

UNIVERSIDADE FEDERAL DA GRANDE DOURADOS
FACULDADE DE CIÊNCIAS EXATAS E TECNOLOGIA
TRABALHO DE CONCLUSÃO DE CURSO
BACHARELADO EM ENGENHARIA DE COMPUTAÇÃO

ATIVIDADES PLUGADAS PARA O ENSINO DO PENSAMENTO
COMPUTACIONAL: UMA ABORDAGEM TAXONÔMICA ALINHADA À
BNCC

Aroldo Rodas Miranda Filho

Orientadora Prof^a Rosenilda Marques da Silva Felipe

DOURADOS – MS

2024

UNIVERSIDADE FEDERAL DA GRANDE DOURADOS

AROLDO RODAS MIRANDA FILHO

**ATIVIDADES PLUGADAS PARA O ENSINO DE PENSAMENTO
COMPUTACIONAL: UMA ABORDAGEM TAXONÔMICA ALINHADA À
BNCC**

Trabalho de Conclusão de Curso de
graduação apresentado para obtenção do
título de Bacharel em Engenharia de
Computação.

Faculdade de Ciências Exatas e Tecnologia
Universidade Federal da Grande Dourados

**Orientadora: Me Rosenilda Marques da
Silva Felipe**

DOURADOS - 2024

2024

AROLDO RODAS MIRANDA FILHO

ATIVIDADES PLUGADAS PARA O ENSINO DE PENSAMENTO COMPUTACIONAL: UMA ABORDAGEM TAXONÔMICA ALINHADA À BNCC

Trabalho de Conclusão de Curso apresentado à Faculdade de Ciências Exatas e Tecnologia da Universidade Federal da Grande Dourados, como requisito à obtenção do título de Bacharel em Engenharia de Computação.

Dourados, 19 de Novembro de 2024.

BANCA EXAMINADORA

Orientadora: Me. Rosenilda Marques da Silva Felipe
FACET - UFGD

Prof. Me. Rodrigo Porfirio da Silva Sacchi
FACET - UFGD

Profa. Dra. Claudia Regina Tinos Peviani
FACET - UFGD

RESUMO

O presente trabalho aborda o desenvolvimento de uma taxonomia voltada ao ensino do Pensamento Computacional na Educação Básica, em alinhamento com as diretrizes da Base Nacional Comum Curricular (BNCC). Dada a crescente relevância do Pensamento Computacional na formação de habilidades cognitivas e técnicas, esta pesquisa visa preencher lacunas relacionadas à ausência de materiais didáticos específicos para o ensino de computação. A metodologia inclui uma Revisão Sistemática da Literatura, pesquisa documental na BNCC e análise de conteúdo de sites e artigos educacionais para identificar as principais ferramentas plugadas utilizadas na educação, como *Scratch* e *Python*, e suas aplicações. Para estruturar esse processo de forma eficaz, segundo Pressman (2009), existem sete etapas para a realização da Engenharia de Requisitos. O escopo deste trabalho contempla as duas primeiras etapas, sendo elas: concepção, que tem como objetivo estabelecer uma compreensão inicial do problema, identificando as partes interessadas e a natureza da solução almejada, identificando os objetivos, as funcionalidades e as necessidades requeridas pelos usuários. A taxonomia criada classifica atividades plugadas com base nas habilidades definidas pela BNCC, organizando-as de maneira hierárquica e progressiva, desde os primeiros anos do ensino fundamental até o ensino médio. Os resultados indicam que essa estrutura facilita a aprendizagem de conceitos de abstração, algoritmos, decomposição de problemas e reconhecimento de padrões, promovendo o desenvolvimento gradual das competências computacionais. A proposta deste trabalho contribui para a Engenharia de Requisitos ao organizar sistematicamente as atividades, auxiliando educadores na seleção de práticas pedagógicas alinhadas aos objetivos da BNCC, além de fornecer uma base estruturada para desenvolvedores interessados em criar novos recursos e materiais didáticos voltados ao ensino do Pensamento Computacional. Dessa forma, o trabalho proporciona aos alunos uma formação mais adequada para os desafios da era digital.

Palavras-chave: Pensamento Computacional; Computação Plugada; BNCC; Taxonomia; Engenharia de Requisitos.

ABSTRACT

This paper addresses the development of a taxonomy aimed at teaching Computational Thinking in Basic Education, in alignment with the guidelines of the National Common Curricular Base (BNCC). Given the growing relevance of Computational Thinking in the formation of cognitive and technical skills, this research aims to fill gaps related to the lack of specific teaching materials for teaching computing. The methodology includes a Systematic Literature Review, documentary research in the BNCC and content analysis of educational websites and articles to identify the main plug-in tools used in education, such as Scratch and Python, and their applications. To structure this process effectively, according to Pressman (2009), there are seven steps to carry out Requirements Engineering. The scope of this paper includes the first two steps, which are: conception, which aims to establish an initial understanding of the problem, identifying the stakeholders and the nature of the desired solution, identifying the objectives, functionalities and needs required by users. The taxonomy created classifies plug-in activities based on the skills defined by the BNCC, organizing them in a hierarchical and progressive manner, from the first years of elementary school to high school. The results indicate that this structure facilitates the learning of concepts of abstraction, algorithms, problem decomposition, and pattern recognition, promoting the gradual development of computational skills. The proposal of this work contributes to Requirements Engineering by systematically organizing the activities, helping educators in the selection of pedagogical practices aligned with the objectives of the BNCC, in addition to providing a structured basis for developers interested in creating new resources and teaching materials aimed at teaching Computational Thinking. In this way, the work provides students with a more appropriate education for the challenges of the digital age.

Palavras-chave: *Computational Thinking; Plug-in Computing; BNCC; Taxonomy; Requirements Engineering.*

SUMÁRIO

1. INTRODUÇÃO	10
2. REFERENCIAL TEÓRICO	12
3. METODOLOGIA	15
4. RESULTADOS E DISCUSSÕES	18
4.1 Pesquisa Documental na BNCC	18
4.2 Levantamento de Atividades Plugadas	19
4.3 Elaboração de Atividades Plugadas	26
4.4 Elaboração da Taxonomia Hierárquica	62
5. CONSIDERAÇÕES FINAIS	69
REFERÊNCIAS	71
APÊNDICE A - Exemplo de código Reco. EF06CO01 - Atividade #01	73
APÊNDICE B - Exemplo de código Alg. EF06CO02 - Atividade #01	74
APÊNDICE C - Exemplo de código Alg. EF06CO03 - Atividade #01	76
APÊNDICE D - Exemplo de código Decomp. EF06CO04 - Atividade #01	77
APÊNDICE E - Exemplo de código Abstra. EF07CO01 - Atividade #01	79
APÊNDICE F - Exemplo de código Abstra. EF07CO03 - Atividade #01	81
APÊNDICE G - Exemplo de código Abstra. EF07CO05 - Atividade #01	82
APÊNDICE H - Exemplo de código Abstra. EF08CO01 - Atividade #01	84
APÊNDICE I - Exemplo de código Decomp. EF08CO02 - Atividade #01	85
APÊNDICE J - Exemplo de código Decomp. EF08CO04 - Atividade #01	86
APÊNDICE K - Exemplo de código Abstra. EF09CO01 - Atividade #01	87

LISTA DE FIGURAS

Figura 1 - Rotas do Avião: Cenário A	20
Figura 2 - Rotas do Avião: Cenário B	21
Figura 3 - Telas do sistema de Ordenação Web	22
Figura 4 - Jogo Pac-Man	23
Figura 5 - Jogo Pac-man no Scratch	24
Figura 6 – Bloco definições iniciais em um dos fantasmas	24
Figura 7 - Código de um dos fantasmas	25
Figura 8 - Código da pontuação	25
Figura 9 - Atividade EF01CO01	27
Figura 10 - Acerto atividade EF01CO01	27
Figura 11 - Atividade EF01CO02	28
Figura 12 - Acerto atividade EF01CO02	28
Figura 13 - Música base EF01CO03	29
Figura 14 - Atividade EF01CO03	29
Figura 15 - Acerto da atividade EF01CO03	30
Figura 16 - Insistência de erro	30
Figura 18 - Atividade EF02CO01	31
Figura 19 - Falha na atividade EF02CO01	32
Figura 20 - Atividade EF02CO01 Parabéns	32
Figura 21 - Atividade EF02CO02	33
Figura 22 - Acerto Atividade EF02CO02	33
Figura 23 - Atividade EF03CO01	34
Figura 24 - Atividade EF03CO01	34
Figura 25 - Final Atividade EF03CO01	35
Figura 26 - Atividade EF03CO02	35
Figura 27 - Avanço atividade EF03CO02	36
Figura 28 - Atividade EF03CO03	36

Figura 29 - Avanço atividade EF03CO03	37
Figura 30 - Exemplo habilidade EF04CO01	38
Figura 31 - Atividade EF04CO01	39
Figura 32 - Direções EF04CO01	39
Figura 33 - Andar atividade EF04CO01	40
Figura 34 - Atividade EF04CO02	41
Figura 35 - Documentos Atividade EF04CO02	41
Figura 36 - Recuperação de dados por nascimento EF04CO02	42
Figura 37 - Atividade EF04CO03	43
Figura 38 - Parte 2 Atividade EF04CO03	43
Figura 39 - Atividade EF05CO01	44
Figura 40 - Atividade EF05CO02	45
Figura 41 - Atividade EF05CO03	46
Figura 42 - Atividade controle EF05CO03	47
Figura 43 - Início Atividade EF05CO03	47
Figura 44 - Congratulação EF05CO03	48
Figura 45 - Tente novamente EF05CO03	49
Figura 46 - Figura geométrica EF06CO04 e EF06CO06	53
Figura 47 - Figura geométrica 2 EF06CO04 e EF06CO06	54
Figura 48 - Composição da classe Principal	63
Figura 49 - Exemplo da composição da classe Abstração	64
Figura 50 - Classificação das atividades Abstração	65
Figura 51 - Classificação das atividades Algoritmo	66
Figura 52 - Classificação das atividades Decomposição de Problemas	67
Figura 53 - Classificação das atividades Reconhecimento de Padrões	68

LISTA DE TABELAS

Tabela 1 - Organização da BNCC Computação	19
Tabela 2 - Atividade EF06CO01	50
Tabela 3 - Atividade EF06CO02	51
Tabela 4 - Atividade EF06CO03 e EF06CO05	52
Tabela 5 - Atividade EF07CO01	55
Tabela 6 - Atividade EF07CO03	56
Tabela 7 - Atividade EF07CO05	57
Tabela 8 - Atividade EF08CO01	59
Tabela 9 - Atividade EF08CO02	60
Tabela 10 - Atividade EF08CO04 e EF09CO02	61
Tabela 11 - Atividade EF09CO01	62

1. INTRODUÇÃO

O pensamento computacional tornou-se uma competência essencial, não apenas para especialistas em ciência da computação, mas para todas as pessoas, pois explora como o uso de conceitos e processos computacionais pode ampliar o poder cognitivo humano. Esse conceito foi incluído como um dos eixos temáticos da Base Nacional Comum Curricular (BNCC), documento que define o conjunto de aprendizagens essenciais para todas as fases da Educação Básica, aprovado entre 2017 e 2018.

Em 2022, a BNCC foi revisada e passou a contar com um documento complementar chamado BNCC Computação, que define diretrizes específicas para o ensino de computação na educação básica. Nesse documento, o Ministério da Educação (2022) detalha as habilidades que devem ser desenvolvidas ao longo da educação básica, organizadas em três eixos temáticos: Cultura Digital, Mundo Digital e Pensamento Computacional.

Segundo Kenski (2018, p. 139), a Cultura Digital é “um termo novo, atual, emergente e temporal”. A expressão integra “perspectivas diversas vinculadas às inovações e aos avanços nos conhecimentos, e à incorporação deles, proporcionados pelo uso das tecnologias digitais e as conexões em rede para a realização de novos tipos de interação, comunicação, compartilhamento e ação na sociedade”.

O Mundo Digital se refere ao conjunto de habilidades que envolvem a interação e participação ativa dos estudantes em um ambiente cada vez mais mediado por tecnologias digitais, não se limitando ao domínio técnico das tecnologias, mas também compreendendo a importância de saber lidar com informações digitais, interpretar dados, reconhecer os impactos das tecnologias na vida social e entender as transformações culturais impulsionadas pela digitalização.

Já o eixo denominado de Pensamento Computacional, segundo Wing (2006), é aquele que engloba a resolução de problemas, o desenvolvimento de sistemas e a compreensão do comportamento humano, por meio da aplicação de conceitos fundamentais da ciência da computação. Esse tipo de pensamento utiliza diversas estratégias mentais que capturam a ampla complexidade da ciência da computação. Pensar e modelar um problema de maneira que um computador seja capaz de resolver,

utilizando conceitos de decomposição, reconhecimento de padrões, abstração e algoritmos.

Para ensinar o pensamento computacional de maneira abrangente e inclusiva, é possível utilizar dois tipos de abordagens: as ferramentas plugadas e desplugadas. As ferramentas plugadas, foco deste trabalho, são dispositivos eletrônicos conectados à internet, como computadores e tablets, que permitem aos alunos explorar conceitos de programação e simulações digitais.

Já as ferramentas desplugadas utilizam recursos físicos, como jogos de tabuleiro e atividades práticas, para promover o entendimento dos princípios fundamentais da computação de maneira analógica. Com a homologação da BNCC e sua implementação, surgiram alguns desafios, como por exemplo a necessidade de formação de professores e a disponibilização de materiais didáticos adequados às necessidades brasileiras (Guarda, 2022).

Visando contribuir para solucionar o problema de ausência de materiais, este trabalho tem o objetivo de desenvolver uma taxonomia que classifique as atividades plugadas comprometidas com o ensino do Pensamento Computacional de acordo com as habilidades contidas na BNCC. Conforme Sommerville (2011), a construção de uma taxonomia é uma etapa essencial na elicitação de requisitos formais.

Essa taxonomia organiza sistematicamente as atividades plugadas, facilitando o entendimento das competências a serem desenvolvidas e orientando a criação de materiais didáticos alinhados às demandas curriculares. Ao estruturá-las conforme as competências da BNCC, a taxonomia permite a identificação de lacunas no conteúdo, orienta o desenvolvimento de novos recursos e melhora a comunicação entre educadores e por fim, pode servir como requisitos formais para desenvolvedores que queiram implementar um sistema com essas informações. Dessa forma, esta taxonomia corrobora as ações governamentais que preveem o ensino do Pensamento Computacional no currículo nacional da educação básica.

2. REFERENCIAL TEÓRICO

Segundo Wing (2006), o pensamento computacional é um conjunto de habilidades cognitivas que inclui a capacidade de decompor problemas complexos, reconhecer padrões, abstrair informações e criar algoritmos. Para Liukas (2015), pensar sobre os problemas de uma forma que permita a compreensão de computadores para resolvê-los é algo que as pessoas fazem, não os computadores.

A capacidade de decompor problemas envolve quebrar um problema complexo em problemas menores e mais gerenciáveis, facilitando o entendimento de problemas complexos, abordando cada subproblema de maneira independente, simplificando a solução geral. “O processo através do qual os problemas são resolvidos em suas partes menores” (Liukas, 2015).

O reconhecimento de padrões consiste em “encontrar semelhanças e padrões para resolver problemas complexos com mais eficiência” (Liukas, 2015), permitindo a reutilização de métodos ou soluções para problemas similares, facilitando a previsão de comportamentos futuros com base em padrões identificados.

Para Liukas (2015) a abstração é “o processo de separar detalhes que não são necessários para se concentrar nas coisas que são necessárias”, isso envolve a redução da complexidade, visando os aspectos mais importantes de um problema, descartando informações irrelevantes, simplificando sua modelagem. Segundo Cormen et al. (2012), um algoritmo consiste em criar uma sequência de instruções finitas para a solução de um problema ou a realização de uma tarefa, permitindo uma implementação de forma eficiente e eficaz, facilitando a automação de tarefas.

Nesse contexto, as ferramentas plugadas oferecem ambientes interativos que favorecem a compreensão de conceitos abstratos por meio de práticas lúdicas e visuais, permitindo que o aluno desenvolva suas habilidades de maneira envolvente e acessível. Essas ferramentas abstraem a dificuldade relacionada à sintaxe das linguagens de programação, permitindo o foco total nos conceitos. Resnick et al. (2009) destacam que o *Scratch*¹, uma dessas ferramentas, facilita a aprendizagem de programação ao utilizar uma interface visual e interativa, que promove a experimentação e a criatividade sem a barreira da sintaxe complexa.

¹ <https://scratch.mit.edu/>

Ferramentas como *Scratch*, *Blockly*², *Code.org*³ e *App Inventor*⁴, utilizam interfaces visuais baseadas em blocos, simplificando a programação eliminando a necessidade de digitar códigos. Resnick et al. (2009) destaca que essas ferramentas são projetadas para incentivar a aprendizagem criativa, permitindo que os alunos criem projetos interativos, como histórias animadas, jogos e simulações, tornando a programação acessível e divertida, estimulando o interesse dos alunos.

Embora essas ferramentas visuais ofereçam uma abordagem simplificada e acessível à programação, linguagens textuais como *Python* também têm conquistado grande popularidade devido à sua sintaxe intuitiva e legível. Para Borges (2021, p. 23), uma das razões para a crescente popularidade da linguagem *Python* é sua sintaxe simples e legível, com comandos que se assemelham ao inglês, o que torna a codificação mais fácil e eficiente em comparação com outras linguagens de programação.

Assim como a acessibilidade e a simplicidade são fatores importantes na escolha de linguagens e ferramentas para o ensino de programação, esses aspectos também desempenham um papel essencial na Engenharia de Requisitos, etapa crucial no processo de desenvolvimento de software, onde o analista de sistemas ou engenheiro de requisitos busca entender as verdadeiras necessidades dos usuários em relação à solução do problema em questão. Nessa fase, são realizadas diversas atividades para identificar, analisar, documentar e validar os requisitos e os artefatos gerados (Kotonya; Sommerville, 1996).

Para estruturar esse processo de forma eficaz, segundo Pressman (2009), existem sete etapas para realização da Engenharia de Requisitos. O escopo deste trabalho contempla as duas primeiras etapas, sendo a primeira concepção, que tem como objetivo estabelecer uma compreensão inicial do problema, identificando as partes interessadas e a segunda concepção, sendo a natureza da solução almejada, que busca indentificar os objetivos, as funcionalidades e as necessidades requeridas pelos usuários.

² <https://developers.google.com/blockly?hl=pt-br>

³ <https://code.org/>

⁴ <https://appinventor.mit.edu/>

A organização das informações desempenha um papel essencial nesse processo, e no contexto deste trabalho, será realizada por meio de uma Taxonomia, que é uma estrutura que permite alocar, recuperar e comunicar informações dentro de um sistema de maneira lógica através de navegação. As taxonomias, enquanto estruturas de classificação, visam promover a organização intelectual em um determinado contexto. Nesse sentido, elas variam conforme o tipo de organização e as informações que buscam representar (Campos; Gomes, 2008).

A análise temática foi o método escolhido para explorar e interpretar os dados qualitativos coletados, com o objetivo de identificar padrões significativos relacionados ao ensino do pensamento computacional no contexto da BNCC. Conforme descrito por Braun e Clarke (2006), a análise temática é um método flexível que permite organizar e descrever dados em detalhes, além de promover a interpretação de significados subjacentes. No presente trabalho, essa abordagem foi empregada com uma perspectiva dedutiva, partindo de categorias pré-definidas baseadas nas competências da BNCC para o ensino de computação.

O processo de análise temática seguiu as seis fases propostas por Braun e Clarke (2006): familiarização com os dados, geração de códigos iniciais, busca por temas, revisão dos temas, definição e nomeação dos temas, e produção do relatório final. A familiarização com os dados foi realizada por meio de uma leitura ativa e repetida do material, permitindo o reconhecimento inicial de padrões e conteúdos relevantes. Em seguida, os dados foram sistematicamente codificados, com a identificação de segmentos que representassem aspectos significativos para o objetivo da pesquisa.

