Universidade Federal de Mato Grosso
Instituto de Ciências Exatas e da Terra
Departamento de Matemática

Pensamento computacional no ensino da Matemática: desafios e possibilidades

Juliano Thadeo Alves da Silva

Mestrado Profissional em Matemática: Profmat/SBM

Orientador: **Prof. Dr. André Krindges**

Trabalho financiado pela Capes

Cuiabá - MT

Dezembro de 2020

Pensamento computacional no ensino da Matemática: desafios e possibilidades

Este exemplar corresponde à redação final da dissertação, devidamente corrigida e defendida por Juliano Thadeo Alves da Silva e aprovada pela comissão julgadora.

Cuiabá, 08 de dezembro de 2020.

Prof. Dr. André Krindges
Orientador

Banca examinadora:

Prof. Dr. André Krindges
Prof. Dr. Nelcileno Virgílio de Souza Araújo
Prof. Dr. André Valente do Couto

Dissertação apresentada ao curso de Mestrado Profissional em Matemática – Profmat, da Universidade Federal de Mato Grosso, como requisito parcial para obtenção do título de **Mestre em Matemática**.

Dados Internacionais de Catalogação na Fonte.

S586p Silva, Juliano Thadeo Alves.
Pensamento computacional no ensino da Matemática: desafios e possibilidades / Juliano Thadeo Alves Silva. -- 2020
xvi, 78 f. : il. color. ; 30 cm.

Orientador: André Krindges.
Dissertação (mestrado profissional) – Universidade Federal de Mato Grosso, Instituto de Ciências Exatas e da Terra, Programa de Pós-Graduação Profissional em Matemática, Cuiabá, 2020.
Inclui bibliografia.

1. Algoritmo. 2. Atividades didáticas. 3. Construcionismo. I. Título.

Ficha catalográfica elaborada automaticamente de acordo com os dados fornecidos pelo(a) autor(a).

MINISTÉRIO DA EDUCAÇÃO

UNIVERSIDADE FEDERAL DE MATO GROSSO

PRÓ-REITORIA DE ENSINO DE PÓS-GRADUAÇÃO

PROGRAMA DE PÓS-GRADUAÇÃO EM MATEMÁTICA EM REDE NACIONAL - PROFMAT

AV. FERNANDO CORRÊA DA COSTA, 2367 - BOA ESPERANÇA - 78.060-900 - CUIABÁ/MT

FONE: (65) 3615-8576 – E-MAIL: PROFMAT@UFMT.BR

FOLHA DE APROVAÇÃO

Título**:** Pensamento computacional no ensino da Matemática: desafios e possibilidades

Autor: mestrando Juliano Thadeo Alves da Silva

Dissertação defendida e aprovada em 8 de dezembro de 2020.

COMPOSIÇÃO DA BANCA EXAMINADORA

1. **Doutor Andre Krindges** (Presidente Banca/orientador)

Instituição: Universidade Federal de Mato Grosso

2. **Doutor Nelcileno Virgílio de Souza Araújo** (Membro Interno)

Instituição: Universidade Federal de Mato Grosso

3. **Doutor André Valente do Couto** (Membro Externo)

Instituição: Instituto Federal de Mato Grosso

Cuiabá, **8/12/2020**.

Documento assinado eletronicamente por **Andre Krindegs**, **Usuário Externo**, em 08/12/2020, às 12:19, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por **André Valente do Couto**, **Usuário Externo**, em 08/12/2020, às 12:21, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por **Nelcileno Virgilio de Souza Araújo**, **Usuário Externo**, em 08/12/2020, às 12:22, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

A autenticidade deste documento pode ser conferida no site http://sei.ufmt.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0, informando o código verificador **3056412** e o código CRC **5C634A60**.

Referência: Processo nº 23108.091096/2020-66 SEI nº 3056412

À minha família.

Agradecimentos

Agradeço primeiramente a Deus, por me dar força e saúde para superar os obstáculos enfrentados ao longo de toda minha trajetória acadêmica, principalmente, por sempre estar presente na minha caminhada.

Aos meus pais, por me orientar e ensinar que nos estudos está a esperança de um futuro melhor. Em especial a minha mãe, Arlete Borges que se dedicou a educação de seus filhos, sendo um exemplo de dignidade, perseverança, amor à família. Aos meus irmãos Marcelo e Caroline, que me ajudaram em todos os momentos de minha vida.

À minha família, agradeço pela compreensão, paciência, apoio ao longo dessa jornada. Meu obrigado a minha esposa, Viviane, pelo suporte, companheirismo e incentivo durante todo esse percurso. Aos meus filhos Arthur, Ivia e Eros que me fazem querer ser uma pessoa melhor a cada dia.

Aos professores do Profmat pelo enriquecimento intelectual proporcionado pelas excelentes aulas. Aos meus colegas do Profmat pela parceria nos estudos tão imprescindíveis durante esse percurso.

Ao meu orientador, professor doutor André Krindges, a quem agradeço primeiro por ter aceito me orientar e depois pela paciência e compreensão no decorrer do percurso.

À SBM e UFMT, pela oportunidade de profissionalizar os educadores matemáticos de todo Brasil.

Ao coordenador do Profmat, professor doutor Geraldo L. Diniz, pela dedicação e profissionalismo que conduz as ações do Programa, contribuindo para a excelência do Profmat.

Agradeço aos membros da banca examinadora pelas contribuições e pela disposição em participar desse processo.

Muito obrigado a todos.

"Ensinar não é transferir conhecimento, mas criar as possibilidades para a sua própria produção ou a sua construção."

Paulo Freire.

Resumo

Neste trabalho é apresentada uma proposta diferenciada para se ensinar o pensamento computacional por meio de uma abordagem construcionista e a aplicação de metodologias ativas, como resolução de problemas e aprendizagem baseadas em jogos. Com o intuito de auxiliar os professores de Matemática que atuam na Educação Básica, se conduziu a pesquisa sobre o pensamento computacional de forma gradual. A partir de uma introdução que busca justificar a necessidade de se trabalhar o pensamento computacional, tendo em vista que a computação e as tecnologias que a permeiam estão no centro da economia e da vida cotidiana. Dessa forma, o pensamento computacional é essencial para que os alunos compreendam o mundo, não apenas como utilizadores de tecnologia, mas também com criadores. Assim, o pensamento computacional e a Ciência da Computação estão no cume das reformas curriculares da Educação Básica no mundo. Enquanto alguns países como o Brasil estão apenas na fase inicial dessa jornada, outros já possuem décadas de implantação do pensamento computacional e Ciências da Computação. Em muitos desses países, hoje o seu estudo é compulsório e com disciplina específica para a computação, pondo em igualdade com as outras disciplinas acadêmicas. O trabalho é finalizado com propostas de atividades, que começam com a utilização dos programas SpriteBox e LightBot, os quais levam o aluno a desenvolver conceitos básicos de programação de forma significativa e prazerosa, e posteriormente, são sugeridas atividades que podem ser trabalhadas no Ensino Fundamental I e II.

Palavras chave: Algoritmo, atividades didáticas, construcionismo

Abstract

This work presents a different proposal to teach computational thinking through a constructionist approach and the application of active methodologies, such as problem solving and game-based learning. In order to assist mathematics teachers working in basic education, research on computational thinking was conducted gradually. From an introduction that seeks to justify the need to work with computational thinking, considering that computing and the technologies that permeate it are at the center of economics and everyday life. Thus, computational thinking is essential for students to understand the world, not only as users of technology, but also with creators. Thus, computational thinking and Computer Science are at the top of the curriculum reforms of Basic Education in the world. While some countries like Brazil are only in the initial phase of this stage, others already have decades of implantation of computational thinking and Computer Science. In many of these countries, today their study is compulsory and with a specific discipline for computing, putting it on an equal footing with other academic disciplines. The work is finished with proposals for activities, which start with the use of the SpriteBox and LightBot programs, which lead the student to develop basic programming concepts in a meaningful and enjoyable way, and later, activities that can be worked on in Elementary School are suggested I and II.

Keywords: Algorithm, teaching activities, constructionism

Sumário

Lista de Figuras

Lista de Tabelas

Lista de abreviaturas

BNCC Base Nacional Comum Curricular

COOPE Instituto Alberto Luiz Coimbra de Pós-Graduação e Pesquisa de Engenharia

CC Ciência da Computação

CIEB Centro de Inovação para a Educação Brasileira

CSTA Computer Science Teacher Association

ESSA Every Student Succeeds Act

EUA Estados Unidos da América

FGV Fundação Getúlio Vargas

GBL Games Based Learning

ISO International Organization for Standardization

ISTE International Society for Technology in Education

LPP Linguagem de Projeto de Programação

OCDE Organisation for Economic Co-operation and Development

OCDE Organização para a Cooperação e Desenvolvimento Econômico

PBL Problem Based Learning

PC Pensamento Computacional

PISA Programme for International Student Assessment

SBC Sociedade Brasileira de Computação

SENAC Serviço Nacional de Aprendizagem Comercial

SMS Short Message Service

TDIC Tecnologias Digitais de Informação e Comunicação

TI Tecnologia da Informação

UFMT Universidade Federal de Mato Grosso

UFRGS Universidade Federal do Rio Grande do Sul

UFRJ Universidade Federal do Rio de Janeiro

Introdução

A sociedade contemporânea nos últimos anos vem vivenciando importantes transformações, cada vez mais temos convivido com os avanços tecnológicos, que tem na computação o eixo central de tais mudanças. Assim, os computadores se tornam ferramentas indispensáveis em quase todos os aspectos de nossa vida, evidenciando a necessidade da elaboração de estratégias que possibilitem aos estudantes o acesso aos conhecimentos em computação.

É notório que a utilização de recursos tecnológicos tem se tornado cada vez mais presente em nosso cotidiano, vivemos cercados por tecnologia e dispositivos digitais. Após a popularização dos computadores e da internet nas últimas décadas, atualmente, estamos vivendo o boom na aquisição dos smartphones. Segundo a pesquisa anual de administração e uso de tecnologia da informação, realizada pela FGVcia (2020), o Brasil tem hoje dois dispositivos digitais por habitante. E com isso, o uso de tecnologia tem mudado o dia a dia de grande parte da população brasileira. Refletindo por exemplo, a forma como as pessoas se comunicam: as ligações telefônicas podem ser substituídas por chamadas de vídeo, com o envio de arquivos, localização; e é claro há também reflexo no mundo do trabalho: como o desenvolvimento de um projeto, que atualmente pode ser feito em home office de maneira colaborativa, devido a diversas plataformas que permitem: reuniões, criação de documentos produzido em cooperação e permitindo interação dentre todos os participantes da equipe.

Certamente, as Tecnologias Digitais de Informação e Comunicação (TDIC) transformam e continuarão transformando o mundo e a força de trabalho profundamente. Como efeito disso, a Ciência da Computação e as tecnologias que ela permite, agora estão no cerne de nossa economia e da maneira como vivemos nossas vidas.

