, 2010.
36. MALONEY, J.; PEPPLER, K.; KAFI, Y.; RESNICK, M.; RABELOTTI, R. Programming by Choice: Urban Youth Learning Programming with Scratch. In: SIGCSE '10: Proceedings of the 41st ACM Technical Symposium on Computer Science Education, Milwaukee, Wisconsin, USA, March 10-13, 2010.
37. MINAYO, M. C. S. O desafio do conhecimento: pesquisa qualitativa em saúde. 11. ed. São Paulo: Hucitec, 2010.
38. OLIVEIRA, E. C.; GOMES, M. L. Desenvolvendo o pensamento computacional por meio da resolução de problemas matemáticos. Anais do XVI Simpósio Brasileiro de Informática na Educação, Recife, 2017.
39. PAPERT, S. Mindstorms: crianças, computadores e aprendizagem. 4. ed. Porto Alegre: Artmed, 1994.
40. RASPBERRY Pi Foundation. Scratch Cards. Disponível em: https://projects.raspberrypi.org/en/projects/scratch-cards. Acesso em: 24/04/2023.
41. RESNICK, Mitchel. Revitalizing kindergarten: A project-based curriculum. New York: National Science Foundation, 2013.
42. RESNICK, M. et al. Scratch: programming for all. Communications of the ACM, v. 52, n. 11, p. 60-67, 2009.

43. RIBEIRO, M. C. B. et al. O Pensamento Computacional na Educação. In: SBSI - Simpósio Brasileiro de Sistemas de Informação, 2017, Florianópolis. Anais do XVII Simpósio Brasileiro de Sistemas de Informação, 2017.
44. ROSENTHAL, R.; ROSNOW, R. L.; RUBIN, D. B. Contrasts and effect sizes in behavioral research: a correlational approach. Cambridge University Press, 2000.
45. SANTOS, L. M. dos; MACHADO, G. F. Introdução à Programação: Conceitos, Algoritmos e Linguagens. Elsevier, 2019.
46. SCRATCH DAY. Disponível em: https://day.scratch.mit.edu/. Acesso em: 18/04/2023.
47. WING, J. M. Computational thinking. Communications of the ACM, v. 49, n. 3, p. 33-35, 2006. DOI: https://doi.org/10.1145/1118178.1118215.