Na etapa de busca por temas, os códigos foram agrupados em categorias maiores, refletindo os eixos temáticos da BNCC, sendo eles: abstração, decomposição de problemas, algoritmos e reconhecimento de padrões. Esses temas foram revisados para garantir coerência interna e diferenças claras entre si, resultando na criação de um mapa temático que orientou a interpretação dos resultados.

3. METODOLOGIA

Com o intuito de desenvolver uma taxonomia que classifique as atividades plugadas comprometidas com o ensino do Pensamento Computacional de acordo com as habilidades contidas na Base Nacional Comum Curricular. Realizou-se uma pesquisa de caráter descritivo, que utiliza uma abordagem qualitativa, envolvendo o levantamento de dados por meio de uma Revisão Sistemática da Literatura, uma pesquisa documental e análise temática.

Levantou-se ferramentas e sites educacionais mais utilizados no ensino do pensamento computacional através de uma revisão sistemática da literatura. Essa revisão incluiu a busca em bases de dados acadêmicos e educacionais para identificar as ferramentas mais relevantes e por meio da análise de conteúdo em sites educacionais online e em artigos levantados na revisão sistemática da literatura. Essa análise permitiu identificar exemplos de atividades que promovem o desenvolvimento das habilidades do Pensamento Computacional de acordo com as diretrizes da BNCC.

Para o levantamento das habilidades do Pensamento Computacional realizou-se uma pesquisa documental na Base Nacional Comum Curricular (BNCC). A pesquisa se constitui em examinar o documento da BNCC a fim de identificar as habilidades específicas relacionadas ao pensamento computacional que devem ser desenvolvidas ao longo da educação básica.

Após o mapeamento das ferramentas e sites educacionais mais utilizados, e a identificação das habilidades do Pensamento Computacional, elaborou-se atividades plugadas. A elaboração de atividades plugadas foi realizada para as habilidades que não foram contempladas na análise de conteúdo. Para a elaboração utilizou-se a plataforma *Scratch*, por ser a plataforma mais citada na revisão sistemática, e também porque essa ferramenta permite a criação de projetos interativos e lúdicos, utilizou-se a linguagem *Python* por ser a linguagem de programação mais utilizada nos últimos 12 meses em relação a data desta pesquisa, segundo o site TIOBE⁵, referência em ranking de linguagens.

Por fim, elaborou-se uma taxonomia hierárquica, com atividades plugadas organizadas de acordo com as habilidades da BNCC.

⁵ <https://www.tiobe.com/tiobe-index/>

A Revisão Sistemática da Literatura foi realizada nos meses de abril, maio e junho de 2024. E teve por meio de testes empíricos as seguintes bases de dados: Periódicos da CAPES, Education Resources Information Center (ERIC) e Biblioteca Digital Brasileira de Teses e Dissertações (BDTD). As quais retornaram números relevantes de estudos. Para garantir a qualidade e relevância dos dados coletados, foram estabelecidos critérios de inclusão e exclusão.

Inicialmente foram realizadas buscas preliminares utilizando um conjunto de palavras-chaves e conectivos lógicos para se chegar às *strings* de busca, as palavras chaves utilizadas foram (“*computational thinking*” *OR* “pensamento computacional”) *AND* (“*technological education*” *OR* “educação tecnológica”), também foi testado a *string* (“*computational thinking*” *OR* “pensamento computacional”) *AND* (“*technology*” *OR* “*ICT*” *OR* “Tecnologia” *OR* “*TIC*”) visando selecionar uma quantidade de pesquisas relevantes em relação aos resultados de busca.

A *string* de busca que mais retornou dados relevantes nas três bases de dados selecionadas foi (“pensamento computacional” *OR* “*computational thinking*”) *AND* (“*ICT*” *OR* “*technology*” *OR* “ensino fundamental” *OR* “ensino médio” *OR* “escola”).

Foram excluídas:

- Pesquisas não acessíveis na íntegra e sem disponibilidade gratuita na internet.
- Pesquisas não pertencentes aos idiomas português e inglês.
- Estudos secundários e terciários.
- Relatórios de workshop, relatórios técnicos, e estudos que tratam do assunto apenas como trabalho futuro.

Critérios de inclusão:

- Pesquisas entre 2014 e 2024, visando informações atualizadas e estudos que refletem os últimos avanços da área.
- Estudos que abordam o uso de computação plugada e o pensamento computacional no ensino fundamental e médio.
- Pesquisas que descrevem métodos e ferramentas para o ensino do pensamento computacional através da computação plugada.

A extração de dados foi realizada seguindo o modelo de Revisão Sistemática da Literatura (RSL) e dividida em três etapas. Primeiro, definiu-se quais dados seriam extraídos, focando nas ferramentas para o ensino do pensamento computacional e nas competências que elas desenvolvem. Na segunda etapa, criou-se um formulário de extração de dados em planilhas para padronizar o processo e garantir a coleta consistente de informações. Por fim, na terceira etapa, os estudos foram lidos detalhadamente, destacando as informações relevantes conforme os campos do formulário, seguido de uma revisão para verificar a integridade dos dados extraídos.

O estudo mapeou métodos e estratégias para integrar ferramentas de computação no ensino do pensamento computacional para crianças do ensino fundamental e médio. Analisando 134 pesquisas, identificou-se que a ferramenta mais utilizada é o *Scratch*, mencionado em 74,24% dos casos. Outras plataformas como *Code.org*, além de linguagens como *Python* e *C*, também foram relevantes.

Para a pesquisa e criação de atividades, visando transformar dados qualitativos em informações comprehensíveis, permitindo a identificação de tendências e padrões em conformidade com a BNCC, realizou-se a análise de conteúdo, que envolve a definição do objeto de estudo e seleção do material a ser analisado, leitura inicial, análise das categorias, interpretação e relato dos resultados.

4. RESULTADOS E DISCUSSÕES

Esse capítulo apresenta os resultados e discussões da pesquisa, iniciando com a análise da BNCC Computação, que organiza as habilidades do Pensamento Computacional em eixos temáticos, com exemplos de atividades. Em seguida, aborda o levantamento e criação de atividades plugadas, destacando ferramentas como *Scratch* e *Python* para o ensino de algoritmos e padrões. Por fim, é apresentada a taxonomia hierárquica desenvolvida, que organiza essas atividades de forma progressiva, facilitando sua aplicação por educadores e desenvolvedores em diferentes níveis escolares.

4.1 Pesquisa Documental na BNCC

A BNCC é organizada em diferentes níveis e etapas de ensino, com cada uma apresentando objetivos de aprendizagem e desenvolvimento específicos. Os principais componentes da BNCC na educação infantil tem como objetivo o desenvolvimento integral da criança, com ênfase em dimensões como a pessoal, social, cognitiva e emocional.

O ensino fundamental encontra-se dividido em dois ciclos, sendo os anos iniciais (1º ao 5º ano), abrange habilidades básicas em diversas áreas, como Língua Portuguesa, Matemática, Ciências, História, Geografia, Educação Física, Arte e Introdução à Computação. Nos anos finais (6º ao 9º ano), amplia o conhecimento nas mesmas áreas e introduz temas mais complexos, incluindo a informática como parte do currículo. O ensino médio, possui ênfase na preparação para o mundo do trabalho e para a continuidade dos estudos, abordando conhecimentos mais aprofundados e interdisciplinares.

Na BNCC Computação, o Pensamento Computacional é tratado como um eixo temático que possui um objeto de conhecimento, ligado à abstração, algoritmos, decomposição de problemas ou reconhecimento de padrões, e esse objeto contempla uma habilidade, explicação da habilidade e exemplo de atividade, como mostrado na Tabela 1.

Tabela 1 - Organização da BNCC Computação

Eixo	Objeto de conhecimento	Habilidade	Explicação da habilidade	Exemplo de Atividade
Pensamento Computacional	Conceituação de Algoritmos	(EF01CO02) Identificar e seguir sequências de passos aplicados no dia a dia para resolver problemas.	O objetivo é que os alunos identifiquem e sigam a sequência de passos para realizar uma tarefa.	O professor pode oferecer sequências de passos para tarefas como montar origamis, seguir caminhos ou fazer uma receita, e pedir que os alunos as realizem.

Fonte: Documento BNCC Computação (2022)

Para cada habilidade existe um código de identificação composto por um conjunto de letras e números. A habilidade da tabela 1, possui a identificação EF01CO02, sendo EF correspondente ao ensino fundamental, 01 ao primeiro ano, CO representa o componente curricular computação e 02 o código da habilidade específica dentro do componente curricular e ano.

Verificando os exemplos de atividade, observou-se que no documento oficial há apenas descrições de como pode ser feito, entretanto, no site oficial⁶, entre o primeiro ano do ensino fundamental e nono ano, há exemplos concretos de atividades que utilizam da abordagem desplugada, por meio de brincadeiras educacionais em sala de aula, envolvendo alunos e professores, e atividades que podem ser impressas, visando desenvolver as competências específicas. Em relação às atividades plugadas, observou-se duas referências ao *Scratch* e um ao *code.org*, como sugestão de plataforma para desenvolver atividades.

4.2 Levantamento de Atividades Plugadas

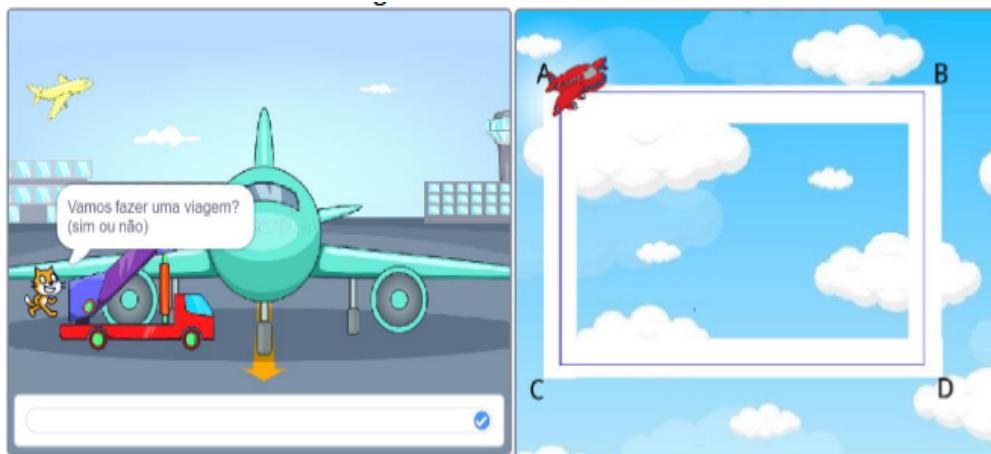
Para nomear as atividades relacionadas às habilidades, foi adotado um sistema de identificação que garante que cada atividade tenha um nome único, de acordo com o objeto de conhecimento e a competência correspondente. Assim, cada atividade é nomeada pela abreviação da competência, seguida da habilidade, acompanhada de um identificador que permite diferenciar várias atividades referentes à mesma habilidade. Por exemplo: Decomp. EF01CO01 - Atividade #01.

⁶ <https://www.computacional.com.br/BNCC/index.php>

Essa sistematização facilita não apenas a organização das atividades, mas também a análise e a documentação dos processos de aprendizado em conformidade com a BNCC. No entanto, levantou-se poucas atividades que vão ao encontro com as necessidades requeridas pela BNCC nos artigos abordados na RSL, e essa ausência de atividades ocorre pelo fato dos estudos abordarem resultados a respeito das atividades por meio das ferramentas, sem evidenciar sistematicamente a atividade.

Para o desenvolvimento da competência EF07CO04, buscando desenvolver o conceito de grafos, Abstra. EF07CO04 - Atividade #01, utiliza a ferramenta *Scratch* para introduzir e elaborar conceitos básicos a respeito de grafos. De acordo com Ferreira, Teixeira e Binotto (2022), é sugerido utilizar o uso do *Scratch*⁷ para que os estudantes investiguem um problema envolvendo rotas de um avião, conforme ilustrado na Figura 1.

Figura 1 - Rotas do Avião: Cenário A

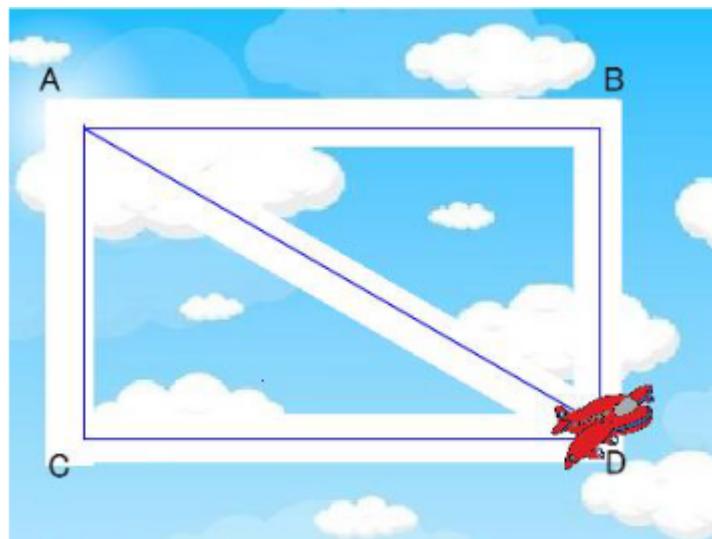


Fonte: Ferreira, Teixeira e Binotto (2022)

A proposta inicia de forma simples e pode solicitar que os estudantes percorram os caminhos conforme mostrados na Figura 2, partindo de cada vértice (A, B, C e D) e passando por todos, uma única vez. Em seguida, podem ser feitas perguntas como: “Será que nesse caso podemos começar de qualquer ponto?” e “O que acontece se adicionarmos mais uma opção de caminho?”. Durante essa etapa, propõe-se que os estudantes reflitam sobre possibilidades de rotas e façam conexões com a ideia de adicionar pontes em situações apresentadas posteriormente.

⁷ Cenário A: <https://scratch.mit.edu/projects/710818916> e cenário B: <https://scratch.mit.edu/projects/710819735>

Figura 2 - Rotas do Avião: Cenário B



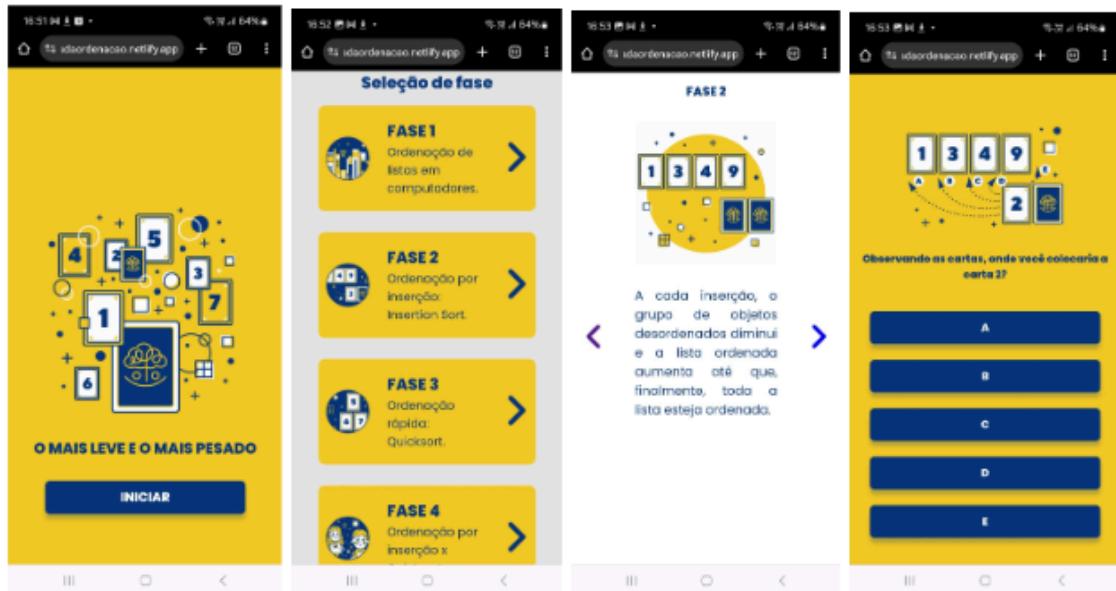
Fonte: Ferreira, Teixeira e Binotto (2022)

A competência EF08CO03 consiste em utilizar algoritmos clássicos de manipulação sobre listas. A Alg. EF08CO03 - Atividade #01, utiliza o sistema "Computação Plugada Ordenação Web⁸" proposto por Ferreira, Gonçalves e Rebouças (2024), com objetivo de apresentar conceitos e atividades relacionadas a algoritmos de ordenação de maneira atrativa e simples para estudantes. A Figura 3 exibe algumas telas do sistema.

⁸

<https://sites.google.com/view/computacaoplugada/aplicativos/cp-ordena%C3%A7%C3%A3o?authuser=0>

Figura 3 - Telas do sistema de Ordenação Web



Fonte: Ferreira, Gonçalves e Rebouças (2024)

Por meio desta aplicação, os usuários percorrem quatro fases distintas. Em cada uma, são apresentados de forma interativa e lúdica alguns métodos de ordenação. Nas segunda e terceira fases, o conhecimento sobre esses métodos é aprofundado, com exemplos práticos usando cartas numeradas e atividades de múltipla escolha. Em certas atividades, os usuários enfrentam desafios para identificar a posição correta de inserção de uma carta, garantindo sua ordenação em relação às demais. Além disso, são exibidos vídeos educativos narrando o processo de ordenação com o método específico em questão.

Na fase final, os usuários assistem a um vídeo explicativo que demonstra a execução de ambos os métodos em uma simulação, permitindo-lhes avaliar o desempenho relativo de cada um para ordenar dados nos exemplos apresentados. Ao concluir, os usuários respondem a perguntas de múltipla escolha sobre o conteúdo abordado no aplicativo e nos vídeos, recebendo uma verificação imediata sobre acertos e erros.

Para desenvolver a habilidade EF09CO03, que tem como exemplo modelar o comportamento de um robô utilizando autômatos, descrevendo eventos acionados a partir da leitura de seus sensores com linguagens baseadas em eventos que permitem descrever sistemas que são orientados pela ocorrência de eventos como cliques de mouse, pressionamento de alguma tecla. A Abstra. EF09CO03 - Atividade #01, busca

desenvolver essa habilidade por meio da construção de um jogo clássico na plataforma *Scratch*.

O *Pac-Man* possui um personagem central cujo objetivo é comer todas as pastilhas do labirinto sem ser atingido pelos fantasmas (Figura 4).

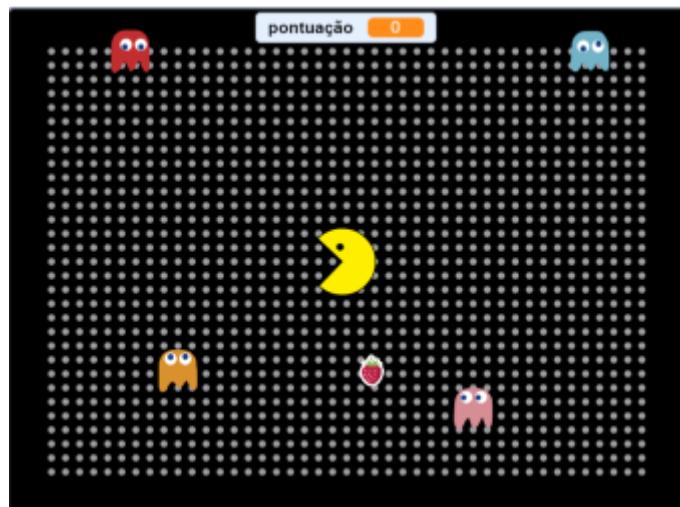
Figura 4 - Jogo *Pac-Man*



Fonte: GratisPNG (2023)

Na construção desenvolvida, as pastilhas foram substituídas por frutas. Ao colidir com uma fruta, o *Pac-Man* a consome e, simultaneamente, emite um som; em seguida, a fruta desaparece do labirinto e a pontuação é incrementada. O personagem move-se nas direções horizontal e vertical, assim como os fantasmas que compõem o cenário do jogo. Na Figura 5, é apresentado o palco da construção final realizada no *Scratch*.