Assim, com o consumo cada vez maior de tecnologia e computação em distintos aspectos de nossa vida, precisamos refletir sobre como os jovens e adolescentes estão sendo

preparados para lidar com tais recursos. Para Computer Science Teacher Association CSTA (2011), com o cenário cada vez mais tecnológico é fundamental que o estudante do século 21 tenha conhecimento dos princípios e prática da Ciência da Computação, independente de seu campo de estudo e ocupação futura.

A Base Nacional Comum Curricular (BNCC) trouxe algumas diretrizes para auxiliar as redes de ensino na construção e revisão dos seus próprios currículos. A competência geral número 5 da BNCC, aponta para a necessidade de se trabalhar Tecnologias Digitais da Informação e Comunicação.

> Compreender, utilizar e criar tecnologias digitais de informação e comunicação de forma crítica, significativa, reflexiva e ética nas diversas práticas sociais (incluindo as escolares) para se comunicar, acessar e disseminar informações, produzir conhecimentos, resolver problemas e exercer protagonismo e autoria na vida pessoal e coletiva (Brasil – MEC, 2018).

Podemos verificar que a BNCC direciona para que os estudantes da Educação Básica sejam capazes não apenas de consumir e utilizar a TDIC, mas também sejam criadores de tecnologias digitais. Assim, coloca os estudantes como atores ativos, protagonistas e não apenas consumidores passivos de tecnologia. E nesse contexto, o desafio dos educadores é enorme, pois devemos questionar e refletir como contribuir para que essa competência geral seja alcançada. Prepará-los para as profissões que ainda nem existem, usar ou talvez inventar novas tecnologias digitais, e resolver problemas que ainda não conhecemos.

Propomos uma discussão em torno do processo de ensino e aprendizagem de competências, que caracterizam conhecimentos e habilidades no que diz respeito à computação e tecnologias digitais da informação e comunicação. Nesse sentido, os referenciais propostos pela BNCC destacam conhecimentos de Ciência da Computação (CC) em três dimensões: pensamento computacional, mundo digital e cultura digital. Possibilitando compreender como funciona e como criar tecnologias digitais, assim como o desenvolvimento de competências indispensáveis para a resolução de problemas.

Diante do exposto, essas três dimensões detalham conhecimentos de computação que consideramos importantes na formação dos estudantes. Buscando conceituar e alinhar as definições das dimensões, consultamos à BNCC e a Sociedade Brasileira de Computação (SBC), e podemos então definir:

– **Cultura digital** é a relação interdisciplinar da tecnologia com outras áreas do co-

nhecimento, buscando promover a fluência no uso do conhecimento computacional, ou seja, compreende as relações humanas mediadas por tecnologias e comunicações digitais;

- **Mundo digital** envolvem conhecimentos de componentes físicos e virtuais e assim, possamos transmitir a informação de maneira segura, compreendendo a importância de codificar, armazenar e proteger a informação. Logo, representa o conjunto de conhecimentos relacionados ao funcionamento dos computadores e suas tecnologias, em especial as redes e internet;

- **Pensamento computacional** envolve capacidade de compreender, sistematizar, representar, analisar e resolver problemas, por meio do desenvolvimento de algoritmo. O pensamento computacional se refere à capacidade de resolver problemas, utilizando para isso, conhecimento e prática de computação.

Haja vista, as definições das dimensões, chegamos ao objeto de estudo desse trabalho, o pensamento computacional e o seu principal foco a resolução de problemas, mediante a elaboração de algoritmo. Para chegar a resolução de um problema, o estudante requer entendimento sobre o assunto e para tal compreensão, os alunos precisam mobilizar conhecimentos e a capacidade de gerenciar informações, sistematizar, analisar, estabelecer estratégias e, dessa forma, utilizar mecanismos que são essenciais na busca por soluções de problemas. E todo esse processo possibilita ao estudante ampliar seu conhecimento matemático, bem como sua visão sobre os problemas, desenvolvendo sua autoconfiança e tornando-o cidadão crítico, ciente de suas responsabilidades sociais.

Podemos destacar como objetivos desta pesquisa, buscar introduzir e aplicar os conteúdos relativos ao pensamento computacional abordados na Educação Básica pela Base Nacional Comum Curricular; trazer propostas de atividades que possam ser aplicadas aos estudantes na construção de algoritmos, para solucionar situações vivenciadas em nosso cotidiano. Nesta dissertação, pretendemos fazer uso de tecnologias digitais para aprimorar os aspectos didáticos envolvendo novas ferramentas metodológicas, como a metodologia de jogos.

No primeiro capítulo, introduziremos um breve relato sobre o construcionismo e o pensamento computacional com seus quatro pilares. Buscando conceituar e mostrar a sua importância no desenvolvimento de competências como a resolução de problemas e,

assim, justificando a sua adoção na Educação Básica brasileira.

No capítulo seguinte, destacamos alguns benefícios de se trabalhar com o pensamento computacional, como por exemplo, a compreensão do mundo e a qualificação de profissionais para suprir a crescente demanda.

No terceiro capítulo, dialogamos sobre o panorama da adoção do pensamento computacional por países no mundo, com as ações de governos, empresas sem fins lucrativos e algumas iniciativas privadas.

No quarto capítulo, discorremos sobre algumas metodologias ativas que poderão ser trabalhadas para introduzir conceitos do pensamento computacional, como a aprendizagem baseada em jogos e resolução de problemas.

No quinto capítulo, falamos sobre o algoritmo, detalhando os conceitos, as características, forma de representá-lo (fluxograma, pseudocódigo) e apresentação do programa Visualg. Nele, é exposta a estrutura básica do algoritmo: declaração de variáveis, comandos de entrada e saída de dados, os desvios condicionais e as estruturas de repetição.

No sexto capítulo, destacamos sugestões de algumas propostas de atividades com o enfoque metodológico destacado neste trabalho, com o objetivo auxiliar o professor de Matemática na aplicação do pensamento computacional em sua sala de aula.

Este trabalho, portanto, procura apresentar ao leitor uma visão geral sobre o pensamento computacional: seus conceitos, benefícios, panorama mundial, sugestões de abordagens metodológicas e propostas de atividades que podem ser trabalhadas no ensino fundamental I e II.

Capítulo 1

Construcionismo e pensamento computacional

Em 1980, Seymour Papert, publicou o livro *"Mindstorms: children, computers, and powerful ideas"*, em português "Tempestades mentais: crianças, computadores e ideias poderosas", ele relata à ascensão do computador pessoal e quantas pessoas o idealizaram permeando a vida doméstica e os negócios. No entanto, Papert (1980) estava pensando além dessas funções, em "como os computadores podem afetar a maneira como as pessoas pensam e aprendem".

Papert, professor PhD em Matemática, foi um dos educadores mais conhecidos pelo uso do computador na educação, criador da linguagem de programação **Logo** em 1967, que procura introduzir conhecimentos de programação para crianças ou adolescentes. O Logo é um ambiente que envolve uma tartaruga gráfica, uma espécie de robô pronto para responder aos comandos do usuário. Sendo adotada em todo o mundo para uso de tecnologias digitais na educação. Alguns anos antes, no período de 1958 a 1963, Papert trabalhou em Genebra na Suíça, com Jean Piaget, um psicólogo do desenvolvimento e aprendizagem das crianças, mais conhecido por ser o pioneiro da teoria da aprendizagem conhecida como construtivismo: em que os alunos constroem novos conhecimentos a partir da interação de suas experiências com conhecimentos anteriores.

Por sua vez, por volta da década de 1980, Papert valendo-se de sua experiência, de suas pesquisas e dos estudos de outros autores, desenvolveu a teoria do construcionismo. Acrescentando à noção de que a aprendizagem é aprimorada quando o aluno constrói seu conhecimento a partir do "fazer", por intermédio de alguma ferramenta, como por

exemplo, o computador. Para ele a educação poderia se favorecer do computador para ensinar não apenas Matemática, mas qualquer disciplina. Assim, muitos conceitos que parecem longínquos do mundo real poderiam ser analisados e materializados por meio da programação. Para construir um algoritmo que resolva um problema é necessário que o estudante conheça o problema nos mínimos detalhes e consiga, a partir disso, criar um modelo de todo o processo de funcionamento, levando assim a uma concepção de conceitos e relações envolvidas.

Para Papert (1972), o computador não é apenas um equipamento para realizar a manipulação de símbolos ou meramente uma máquina instrucional, ele considera que o computador permite a construção do conhecimento por meio do aprender - fazendo e pensar sobre o que está fazendo, possibilitando uma ação reflexiva do estudante sobre um resultado alcançado e sobre o seu próprio pensamento.

No entanto, o termo pensamento computacional(PC) popularizou-se com a publicação de um artigo na revista *Communications of the ACM*, muito influente no âmbito acadêmico da Computação, por Jeannette Wing em 2006, em que ela trouxe um olhar como cientista da computação. Wing (2006), em seu texto, apresenta uma comparação da Ciência da Computação com as habilidades formativas básicas de leitura, escrita e aritmética, colocando-a na categoria de conhecimento básico. Assim, a Ciência da Computação é um conjunto de habilidades e atitudes que todos deveriam aprender a utilizar, ou seja, é uma habilidade fundamental para qualquer um, não apenas para o cientista da computação. Com ênfase na utilização de conceitos fundamentais da computação (abstração, decomposição, recursão, algoritmo, simulação, entre outros) para a resolução de problemas.

Assim, vale destacar que o termo pensamento computacional não pode ser confundido com a simples aptidão de manuseio de tecnologias digitais, o que poderia ser definido como alfabetismo digital. Logo pensamento computacional não se trata, por exemplo, de saber verificar ou enviar e-mail, Short Message Service (SMS) ou mensagem por aplicativo; realizar a digitação de um documento ou criar uma planilha eletrônica com fórmulas e gráficos. Blikstein (2008), traz um pensamento mais crítico sobre a utilização da TDIC na educação.

> Infelizmente, na maioria de nossas escolas, o que se faz é "adestramento digital" e ao custo de milhões de reais. Pior, estamos ensinando nossos alunos que a tecnologia serve para recombinar informações já existentes, e não para criar conhecimento novo. E o conhecimento novo não está na internet, facilmente encontrável em um mecanismo de busca com meia dúzia de palavras-chave. Ele está por ser descoberto (Blikstein, 2008).

Nesse sentido, e diante da complexidade nos processos utilizados pelas indústrias e comércios, o profissional precisa de habilidades e atitudes que vão muito além da alfabetização digital, precisam estar preparados para criar novas tecnologias e possibilidades. Quando é ensinado linguagem para um estudante do ensino básico, não temos a intenção de que ele seja um escritor, romancista, poeta, mas estamos propiciando habilidades essenciais para que ele seja um cidadão participativo e crítico. É senso comum, todos concordam que saber ler e escrever é fundamental, assim como as habilidades relacionadas à Matemática. O problema é que o mundo atual exige muito mais do que ler, escrever, adicionar e subtrair. Para Barba e Solomon (2016), ainda que a maioria das pessoas não queiram ser um cientista da computação, no entanto, todos podem usar computadores como uma extensão de sua mente, para experimentar, criar e descobrir.

Segundo Freire (2019), a educação deve atender as necessidades de independência da consciência do estudante, possibilitando a sua inserção na cultura e dando-lhe o direito de se apropriar das tecnologias, para que possa utilizá-las também como um instrumento de sua emancipação enquanto ser humano crítico.

Nesse sentido, a escola deve ser um local para trabalhar em prol da equidade, permitindo aos estudantes que não tenham acesso à tecnologia, consigam na escola desenvolver habilidades e conhecimentos necessários para apoderar-se dessas tecnologias. Oportunizando, o ensino democrático e, dessa forma, possibilitando que os alunos sejam capazes de agir no mundo cada vez mais digital. Equipar os jovens com a habilidade de enfrentar problemas complexos, com ferramentas inteligentes, irá equipá-los para uma vida inteira de contribuição na construção de uma sociedade mais justa e igualitária.

O pensamento computacional pode ser empregado tanto na resolução de problemas básicos do cotidiano, como a elaboração de uma lista de compras ou colocar as roupas sujas para lavar, quanto na resolução de problemas mais complexos, como por exemplo: quais rotas podem ser feitas para obtermos um menor deslocamento possível em entregas para uma empresa de logística. Segundo Blikstein (2008), pensamento computacional é saber utilizar o computador como um instrumento para aumentar o poder cognitivo e

operacional humano, aumentando a nossa produtividade, inventividade e criatividade.

Wing (2011), almejando criar uma definição mais clara para o termo PC, publicou outro artigo, que definiu assim: pensamento computacional são os processos de pensamentos envolvidos na formulação de problemas e suas soluções, para que estas sejam representadas de uma maneira que possam ser efetivamente executadas por um agente de processamento de informações.

Outra definição importante do termo pensamento computacional, foi na publicação da *International Society for Technology in Education*, ISTE (2016), que define o pensamento computacional como uma forma de desenvolver e empregar estratégias para compreender e resolver problemas de forma a aproveitar o poder dos métodos tecnológicos para desenvolver e testar soluções.

Para Brackmann (2017) em sua tese de doutorado da Universidade Federal do Rio Grande do Sul (UFRGS), propôs a seguinte definição para o termo Pensamento Computacional (PC):

> O pensamento computacional é uma distinta capacidade criativa, crítica e estratégica humana de saber utilizar os fundamentos da Computação, nas mais diversas áreas do conhecimento, com a finalidade de identificar e resolver problemas, de maneira individual ou colaborativa, através de passos claros, de tal forma que uma pessoa ou uma máquina possam executá-los eficazmente (Brackmann, 2017).

Segundo o Centro de Inovação para a Educação Brasileira, CIEB (2018), pensamento computacional refere-se à capacidade de resolver problemas a partir de conhecimentos e práticas da computação, englobando sistematizar, representar, analisar e resolver problemas.

A BNCC publicada em Brasil – MEC (2018), nos traz uma definição do termo pensamento computacional: "envolve as capacidades de compreender, analisar, definir, modelar, resolver, comparar e automatizar problemas e suas soluções, de forma metódica e sistemática, por meio do desenvolvimento de algoritmos".

É necessário utilizar o pensamento computacional para que possamos estruturar os problemas e encontrarmos soluções para os mesmos. Para nos auxiliar nesse processo, descreveremos os quatro pilares do PC: decomposição, reconhecimento de padrões, abstração e algoritmo. Assim, para se alcançar a resolução de problemas, torna-se necessário identificá-lo como um todo, dividir em partes menores e mais fáceis de solucionar (decomposição). Posteriormente, identificar nas partes menores os problemas que são

semelhantes, e assim, aplicar a mesma solução (reconhecimento de padrões), para isso devemos utilizar a capacidade abstração e identificar as informações que são realmente relevantes para o problema. Finalizando com os passos e etapas (algoritmo) que precisam ser executados para a solução de cada um dos subproblemas.

Na Figura 1.1, podemos observar os quatro pilares que formam a estrutura do pensamento computacional.

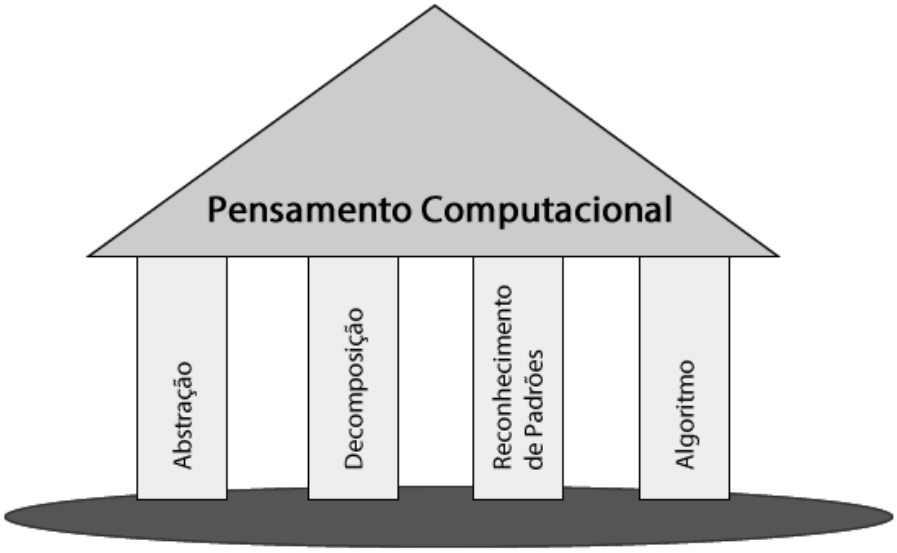


Figura 1.1: Os quatro pilares do pensamento computacional.

Todos os pilares têm a sua importância durante o percurso de resolução de um determinado problema, auxiliando para que alcancemos uma solução computacionalmente viável. Assim, resta nos esclarecer qual o papel de cada um dos pilares.

1.1 Abstração

Em nosso cotidiano, frequentemente precisamos organizar objetos ou processos de forma conveniente, para que se atinja o determinado objetivo, e a abstração é um elemento fundamental para isso. Assim, podemos denominar a abstração como uma ação de selecionar características de um objeto ou processo que devem ser considerados para satisfazer uma finalidade. A habilidade está em escolher as características corretas para que o problema se torne mais fácil, sem perder nada que seja importante.

Assim, esse pilar envolve a filtragem dos dados e sua classificação, desconsiderando elementos que não são essenciais e focando apenas nos que são relevantes, tornando os

problemas mais fácies de serem solucionados. Através dessa técnica, consegue-se criar uma representação abstrata do que se quer resolver. Segundo CAS (2014), o difícil é escolher o que será considerado irrelevante, para que o problema se torne mais fácil de ser compreendido, sem perder nenhuma informação relevante.

Podemos entender a abstração como uma etapa fundamental para alcançarmos a solução de um problema, tendo em vista que permite simplificar a realidade e representar apenas aspectos relevantes do problema. Segundo Raabe et al. (2020), no contexto do pensamento computacional, a abstração compreende os seguintes aspectos:

- Dados: informações de entrada e saída, que permitem descrever os dados envolvidos na solução de um problema;
- Processos: etapa e procedimento necessários para definir os algoritmos que descrevem a solução de um problema;
- Técnicas de construção de algoritmos: permitem construir a solução de problemas mais complexos.

Para Liukas (2016), a abstração é o processo de separar detalhes que não são necessários para se concentrar nas coisas que são precisas. Por exemplo, um mapa do metrô é uma abstração do mundo real e complexo, da mesma forma um calendário é uma abstração do tempo.

1.2 Decomposição

Muitas vezes, para chegarmos a solução de um problema é mais fácil quando o dividimos em subproblemas, ou melhor dizendo, em problemas menores. Liukas (2016) define a decomposição como um processo pelo qual os problemas são quebrados em partes menores. Ela exemplifica isto através da decomposição de refeições, receitas culinárias e as fases que compõem um jogo. Assim, os programadores costumam quebrar um problema complexo em partes menores, que são mais fáceis de entender e manter.

Contudo, o processo de decomposição pode ser utilizado em nosso cotidiano, como por exemplo, quando um professor precisa realizar um planejamento de uma aula. Se dividirmos o planejamento em partes, ele pode ser facilmente elaborado. Vejamos algumas possíveis divisão:

– Identificação do tema da aula;

– Definição de objetivos educacionais;

– Identificação dos conteúdos;

– Estabelecer a metodologia;

– Definir o cronograma;

– Levantamento do conhecimento prévio dos alunos;

– Proposta de atividades individuais e cooperativas;

– Planejamento da avaliação das aprendizagens.

Quando um problema não está decomposto, sua resolução é muito mais difícil. Segundo Brackmann (2017), quando a decomposição é aplicada a elementos físicos, como por exemplo, consertar uma bicicleta através da decomposição de suas partes, à manutenção torna-se mais fácil. Haja vista, que se o objeto não pudesse ser dividido em partes menores o conserto ficaria inviável, sendo somente possível a substituição da bicicleta como um todo.

Para Raabe et al. (2020), a decomposição é a técnica mais importante para solucionar um problema. E consiste em separar o problema em problemas menores, solucionando e depois combinando as soluções para chegarmos à solução original do problema. Na figura 1.2, podemos observar o processo de decomposição de um problema em problemas menores, até que todos os subproblemas sejam resolvidos e, assim, tenhamos a solução do problema como um todo.

Figura 1.2: Processo de decomposição - divisão e conquista.

Essa técnica em computação é conhecida como divisão e conquista, foi utilizada pela primeira vez por Anatolii Karatsuba em 1960 no algoritmo de karatsuba. Ele criou o primeiro método para a multiplicação de números, especialmente números grandes. Quebrando o problema em subproblemas que são similares ao problema original, recursivamente resolve os subproblemas e, finalmente, combina as soluções para resolver o problema original.

1.3 Reconhecimento de padrões

O hábito de identificar padrões nos acompanha desde o nosso nascimento. Trata-se de um processo de construção continuada e o nosso repertório de padrões é crescente. Assim, ao nos depararmos com um novo problema é muito comum adotarmos um padrão de comportamento para abordá-lo e, com isso, obtemos uma maneira de resolver novos problemas rapidamente com base em problemas anteriores que resolvemos. Desta forma, sempre que tivermos que resolver um problema novo, aplicamos os mesmos processos já utilizados em soluções genéricas. Podemos encontrar na literatura a associação de reconhecimento de padrões ao termo generalização.

Reconhecimento de padrões permite construir uma solução mais genérica a partir

de outra e, dessa forma, um algoritmo pode ser utilizado em outro contexto. Segundo Raabe et al. (2020), reutilizar e adaptar algoritmos é fundamental e exige um grande poder de abstração. Problemas que à primeira vista parece totalmente diferentes podem ser solucionados pelo mesmo algoritmo, apenas com pequenas adaptações.

Para Liukas (2016), reconhecimento de padrões é o processo de encontrar semelhanças e padrões para resolver problemas complexos com mais eficiência. Para encontrar padrões nos problemas procuramos coisas que são iguais ou muito semelhantes em cada problema. Assim, ao realizar à decomposição de um problema complexo, provavelmente encontraremos semelhanças entre os subproblemas gerados. Dessa forma, podemos entender o processo de reconhecimento de padrões como a associação de algum objeto ou parte dele, palpável ou não, com padrões familiares que permitam identificar e classificar.