Figura 5 - Jogo *Pac-man* no *Scratch*



Fonte: Padilha et al. (2023)

Para iniciar a construção do jogo no *Scratch*, foi definido como a criação de um jogo em que um personagem se move pela tela nas direções horizontal e vertical, com o objetivo de capturar frutas enquanto tenta escapar da perseguição de quatro fantasmas. Para exemplificar o tratamento dado ao problema de construir o movimento de quatro monstros pela tela, é apresentada a programação do código utilizado para um dos fantasmas que se deslocam na tela. Para o código desse fantasma, foi criado um novo bloco nomeado definições iniciais (Figura 6), que contém a programação responsável por posicionar o fantasma em uma localização específica do palco e por alterar sua fantasia toda vez que o jogo é iniciado.

Figura 6 – Bloco definições iniciais em um dos fantasmas

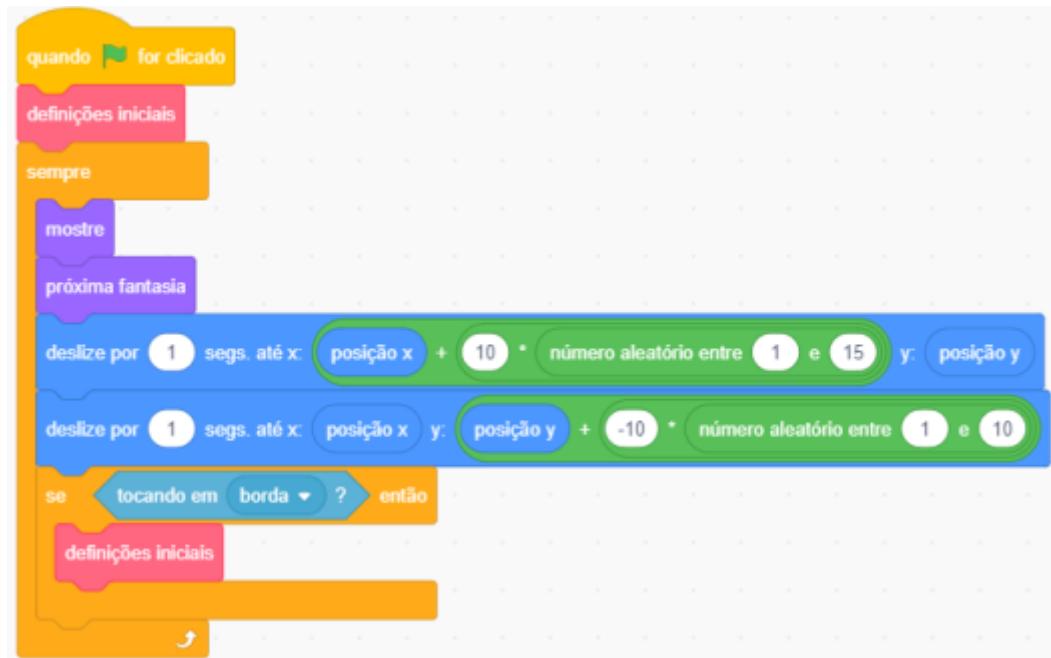


Fonte: Padilha et al. (2023)

O bloco de evento para iniciar o jogo (Figura 7) foi programado para fazer o fantasma deslizar pelo palco ao longo dos eixos X e Y, partindo de sua posição inicial

definida pelo bloco definições iniciais. Quando o fantasma atinge a borda do palco, ele aciona novamente o bloco definições iniciais e reinicia o processo.

Figura 7 - Código de um dos fantasmas



Fonte: Padilha et al. (2023)

Para ilustrar um dos itens mencionados, foi selecionada a programação do bloco de pontuação do jogo. Essa programação foi inserida no ator fruta, de modo que, sempre que o ator *Pac-Man* tocar a fruta, um ponto seja adicionado. Além disso, a cada início de jogo, a pontuação é reiniciada para zero (Figura 8).

Figura 8 - Código da pontuação



Fonte: Padilha et al. (2023)

O processo de construção do jogo *Pac-Man* no *Scratch* possibilitou a exploração dos pilares do Pensamento Computacional, bem como a aplicação de autômatos. Essa abordagem não apenas facilitou a compreensão dos conceitos fundamentais de programação, mas também permitiu modelar e automatizar comportamentos de forma estruturada e eficiente.

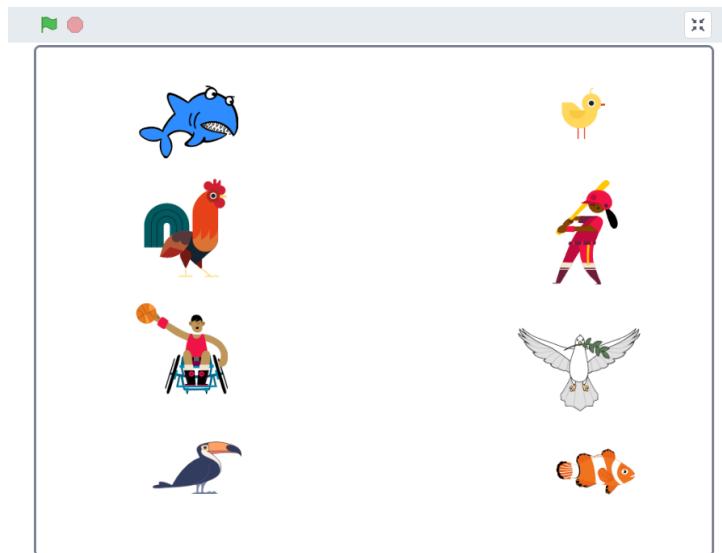
4.3 Elaboração de Atividades Plugadas

As atividades foram elaboradas pelo autor, em função das atividades que não foram levantadas, com o objetivo de desenvolver as competências previstas pela BNCC, tomando como referência o documento oficial e o site Computacional BNCC, que utiliza a mesma base e inclui exemplos complementares. Os exemplos que o site disponibiliza contempla do primeiro ao quinto ano do ensino fundamental, com atividades desplugadas.

De acordo com a sugestão de atividade para desenvolver a habilidade EF01CO01, que busca classificar objetos físicos ou digitais com base em características específicas, destacando padrões e diferenças entre eles. A Reco. EF01CO01 - Atividade #01⁹ consiste em um simples jogo, em que o aluno assimila os elementos que se relacionam (Figura 9).

⁹ <https://scratch.mit.edu/projects/1068100486>

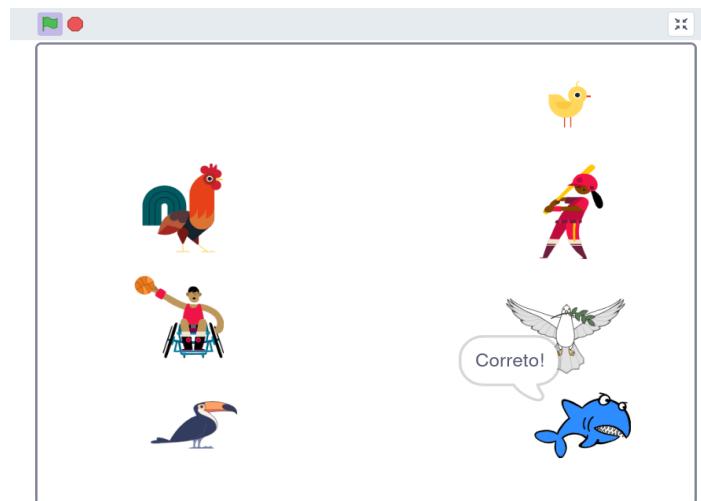
Figura 9 - Atividade EF01CO01



Fonte: Autor (2024)

A Figura 10 exibe o resultado após o elemento ser assimilado de maneira correta, respeitando o padrão.

Figura 10 - Acerto atividade EF01CO01



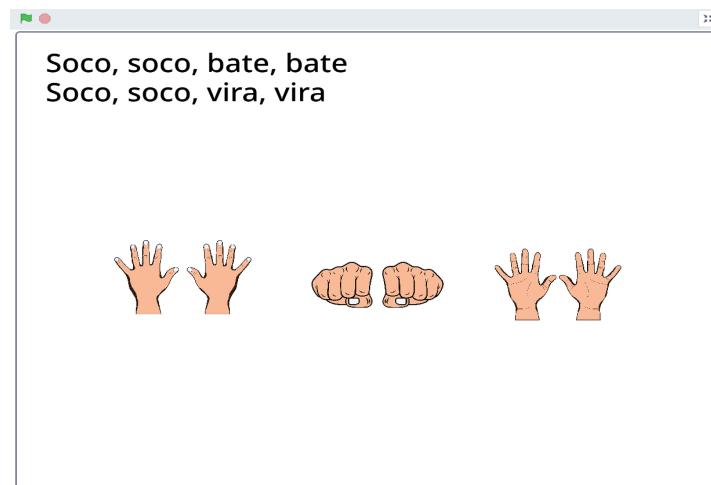
Fonte: Autor (2024)

Para o desenvolvimento da habilidade EF01CO02, no qual os alunos possam identificar passos que fazem parte da execução de uma tarefa, bem como seguir uma sequência de passos para realizar uma tarefa. A Alg. EF01CO02 - Atividade #01¹⁰, conforme sugerido pelo site, foi desenvolvido um jogo, em que o aluno segue o refrão

¹⁰ <https://scratch.mit.edu/projects/1069136760>

de uma música, configurando uma sequência de passos para determinada brincadeira (Figura 11).

Figura 11 - Atividade EF01CO02



Fonte: Autor (2024)

A Figura 12 exibe o resultado final obtido quando o aluno segue corretamente a sequência de passos proposta na atividade, mostrando o progresso alcançado ao longo do processo.

Figura 12 - Acerto atividade EF01CO02

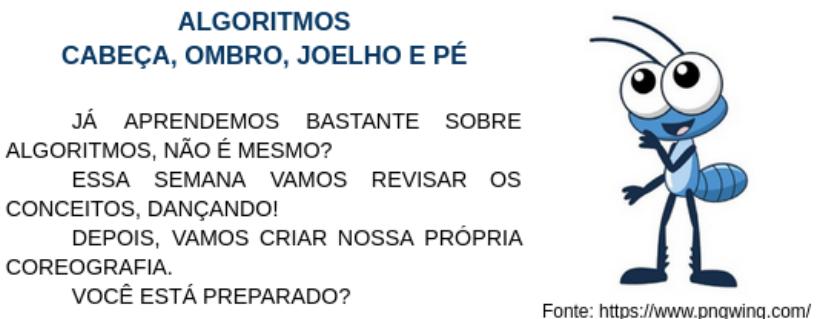


Fonte: Autor (2024)

A última atividade referente ao primeiro ano do ensino fundamental EF01CO03, que busca criar e organizar sequências de passos em formatos físicos ou digitais,

associando essas sequências a "Algoritmos", foi desenvolvida com base em uma música referenciada no site BNCC, mostrada na Figura 13.

Figura 13 - Música base EF01CO03

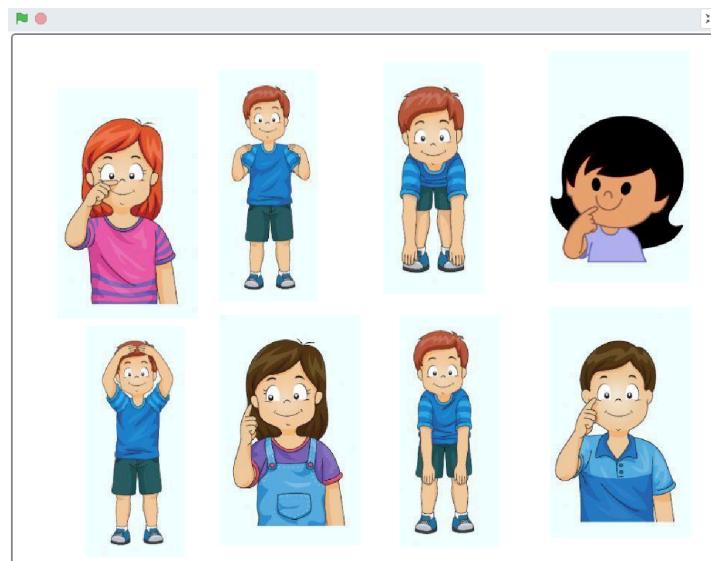


CABEÇA, OMBRO, JOELHO E PÉ

Fonte: BNCC Computação (2022)

A Alg. EF01CO03 - Atividade #01¹¹, um jogo baseado no algoritmo “cabeça, ombro, joelho e pé”, consiste no aluno ir selecionando as etapas até encontrar a sequência correta (Figura 14).

Figura 14 - Atividade EF01CO03

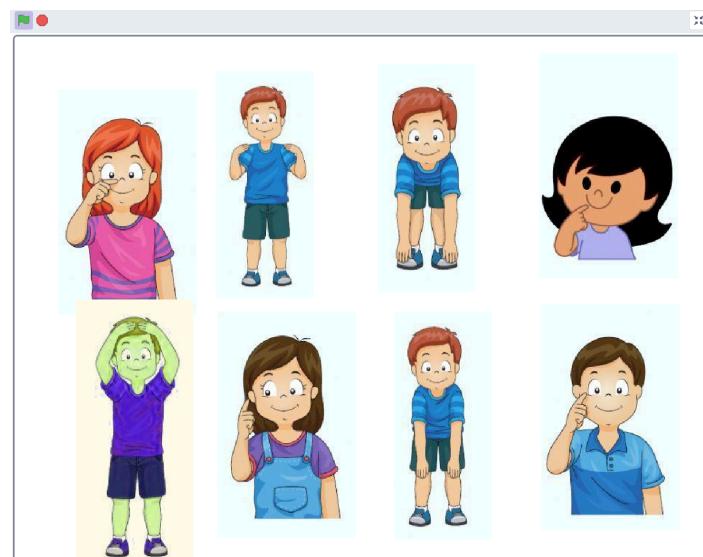


Fonte: Autor (2024)

Conforme o aluno acerta a sequência de passos, os personagens da cena ficam verdes, sendo um reforço positivo durante a execução da atividade, como mostrado na Figura 15.

¹¹ <https://scratch.mit.edu/projects/1072413639>

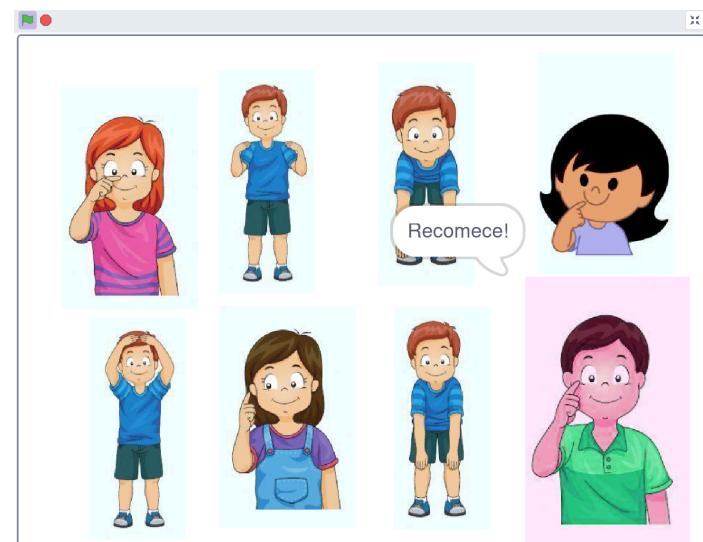
Figura 15 - Acerto da atividade EF01CO03



Fonte: Autor (2024)

Quando o aluno erra a sequência, os personagens recebem a coloração avermelhada, denotando que não estão no caminho correto e um aviso de recomeço conforme insistam em continuar, como exibido na Figura 16.

Figura 16 - Insistência de erro



Fonte: Autor (2024)

Na condição onde o aluno consegue atingir o objetivo completando a atividade, ele recebe as parabenizações, apresentado na Figura 17.

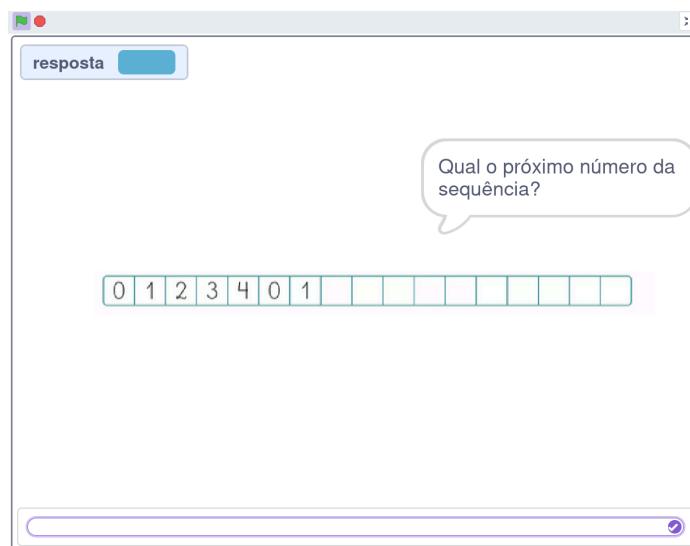
Figura 17 -Parabenização EF01CO03



Fonte: Autor (2024)

A primeira habilidade do segundo ano do ensino fundamental EF02CO01, consiste em criar e comparar modelos de objetos, destacando padrões e atributos essenciais. A Reco. EF02CO01 - Atividade #01¹², baseada na sugestão do site da BNCC, envolve identificar o padrão proposto na imagem e inserir os valores conforme o padrão, como proposto na Figura 18.

Figura 18 - Atividade EF02CO01

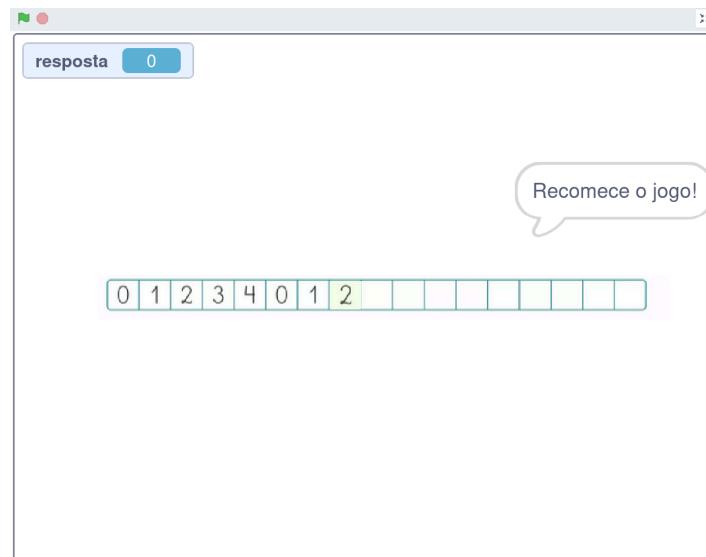


Fonte: Autor (2024)

A Figura 19 representa o caso em que o aluno não conseguiu completar e reconhecer o padrão proposto pelo jogo.

¹² <https://scratch.mit.edu/projects/1072734103>

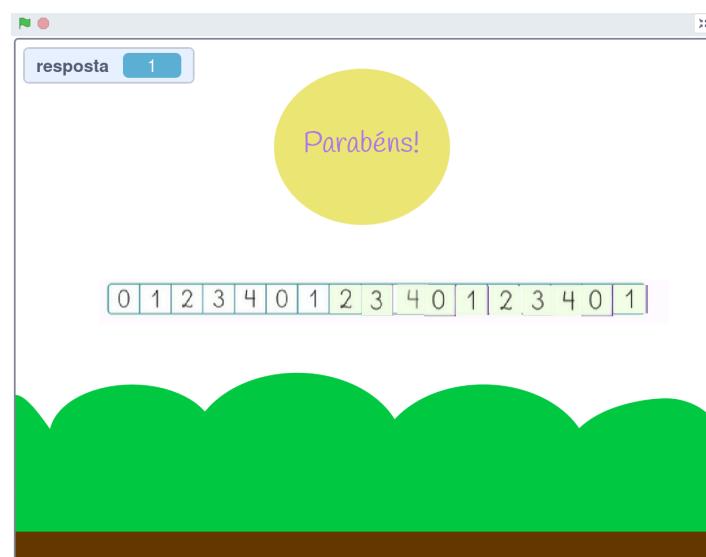
Figura 19 - Falha na atividade EF02CO01



Fonte: Autor (2024)

Quando o aluno consegue atingir o objetivo terminando o jogo e reconhecendo seu respectivo padrão, é emitido uma parabenização conforme a Figura 20.