Brackmann (2017), traz um exemplo prático da aplicação da técnica de reconhecimento de padrões, identificando semelhanças entre raça de cachorros. Eles possuem alguns atributos como os olhos, pelo, rabo, entretanto, suas características podem ser diferentes como a cor dos olhos, a tipologia da pelagem e o comprimento do rabo. No pensamento computacional, estas características são chamadas de padrões.

1.4 Algoritmo

A palavra algoritmo, segundo o dicionário Oxford é aplicada nas áreas de Matemática e Computação. Na esfera da Matemática é definida como sequência finita de regras, raciocínios ou operações, que aplicada a um número finito de dados, permite solucionar classes semelhantes de problemas. Na Computação, está associada a um conjunto de regras e procedimentos lógicos perfeitamente definidos que levam à solução de um problema, em um número finito de etapas.

Dessa forma, podemos definir um algoritmo como um conjunto de etapas específicas que você pode seguir para resolver um problema, sendo que para chegarmos à solução do problema já passamos pelo processo de abstração, decomposição e reconhecimento de padrões. O algoritmo é um pilar que agrega todos os demais, assim podemos entender como uma forma de chegar a uma solução por meio da definição clara das etapas necessárias - nada acontece por acaso.

Para Liukas (2016), algoritmo é um conjunto de passos específicos usado para

solucionar um problema e não deve ser confundido com o termo programa que foi definido como sendo "uma sequência de instruções precisas escritas em uma linguagem que computadores compreendam" (Liukas, 2016). O algoritmo pode ser entendido como sendo um conjunto de regras ou instruções que se forem seguidas com precisão, levam à resposta tanto do problema original, como de problemas similares. Em um algoritmo, as instruções são ordenadas e descritas para que seja atingido um determinado objetivo, podendo para tal, ser escrita em formato de diagrama ou pseudocódigo (linguagem humana).

Na escola, desde os anos iniciais aprendemos a utilizar algoritmos como por exemplo, algoritmo da adição, entre tantos outros. E se nós conseguirmos seguir os passos de execução do algoritmo, chegaremos a resposta para qualquer problema que necessite somar dois números quaisquer. Logo, se soubermos o algoritmo, não precisamos descobrir como realizar uma adição sempre que nos deparamos com um novo problema. E é isso que torna o algoritmo tão importante, a sua capacidade de automatizar as tarefas. Na Figura 1.3, é apresentado um exemplo de adição de dois números, em que é possível identificar uma sequência de passos necessários para atingir o resultado.

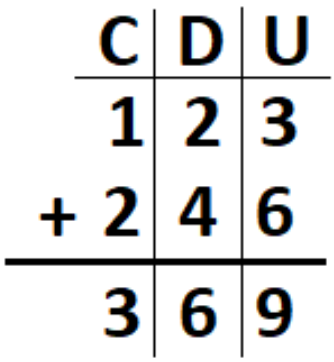


Figura 1.3: Algoritmo da adição.

Essencialmente, os algoritmos são procedimentos que realizados passo a passo e ordenadamente, pode solucionar um problema. Utilizado em diversas áreas do conhecimento, é uma estratégia muito importante, tendo em vista que o computador pode realizar comparações e operações matemáticas em uma velocidade gigantesca e, assim, com a utilização de um algoritmo adequado podemos chegar a resultados almejados.

Capítulo 2

Benefícios do pensamento computacional

Para Brackmann (2017), os esforços para tornar o pensamento computacional uma habilidade básica para qualquer pessoa surgiu devido a diferentes razões. Como por exemplo: emprego, compreender o mundo, alfabetização digital, produtividade, transversalidade em diferentes áreas, inclusão de minorias, trabalhar em equipe, entre outras.

Atualmente, empresas líderes no mercado de tecnologia necessitam cada vez mais de profissionais com conhecimento em computação. Profissões que antigamente não necessitavam de conhecimentos sólidos em computação como: Ciências, Engenharia, Matemática, Biologia e Medicina, hoje, precisam cada vez mais.

Nesse contexto, o tema tem despertado interesse de governantes e de empresas de tecnologias, como Microsoft, Google e Facebook; que procuram fomentar o pensamento computacional mediante diferentes iniciativas. Podemos citar um evento promovido pelo então presidente dos Estados Unidos da América (EUA), Barack Obama, *"Computer Science For All"* em 2016, que visava formar estudantes americanos em habilidades computacionais de que precisam para prosperar em uma economia digital.

Por outro lado, a comunidade acadêmica tem vivenciado uma urgência em preparar os estudantes para um mundo cada vez mais digital, em que é necessário não apenas utilizar os recursos tecnológicos como também ser capazes de compreender e criar novas tecnologias. Logo, iniciativas ligadas somente ao ensino de programação usando apenas os argumentos relacionados a preencher vagas no mercado de trabalho trata-se de uma visão muito limitada. Considerando que podemos também destacar como benefício do

pensamento computacional a compreensão do mundo.

Grover e Pea (2013), após ampla revisão bibliográfica, resumiram em nove elementos que o PC tende a atender para apoiar a aprendizagem dos alunos de forma interdisciplinar, bem como avaliar o seu desenvolvimento, no que se refere à:

- Abstração e reconhecimento de padrões (incluindo modelos e simulações);
- Processamento sistemático da informação;
- Sistema de símbolos e representações;
- Noções de controle de fluxo em algoritmos;
- Decomposição de problemas estruturados (modularização);
- Pensamento iterativo, recursivo e paralelo;
- Lógica condicional;
- Eficiência e restrições de desempenho;
- Depuração e detecção de erro sistemático.

O pensamento computacional está se tornando a nova alfabetização do século 21. Podemos dizer que se apropriando do pensamento computacional é possível adaptar a computação às nossas necessidades. Segundo Cuny et al. (2010), os benefícios do pensamento computacional para todos significam ser capaz de:

- Entender quais aspectos de um problema são passíveis de computação;
- Avaliar a correspondência entre as ferramentas e técnicas computacionais e um problema;
- Compreender as limitações e o poder das ferramentas e técnicas computacionais;
- Aplicar ou adaptar uma ferramenta ou técnica computacional para um novo uso;
- Reconhecer uma oportunidade de usar a computação de uma nova maneira;
- Aplicar estratégias computacionais como dividir e conquistar em qualquer domínio.

Capítulo 3

Panorama do pensamento computacional

Após os avanços tecnológicos e automatização ocorrida na indústria e na agricultura, estudos direcionam para que o próximo setor que será afetado é o de prestação de serviço. Segundo pesquisas publicadas pela Mckinsey e Company (2020) apontam que mais da metade dos postos de trabalho correm alto risco de serem automatizados na próxima década. E isso é um processo que está ocorrendo em todo o mundo, e temos como grande desafio a recolocação desses trabalhadores em profissões que estão surgindo, logo, o dilema que se apresenta é preparar esses trabalhadores e estudantes para essas novas profissões.

O interesse e entusiasmo em torno do pensamento computacional cresceu além da educação superior. Assim projetos mais recentes estão focados em incorporar o pensamento computacional desde o jardim de infância e por todo o Ensino Básico.

Em muitos países do mundo, o pensamento computacional e a Ciência da Computação estão no topo da agenda de reforma curricular na Educação Básica. Haja vista, que o uso de habilidades da área da Computação possui benefícios educacionais e econômicos, pois além de atender a demanda por profissionais com boa formação, permite que os alunos compreendam o mundo, desenvolvendo habilidades de reflexão e resolução de problemas.

Nos últimos anos, vários países tomaram medidas para incorporar de alguma forma a Ciência da Computação ou o pensamento computacional em seus currículos escolares. Como por exemplo, EUA, Canadá, Finlândia, Reino Unido, Austrália, Argentina,

Chile, África do Sul, Emirados Árabes, Iraque, Singapura, China, Coréia do Sul, Turquia e Portugal. Sendo que alguns deles, colocaram a Ciência da Computação em condições de igualdade com outras disciplinas acadêmicas, outros países buscam implementá-las em conjunto com outras disciplinas.

Nos Estados Unidos, conhecimentos relacionados com a computação, inicialmente eram trabalhados pela disciplina de Matemática, porém no ano de 2015 foi assinado a lei federal *Every Student Succeeds Act (ESSA)*, responsável pelas políticas públicas na área educacional no país. ESSA busca incluir disposições para ajudar a garantir o sucesso de alunos e escolas. Dentre diversos pontos, a lei coloca a Ciência da Computação em condições de igualdade com outras disciplinas acadêmicas, como Matemática, Geografia, História e Ciências.

No Reunido Unido desde 2014, o ensino de programação é obrigatório no Ensino Básico. Os alunos têm contato com a nova disciplina a partir dos cinco anos de idade, cujo objetivo nessa faixa etária não é aprender a programar, mas conhecer a lógica de programação para realizar tarefas simples. Na tabela 3.1, trouxemos um pequeno resumo da implementação da computação na modalidade do Ensino Básico, em diversos países.

Tabela 3.1: Adoção do pensamento computacional

País	**Ano**	**Ensino Fundamental**		**Ensino Médio**	
		Facultativo	**Compulsório**	**Facultativo**	**Compulsório**
Alemanha	2004	X		X	
Argentina	2015	X		X	
Austrália	2015		X		
Coréia do Sul	2007	X		X	
Estados Unidos	2015		X		X
Finlândia	2016		X		
França	2016		X		X
Grécia	1993		X		X
Iraque	1976	X		X	
Portugal	2012		X		
Reino Unido	2014		X	X	

Fonte: Adaptação de (Raabe et al., 2020, pg.42).

No Brasil, desde publicação da BNCC (2018), a Computação (pensamento computacional, mundo digital e a cultura digital) não aparece como uma disciplina, mas está elencada de uma maneira transversal, sendo principalmente citada na disciplina de Matemática. No entanto, a BNCC não detalha como será realizada essa implementação da computação nas disciplinas.

Em meio ao processo de aprimoramento da Educação Básica no Brasil, norteadas pelas propostas da BNCC, o Centro de Inovação para a Educação Brasileira está propondo um Currículo de Referência em Tecnologia e Computação para as escolas de Ensino Fundamental. Tal currículo, tem como objetivo apoiar as redes de ensino e as escolas incluir os temas tecnológicos e computacionais em suas propostas curriculares, buscando alinhar às competências gerais e às habilidades da BNCC. O Currículo de referência do Centro de Inovação para a Educação Brasileira (CIEB) está organizado em três eixos, como podemos observar na figura 3.1, cada um com seus conceitos.