Figura 20 - Atividade EF02CO01 Parabéns



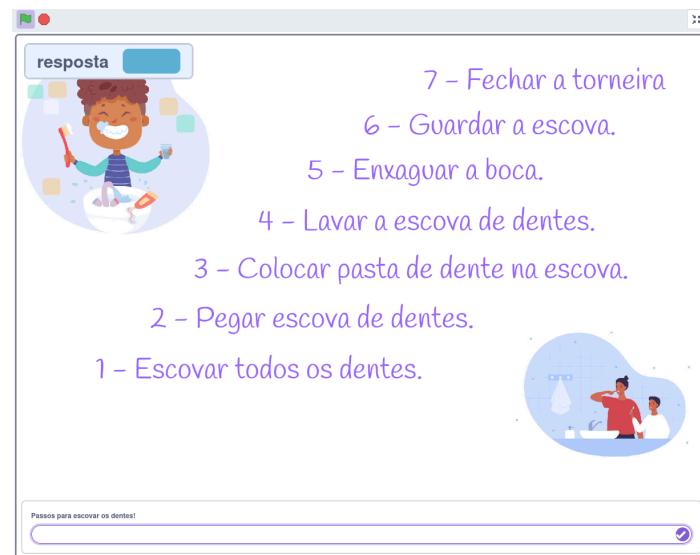
Fonte: Autor (2024)

Para desenvolver a habilidade EF02CO02, na qual alunos podem construir algoritmos com conjuntos de instruções pré-definidas, a Alg. EF02CO02 - Atividade #01¹³, assim como a primeira atividade do segundo ano, utiliza do recurso de entrada de

¹³ <https://scratch.mit.edu/projects/1073018552/>

texto, no qual o aluno insere o valor correspondido a ação a ser realizada de acordo com o algoritmo, como mostrado na Figura 21.

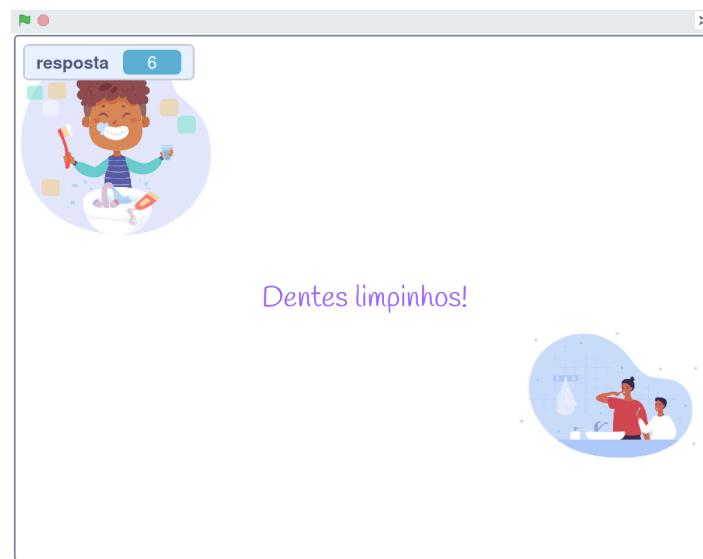
Figura 21 - Atividade EF02CO02



Fonte: Autor (2024)

Nessa atividade, as instruções são pré-definidas, portanto o professor é livre para explorar algoritmos diferentes para realizar a mesma tarefa. Ao final da atividade, a tela de conclusão será apresentada como mostra a Figura 22.

Figura 22 - Acerto Atividade EF02CO02

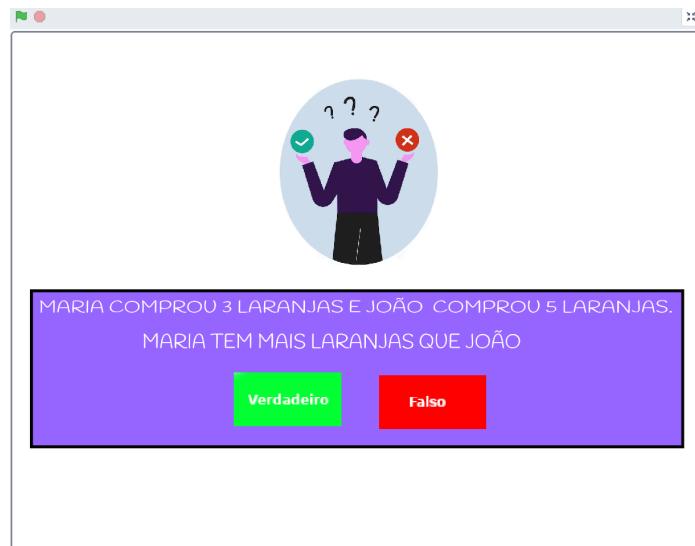


Fonte: Autor (2024)

A primeira habilidade do terceiro ano do ensino fundamental, EF03CO01 tem o objetivo de associar os valores “verdadeiro” e “falso” a sentenças lógicas que dizem respeito a situações do dia a dia. Ao associar valores "verdadeiro" e "falso" a sentenças

lógicas, o foco está em identificar e reconhecer padrões de verdade e falsidade em situações cotidianas. A Reco. EF03CO01 - Atividade #01¹⁴, por meio de uma interface, exibe uma sentença que o aluno deve classificá-la, como mostrado na Figura 23.

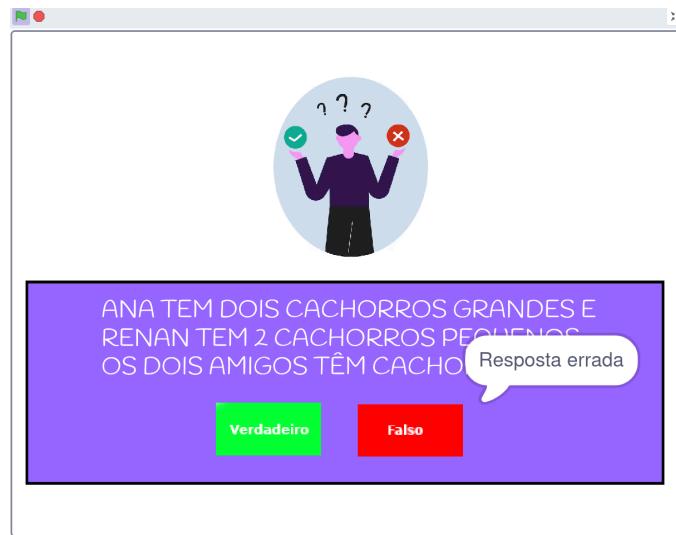
Figura 23 - Atividade EF03CO01



Fonte: Autor (2024)

Quando ocorre o acerto, a próxima sentença irá ser exibida, em caso de falha uma mensagem que indica o erro é emitida (Figura 24).

Figura 24 - Atividade EF03CO01

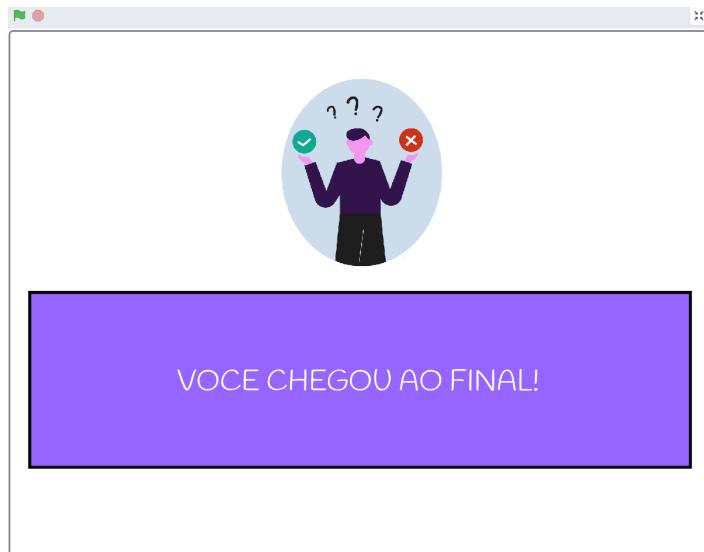


Fonte: Autor (2024)

¹⁴ <https://scratch.mit.edu/projects/1073264906>

Após a conclusão da atividade, uma mensagem é exibida na tela indicando o término do processo, confirmando que todos os passos foram concluídos com sucesso, como na Figura 25.

Figura 25 - Final Atividade EF03CO01



Fonte: Autor (2024)

Para desenvolver a habilidade EF03CO02, que visa simular algoritmos representados em linguagem oral, escrita ou pictográfica, a Alg. EF03CO02 - Atividade #01¹⁵, exibe possíveis escolhas para completar o algoritmo e dar continuidade ao problema proposto, como na Figura 26.

Figura 26 - Atividade EF03CO02



Fonte: Autor (2024)

¹⁵ <https://scratch.mit.edu/projects/1073485989>

Quando a escolha correta é selecionada, outras situações do dia a dia são apresentadas, como mostra a Figura 27.

Figura 27 - Avanço atividade EF03CO02



Fonte: Autor (2024)

Para o desenvolvimento da habilidade EF03CO03, busca-se aplicar a estratégia de decomposição para resolver problemas complexos, dividindo esse problema em partes menores, resolvendo-as e combinando suas soluções. A Decomp. EF03CO03 - Atividade #01¹⁶, utiliza do recurso de entradas de texto para ajudar a decompor e montar uma sequência de passos para ajudar colegas a plantar uma árvore (Figura 28).

Figura 28 - Atividade EF03CO03



Fonte: Autor (2024)

¹⁶ <https://scratch.mit.edu/projects/1073937037>

Conforme o aluno insere o passo ou etapa do problema, ele é exibido na tela, conforme a Figura 29.

Figura 29 - Avanço atividade EF03CO03



Fonte: Autor (2024)

Essa atividade permite diversas soluções possíveis, incentivando a discussão entre alunos e o professor, além de promover comparações e sugestões de melhorias. Esse ambiente colaborativo estimula o pensamento crítico, a criatividade e a troca de conhecimentos, ajudando os alunos a refletirem sobre diferentes abordagens e a desenvolverem habilidades de resolução de problemas de forma conjunta.

A primeira atividade do quarto ano do ensino fundamental, traz introdução aos conceitos de matrizes, buscando reconhecer objetos que podem ser tanto do mundo real quanto digital que podem ser representados por matrizes, em que cada componente ocupa uma posição específica definida por coordenadas, permite manipulações simples dessas representações.

Matrizes são estruturas de dados organizadas em linhas e colunas, semelhantes a tabelas, com um tamanho pré-definido e compostas por dados do mesmo tipo. O acesso a um dado específico em uma matriz ocorre por meio de coordenadas (x, y), que indicam a linha e a coluna onde ele se localiza. Matrizes que contêm apenas uma linha são chamadas de vetores. A proposta é que os alunos sejam capazes de identificar

objetos estruturados no mundo real que possam ser caracterizados como matrizes e utilizem representações visuais para ilustrá-los.

Para o desenvolvimento da habilidade EF04CO01, com base no documento disponibilizado no site BNCC Computação, buscou-se desenvolver uma atividade similar e mais interativa. Utilizando o conceito de matrizes que representam o tabuleiro, onde existe um robô como ator principal e alguns obstáculos, deve-se conduzir o robô até a saída, como mostrado na Figura 30.

Figura 30 - Exemplo habilidade EF04CO01

O robô foi desprogramado e agora não consegue mais sair da sala de comando. Você poderia ajudá-lo?

Qual a sequência de comandos necessária para que ele encontre uma saída? Lembre-se de evitar os obstáculos.

a)

saída

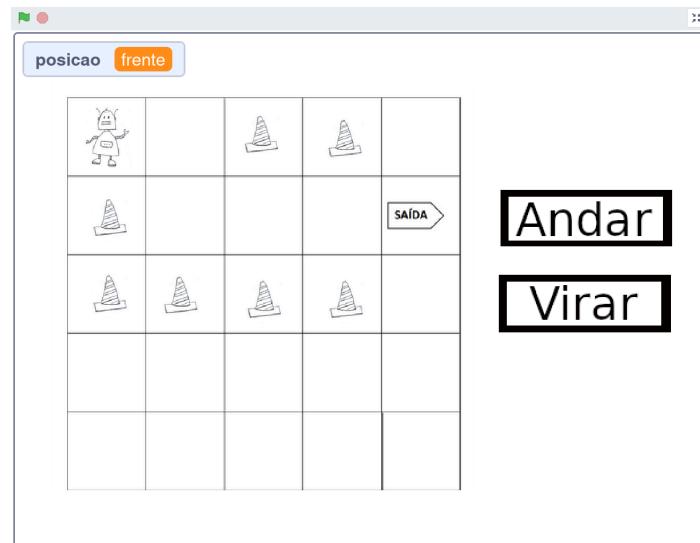
Ínicio

- Virar à esquerda**
- Avançar 1 casa**
- Virar à direita**
- Avançar 1 casa**
- Virar à esquerda**
- Avançar 3 casas**
- Fim**

Fonte: BNCC Computação (2022)

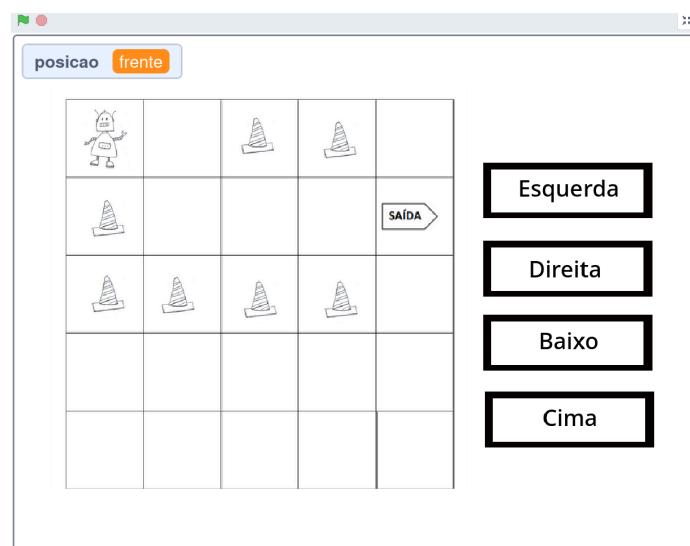
A Abstra. EF04CO01 - Atividade #01¹⁷, utiliza-se de uma interface interativa onde o aluno exerce instruções ao robô para encontrar a saída (Figura 31). A primeira interação configura em definir a orientação do robô, ou seja, para qual lado ele irá seguir.

¹⁷ <https://scratch.mit.edu/projects/1073953443>

Figura 31 - Atividade EF04CO01

Fonte: BNCC Computação (2022)

A primeira interação configura em definir a orientação do robô, ou seja, para qual sentido ele irá seguir. Para que o ator principal consiga andar por meio das coordenadas é necessário que a posição para qual ele irá se deslocar seja configurada, inicialmente a posição nula é definida, representando “frente”. A Figura 32 exibe as possíveis direções.

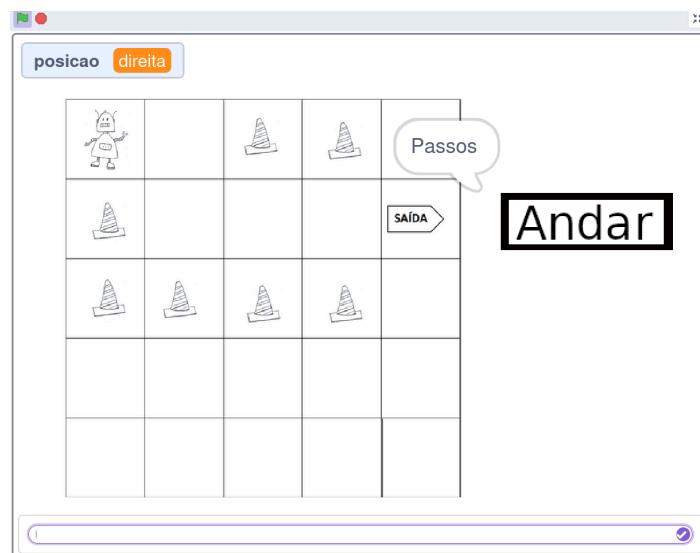
Figura 32 - Direções EF04CO01

Fonte: Autor (2024)

Com a direção definida é possível selecionar a opção “Andar”, que irá exibir uma caixa de entrada de dados em que o aluno será capaz de informar a quantidade de

passos a serem executados pelo robô, sendo um passo equivalente a um bloco no tabuleiro, como exibido na Figura 33.

Figura 33 - Andar atividade EF04CO01



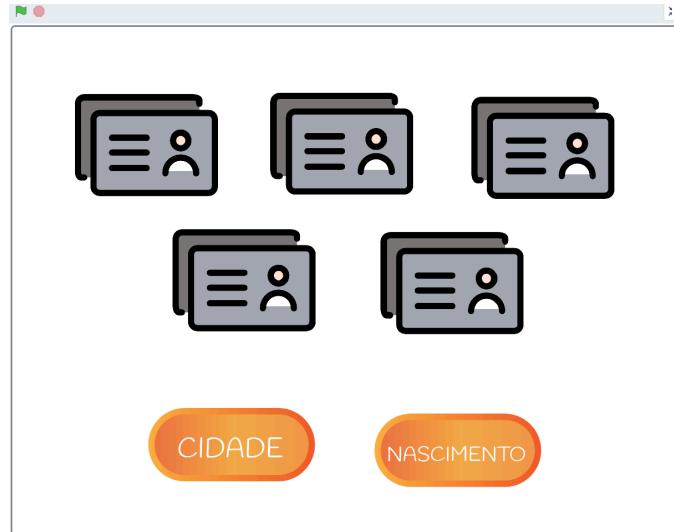
Fonte: Autor (2024)

Após a resolução dessa atividade, também é possível trazer debates de situações similares, com foco na digitalização para ambientes modelados por meio de matrizes, como sistemas de mapas em jogos eletrônicos, organização de assentos em auditórios ou cinemas, e até a distribuição de recursos em uma grade urbana. Esses exemplos ajudam a aprofundar a compreensão sobre como as matrizes podem representar de forma eficiente dados complexos, permitindo manipulação e análise de informações estruturadas para resolver problemas do cotidiano.

A habilidade EF04CO02, busca reconhecer objetos do mundo real e/ou digital que podem ser representados através de registros que estabelecem uma organização na qual cada componente é identificado por um nome, fazendo manipulações sobre estas representações. A Abstra. EF04CO02 - Atividade #01¹⁸, com base na sugestão do site da BNCC, reúne um conjunto de informações pessoais em cada documento (Figura 34). Essas informações podem ser recuperadas e filtradas de acordo com a cidade ou data de nascimento.

¹⁸ <https://scratch.mit.edu/projects/1074236891>

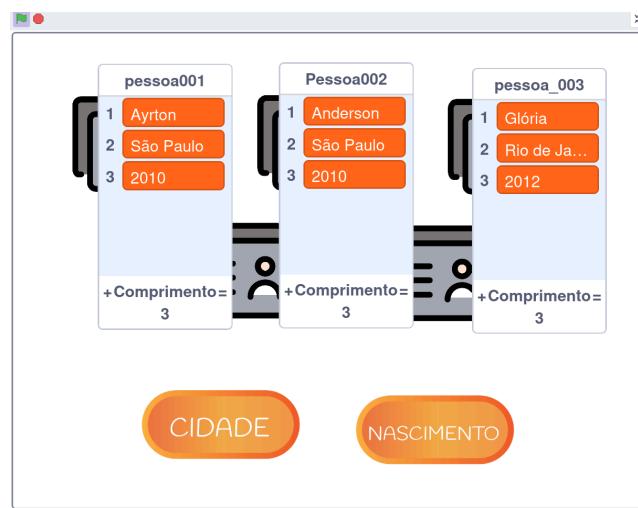
Figura 34 - Atividade EF04CO02



Fonte: Autor (2024)

Selecionando um ou mais documentos, dados como nome, cidade e data de nascimento são emitidos sobre cada documento, como mostrado na Figura 35.

Figura 35 - Documentos Atividade EF04CO02

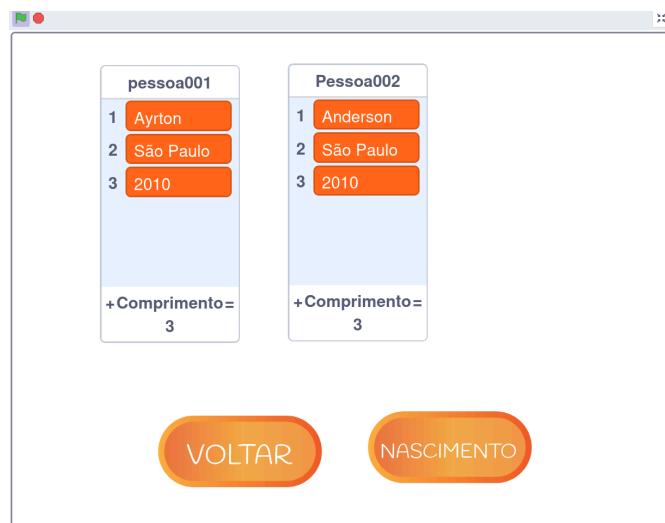


Fonte: Autor (2024)

Para manipulação e recuperação dos dados, o sistema oferece duas opções de busca: por cidade ou por data de nascimento. Ao escolher uma dessas opções, o aluno utiliza uma caixa de texto para inserir o termo desejado. Caso o valor digitado seja válido, o sistema localiza e exibe os documentos correspondentes que atendem a esses critérios específicos. Essa funcionalidade permite uma navegação prática e direcionada,

facilitando a consulta de informações. A Figura 36 ilustra o processo de como os resultados são recuperados e apresentados, promovendo uma experiência interativa.

Figura 36 - Recuperação de dados por nascimento EF04CO02



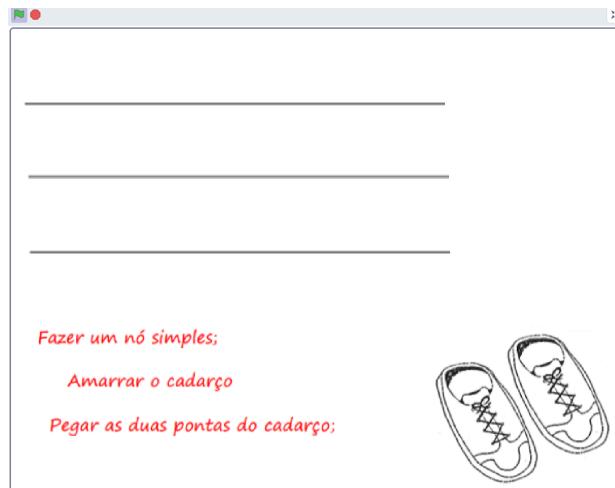
Fonte: Autor (2024)

A atividade permite que os alunos identifiquem objetos do mundo real e digital que podem ser representados através de registros organizados, nos quais cada componente é associado a um nome específico. Esse exercício facilita a compreensão sobre como utilizar registros para estruturar dados, simplificando a manipulação e recuperação de informações.

Para o desenvolvimento da habilidade EF04CO03, que visa trabalhar a criação e simulação de algoritmos representados em linguagem oral, escrita ou pictográfica, que incluem sequências e repetições simples, para resolver problemas de forma independente e em colaboração. A Alg. EF04CO03 - Atividade #01¹⁹, por meio de ações pré-definidas e uma interface interativa, proporciona a possibilidade do aluno desenvolver um algoritmo para problemas do cotidiano, selecionando e arrastando cada etapa até a linha correspondente (Figura 37).

¹⁹ <https://scratch.mit.edu/projects/1075019247>

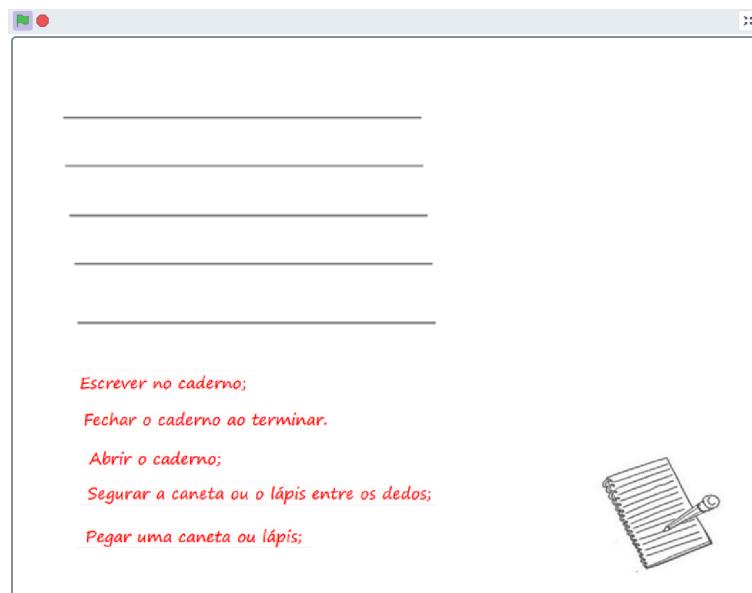
Figura 37 - Atividade EF04CO03



Fonte: Autor (2024)

A outra etapa da atividade consiste em um algoritmo com mais passos, usando o mesmo conceito de arrastar as etapas para as respectivas linhas, como mostrado na Figura 38.

Figura 38 - Parte 2 Atividade EF04CO03



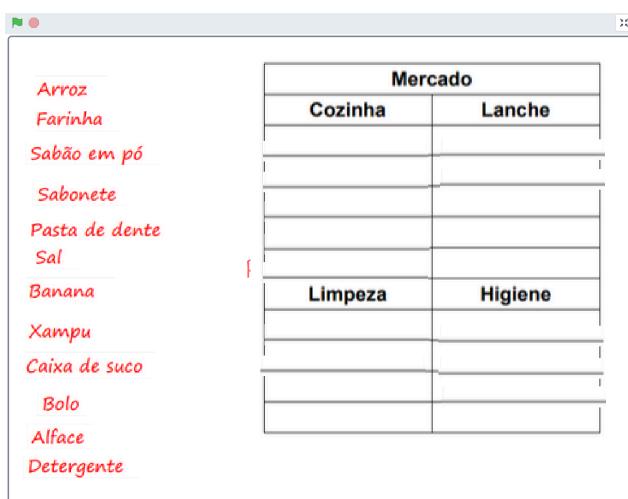
Fonte: Autor (2024)

O formato de arrastar e soltar das etapas permite que os estudantes representem as instruções em linguagem pictográfica, aproximando-se da lógica algorítmica de forma intuitiva e acessível. A atividade também incentiva a colaboração entre os alunos,

que para discutir ideias e estratégias, promovendo um aprendizado cooperativo e fortalecendo sua autonomia na resolução de problemas.

A primeira habilidade do quinto ano do ensino fundamental EF05CO01, busca desenvolver a capacidade de reconhecer objetos tanto dos cenários reais quanto digitais, que podem ser representados através de listas que estabelecem uma organização na qual há um número variável de itens dispostos em sequência, fazendo manipulações simples sobre estas representações. A Reco. EF05CO01 - Atividade #01²⁰, tem o objetivo de exibir um conjunto de itens e que o aluno classifique-os conforme as categorias corretas, como mostrado na Figura 39.

Figura 39 - Atividade EF05CO01



The image shows a Scratch script titled "Mercado". On the left, there is a vertical list of grocery items: Arroz, Farinha, Sabão em pó, Sabonete, Pasta de dente, Sal, Banana, Xampu, Caixa de suco, Bolo, Alface, and Detergente. On the right, there is a 4x2 grid table with the following structure:

Mercado	
Cozinha	Lanche
Limpeza	Higiene

Fonte: Autor (2024)

Ao classificar itens conforme as categorias corretas, os alunos praticam a organização e categorização de dados, o que aprimora sua capacidade de identificar padrões e estabelecer critérios para agrupamento. Essa atividade também ajuda a desenvolver a flexibilidade cognitiva, pois o aluno precisa entender que listas podem variar em tamanho e ser reorganizadas de acordo com as necessidades da tarefa.

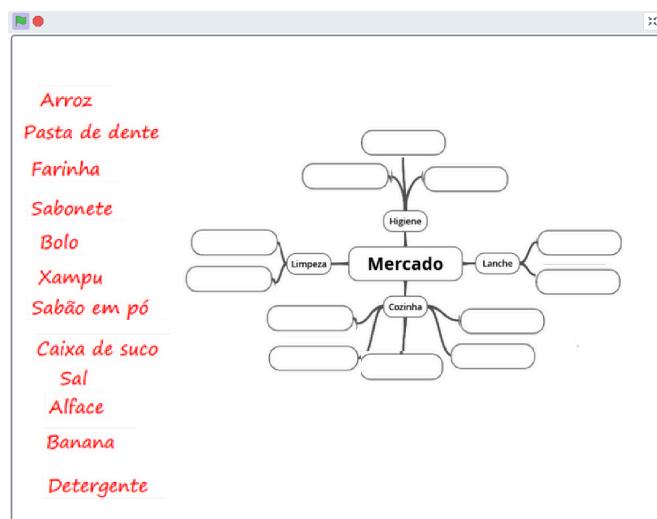
A habilidade EF05CO02 consiste em reconhecer objetos do mundo real e digital que podem ser representados através de grafos que estabelecem uma organização com uma quantidade variável de vértices conectados por arestas, fazendo manipulações simples sobre estas representações. A Abstra. EF05CO02 - Atividade #01²¹, estimula a organização de um grafo, introduzindo conceitos básicos. Usando uma interface

²⁰ <https://scratch.mit.edu/projects/1075865132>

²¹ <https://scratch.mit.edu/projects/1078714109>

interativa as palavras ficam no canto esquerdo e o grafo à direita, o intuito é levar as palavras até seus respectivos lugares, como mostrado na Figura 40.

Figura 40 - Atividade EF05CO02



Fonte: Autor (2024)

A interface interativa facilita a compreensão dos conceitos básicos de grafos, como vértices e arestas. Os alunos são desafiados a associar as palavras aos seus respectivos lugares no grafo, promovendo uma compreensão prática e visual das relações entre os elementos. Os alunos podem experimentar manipulações básicas no grafo, fortalecendo a capacidade de organizar e compreender dados estruturados.

A EF05CO03, última habilidade do quinto ano do ensino fundamental, em que o professor pode apresentar diferentes sentenças lógicas e solicitar que os alunos determinem seus valores verdade, desenvolvida por meio da Abstra. EF05CO03 - Atividade #01²², traz um “gatinho” como protagonista de quiz de verdadeiro e falso (Figura 41).

²² <https://scratch.mit.edu/projects/1079674165>

Figura 41 - Atividade EF05CO03



Fonte: Autor (2024)

O controle do ator principal é realizado através das setas direcionais do teclado, permitindo que o usuário o movimente de forma intuitiva e responsiva. Ao utilizar as teclas, o personagem pode não apenas se deslocar para a esquerda e para a direita, mas também executar movimentos de pulo, elevando-se verticalmente. Dessa forma, o jogador possui controle total sobre as ações básicas de movimentação, possibilitando interações variadas dentro do cenário, como ilustrado na Figura 42.

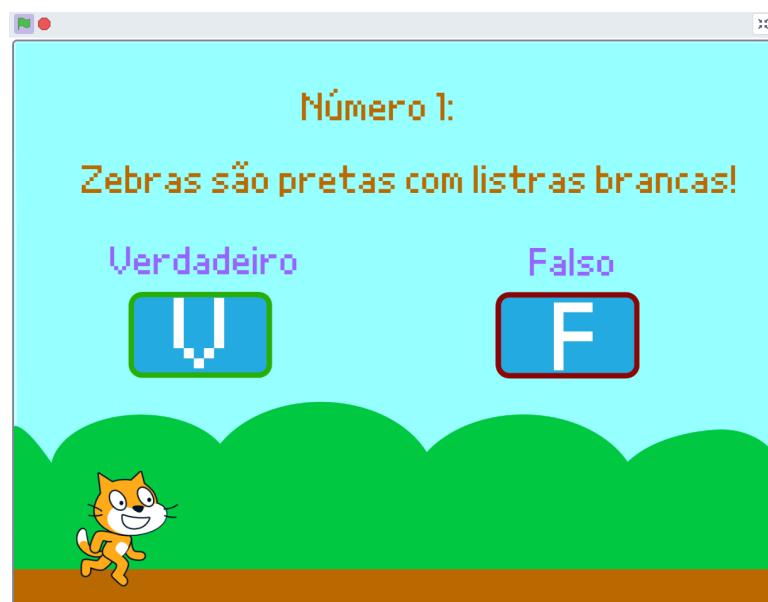
Figura 42 - Atividade controle EF05CO03



Fonte: Autor (2024)

Inicialmente o ator encontra-se no canto esquerdo da tela, uma sentença lógica no topo é apresentada e 2 caixas com respectivos valores verdadeiro e falso são exibidos na tela, como mostrado na Figura 43.

Figura 43 - Início Atividade EF05CO03



Fonte: Autor (2024)

O aluno, ao conduzir o personagem através do ambiente interativo, até ultrapassar a última sentença lógica, ao final dessa trajetória, recebe uma mensagem de congratulações, parabenizando o aluno por ter concluído com sucesso todas as etapas do desafio e reconhecendo seu esforço e atenção ao longo da atividade (Figura 44).

Figura 44 - Congratulação EF05CO03



Fonte: Autor (2024)

Caso o aluno selecione um valor incorreto para a sentença lógica, ele é imediatamente direcionado para uma tela, onde aparece a mensagem “tente novamente”, incentivando-o a revisar sua escolha. Esse redirecionamento serve como uma oportunidade para o aluno refletir sobre a sentença e corrigir o erro, como na Figura 45.

Figura 45 - Tente novamente EF05CO03



Fonte: Autor (2024)

Ao conduzir o personagem pelo ambiente interativo e interagir com as sentenças, o aluno não apenas aplica conhecimentos prévios, mas também é incentivado a refletir criticamente sobre suas escolhas.

Nas atividades do sexto ao nono ano do ensino fundamental, espera-se o uso de uma linguagem de programação para a resolução de problemas, o *Python* foi adotado como linguagem principal. Nessa etapa, as atividades tornam-se mais abstratas, incentivando os alunos a aplicar conceitos de forma criativa e flexível. Exemplos de códigos foram incluídos no apêndice deste trabalho para auxiliar o docente e os alunos, mas ambos são incentivados a ir além dos exemplos fornecidos e explorar diferentes abordagens para os mesmos conceitos, promovendo uma compreensão mais profunda e adaptativa.

Para o desenvolvimento da habilidade EF06CO01, com o objetivo de classificar informações, agrupando-as em coleções e associando cada coleção a um tipo de dados, a Reco. EF06CO01 - Atividade #01, propõe um enunciado conforme a Tabela 2.

Tabela 2 - Atividade EF06CO01

Imagine que você está no "Mercado Mágico", onde pode encontrar uma mistura de itens como maçãs, bananas, números, e até mesmo criaturas mágicas. Sua tarefa é separar esses itens em diferentes listas de acordo com o tipo deles:

1. Crie três listas chamadas inteiro, real, e string.
2. Use a lista de dados fornecida, que contém uma mistura de números e palavras.
3. Classifique cada item dessa lista em uma das três listas:
 - Coloque números inteiros na lista inteiro.
 - Coloque números com vírgula (reais) na lista real.
 - Coloque palavras na lista string.

No final, mostre cada uma das listas organizadas com uma mensagem divertida.

Fonte: Autor (2024)

Essa atividade incentiva os alunos a reconhecer diferentes tipos de dados e organizá-los em listas específicas. Ao separar os itens mistos em categorias distintas, os alunos exercitam o conceito de coleções associadas a tipos de dados, que é fundamental para o entendimento da classificação e organização de informações no mundo da programação, evidenciando a importância de agrupar dados semelhantes, facilitando o armazenamento, a manipulação e o acesso a essas informações de forma organizada e eficiente.

A habilidade EF06CO02, explorando a função de elaborar algoritmos que envolvam instruções sequenciais, de repetição e de seleção usando uma linguagem de programação, introduz os principais conceitos de laço de repetição. A Alg. EF06CO02 - Atividade #01, sugere um cálculo de média simples, como mostra a Tabela 3.

Tabela 3 - Atividade EF06CO02

Imagine que você é o professor de uma turma e precisa calcular a média das notas de seus alunos para ver como foi o desempenho da turma nas provas. Para isso, você vai criar um programa em *Python* que calcula a média das notas usando uma lista.

Passos para Resolver:

1. Criar a lista de notas: Crie uma lista chamada *notas* que contém as notas de três alunos (valores como 7.5, 8.0, 9.0).
2. Calcular a soma das notas: Some todas as notas dessa lista. Para isso, você pode usar um laço de repetição (o *for* ou *while*).
3. Calcular a média das notas: Divida a soma das notas pelo número total de alunos (no caso, 3).
4. Exibir a média na tela: Mostre a média das notas com uma mensagem explicativa.

Perguntas para Reflexão:

- Qual foi a média obtida? Isso indica um bom desempenho da turma?
- Como você poderia modificar o programa para funcionar com uma quantidade diferente de alunos?

Dicas Extras:

- Experimente alterar a lista de notas para ver como a média muda com diferentes valores.

Fonte: Autor (2024)

A atividade de cálculo da média das notas possibilita aos alunos criar um algoritmo em Python que incorpora instruções sequenciais, de repetição e de seleção. Os alunos utilizam instruções sequenciais para definir e somar as notas, aplicando laços de repetição para iterar sobre a lista de notas. Além disso, podem implementar estruturas de seleção para validar os valores, garantindo que estejam dentro de um intervalo aceitável. Dessa forma, a atividade proporciona uma experiência prática completa na elaboração de algoritmos, explorando os conceitos fundamentais da programação.

Para o desenvolvimento da habilidade EF06CO03 e EF06CO05, com o objetivo de descrever com precisão a solução de um problema, construindo o programa que implementa a solução descrita. A BNCC Computação deixa claro que o aluno precisa entender que o mais importante é a construção do algoritmo, a ideia aqui não é apenas descrever as linhas de código, mas sim descrever em um alto nível de abstração como o problema é resolvido. A Alg. EF06CO03 - Atividade #01, promove a construção de um algoritmo, sendo possível desenvolver a capacidade de trabalhar com listas, funções e condicionais (Tabela 4).

Tabela 4 - Atividade EF06CO03 e EF06CO05

<p>Os alunos devem imaginar que têm uma pilha de cartas e precisam verificar se há um ás (cartas específicas) na pilha. A pilha é uma estrutura de dados onde a última carta a ser colocada é a primeira a ser removida.</p>
<ol style="list-style-type: none"> 1. Antes de programar, os alunos devem escrever, em português, um algoritmo que descreva como resolver o problema. Eles devem considerar as seguintes etapas: <ul style="list-style-type: none"> • Verificar se a pilha está vazia. • Se a pilha estiver vazia, informar que não há ás na pilha. • Se a primeira carta na pilha for um ás, informar que há ás na pilha. • Caso contrário, remover a primeira carta e repetir o processo até que a pilha esteja vazia ou um ás seja encontrado. 2. Usando a descrição do algoritmo que construíram, os alunos irão implementar a solução em <i>Python</i>. Eles devem usar uma função para verificar a presença de ás na pilha, como demonstrado no código fornecido no apêndice. O código deve ser adaptado para seguir as etapas do algoritmo que eles escreveram. 3. Após a construção do algoritmo e a implementação do código, os alunos devem testar a função com diferentes pilhas de cartas, adicionando e removendo cartas conforme necessário, para observar como a função se comporta. 4. Após a execução do código, os alunos devem responder a algumas perguntas: <ul style="list-style-type: none"> • O que o algoritmo fez corretamente? • Como a programação ajudou a resolver o problema? • Quais partes do algoritmo foram mais desafiadoras de implementar?