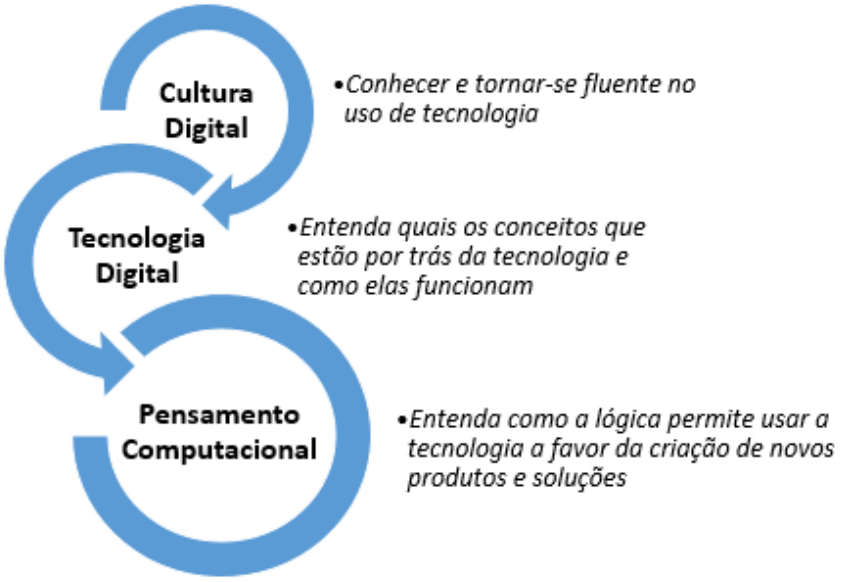


Figura 3.1: Três eixos tecnológicos. Fonte: Currículo de Referência CIEB (2018).

Iniciativas globais de organizações sem fins lucrativos, como o Code.org que organiza a Hora do Código, buscam promover à Ciência da Computação, por meio da introdução de conceitos de programação, em que os participantes buscam desenvolver um programa (código) em uma hora. O objetivo do programa é levar a CC até as escolas, desmistificando a programação e mostrando que qualquer pessoa pode aprender os fundamentos básicos desta ciência, começando com uma hora de código. O programa atende mais de 180 países e já participaram do evento mais de 100 milhões de alunos.

Iniciativas privadas, como por exemplo o Supergeeks, que foi a primeira escola de programação e robótica para crianças a partir dos 7 anos de idade. Atualmente o Supergeeks conta com 54 unidades ou franquias, suas aulas se baseiam na metodologia ativa para a criação de jogos, aplicações, robótica e empreendedorismo. Podemos destacar

também, Happy Code, Buddys, Crtl + Play, DragonByte, MadCode, entre outras tantas que buscam trabalhar conceitos de computação.

Em 2021, na próxima edição do Programme for International Student Assessment (PISA), o pensamento computacional já estará presente. O programa mede o desempenho dos estudantes de 15 anos, em três domínios básicos: leitura, matemática e ciências. Vale lembrar que o seu objetivo é ser uma avaliação comparativa que reflete a realidade da educação mundial e, principalmente, servindo para nortear as novas ações educacionais que permitam melhorar as práticas pedagógicas. E com isso, possibilitar a plena participação dos alunos em uma sociedade baseada no conhecimento e dependente das tecnologias digitais.

Segundo a Organização para a Cooperação e Desenvolvimento Econômico (OCDE), o PISA "analisa a capacidade dos alunos de aplicar conhecimentos e as habilidades de analisar, raciocinar e comunicar-se com eficácia à medida que examinam, interpretam e resolvem problemas" (OCDE, 2020). Dessa forma, o programa busca diagnosticar a capacidade dos alunos de não apenas repetir o que aprenderam na escola, mas empregar conhecimento e habilidade de forma criativa, interdisciplinar e prática.

O Brasil infelizmente não obteve nenhum avanço no desempenho dos estudantes na disciplina de Matemática no PISA, que é o ranking mais importante da educação mundial, na verdade, o resultado aponta para uma estagnação na última década. Segundo o resultado do PISA divulgado no dia 03/12/2019, o Brasil teve um ligeiro aumento da nota média, mas os estudantes brasileiros seguem entre os últimos 10 colocados na avaliação de matemática.

Isso nos serve de alerta para que possamos repensar as nossas práticas pedagógicas e currículos escolares e compreendermos que se faz necessário, urgentemente, a implementação de novas abordagens no processo de ensino aprendizagem. Para Backmann (2020), um dos obstáculos que deve ser superado para colocar em prática competências relacionadas à computação, baseia-se na formação de professores. E que alguns países como o EUA e a Alemanha, encontraram à solução para a falta de professores habilitados para lecionar a disciplina de computação nas aulas de Matemática, e esse modelo pode ser seguido pelo Brasil.

Segundo o referido autor, o próximo momento seria a oferta de disciplinas voltadas para a computação de maneira opcional, possibilitando o acesso dos estudantes aos

conceitos do pensamento computacional, para que em uma etapa posterior, essa disciplina faça parte do currículo obrigatório nacional. E conclui que desperdiçar essa oportunidade seria desfavorável para o nosso país e acarretaria um prejuízo incalculável para o nosso futuro, tendo em vista que as oportunidades de emprego exigem como requisito, um conjunto de habilidades técnicas inerentes à computação.

Capítulo 4

Metodologia de aprendizagem

A docência requer um constante processo de reflexão sobre sua prática pedagógica, gerando autocrítica sobre o fazer profissional, e assim, criando novas possibilidades e metodologias para que aconteça o processo de ensino e aprendizado.

De acordo com Moran e Bacich (2018), desde de que nascemos aprendemos a partir de situações concretas, que aos poucos conseguimos ampliar e generalizar (processo indutivo), e também aprendemos por meio de ideias ou teorias para testá-las depois no concreto (processo dedutivo). Nesse sentido, a aprendizagem por meio da transmissão é importante, no entanto, o processo de ensino aprendizagem por questionamento e experimentação é mais relevante para uma concepção mais ampla e profunda.

Freire (2019) reforça que "ensinar não é simplesmente transferir conhecimento, mas criar as possibilidades para a sua produção ou a sua construção", respeitando a autonomia do educando. De acordo com o próprio autor, "é preciso aprender a ser coerente. De nada adianta o discurso competente se a ação pedagógica é impermeável às mudanças". Diante do exposto, existe uma necessidade dos educadores de criarem as possibilidades para a produção ou construção do conhecimento pelos alunos. Dessa forma, o professor pesquisador deve repensar e buscar constantemente melhorias nas práticas que norteiam o campo da Matemática, como em qualquer outra área do conhecimento.

Mediante o cenário atual, com os avanços na TDIC e a popularização da internet, possibilitou que o acesso à informação aconteça em qualquer hora e em qualquer lugar. Assim, métodos tradicionais que privilegiam apenas à transmissão de informações pelos professores, não fazem mais sentido. Devemos buscar novas metodologias de ensino aprendizagem que se alicerçam na concepção do aluno como sujeito central e protagonista

de seu desenvolvimento educacional.

Nesse sentido, podemos caracterizar as metodologias ativas pela relação entre cultura, educação e sociedade, sendo desenvolvida por meio de métodos criativos e, principalmente, centrado na ação do fazer do aluno. Assim, se temos o objetivo de propiciar uma aprendizagem significativa, as metodologias devem estar alinhadas com os objetivos pretendidos. Se aspiramos que os alunos sejam proativos, necessitamos de práticas docentes pautadas em metodologias que tragam o aluno para o centro do processo pedagógico. Envolvendo-o em atividades relacionadas a resolução de problemas cada vez mais complexos, capacitado à tomar decisões e avaliar os resultados com apoio da TDIC, tornando-se assim, protagonista do processo de ensino e aprendizagem. Dentre as diversas metodologias ativas, destacamos aqui a aprendizagem baseada em jogos e a aprendizagem baseada em problemas por estarem mais alinhadas aos objetivos do nosso trabalho.

4.1 Aprendizagem baseada em jogos

Aprendizagem baseada em jogos (Games Based Learning (GBL)), trata-se de uma abordagem que utiliza os jogos, em sua perspectiva lúdica, que transformam os objetos de aprendizagem em atividades educacionais mais atraentes e prazerosas. Assim, com a ajuda dos jogos, os alunos são desafiados a testar limites, formular hipóteses, solucionar problemas, vencer etapas. Competindo sob normas que regem o jogo, e ditando como cada jogador deve se comportar para alcançar os objetivos propostos.

É importante destacar aqui a diferença entre Gamification, que é transformar o processo de aprendizagem como um todo em um jogo, e a aprendizagem baseada em jogos que utiliza um jogo como parte do processo de aprendizagem. Apesar da diferença, para o autor de The Gamification of Learning and instruction, (Kapp, 2012), o objetivo de ambos é o mesmo tentando resolver um problema, motivar e promover o aprendizado usando o pensamento e as técnicas baseadas em jogos.

Na aprendizagem baseada em jogos, existem diferentes formas de aplicação dos jogos em contextos pedagógicos, o docente pode utilizar os jogos nos formatos digitais (computadores, tablet e smartphones) ou físico, plugados ou desplugados, dependendo dos recursos disponíveis.

Cada vez mais presente nas escolas, essa estratégia de aprendizagem encanta e

motiva, tornando o processo mais rápido e prazeroso. A aprendizagem baseada em jogos é considerada uma forma eficaz de ensinar novos conceitos às pessoas, dada a natureza interativa dos jogos, os desafios, os níveis de dificuldades e a progressão. Assim, os jogos trazem as competências necessárias para cada etapa, dando informações indispensáveis para que os alunos consigam realizar determinada tarefa.

A área de educação tem elevado potencial de implementação desse conceito, tendo em vista que a GBL procura promover a motivação e o envolvimento dos alunos. No entanto, algumas questões que envolvem a aprendizagem baseada em jogos devem ser consideradas para determinar a abordagem mais apropriada e que maximize as experiências de aprendizagem. Nesse contexto, SENAC – Departamento Nacional (2018), nos traz um alerta que primeiramente é indispensável analisar o grau de contribuição dos jogos para o desenvolvimento do processo de aprendizagem, de modo que não sejam desinteressantes e inflexíveis a ponto de perder o seu caráter lúdico, tampouco, solto ou de forma descontextualizada que dificulte a reflexão sobre qual objeto de aprendizagem.

4.2 Aprendizagem baseada em problemas

A aprendizagem baseada em problemas, em inglês Problem Based Learning (PBL) é um método de aprendizagem centrado no aluno, busca mobilizá-lo a encontrarem possíveis soluções, quando estes se deparam com um problema. Em outras palavras, PBL é uma abordagem pedagógica que possibilita que os alunos aprendam enquanto se envolvem ativamente com problemas significativos. E, dessa forma, prepara os alunos para um mundo em constante mudança, que exige pessoas dispostas a resolver novos problemas.

Atualmente, a aprendizagem baseada em problemas tem sido amplamente utilizada em diversos campos e contextos educacionais buscando além da resolução de problemas em situações de aprendizagem reais, o desenvolvimento do pensamento crítico. Nessa direção, a BNCC enfatiza como uma das finalidades do Ensino Médio o aprimoramento do educando como pessoa humana, considerando sua formação ética, o desenvolvimento da autonomia intelectual e do pensamento crítico.

Assim, o desenvolvimento da independência propiciado por atividades de resoluções de problemas, poderá contribuir na formação de um ser humano crítico, explorador, criativo e autônomo em suas iniciativas. Dante (2003) destaca que estimular o

aluno a pensar, desenvolver o raciocínio lógico, ensinar a enfrentar situações novas, torna as aulas mais interessantes e motivadoras.

No entanto, é importante também deixar claro a diferenciação entre o exercício que serve para praticar determinado processo ou algoritmo e a atividade baseada em resolução de problemas, que envolve uma situação em que se busca algo desconhecido e não temos previamente nenhum algoritmo que garanta a solução. Assim, o processo de resolução de problemas exige muita criatividade, investigação, aliada ao conhecimento de algumas estratégias.

Segundo Dante (2003), os problemas propostos aos alunos precisam ser bem escolhidos, e para auxiliar devemos observar algumas características: ser desafiador, real e interessante, sendo que sua resolução pode ter um ou mais algoritmos. Devemos salientar ainda que o tempo deve ser o suficiente para que os alunos realizem o processo completo de resolução do problema. Logo, o professor não deve dar respostas diretas, precisa incentivar no aluno, o interesse pela investigação e resolução do problema. Nessa perspectiva didática, a força motriz da aprendizagem consiste em despertar o interesse do aluno a partir de situações cotidianas, de forma a provocar a reflexão.

Conforme a Base Nacional Comum Curricular, a Matemática é uma área do conhecimento, que dispõe de várias articulações com o intuito de garantir o relacionamento do aluno com o mundo real. No Ensino Fundamental, centraliza na compreensão de conceitos e procedimentos em seus diferentes campos e no desenvolvimento do pensamento computacional, visando à resolução e formulação de problemas em contextos diversos. No Ensino Médio, na área de Matemática e suas tecnologias, os estudantes devem consolidar os conhecimentos desenvolvidos na etapa anterior e agregar novos, ampliando o leque de recursos para resolver problemas mais complexos, que exijam maior reflexão e abstração.

O desenvolvimento dessas habilidades está intrinsecamente relacionado com os métodos de aprendizagem, estratégias de ensino e as situações da vida cotidiana. Nesse sentido, PBL utiliza um problema como ponto de partida, buscando motivação para que o aprendizado aconteça. O objetivo final não é apenas ter o problema resolvido, mas enfatizar todo o processo de aprendizagem, ou seja, os caminhos e etapas na busca de uma solução viável do problema.

Capítulo 5

Algoritmo

A Base Nacional Comum Curricular prevê a utilização do pensamento computacional e de seus pilares na disciplina de Matemática, auxiliando na investigação e resolução de problemas. Em contrapartida, a aprendizagem de álgebra pode contribuir para o desenvolvimento do PC dos alunos, considerando que eles devem ser capazes de traduzir uma determinada situação em outras linguagens, como por exemplo, língua materna, fórmulas, tabelas e gráficos e vice-versa.

> Associado ao pensamento computacional, cumpre salientar a importância dos algoritmos e de seus fluxogramas, que podem ser objetos de estudo nas aulas de Matemática. Um algoritmo é uma sequência finita de procedimentos que permite resolver um determinado problema. Assim, o algoritmo é a decomposição de um procedimento complexo em suas partes mais simples, relacionando-as e ordenando-as, e pode ser representado graficamente por um fluxograma (Brasil – MEC, 2018).

Embora tenhamos resolvido algoritmo por milhares de anos, fazer isso manualmente pode consumir uma grande quantidade de tempo e requerer muitos cálculos, dependendo da complexidade do problema. Segundo Mueller e Massaron (2018), os algoritmos dizem respeito a encontrar soluções, e quanto mais rápido e fácil, melhor. Assim, utilizar os computadores para resolver problemas empregando algoritmo apropriado acelera a tarefa significativamente, e é a razão pela qual o desenvolvimento de novos algoritmos progrediu tão rapidamente desde o surgimento de sistemas computacionais potentes.

De maneira geral, os sistemas computacionais são compostos por 3 etapas basicamente: entrada, processamento e saída. Dessa forma, para que seja realizada uma transformação nos dados de entrada, é preciso que ocorram procedimentos que realizem o processamento e retorne à informação desejada. Para descrever esses procedimentos

(processamento) podemos utilizar o algoritmo (representação textual) ou o fluxograma (representação gráfica).

Na criação de um algoritmo que resolva um determinado problema, os procedimentos, rotinas ou métodos devem ser bem definido e, desse modo, podemos identificar cinco características indispensáveis para criação de um algoritmo.

- **Finitude**: um algoritmo deve sempre terminar após um número finito de passos, ou seja, deve ter fim;
- **Assertividade**: as ações devem ser definidas rigorosamente e nunca de forma ambígua e, assim, não deve se dar margem à dupla interpretação;
- **Entrada**: é a capacidade de receber dados de entrada do mundo externo, logo, um algoritmo pode ter nenhum ou mais dados de entrada;
- **Saídas**: resultado do processamento do algoritmo, são os dados de saída para o mundo externo, podendo ter um ou mais dados de saída;
- **Efetividade**: as operações devem ser suficientemente simples de modo que possam ser, em princípio, executadas manualmente, assim, todas as etapas especificadas no algoritmo devem ser alcançáveis em um tempo finito.

5.1 Fluxograma

Fluxograma é um tipo de diagrama, empregado para representar um processo ou algoritmo, ou seja, é uma sequência operacional para o desenvolvimento de um determinado processo. Muitas vezes, é utilizado para modelar e documentar um sistema computacional.

Para Manzano et al. (2018), o fluxograma é um conjunto de símbolos (representação gráfica) que retratam todos os passos do algoritmo. Sendo que cada símbolo possui uma ação específica, dessa forma, os fluxogramas (diagrama de blocos) são compostos por símbolos que mostram a linha de raciocínio usada para resolver o problema proposto.

Os símbolos adotados no diagrama de blocos estão normatizados pela International Organization for Standardization (ISO) 5807:1985 (E). Assim, o diagrama é cons-

truído de modo que qualquer profissional da área de Tecnologia da Informação (TI) possa entender o que representa cada símbolo, como podemos observar na figura 5.1.

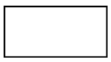
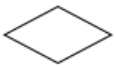
Símbolo	Significado	Descrição
	Terminal	Representa a definição de início e fim do fluxo lógico de um programa. Também é utilizado na definição de sub-rotinas ou função.
	Entrada manual	Representa a entrada manual de dados, normalmente efetuada em um teclado.
	Processamento	Representa a execução de uma operação ou grupo de operações que estabelecem o resultado de uma operação lógica ou matemática.
	Exibição	Representa a execução da operação de saída visual de dados em um monitor de vídeo conectado ao console do computador.
	Decisão	Representa o uso de desvios condicionais para outros pontos do programa de acordo com situações variáveis.
	Preparação	Representa a modificação de instruções ou grupo de instruções existentes em relação à ação de sua atividade subsequencial.
	Conector	Representa a entrada ou saída em outra parte do diagrama de blocos. Pode ser usado na definição de quebras de linha e na continuação da execução de decisões
	Linha	Representa a ação de vínculo existente entre os vários símbolos de um diagrama de blocos. Normalmente possui a ponta de uma seta indicando a direção do fluxo de ação

Figura 5.1: Diagrama de bloco. Fonte: Adaptado de Manzano e Oliveira (2019).

Podemos destacar algumas vantagens na utilização do fluxograma, que por ser uma ferramenta global e muito conhecida, possui padrão mundial, e os símbolos dizem mais do que palavras. Entre as desvantagens, dá pouca atenção aos dados não oferecendo recursos para descrevê-los ou representá-los, tornando complexo sua estruturação à medida que o algoritmo cresce. Na figura 5.2 temos um exemplo de fluxograma para calcular o dobro de um número.

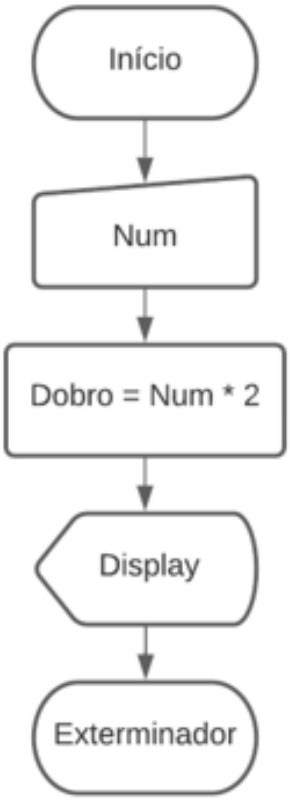


Figura 5.2: Exemplo de fluxograma (calcula o dobro de um número).

5.2 Pseudocodigo

Um algoritmo é a descrição de um conjunto de comandos que, quando executados, resultam em uma sucessão finita de procedimentos, que retornam ou não valores. Em vez de implementar um método diretamente em uma linguagem de programação, é preferível descrevê-lo por meio de uma notação algorítmica.

Manzano e Oliveira (2019) nos alerta que o pseudocódigo é uma linguagem de projeto de programação(LPP), e não uma linguagem de programação real. O pseudocódigo permite descrever de forma simples, sem o rigor técnico de uma linguagem de programação formal (uso de parênteses, pontuações e parâmetros) as etapas que o programa de computador deve executar, desde que essas etapas estejam definidas e delineadas, e não com uma das ferramentas gráficas existentes como o fluxograma. Tornando possível abstrair dos detalhes de uma linguagem de programa e, com isso, permite concentrar apenas nos aspectos matemáticos do método.

As linguagens de projeto de programação possuem como característica principal o

regionalismo, pois são expressas no idioma oficial do pais em que são utilizadas. No Brasil, denominamos de portugol ou português estruturado. Visando ser uma forma preliminar de escrever um programa de computador sem se preocupar com a sintaxe dos comandos de uma linguagem de programação. Além do mais, a descrição do método em uma notação algorítmica facilitará uma futura implementação do código em uma linguagem de programação. Na figura 5.3, podemos observar um algoritmo escrito em portugol, que calcula e retorna o dobro de um número.

```
1  Algoritmo "Algoritmo que calcula o dobro de um número"
2  // Autor     : Juliano Thadeo Alves da Silva
3  Var
4     num, dobro: real
5
6  Inicio
7     Escreva ("Digite um número:")
8     Leia (num)
9     dobro <- num * 2
10    Escreval ("O Dobro do número", num, " é", dobro)
11 Fimalgoritmo
```

Figura 5.3: Exemplo de pseudocódigo (calcula o dobro de um número) usando o VisuAlg.

O pseudocodigo pode ser criado sem a utilização de dispositivos digitais, no entanto, o processo de ensino aprendizagem se torna mais dinâmico com o uso de programas. É sugerido neste trabalho o software VisuAlg para computadores, e também pode ser utilizada uma versão para smartphone: o App PseudoCode, o que permite aproveitar desse recurso mesmo quando o laboratório de informática não está disponível.

5.3 VisuAlg

O VisuAlg é um programa de uso livre e distribuído gratuitamente, que permite criar, editar, interpretar e executar os algoritmos em português estruturado. É utilizado por diversas instituições no Brasil e exterior, para o ensino de lógica de programação.

O programa foi desenvolvido originalmente e mantido até a versão 2.5 pelo professor Cláudio Morgado de Souza, com o intuito de ajudar os estudantes a aprender conceitos de programação, em uma linguagem mais próxima a língua materna. Morgado se desligou do projeto e convidou o professor Antonio Carlos Nicologi para que o substituísse no desenvolvimento e aprimoramento do projeto, passando os códigos fontes do programa.