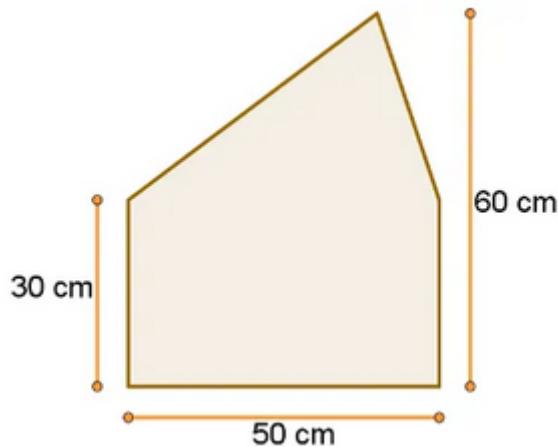
Fonte: Autor (2024)

A atividade visa fortalecer a compreensão dos alunos sobre como descrever um problema, a construção de algoritmos e a implementação desses algoritmos em uma linguagem de programação, incentivando os alunos a formular a solução em português antes de implementá-la em código, o que reforça a importância do pensamento algorítmico.

Para o desenvolvimento da habilidade EF06CO04 e EF06CO06, com objetivo de construir soluções de problemas usando a técnica de decomposição e automatizar tais soluções usando uma linguagem de programação. A Decomp. EF06CO04 - Atividade #01, propõe cálculos de áreas de figuras geométricas que necessitam ser

divididas para que possam ser calculadas separadamente, fomentando a ideia de decompor problemas maiores em menores, ao final juntando o todo e solucionando o respectivo problema, como mostrado na Figura 46.

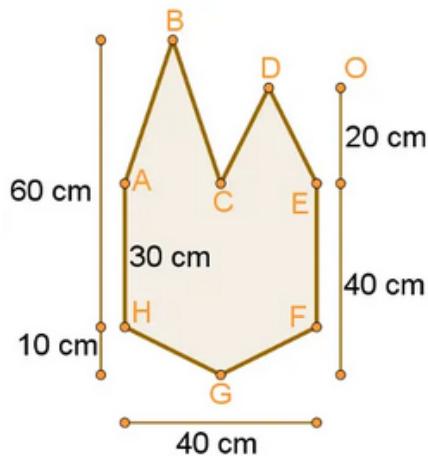
Figura 46 - Figura geométrica EF06CO04 e EF06CO06



Fonte: Mundo Educação (2016)

Para o cálculo da figura geométrica contida da Figura 46, cabe o aluno desenvolver a habilidade de raciocínio para dividir o elemento e aplicar a fórmula da área do triângulo e área do retângulo. A outra figura geométrica, contida na Figura 47, utiliza o mesmo conceito, sendo porém, mais elaborada.

Figura 47 - Figura geométrica 2 EF06CO04 e EF06CO06



Fonte: Mundo Educação (2016)

Essa atividade busca comparar diferentes instâncias do problema de calcular a área de uma figura geométrica, identificando que o que varia entre elas são as medidas que podem ser alteradas e automatizadas com a linguagem de programação.

A EF07CO01, primeira habilidade do sétimo ano do ensino fundamental, com objetivo de criar soluções de problemas para os quais seja adequado o uso de registros e matrizes unidimensionais para descrever suas informações e automatizá-las usando uma linguagem de programação. A Abstra. EF07CO01 - Atividade #01, tem um objetivo de desenvolver um programa que permita o cadastro de animais de estimação e a consulta por tipo de animal. Os alunos podem utilizar listas (matrizes unidimensionais) para armazenar informações e realizar buscas (Tabela 5).

Tabela 5 - Atividade EF07CO01

<p>Os alunos criarão um programa que armazena dados de animais de estimação, incluindo nome, tipo (cachorro, gato, pássaro, etc.) e idade. O programa deve permitir que o usuário consulte os animais cadastrados por tipo.</p>
<p>Requisitos:</p> <ul style="list-style-type: none"> • O programa deve permitir que o usuário cadastre pelo menos 5 animais. • Cada animal deve ter um nome, um tipo e uma idade. • O programa deve ter um menu que permita ao usuário: • Cadastrar um novo animal. • Consultar todos os animais de um tipo específico (por exemplo, todos os cachorros). • Exibir todos os animais cadastrados.
<p>Sugestões de Implementação:</p> <ul style="list-style-type: none"> • Crie três listas: nomes, tipos e idades para armazenar as informações dos animais. • Utilize um loop while para apresentar o menu até que o usuário escolha sair. • Implemente funções para cadastrar um animal e para consultar os animais por tipo.

Fonte: Autor (2024)

Com essa atividade os alunos aprendem a organizar dados, como nome, tipo e idade dos animais, utilizando listas para armazenar essas informações de forma estruturada. Isso permite que desenvolvam habilidades de lógica de programação ao implementar funções de cadastro e consulta, reforçando a importância de escolher a estrutura de dados adequada para resolver problemas de maneira eficiente. Assim, a atividade promove o pensamento crítico sobre a organização de informações, uma competência essencial.

Para o desenvolvimento da habilidade EF07CO02, que busca introduzir conceitos de depuração de código, e usar aplicativos disponíveis que permitem ao programador monitorar a execução de um programa, pará-lo e reiniciá-lo, ativar pontos de parada, entre outros, a Reco. EF07CO02 - Atividade #01, tem como sugestão a utilização da ferramenta *Google Colab*²³, tanto para execução dos códigos propostos quanto para a depuração do mesmo. Sendo uma ferramenta de fácil acesso e aprendizado. Para resolução da atividade, fica a critério do professor escolher algum código trabalhado nas atividades anteriores ou criar um novo para a atividade.

²³ <https://colab.research.google.com/>

A habilidade EF07CO03 tem o objetivo de analisar a proposição e os requisitos de um programa e identificar qual a estrutura de dados adequada a ser empregada. A Abstr. EF07CO03 - Atividade #01, propõe um desafio aos alunos (Tabela 6).

Tabela 6 - Atividade EF07CO03

Explique aos alunos que o pirata tem um baú com vários compartimentos, e cada compartimento contém uma certa quantidade de moedas. O pirata precisa saber quantas moedas ele tem no total e a média de moedas por compartimento.

Estrutura de Dados:

- Lista: A lista será usada para armazenar o número de moedas em cada compartimento.
- Operações: Somar todas as moedas e calcular a média de moedas por compartimento.

Desenvolvimento:

- Crie uma lista para armazenar o número de moedas em cada compartimento.
- Peça para os alunos inserirem manualmente a quantidade de moedas em 5 compartimentos.
- Escreva um algoritmo que some o número de moedas e calcule a média.

Fonte: Autor (2024)

Essa atividade de criar uma lista para somar e calcular a média de moedas em compartimentos atende ao objetivo de construir soluções computacionais para problemas ao ensinar os alunos a selecionar e aplicar estruturas de dados apropriadas e operações básicas. Ela incentiva a compreensão de como dados podem ser organizados e manipulados, além de promover a colaboração e a prática de algoritmos.

Para o desenvolvimento da habilidade EF07CO05, que possui o objetivo criar algoritmos fazendo uso da decomposição e do reúso no processo de solução de forma colaborativa e automatizá-los usando uma linguagem de programação. A Decomp. EF07CO05 - Atividade #01, consiste em um desafio para ajudar a organizar uma biblioteca, fomentando os conceitos de decomposição, como mostrado na Tabela 7.

Tabela 7 - Atividade EF07CO05

<p>Os alunos precisam organizar os livros da biblioteca da escola. Eles devem dividir os livros por gênero (Ficção, Não Ficção, Ciências, História, etc.) e, dentro de cada gênero, organizar os livros por título em ordem alfabética. O objetivo é criar um algoritmo para realizar essa tarefa, incentivando o uso da decomposição e do trabalho cooperativo.</p>
<p>A atividade ensina como dividir um problema maior em subtarefas simples que podem ser resolvidas separadamente, permitindo que diferentes grupos trabalhem em conjunto.</p>
<p>1. Divisão do Problema:</p> <ul style="list-style-type: none"> • Subproblema 1: Separar os livros por gênero. • Subproblema 2: Ordenar os livros de cada gênero em ordem alfabética pelo título. • Subproblema 3: Juntar os livros organizados de todos os gêneros em uma lista final.
<p>2. Reúso:</p> <ul style="list-style-type: none"> • O subproblema de ordenar os livros por título é repetido para cada gênero.
<p>3. Interfaces das Soluções:</p> <ul style="list-style-type: none"> • Separação dos livros por gênero recebe uma lista de todos os livros e retorna dicionários com os livros separados por gênero. • A ordenação recebe uma lista de livros de um gênero e retorna essa lista ordenada. • A junção combina todas as listas organizadas de cada gênero.
<p>4. Distribuição das Tarefas:</p> <ul style="list-style-type: none"> • Grupo 1: Separar os livros por gênero. • Grupo 2: Ordenar os livros dentro de cada gênero. • Grupo 3: Juntar todas as listas em uma lista final.
<p>5. Composição Coletiva:</p> <ul style="list-style-type: none"> • Ao final, todas as partes são unidas para criar uma lista de livros organizada e separada por gênero e título.
<p>6. Explicação:</p> <ul style="list-style-type: none"> • <code>separar_genros</code>: Agrupa os livros por gênero em dicionários. • <code>ordenar_por_titulo</code>: Ordena os livros de cada gênero pelo título. • <code>juntar_genros</code>: Combina todos os gêneros organizados em uma lista final.
<p>7. Tarefa para os Alunos:</p> <p>Os alunos devem criar listas de livros com diferentes gêneros e usar o algoritmo para organizá-los. Eles trabalharão em grupos, cada um implementando uma parte da solução, e depois combinarão tudo para gerar uma lista final organizada de todos os livros.</p>

A Decomp. EF07CO05 - Atividade #01 ensina os alunos a dividir um problema grande, como organizar livros por gênero e título, em subtarefas que podem ser resolvidas separadamente. Ao identificar subproblemas como separação, ordenação e junção de listas, os alunos aprendem a decompor tarefas complexas em partes gerenciáveis, com cada grupo contribuindo para uma etapa específica da solução. O reúso ocorre ao aplicar o mesmo método de ordenação para cada gênero, mostrando como operações podem ser repetidas com diferentes dados. A atividade reforça também a integração dos módulos para formar uma solução completa, promovendo o trabalho colaborativo e a construção de um algoritmo que, ao final, organiza os livros de forma sistemática e eficiente.

A habilidade EF08CO01 visa desenvolver a habilidade de construir soluções de problemas usando a técnica de recursão e automatizar tais soluções usando uma linguagem de programação. A Abstra. EF08CO01 - Atividade #01, utiliza do conceito clássico no ensino de recursão, a sequência de Fibonacci como mostrado na Tabela 8.

Tabela 8 - Atividade EF08CO01

Explique aos alunos que a recursão é uma técnica de programação onde uma função chama a si mesma para resolver subproblemas menores. Apresente o exemplo da sequência de Fibonacci, em que cada número é a soma dos dois números anteriores.

Exploração da Sequência de Fibonacci:

- Mostre o código fornecido e explique o que cada linha faz.
- Explique as condições base da função ($n == 0$ e $n == 1$) e como a função usa essas condições para calcular números maiores da sequência.

Atividade Prática:

- Passo 1: Peça aos alunos para modificar o código para que exiba os primeiros 15 números da sequência, em vez de 10.
- Passo 2: Eles devem testar o programa com valores maiores para n e observar o tempo de execução, discutindo o que acontece quando a função faz muitas chamadas recursivas.
- Passo 3: Proponha que calculem manualmente alguns números pequenos da sequência de Fibonacci para entender melhor como a função recursiva trabalha.

Reflexão:

- Questione os alunos sobre como o programa faz uso da recursão e o que acontece quando n aumenta.
- Discuta alternativas para otimizar o cálculo (como usar uma abordagem iterativa ou memorizar resultados), incentivando a reflexão sobre quando usar recursão ou uma outra abordagem.

Fonte: Autor (2024)

A atividade para trabalhar a habilidade EF08CO01 ajuda os alunos a entender o conceito de recursão e raciocínio indutivo. Eles veem como um problema maior pode ser resolvido ao resolver subproblemas menores, permitindo a construção de algoritmos que atuam em níveis de abstração mais altos.

Para desenvolver a habilidade EF08CO02, com o objetivo de desenvolver a capacidade de criar soluções de problemas para os quais seja adequado o uso de listas para descrever suas informações e automatizá-las usando uma linguagem de programação, a Decomp. EF08CO02 - Atividade #01 convida o aluno a usar listas para resolver um problema real, analisando como combinar duas listas ordenadas em uma única lista. Eles também terão a oportunidade de utilizar a recursão como técnica

opcional e identificar a importância do caso base e da convergência em um programa recursivo (Tabela 9).

Tabela 9 - Atividade EF08CO02

<p>Os alunos do oitavo ano estão encarregados de unir duas pilhas de cartas, cada uma já ordenada por número. O objetivo é juntar essas duas pilhas de forma que a pilha final também esteja em ordem. Eles deverão desenvolver uma solução para esse problema utilizando listas e, opcionalmente, recursão.</p>
<p>Problema:</p> <ul style="list-style-type: none"> • Você tem duas pilhas de cartas, cada uma ordenada em ordem crescente. Seu objetivo é criar um programa que junte as duas pilhas em uma única pilha, mantendo a ordem crescente. <p>Etapas:</p> <ol style="list-style-type: none"> 1. Compreensão do Problema: <ul style="list-style-type: none"> • A primeira pilha contém números ordenados de cartas. • A segunda pilha também contém números ordenados de cartas. • O objetivo é unir as duas pilhas de cartas em uma só, preservando a ordem dos números. 2. Usando Listas e Recursão: <ul style="list-style-type: none"> • Uma forma de resolver o problema é usar listas para representar as pilhas.* A recursão pode ser usada para comparar os elementos no topo das duas listas e construir a lista final, unindo as duas de maneira ordenada. 3. Casos Base e Convergência: <ul style="list-style-type: none"> • O caso base é quando uma das listas está vazia, pois nesse caso basta retornar a outra lista. • A convergência é garantida quando, a cada passo, removemos o menor elemento de uma das listas e o adicionamos à lista final.

Fonte: Autor (2024)

A Decomp. EF08CO02 - Atividade #01 visa ensinar os alunos a criar soluções para problemas que envolvem o uso de listas para armazenar e organizar informações, além de apresentar a recursão como uma técnica de resolução. Ao pedir que unam duas pilhas de cartas ordenadas, os alunos exploram conceitos fundamentais como o uso de listas para representar estruturas de dados ordenadas e a aplicação de recursão para construir uma lista final ordenada.

A habilidade EF08CO04 e EF09CO02, tem como foco em construir soluções computacionais de problemas de diferentes áreas do conhecimento, de forma individual e colaborativa, selecionando as estruturas de dados e técnicas adequadas. Para desenvolver essa habilidade a Decomp. EF08CO04 - Atividade #01, convida os alunos

a desenvolver uma solução computacional para registrar e classificar as temperaturas diárias de uma semana em uma cidade, como mostra a Tabela 10.

Tabela 10 - Atividade EF08CO04 e EF09CO02

Nesta atividade, você irá desenvolver uma solução computacional para registrar e classificar as temperaturas diárias de uma semana em uma cidade. Utilize listas como estrutura de dados para armazenar as informações. Ao final, você deverá manipular os dados para encontrar a maior e a menor temperatura da semana, além de calcular a média das temperaturas registradas.

Objetivos da Atividade:

- Aprender a selecionar a estrutura de dados adequada para armazenar informações.
- Desenvolver um programa que capture dados de entrada, manipule-os e forneça resultados úteis, como a temperatura mais alta, mais baixa e a média.
- Promover o trabalho individual e colaborativo na análise de dados e resolução do problema.

Etapas:

1. Compreensão do Problema:

- O programa deve armazenar as temperaturas de 7 dias.
- Ele deve determinar a maior e a menor temperatura.
- O programa também deve calcular a média das temperaturas.

2. Seleção da Estrutura de Dados:

- Utilizar listas para armazenar as temperaturas.
- Implementar funções que manipulam a lista para encontrar os valores necessários.

3. Implementação do Programa:

- Entrada: O aluno deverá inserir as temperaturas diárias.
- Saída: O programa deve exibir a maior, a menor e a média das temperaturas.

Fonte: Autor (2024)

A Decomp. EF08CO04 - Atividade #01 promove a compreensão de conceitos fundamentais de programação, a decomposição de problemas em etapas gerenciáveis e a colaboração entre os alunos, estimulando o pensamento crítico e o raciocínio algorítmico. Dessa forma, a atividade integra práticas de aprendizado individual e colaborativo, preparando os alunos para resolver problemas em diferentes áreas do conhecimento.

A última atividade elaborada desenvolve a habilidade EF09CO01, com objetivo de criar soluções de problemas para os quais seja adequado o uso de árvores e grafos para descrever suas informações e automatizá-las usando uma linguagem de programação. A Abstra. EF09CO01 - Atividade #01, convida o aluno a usar grafos para

representar mapas de cidades e a construir um algoritmo para encontrar o caminho entre duas cidades. O objetivo é aplicar o conceito de grafos para resolver problemas práticos do dia a dia, como navegação e busca de rotas, como exemplificado na Tabela 11.

Tabela 11 - Atividade EF09CO01

Dado um mapa de cidades conectadas por estradas (grafo), você deve encontrar o menor caminho de uma cidade de origem até uma cidade de destino. Cada cidade será representada como um nó do grafo, e cada estrada entre duas cidades será uma aresta. As arestas podem ter pesos, representando a distância ou tempo de viagem entre as cidades.

1. Compreensão do Problema:

- Representar as cidades e as estradas como um grafo, onde as cidades são os nós e as estradas são as arestas.
- Encontrar um caminho eficiente entre duas cidades.
- Implementar um algoritmo que percorra o grafo e encontre a rota.

2. Construindo o Grafo:

- Cada cidade será um nó do grafo.
- As estradas entre cidades serão as arestas do grafo.
- O algoritmo deve usar busca em largura (BFS) ou busca em profundidade (DFS) para encontrar o caminho.

Fonte: Autor (2024)

A atividade que desenvolve a habilidade EF09CO01 explora a representação de cidades e estradas em um grafo, em que cada cidade é um nó e cada estrada é uma aresta com peso. Ao implementar algoritmos de busca, como BFS (Busca em Largura) ou DFS (Busca em Profundidade), os alunos conseguem explorar o grafo para encontrar o menor caminho entre duas cidades. Esse processo requer a construção de uma solução computacional que usa grafos para organizar dados e algoritmos para automatizar a busca pela rota mais eficiente, alinhando-se perfeitamente com o objetivo proposto.

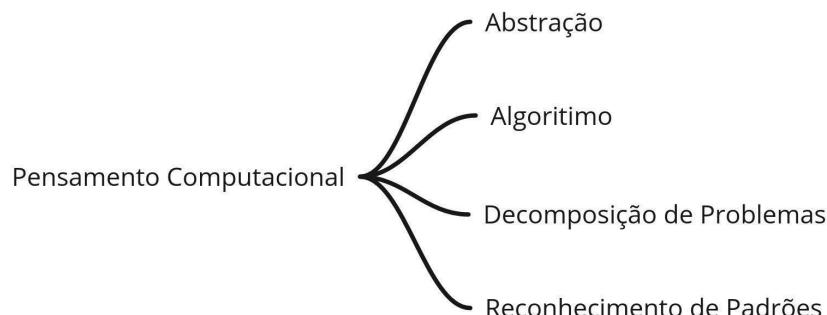
4.4 Elaboração da Taxonomia Hierárquica

Segundo Furgeri (2006), taxonomias são utilizadas em diversas áreas, incluindo a Ciência da Computação, para organizar informações destinadas aos usuários. Elas são consideradas ferramentas importantes para facilitar a recuperação de informações por meio da navegação. Para a elaboração da taxonomia, foram estabelecidos cânones e princípios para a organização e estruturação. De acordo com Campos e Gomes (2008), esses princípios orientam o processo de classificação, minimizando a subjetividade

inerente a qualquer processo classificatório. Além disso, definiu-se o Pensamento Computacional como a principal classe

Em relação a classe Pensamento Computacional, estabeleceu-se o Princípio da Ordem Alfabética para ordenar suas subclasses, sendo elas: Abstração, Algoritmos, Decomposição de Problemas e Reconhecimento de Padrões. Para a taxonomia, utilizou-se o Cânone da Extensão Decrescente, que ajuda a organizar as classes de forma hierárquica, das mais gerais às mais específicas. Além disso, utilizou-se o Cânone da Sequência Consistente, promovendo paralelismo, ou seja, as classes mantêm uma estrutura similar e ordenada ao longo da taxonomia (Figura 48).

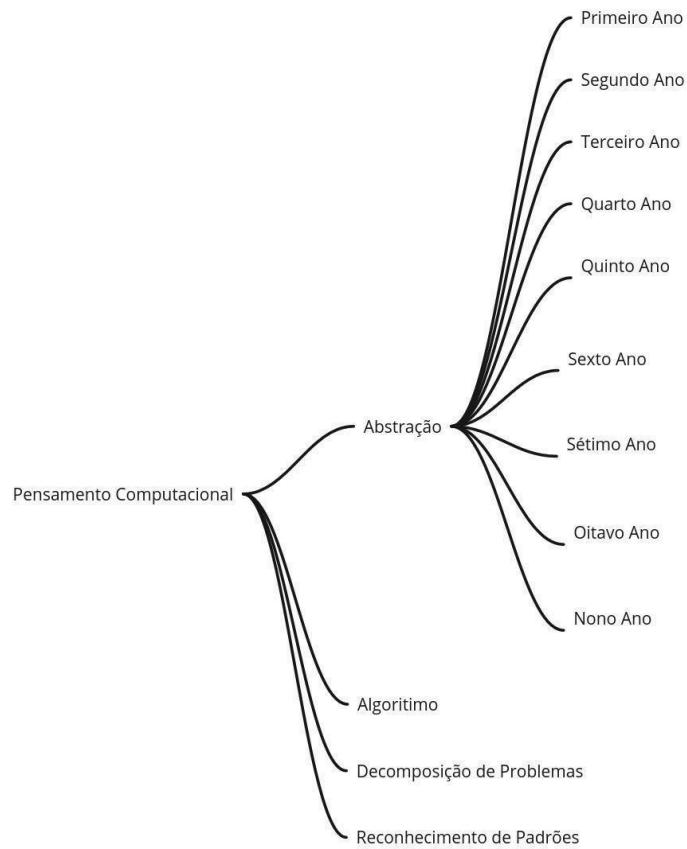
Figura 48 - Composição da classe Principal



Fonte: Autor (2024)

As classes Abstração, Algoritmos, Decomposição de Problemas e Reconhecimento de Padrões, respectivamente, possuem as subclasses; Primeiro, Segundo, Terceiro, Quarto, Quinto, Sexto, Sétimo, Oitavo e Nono Ano. Para a organização dessas subclasses, utilizou-se o Princípio da Complexidade Crescente, esse princípio orienta a taxonomia para que os tópicos avancem em complexidade conforme os anos escolares. Aplicou-se também o Cânone da Sequência Útil, garantindo que a ordem das classes tenha uma utilidade prática, facilitando o uso e a navegação para quem consulta a taxonomia. A Figura 49 exemplifica a composição da classe Abstração, para cada subclass de Pensamento Computacional o padrão de composição se mantém.

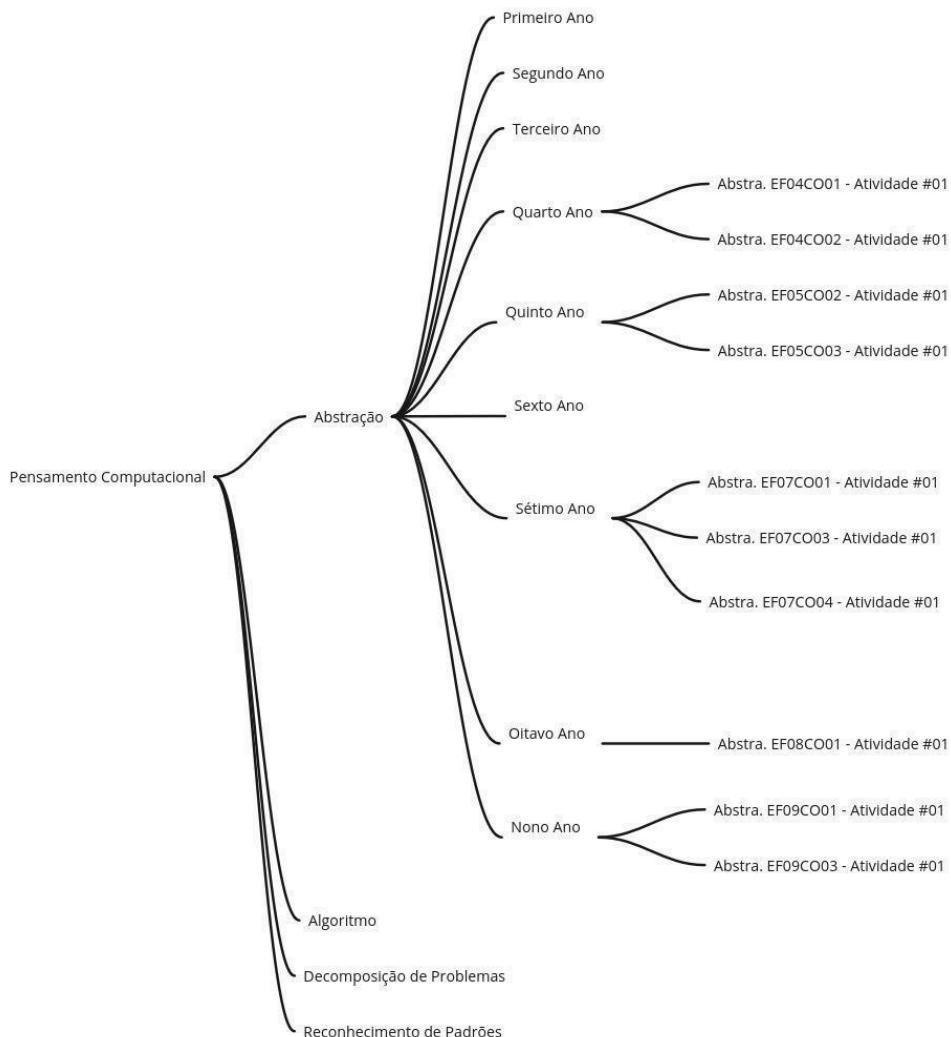
Figura 49 - Exemplo da composição da classe Abstração



Fonte: Autor (2024)

Para cada uma das classes, do Primeiro ao Nono Ano, há uma classe terminal composta pelas atividades identificadas e desenvolvidas ao longo deste trabalho. Essas classes terminais representam o nível mais específico da taxonomia, detalhando as atividades propostas para cada ano escolar. A Figura 50 mostra a distribuição e classificação das atividades da competência Abstração ao longo do ensino fundamental.

Figura 50 - Classificação das atividades Abstração

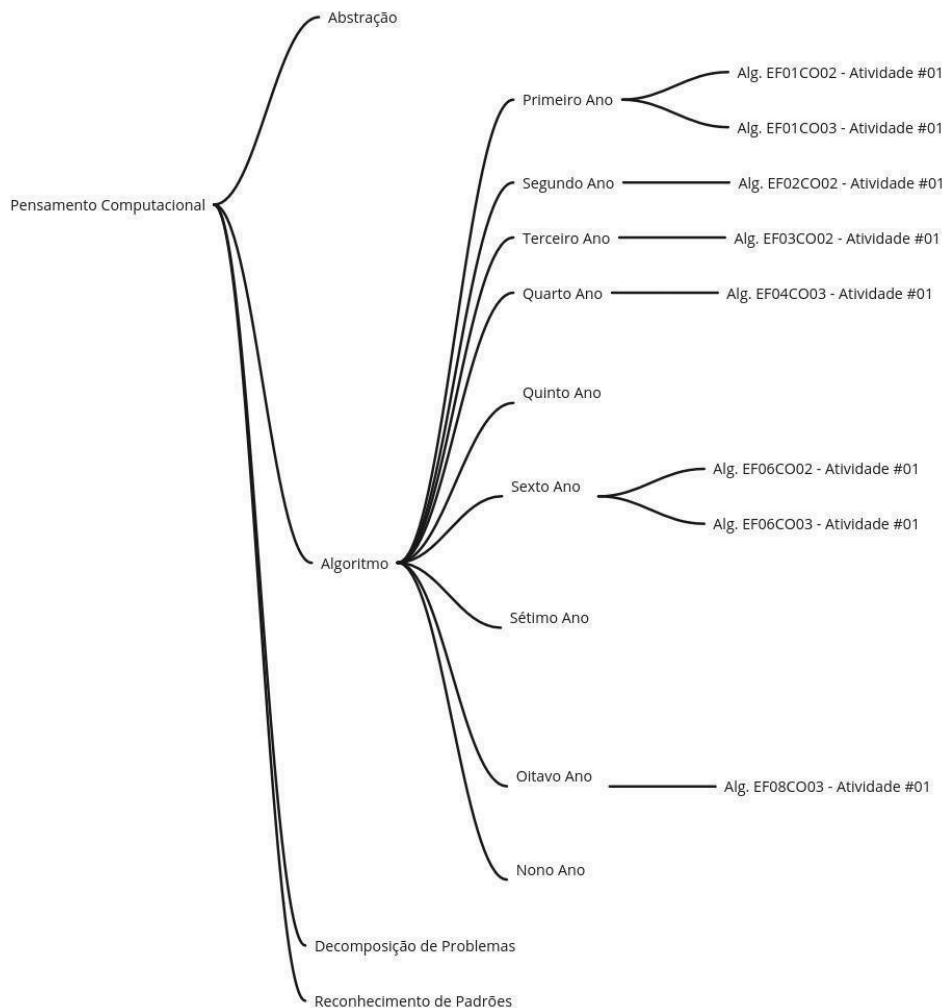


Fonte: Autor (2024)

A competência de abstração é introduzida a partir do quarto ano do ensino fundamental por ser uma habilidade que exige um nível mais avançado de desenvolvimento cognitivo. Ao atingir o quarto ano, os alunos começam a demonstrar uma maior habilidade para lidar com conceitos mais complexos e para analisar informações de forma estruturada. A introdução da abstração neste momento permite que eles começem a desenvolver uma compreensão mais profunda de como resolver problemas de forma eficiente, facilitando a organização de ideias e a simplificação de tarefas.

A Figura 51 exibe a classificação da competência Algoritmo, sendo introduzida no primeiro ano do ensino fundamental para que os alunos desenvolvam, desde cedo, habilidades essenciais de sequência lógica e ordenação de ações.

Figura 51 - Classificação das atividades Algoritmo



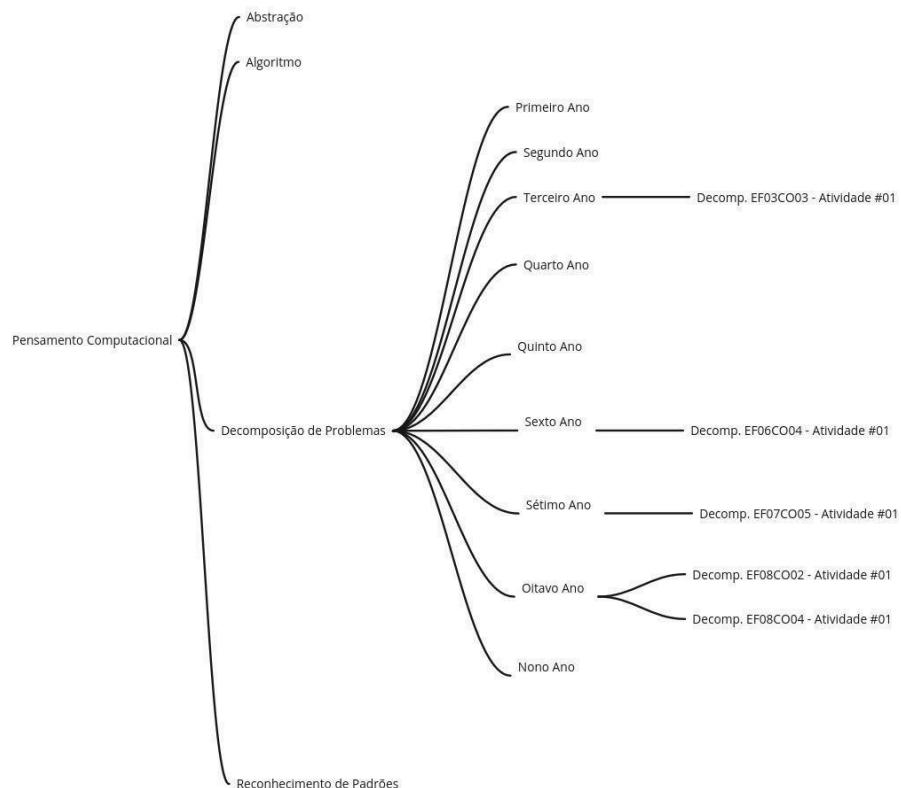
Fonte: Autor (2024)

Ensinar algoritmos no início da vida escolar é apropriado porque, nessa fase, os alunos conseguem compreender e aplicar noções básicas de sequência, instrução e repetição por meio de atividades concretas e práticas. Ao focar no ensino de algoritmos, é possível construir uma base sólida em que os alunos aprendem a descrever processos simples e a resolver problemas em etapas ordenadas. Esse foco inicial facilita a compreensão de tarefas mais complexas nas etapas seguintes, quando serão introduzidas habilidades mais abstratas, como a abstração e a decomposição de problemas.

O ensino da competência Decomposição de Problemas é introduzida no terceiro ano do ensino fundamental para que os alunos comecem a desenvolver habilidades básicas de segmentação de tarefas, essenciais ao pensamento computacional. Nessa fase entre 7 e 9 anos, segundo Piaget (1992), as crianças ainda estão em um estágio inicial

de desenvolvimento cognitivo, no qual podem entender problemas concretos, mas apresentam dificuldades com abstrações mais complexas. A Figura 52 exibe a classificação das atividades da classe Decomposição de Problemas.

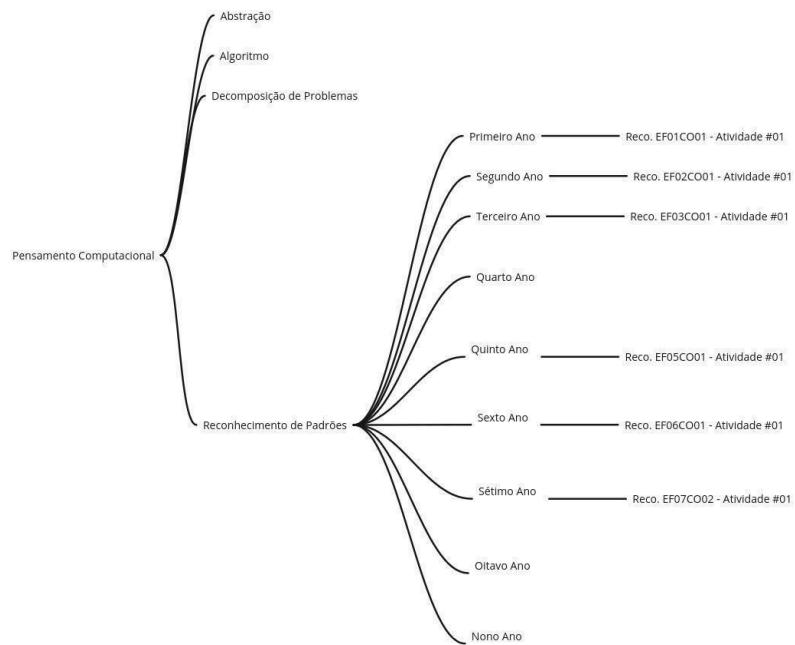
Figura 52 - Classificação das atividades Decomposição de Problemas



Fonte: Autor (2024)

A classificação das atividades referentes à competência Reconhecimento de Padrões (Figura 53), mostra o ensino da competência já no primeiro ano do ensino fundamental, essa habilidade é essencial para o desenvolvimento inicial do pensamento computacional e está alinhada com o estágio cognitivo das crianças mais jovens. De acordo com Piaget (1992), crianças de 6 a 7 anos já conseguem identificar e responder a padrões básicos, o que contribui para habilidades de categorização e organização de informações.

Figura 53 - Classificação das atividades Reconhecimento de Padrões



Fonte: Autor (2024)

A taxonomia representa um pilar fundamental na engenharia de requisitos. Ao estruturar o processo de ensino-aprendizagem com base nos pilares do pensamento computacional, os educadores podem realizar uma análise mais precisa das necessidades e capacidades dos alunos, garantindo que as atividades propostas sejam desafiadoras e significativas. Essa abordagem permite a construção de um plano de ensino mais eficaz, facilitando a identificação de lacunas no aprendizado e a adaptação das atividades. Além disso, a taxonomia auxilia na criação de atividades que preparem os alunos para o futuro, desenvolvendo habilidades essenciais em um mundo cada vez mais tecnológico.

5. CONSIDERAÇÕES FINAIS

A taxonomia proposta organiza e classifica atividades plugadas, alinhando-se às diretrizes educacionais e habilidades específicas da BNCC, com o intuito de facilitar a compreensão, ensino e prática de conceitos fundamentais como algoritmos, abstração, decomposição e reconhecimento de padrões.

Ao longo do trabalho, observou-se que a inclusão do Pensamento Computacional no currículo escolar brasileiro, com a homologação da BNCC e do documento complementar BNCC Computação, reflete uma importante evolução no sistema educacional. Isso é particularmente relevante em uma sociedade cada vez mais mediada por tecnologias digitais, onde habilidades como resolução de problemas, abstração e análise crítica são cruciais. A taxonomia desenvolvida atende a essas demandas ao estruturar atividades práticas, didáticas e interativas, que ajudam a concretizar os princípios do Pensamento Computacional para os alunos desde os primeiros anos escolares até o ensino médio.

A metodologia aplicada no estudo compreendeu uma revisão sistemática da literatura, pesquisa documental na BNCC e análise de conteúdo de sites e artigos educacionais. Esse rigor metodológico assegurou a qualidade e relevância dos dados coletados e permitiu identificar práticas efetivas no ensino de Pensamento Computacional, além de desenvolver atividades que integram o uso de ferramentas como Scratch e Python. Essas ferramentas não apenas incentivam o aprendizado interativo, mas também promovem o envolvimento dos alunos de maneira lúdica, reforçando os conceitos de programação e resolução de problemas.

A taxonomia hierárquica, resultado deste estudo, facilita o processo de ensino ao classificar as atividades de forma progressiva e organizada. Cada competência da BNCC foi cuidadosamente associada a atividades específicas que visam consolidar o aprendizado dos conceitos fundamentais de computação. Essa estruturação não só orienta educadores na escolha das atividades mais apropriadas para cada nível de ensino, mas também oferece uma visão abrangente sobre o desenvolvimento das habilidades de Pensamento Computacional ao longo da educação básica. Dessa forma, a taxonomia criada preenche lacunas de materiais didáticos e colabora para uma formação educativa mais completa e alinhada às necessidades do século XXI.

Com este estudo pretende-se contribuir para com o ensino de computação na educação básica, de maneira a facilitar a localização das atividades de acordo com as habilidades, permitindo que os professores possam melhorar suas práticas pedagógicas de maneira a promover um ambiente controlado e estimulante. Observou-se também que, embora as atividades desplugadas tenham seu valor pedagógico, o uso de dispositivos eletrônicos e plataformas interativas como o *Scratch* enriquece o processo de aprendizado, tornando-o mais acessível e adaptado ao contexto digital.