Atualmente, o programa VisuAlg está na versão 3.0, possui uma interface agradável e intuitiva, composta pela barra de tarefas, área do editor de código fonte, área das variáveis de memória, área do simulador de saída e da barra de status. Quando o programa é carregado, já apresenta no editor um "esqueleto" de pseudocódigo, com a intenção de poupar trabalho ao usuário e de mostrar o formato básico que deve ser seguido.

5.3.1 Estrutura básica

A linguagem permite apenas um comando por linha: desse modo, não há necessidade de tokens separadores de estruturas, como o ponto e vírgula em Pascal. E para facilitar a digitação e evitar confusões, todas as palavras-chave do VisuAlg foram implementadas sem acentos e cedilha, quando estiverem em minúsculos. O VisuAlg também não distingue maiúsculas e minúsculas no reconhecimento de palavras-chave e nomes de variáveis. Na figura 5.4, é mostrado o formato básico do VisuAlg.

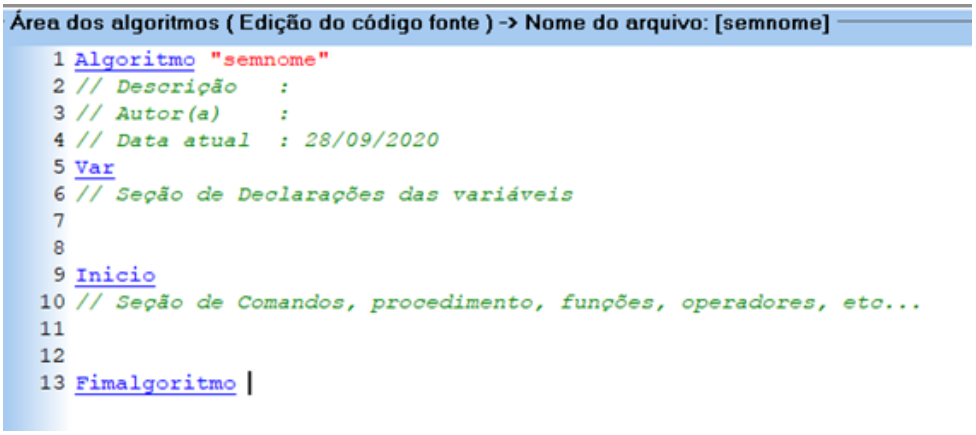


Figura 5.4: Estrutura básica do VisuAlg.

A primeira linha é composta pela palavra-chave "algoritmo" seguida do seu nome delimitado por aspas duplas. Este nome será usado como título nas janelas de leitura de dados. A seção que se segue é a de declaração de variáveis, que termina com a linha que contém a palavra-chave "inicio". Deste ponto em diante está a seção de comandos, que continua até a linha em que se encontre a palavra-chave "fimalgoritmo". Esta última linha marca o final do pseudocódigo: todo texto existente a partir dela é ignorado pelo interpretador.

5.3.2 Declaração de variável

Na Seção de Declaração, o VisuAlg prevê quatro tipos de dados: inteiro, real, cadeia de caracteres e lógico. Observe as palavras-chave que definem as variáveis, não possuem acentuação. Na tabela 5.1, seguem definições referentes a cada tipo de dados.

Tabela 5.1: Tipo de variável.

Tipo de dado	**Descrição**
inteiro	define variáveis numéricas do tipo inteiro
real	define variáveis numéricas do tipo real
caractere ou caracter	define variáveis do tipo string, ou seja, cadeia de caracteres
logico (sem acento)	define variáveis do tipo booleando (Verdadeiro ou Falso)

Toda variável possui um nome ou identificador, cujas regras variam de acordo com a linguagem de programação escolhida. No Visual, os nomes das variáveis devem começar por uma letra e depois conter letras, números ou underline, e tem limite de 30 caracteres. A seção de declaração de variáveis começa com a palavra-chave var e continua com o nome-da-variável, depois o caractere dois pontos, finalizando o tipo da variável, como podemos observar na figura 5.5.

```
Var
// Seção de Declarações das variáveis
nota1, nota2: real
quantidade: inteiro
nome_aluno: caracter
sinalizador: logico
```

Figura 5.5: Declaração de variável.

5.3.3 Atribuição de valores

A atribuição de valores a variáveis pode ser feita com os operadores $<-$ (menor, traço) ou := (dois pontos, igual). Do seu lado esquerdo fica a variável à qual está sendo atribuído o valor, e à sua direita pode-se colocar qualquer expressão (constantes, variáveis, expressões numéricas), desde que seu resultado tenha tipo igual ao da variável. Como por exemplo, $quantidade <- 15$ (variável quantidade recebe o valor 15).

5.3.4 Operadores aritméticos

Os símbolos utilizados para realização de cálculos matemáticos, são os tradicionais operadores aritméticos: adição (+), subtração (−), multiplicação (∗) e divisão (/); além dos operadores de potenciação e divisão inteira (\). Por convenção, as prioridades das operações são as mesmas utilizadas na matemática. Da mesma forma, para modificar a ordem de execução das operações é necessário usar parênteses como em qualquer expressão aritmética.

5.3.5 Operadores relacionais

Os símbolos utilizados em expressões lógicas para testar ou definir uma relação entre dois valores do mesmo tipo. O Visualg trabalha com os operadores: igual (=), menor que (<), maior que (>), menor ou igual a (<=), maior ou igual a (>=) e diferente de (<>). Por exemplo: $3 > 2$ (3 é maior do que 2) retorna Verdadeiro.

5.3.6 Comandos de entrada e saída de dados

A entrada de dados no VisuAlg é feita através do comando leia (<lista-de-variáveis>). Este comando recebe valores digitados pelos usuários, atribuindo-os às variáveis cujos nomes estão em <lista-de-variáveis>. É respeitada a ordem especificada nesta lista. Por exemplo: leia (nota1).

O comando de saída "escreva", escreve no dispositivo de saída padrão o conteúdo de cada uma das expressões que compõem <lista-de-expressões>. As expressões dentro desta lista devem estar separadas por vírgulas; depois de serem avaliadas, seus resultados são impressos na ordem indicada. A sintaxe do comando escreva é: escreva (<lista-de-expressões>). É possível utilizar também o comando "escreval", com a única diferença que após escrever o resultado solicitado, ele pulará uma linha.

Para variáveis reais, pode-se também especificar o número de casas decimais que serão exibidas. Por exemplo, considerando "media" como uma variável real, o comando escreva (media:4:2) escreve seu valor em 4 espaços colocando 2 casas decimais. Como podemos observar na figura 5.6.

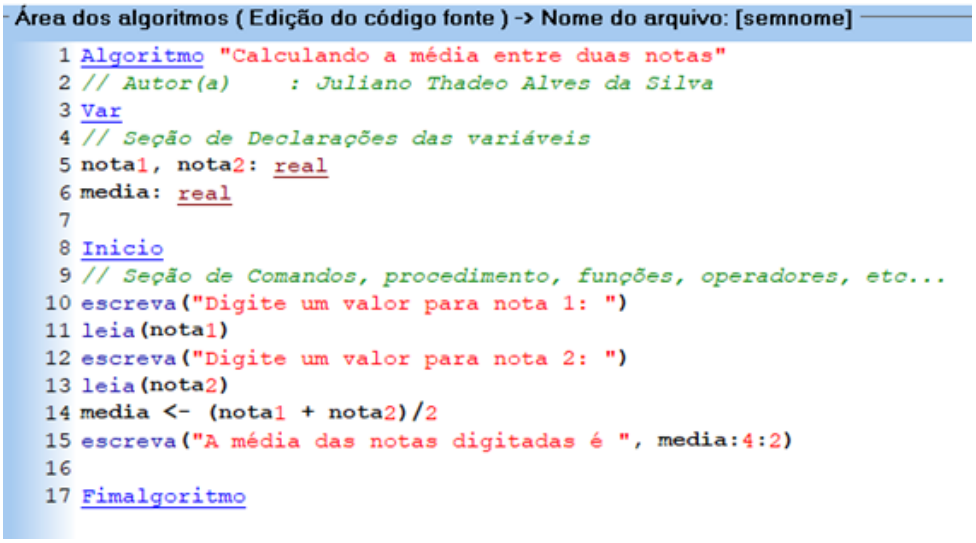


Figura 5.6: Comando de entrada e saída de dados.

5.3.7 Desvio condicional simples e composto

Muitas vezes é necessário que uma parte do código só seja executada se determinada condição for satisfeita. Essa condição é definida por uma expressão lógica cujo resultado sempre será um valor booleano, ou seja, verdadeiro ou falso. O desvio condicional tem por finalidade tomar uma decisão de acordo com o resultado de uma condição ou critério (teste lógico), e executar um bloco de códigos dependendo do resultado dessa decisão. Segue sintaxe na figura 5.7 do desvio condicional simples.

```
se <expressão−lógica> entao
    <sequência−de−comandos>
fimse
```

Figura 5.7: Sintaxe do desvio condicional simples.

A expressão lógica que testará as condições ou critérios impostos pelo algoritmo, utiliza uma relação entre um par de elementos, o qual pode ser composto por variável, constante e um operador relacional. Dessa forma, ao encontrar este comando, o VisuAlg analisa a <expressão-lógica>. Se o seu resultado for **verdadeiro**, todos os comandos da <sequência-de-comandos> são executados até a instrução "fimse".

No desvio condicional composto, se o resultado da avaliação de <expressão-lógica> for **verdadeiro**, todos os comandos da <sequência-de-comandos-1> são executados, e a execução contínua depois a partir da primeira linha depois do "fimse". Se

o resultado for **falso**, estes comandos são desprezados e o algoritmo continua a ser executado a partir da primeira linha depois do "senão", executando todos os comandos da <sequência-de-comandos-2>. Segue na figura 5.8 sintaxe para o desvio condicional composto, que deve ser utilizado quando temos duas alternativas possíveis.

```
se <expressão−lógica> entao
    <sequência−de−comandos−1>
senao
    <sequência−de−comandos−2>
fimse
```

Figura 5.8: Sintaxe do desvio condicional composto.

A seguir, na figura 5.9, vamos acompanhar um exemplo da aplicação com comando de desvio condicional composto. O algoritmo solicita duas notas e calcula a média aritmética das notas. Se a média for maior ou igual a sete, ele retorna a mensagem "O aluno está **aprovado**", caso contrário, ele retorna "O aluno está **reprovado**".

Área dos algoritmos (Edição do código fonte) -> Nome do arquivo: [calculo da média.ALG]

```
 1 Algoritmo "Calculando a média entre duas notas"
 2 // Autor(a)     : Juliano Thadeo Alves da Silva
 3 Var
 4 // Seção de Declarações das variáveis
 5 nota1, nota2: real
 6 media: real
 7
 8 Inicio
 9 // Seção de Comandos, procedimento, funções, operadores, etc...
10 escreva("Digite um valor para nota 1: ")
11 leia(nota1)
12 escreva("Digite um valor para nota 2: ")
13 leia(nota2)
14 media <- (nota1 + nota2)/2
15 escreval("A média das notas digitadas é ", media:4:2)
16 se media >=7 entao
17     escreva("O aluno está APROVADO")
18 senao
19     escreva("O aluno está REPROVADO")
20 fimse
21
22 Fimalgoritmo
```

Figura 5.9: Exemplo de aplicação do desvio condicional composto.