A taxonomia desenvolvida ainda contribui para a Engenharia de Requisitos, organizando informações de forma lógica e prática, o que facilita a comunicação entre educadores e desenvolvedores de conteúdo. Essa abordagem permite que o ensino do Pensamento Computacional seja mais eficaz, pois transforma os objetivos educacionais em atividades práticas e direcionadas, que são facilmente aplicáveis em sala de aula. Com isso, espera-se que este estudo sirva como referência para futuros trabalhos, que possam expandir e adaptar a taxonomia de acordo com novas necessidades educacionais e avanços tecnológicos.

Durante o desenvolvimento deste trabalho, enfrentaram-se desafios relacionados à escassez de atividades plugadas sistematicamente alinhadas às habilidades da BNCC nos materiais pesquisados. Essa lacuna evidenciou a necessidade de criação e adaptação de atividades específicas para atender às diretrizes educacionais, exigindo esforço adicional na análise e na elaboração de propostas pedagógicas. Além disso, a limitada integração entre ferramentas plugadas e práticas de ensino mais amplas demonstrou ser um obstáculo para uma implementação uniforme no ambiente educacional.

Para estudos futuros, sugere-se expandir a pesquisa para abranger a aplicação prática das atividades propostas em diferentes contextos escolares, bem como a exploração de novas plataformas e tecnologias que possam enriquecer o ensino do pensamento computacional, considerando também abordagens híbridas que combinem ferramentas plugadas e desplugadas. Adicionalmente, propõe-se o desenvolvimento de um *software* para auxiliar os professores, que integre um banco de atividades plugadas alinhadas à BNCC, com recursos interativos, relatórios de desempenho e tutoriais, promovendo maior eficiência e padronização no ensino do Pensamento Computacional.

REFERÊNCIAS

- BORGES, Brunno de Oliveira. **Linguagem Python como ferramenta no Ensino Básico.** Orientador: Leonardo de Amorim e Silva. 2021. 84 f. Dissertação (Mestrado em Matemática) - Faculdade de Matemática, Universidade Federal do Triângulo Mineiro, Uberaba-MG, 2021.
- BRASIL. **Ministério da Educação.** Base Nacional Comum Curricular da Computação. Brasília, DF: MEC, 2022. Disponível em: <http://portal.mec.gov.br/docman/fevereiro-2022-pdf/236791-anexo-ao-parecer-cneceb-n-2-2022-bncc-computacao/file>. Acesso em: 19 out. 2024.
- BRAUN, V.; CLARKE, V. (2006). **Using thematic analysis in psychology. Qualitative Research in Psychology**, 3(2), 77-101. DOI: <https://doi.org/10.1191/1478088706qp063oa>
- CAMPOS, M. L. A.; GOMES, H. E. **Taxonomia e classificação: a categorização como princípio.** DataGramZero, João Pessoa, v. 9, n. 4, p. 1-22, 2008.
- CORMEN, Thomas H.; LEISERSON, Charles E.; RIVEST, Ronald L.; STEIN, Clifford. **Algoritmos: Teoria e Prática.** 3. ed. Rio de Janeiro: Elsevier, 2012.
- COSCARELLI; Carla Viana. Alfabetização e letramento digital. In: COSCARELLI, Carla; RIBEIRO, Ana Elisa (orgs). **Letramento digital: aspectos sociais e possibilidades pedagógicas.** 2. ed. Belo Horizonte: Ceale: Autêntica, 2007.
- FERREIRA, Ingrid Oara Lopes; TEIXEIRA, Franciele Santo; BINOTTO, Rosane Rossato. **O uso do Scratch para abordar grafos no Ensino Fundamental.** Simpósio Internacional de Tecnologias em Educação Matemática, [s. l.], 6 out. 2022.
- FERREIRA, Marcos Vinicius; GONÇALVES, Sabrina; REBOUÇAS, Ayla Dantas. **Utilizando o Sistema "Computação Plugada Ordenação Web" para Ensinar sobre Algoritmos de Ordenação na Educação Básica.** In: SIMPÓSIO BRASILEIRO DE COMPUTAÇÃO NA EDUCAÇÃO BÁSICA (SBC-EB), 1. , 2024, Porto Alegre/RS. Anais [...]. Porto Alegre: Sociedade Brasileira de Computação, 2024 . p. 151-155. DOI: <https://doi.org/10.5753/sbceb.2024.1740>.
- FURGERI, Sérgio. **Representação de Informação e Conhecimento: Estudo Entre Diferentes Abordagens Entre a Ciência da Informação e Ciência da Computação.** 2006. 161 f. Dissertação (Mestrado em Computação) - Pontifícia Universidade Católica de Campinas, [S. l.], 2006.
- GUARDA, G. F. **Um Framework pedagógico desplugado para a prática das habilidades do Pensamento Computacional no Ensino Fundamental.** 2022. 141 f. Tese (Doutorado em Ciências, Tecnologias e Inclusão). Instituto de Biologia, Universidade Federal Fluminense, Niterói, 2022. Disponível em: https://bdtd.ibict.br/vufind/Record/UFF-2_3baf57867fc85dd19c36e8c8282c573c. Acesso em: 20 out. 2024.
- KENSKI, Ivani M. Cultura Digital. In: MILL, Daniel. **Dicionário crítico de Educação e tecnologias e de educação a distância.** Campinas, SP: Papirus, 2018. p. 139-144.

KOTONYA, G.; SOMMERVILLE, I. **Requirements engineering with viewpoints**. Software Engineering Journal, [s.l.], p. 5 18, 1996.

LIUKAS, Linda. **Hello Ruby: Adventures in Coding**: 1. 1. ed. [S. l.]: Feiwel & Friends, 2015. 112 p. v. 1. ISBN 1250065003.

PADILHA, Luan; PRADO, Suzana Pereira do; ALMEIDA, Noeli Terezinha de; SOUZA, Suely Maria de; DANTAS, Sérgio Carrazedo. **Pensamento Computacional: A Construção do jogo Pac-Man No Scratch**. Revista Eletrônica Debates Em Educação Científica E Tecnológica, [s. l.], v. 13, ed. 1, 6 dez. 2023. DOI <https://doi.org/10.36524/dect.v13i1.2455>. Disponível em: <https://ojs.ifes.edu.br/index.php/dect/article/view/2455>. Acesso em: 10 nov. 2024.

PIAGET, Jean. **The Origins of Intelligence in Children**. 2. ed. [S. l.]: Intl Universities Pr Inc, 1992. 419 p. ISBN 0823682072.

PRESSMAN, Roger S.; MAXIM, Bruce R. **Engenharia de Software: Uma Abordagem Profissional**. 8. ed. [S. l.]: AMGH, 2016. 968 p. ISBN 8580555337.

RESNICK, M., MALONEY, J., MONROY-HERNÁNDEZ, A., RUSK, N., EASTMOND, E., BRENNAN, K., ... & KAFAI, Y. (2009). **Scratch: Programming for All**. Communications of the ACM, 52(11), 60-67.

SOMMERVILLE, Ian. **Engenharia de Software**. 9. ed. [S. l.]: Pearson Universidades, 2011. 529 p. ISBN 8579361087.

WING, Jeannette. **Computational Thinking**. ACM Digital Library, [s. l.], 2006. DOI 10.1145/1118178.1118215. Disponível em: <https://dl.acm.org/doi/10.1145/1118178.1118215>. Acesso em: 10 nov. 2024.

WING, J. **Computational thinking. What and why?** Carnegie-Mellon School of Computer Science Research Notebook (Mar. 2011). <https://www.cs.cmu.edu/link/research-notebookcomputational-thinkingwhat-and-why>. Acesso em: 19 out. 2024.

APÊNDICE A - Exemplo de código Reco. EF06CO01 - Atividade #01

```
# Criação de listas vazias
inteiro = []
real = []
string = []
dados = [10, "maçã", 3.5, "banana", 25, 4.2, "gato"]

# Organize os itens de "dados" nas listas apropriadas
for item in dados:
    if type(item) == int:
        inteoiro.append(item)
    elif type(item) == float:
        real.append(item)
    elif type(item) == str:
        string.append(item)

# Exiba os resultados
print(f"Lista de inteiros {inteiro}")
print(f"Lista de reais {real}")
print(f"Lista de strings {string}")
```

APÊNDICE B - Exemplo de código Alg. EF06CO02 - Atividade #01

O primeiro exemplo trabalha com o laço de repetição *while*, esse laço é utilizado normalmente quando não se sabe a quantidade exata de interações sobre um determinado item.

```
#Lista de notas e variáveis
valor_da_soma = 0
contador = 0
quantidade_de_alunos = 3
notas = [5.5, 10, 8.5]

#Laço de repetição - Inicia um contador valendo 0, após cada intereração do laço, soma + 1 ao contador
while(contador != 3):
    valor_da_soma = valor_da_soma + notas[contador]
    contador = contador + 1;

#Calculo da média
media = valor_da_soma / quantidade_de_alunos

#Exibir média
print(f"A média das provas é {media}")
```

O segundo exemplo trabalha com o laço de repetição *for*, esse laço é utilizado normalmente quando se sabe a quantidade exata de interações sobre um determinado item.

```
#Lista de notas e variáveis
valor_da_soma = 0
quantidade_de_alunos = 5
notas = [8, 6, 4.5, 10, 8.5]

#Laço de repetição - Para cada nota contida na lista de notas, soma-se a nota + o valor da soma, que começa inicialmente com 0.
for nota in notas:
    valor_da_soma = valor_da_soma + nota
```

```
#Calcula a média das provas - Valor da soma dividido pela
quantidade de alunos
media = valor_da_soma / quantidade_de_alunos

#Exibir média das provas
print(f"A média das provas é {media}")
```

APÊNDICE C - Exemplo de código Alg. EF06CO03 - Atividade #01

```
#Função para verificar se há Ás na pilha de cartas e remover
def remover_carta_e_verificar(pilha):
    if len(pilha) == 0:
        return "Não há Ás na pilha."
    if pilha[0] == "Ás de Ouros" or pilha[0] == "Ás de Copas" or pilha[0] == "Ás de Espadas" or pilha[0] == "Ás de Paus":
        return "Há Ás na pilha."
    else:
        pilha.pop(0) # Remove a primeira carta da pilha
        return "Carta removida."

#Inicializando a pilha de cartas
pilha_de_cartas = ["3 de Espadas", "Ás de Ouros", "5 de Paus", "2 de Copas"]

#Verificando a pilha de cartas
resultado = remover_carta_e_verificar(pilha_de_cartas)
print(resultado)
```

APÊNDICE D - Exemplo de código Decomp. EF06CO04 - Atividade #01

O primeiro exemplo de código é pertinente a Figura 46.

```
#Variáveis do triângulo
base_triangulo = 50
altura_triangulo = 30

#Variáveis do retângulo
base_retangulo = 50
altura_retangulo = 30

#Calculando área do triângulo
area_triangulo = base_triangulo * altura_triangulo / 2

#Calculando área do retângulo
area_retangulo = base_retangulo * altura_retangulo

#Calculando área total
area_total = area_triangulo + area_retangulo

#Exibindo as áreas da figura
print(f"Área do triângulo {area_triangulo} cm²")
print(f"Área do retângulo {area_retangulo} cm²")
print(f"Área total {area_total} cm²")
```

O segundo exemplo de código é pertinente a Figura 47.

```
#Variáveis do triângulo ABC
base_abc = 20
altura_abc = 30

#Variáveis do triângulo CDE
base_cde = 20
altura_cde = 20

#Variáveis do triângulo HFG
base_hfg = 40
```

```
altura_hfg = 10

#Variáveis do retângulo
base_retangulo = 40
altura_retangulo = 30

#Área dos triângulos
area_abc = base_abc * altura_abc / 2

area_cde = base_cde * altura_cde / 2

area_hfg = base_hfg * altura_hfg / 2

area_retangulo = base_retangulo * altura_retangulo

#Área total
area_total = area_abc + area_cde + area_hfg + area_retangulo

#Exibindo as áreas da figura
print(f"Área do triângulo ABC {area_abc} cm²")
print(f"Área do triângulo CDE {area_cde} cm²")
print(f"Área do triângulo HFG {area_hfg} cm²")
print(f"Área do retângulo {area_retangulo} cm²")
print(f"Área total {area_total} cm²")
```

APÊNDICE E - Exemplo de código Abstra. EF07CO01 - Atividade #01

```

#Listas para armazenar os dados dos animais
nomes = []
tipos = []
idades = []

#Função para cadastrar um novo animal
def cadastrar_animal():
    nome = input("Digite o nome do animal: ")
    tipo = input("Digite o tipo do animal (cachorro, gato, pássaro, etc.): ")
    idade = int(input("Digite a idade do animal: "))

    nomes.append(nome)
    tipos.append(tipo)
    idades.append(idade)
    print("Animal cadastrado com sucesso!\n")

#Função para consultar animais por tipo
def consultar_animal_por_tipo(tipo_consulta):
    print(f"Animais do tipo {tipo_consulta}:")
    for i in range(len(tipos)):
        if tipos[i].lower() == tipo_consulta.lower():
            print(f"Nome: {nomes[i]}, Idade: {idades[i]} anos")
    print()

#Menu principal
while True:
    print("Menu:")
    print("1. Cadastrar um animal")
    print("2. Consultar animais por tipo")
    print("3. Exibir todos os animais cadastrados")
    print("4. Sair")

    escolha = input("Escolha uma opção: ")

```

```
if escolha == "1":  
    cadastrar_animal()  
elif escolha == "2":  
    tipo_consulta = input("Digite o tipo de animal que deseja  
consultar: ")  
    consultar_animal_por_tipo(tipo_consulta)  
elif escolha == "3":  
    print("Todos os animais cadastrados:")  
    for i in range(len(nomes)):  
        print(f"Nome: {nomes[i]}, Tipo: {tipos[i]}, Idade:  
{idades[i]} anos")  
    print()  
elif escolha == "4":  
    print("Saindo do programa.")  
    break  
else:  
    print("Opção inválida! Tente novamente.\n")
```

APÊNDICE F - Exemplo de código Abstra. EF07CO03 - Atividade #01

```
#Incializando a lista de moedas em cada compartimento
compartimentos = [10, 15, 8, 20, 12]

#Calculando o total de moedas
total_moedas = sum(compartimentos)

#Calculando a média
media_moedas = total_moedas / len(compartimentos)

#Exibindo as informações
print(f"Total de moedas: {total_moedas}")
print(f"Média de moedas por compartimento: {media_moedas:.2f}")
```

APÊNDICE G - Exemplo de código Abstra. EF07CO05 - Atividade #01

```
#Função para separar os livros por gênero
def separar_generos(livros):
    generos = {}
    for livro in livros:
        genero = livro['genero']
        if genero not in generos:
            generos[genero] = []
        generos[genero].append(livro)
    return generos

#Função para ordenar os livros de um gênero por título
def ordenar_por_titulo(livros):
    return sorted(livros, key=lambda x: x['titulo'])

#Função para juntar todos os livros organizados
def juntar_generos(generos_organizados):
    lista_final = []
    for genero in generos_organizados:
        lista_final.extend(generos_organizados[genero])
    return lista_final

#Lista de livros (exemplo)
livros = [
    {'titulo': 'Aventuras na Floresta', 'genero': 'Ficção'},
    {'titulo': 'A História do Mundo', 'genero': 'História'},
    {'titulo': 'Zoologia Básica', 'genero': 'Ciências'},
    {'titulo': 'Física para Todos', 'genero': 'Ciências'},
    {'titulo': 'Mistérios do Passado', 'genero': 'Ficção'},
]

#Separar os livros por gênero
generos = separar_generos(livros)

#Ordenar os livros dentro de cada gênero
```

```
generos_organizados = {genero:
ordenar_por_titulo(generos[genero]) for genero in generos}

#Juntar todos os livros organizados
lista_final = juntar_generos(generos_organizados)

#Exibir o resultado
for livro in lista_final:
    print(f"Título: {livro['titulo']}, Gênero:
{livro['genero']}")
```

APÊNDICE H - Exemplo de código Abstra. EF08CO01 - Atividade #01

```
#Função recursiva para calcular o n-ésimo número de Fibonacci
def fibonacci(n):
    if n == 0:
        return 0
    elif n == 1:
        return 1
    else:
        return fibonacci(n-1) + fibonacci(n-2)

#Exibir os primeiros 10 números da sequência de Fibonacci
for i in range(10):
    print(f"fibonacci({i}) = {fibonacci(i)}")
```

APÊNDICE I - Exemplo de código Decomp. EF08CO02 - Atividade #01

```
#Função para juntar duas listas ordenadas recursivamente
def juntar_pilhas(pilha1, pilha2):
    # Caso base: Se uma das pilhas está vazia, retorna a outra
    if not pilha1:
        return pilha2
    elif not pilha2:
        return pilha1

    #Comparar o primeiro elemento de cada pilha e
    recursivamente chamar a função
    if pilha1[0] < pilha2[0]:
        return [pilha1[0]] + juntar_pilhas(pilha1[1:], pilha2)
    else:
        return [pilha2[0]] + juntar_pilhas(pilha1, pilha2[1:])

#Duas pilhas de cartas ordenadas
pilha1 = [2, 5, 8]
pilha2 = [1, 3, 7, 9]

#Juntar as pilhas ordenadas
pilha_final = juntar_pilhas(pilha1, pilha2)

#Exibir a pilha final ordenada
print("Pilha final ordenada:", pilha_final)
```

APÊNDICE J - Exemplo de código Decomp. EF08CO04 - Atividade #01

```
#Função para encontrar a maior temperatura
def encontrar_maior(temperaturas):
    return max(temperaturas)

#Função para encontrar a menor temperatura
def encontrar_menor(temperaturas):
    return min(temperaturas)

#Função para calcular a média das temperaturas
def calcular_media(temperaturas):
    return sum(temperaturas) / len(temperaturas)

#Lista para armazenar as temperaturas dos 7 dias
temperaturas = []

#Entrada: Ler as temperaturas para cada dia
for i in range(7):
    temp = float(input(f"Digite a temperatura do dia {i+1}:"))
    temperaturas.append(temp)

#Saída: Exibir a maior, menor e a média das temperaturas
print("Maior temperatura:", encontrar_maior(temperaturas))
print("Menor temperatura:", encontrar_menor(temperaturas))
print("Temperatura média:", calcular_media(temperaturas))
```

APÊNDICE K - Exemplo de código Abstra. EF09CO01 - Atividade #01

```

from collections import deque

#Função para criar o grafo
def criar_grafo():
    grafo = {
        'A': ['B', 'C'],
        'B': ['A', 'D', 'E'],
        'C': ['A', 'F'],
        'D': ['B'],
        'E': ['B', 'F'],
        'F': ['C', 'E']
    }
    return grafo

#Função para encontrar o caminho entre duas cidades usando BFS
def bfs(grafo, inicio, destino):
    #Fila para armazenar os caminhos
    fila = deque([[inicio]])
    visitados = set()

    while fila:
        #Pegamos o caminho na frente da fila
        caminho = fila.popleft()
        cidade_atual = caminho[-1]

        #Verifica se chegamos no destino
        if cidade_atual == destino:
            return caminho

        #Se a cidade não foi visitada
        if cidade_atual not in visitados:
            vizinhos = grafo[cidade_atual]
            for vizinho in vizinhos:
                novo_caminho = list(caminho)
                novo_caminho.append(vizinho)
                fila.append(novo_caminho)
                visitados.add(cidade_atual)

```

```
#Cria um novo caminho
    novo_caminho.append(vizinho)
    fila.append(novo_caminho)

#Adiciona o novo caminho à fila

    visitados.add(cidade_atual)

return None

#Cria o grafo
grafo = criar_grafo()

#Entrada: Cidade inicial e destino
inicio = input("Digite a cidade inicial: ")
destino = input("Digite a cidade de destino: ")

#Busca o caminho
caminho = bfs(grafo, inicio, destino)

#Saída: Exibe o caminho encontrado
if caminho:
    print("Caminho encontrado: ", " -> ".join(caminho))
else:
    print("Não foi possível encontrar um caminho.")
```