5.3.8 Comando de seleção múltipla

Em casos que temos uma condição que necessite ser avaliada a partir de múltiplos valores, o uso de estruturas de decisão encadeadas pode se tornar trabalhoso. É recomen-

dado, nesses casos, o uso de estrutura de desvio de seleção múltipla (caso). Essa estrutura possui um pequeno inconveniente, ela só pode ser utilizada nos casos em que as condições de seleção são sempre do tipo "igual a". Na figura 5.10, observamos a sintaxe do comando.

```
escolha <expressão−de−seleção>
caso <exp11>, <exp12>, ..., <exp1n>
    <sequência−de−comandos−1>
caso <exp21>, <exp22>, ..., <exp2n>
    <sequência−de−comandos−2>
...
outrocaso
    <sequência−de−comandos−extra>
fimescolha
```

Figura 5.10: Sintaxe do comando de seleção múltipla.

No exemplo a seguir (figura 5.11), podemos verificar o funcionamento do comando de seleção múltipla.

Área dos algoritmos (Edição do código fonte) -> Nome do arquivo: [conceito.ALG]

```
1  Algoritmo "Conceito"
2  // Autor(a)     : Juliano Thadeo Alves da Silva
3
4  Var
5  // Seção de Declarações das variáveis
6  conceito: caracter
7
8  Inicio
9  // Seção de Comandos, procedimento, funções, operadores, etc...
10 escreva("Digite o seu conceito na disciplina de matemática: ")
11 leia(conceito)
12 escolha conceito
13 caso "A"
14 escreva("Parabéns excelente!")
15 caso "B"
16 escreva("Muito bom!")
17 caso "C"
18 escreva("Pode melhorar!")
19 outrocaso
20 escreva("Estude mais um pouco")
21 fimescolha
22 Fimalgoritmo
```

Figura 5.11: Exemplo de comando para seleção múltipla.

5.3.9 Comando de repetição - loop (laços)

Ao escrevermos um algoritmo, muito vezes precisamos que ele execute o mesmo trecho de código mais de uma vez. Em muitos casos, o número de vezes necessário para

efetuar a repetição é elevado e inviabiliza qualquer tentativa de reescrever aquele trecho de código tantas vezes quanto necessárias.

Com a técnica de laços resolvemos esse problema, permitindo que o usuário controle quantas vezes determinada porção de instrução deve ser repetida. Assim, uma estrutura de repetição faz com que uma sequência de comandos seja executada repetidamente até que uma dada condição de interrupção seja satisfeita. Basicamente, existem dois tipos de estrutura de repetição, dependendo se conhecemos o número de repetições ou não.

5.3.9.1 Estrutura de repetição: para ... faca

Esta estrutura repete uma sequência de comandos um determinado número de vezes.

```
para <variável> de <valor−inicial> ate <
    valor−final> [passo <incremento>]
    faca
   <sequência−de−comandos>
fimpara
```

Figura 5.12: Sintaxe do comando de repetição: para...faca.

O argumento "variável" é responsável por realizar uma contagem de repetições. Tal contagem tem início num "valor-inicial" e termina num "valor-final", também descritos na primeira linha da sintaxe. É importante ressaltar que quando na atribuição "ate <valor-final>", não significa que esse valor será exatamente atingido, mas que as instruções ocorrerão enquanto o contador for menor ou igual a esse valor. O comando "passo" é opcional. Serve para informar qual será o valor do incremento da contagem de repetições.

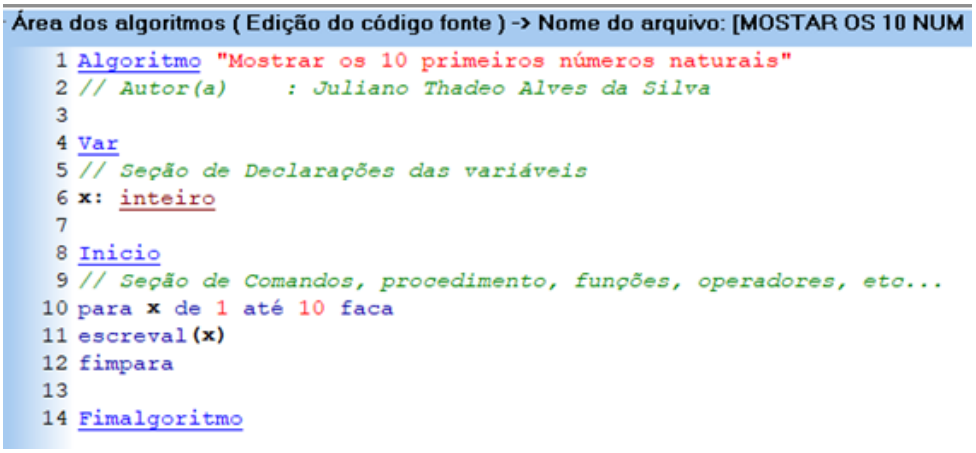


Figura 5.13: Exemplo de loop - para...faca.

No exemplo da figura 5.13, utilizamos a estrutura de repetição para criar um loop na impressão dos dez primeiros números naturais.

5.3.9.2 Estrutura de repetição: enquanto ... faca

Esta estrutura repete uma sequência de comandos enquanto uma determinada condição (especificada através de uma expressão lógica) for satisfeita.

enquanto <expressão−lógica> faca
<sequência−de−comandos>
fimenquanto

Figura 5.14: Sintaxe do comando de repetição: loop - enquanto...faca.

Esta expressão que é avaliada antes de cada repetição do laço, quando seu resultado for **verdadeiro**, <sequência-de-comandos> é executada. Podemos observar na figura 5.15, uma adaptação do algoritmo anterior com o comando “enquanto...faca”.

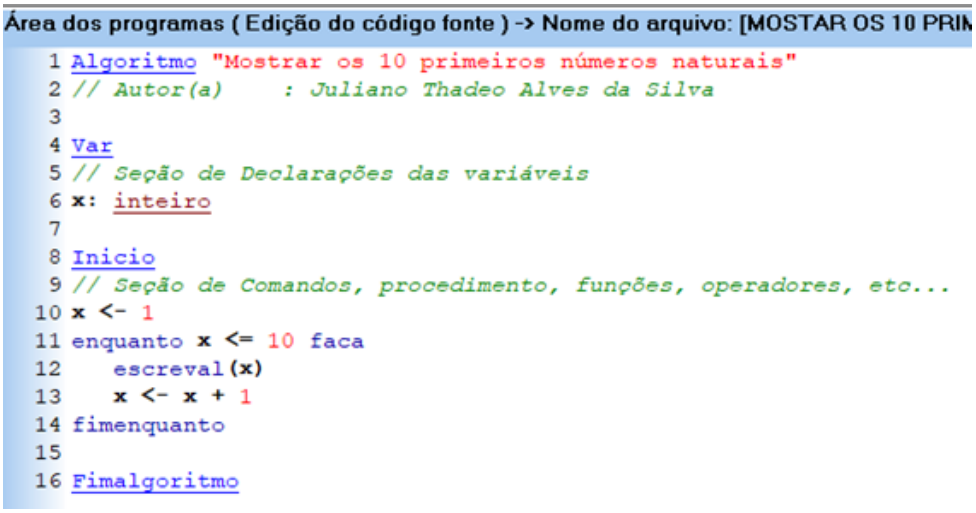


Figura 5.15: Exemplo de loop - enquanto...faca

Capítulo 6

Proposta de atividades didáticas

Baseando nos conceitos do construcionismo e aprendizagem centrada no aluno como abordagens de ensino, optamos pelas duas metodologias de ensino apresentadas nesse trabalho: aprendizagem baseada em problemas e a aprendizagem baseada em jogos. Criamos algumas propostas de atividades que visam trabalhar, de forma didática e gradual, conhecimentos necessários tanto para o uso de tecnologias digitais quanto para a sua criação, desenvolvendo lógica de resolução de problemas.

Dessa forma, a proposta de atividade didática traz um modelo de ensino e aprendizagem que foi pensado em promover o desenvolvimento de competências, com atividades que os alunos possam tomar decisão, resolver problemas e socializar com os colegas. Assim, essas práticas pedagógicas buscam formar cidadãos responsáveis e críticos em uma sociedade digital e, com isso, tornando-os aptos a exercer sua cidadania.

Nesse viés, a proposta de atividade didática foi elaborada com o intuito de auxiliar os professores da rede de ensino no Brasil a incluir práticas no cotidiano escolar, que desenvolvam o pensamento reflexivo e crítico, assim como o protagonismo e a autonomia nos estudantes.

Em meio a uma necessidade urgente de adoção de práticas pedagógicas que contemplem o eixo do pensamento computacional, e levando em consideração todas as dificuldades com estrutura e tecnologia das nossas escolas, buscamos abordar atividades que possam ser trabalhadas tanto no computador quanto no smartphone. Vale ainda destacar, que as atividades podem ser realizadas de maneira desplugada, caso não se tenha acesso a nenhum dos dispositivos. Por exemplo, o docente pode confeccionar um tabuleiro inspirado no jogo LigthBot, com regras adaptadas e contendo como adicional a possibilidade de

definir o grau de complexidade que permita ajustar a dificuldade conforme a experiência dos estudantes.

Nas atividades 1 e 2, buscamos abordar conceito de programação de uma maneira lúdica e diferenciada, para isso utilizamos a metodologia de aprendizagem baseada em jogos. Os aplicativos educacionais aqui sugeridos (LightBot e SpriteBox) visam proporcionar uma experiência divertida ao usuário, levando o raciocínio lógico e conceitos básicos de programação de maneira despretensiosa. Tendo em vista, que os games utilizam comandos que fazem referência direta aos algoritmos computacionais, explorando as estruturas sequenciais, loop (repetição) e de funções/procedimentos sem a necessidade de escrever uma linha de pseudocódigos.

Os aplicativos aqui sugeridos fazem uso de blocos para ensino de programação, que é uma prática conhecida e adotada em ferramentas famosas na área de educação, como o Scratch. A sua utilização e interação (arrastar e soltar com o dedo) é bem simples, e se mostram adequados diante das limitações no tamanho da tela e entrada de dados dos dispositivos móveis, favorecendo a jogabilidade.

Vale reforçar que é fundamental que o docente conheça os aplicativos e conheça as instruções que resolvam determinadas situações, para que assim ele tenha condições de direcionar o processo de ensino e aprendizagem por meio de jogos eletrônicos.

Para desenvolvemos as atividades de codificação em pseudocódigo, foi sugerido a utilização do VisuAlg no computador, ou Pseudocode para smartphone. Assim, todas as atividades aqui propostas poderão ser realizadas, tanto no computador, como no smartphone.

6.1 Atividade App SpriteBox

SpriteBox é um jogo de quebra-cabeça lógico dos criadores do LightBot que combina a diversão de jogos de "plataforma", como por exemplo o Super Mario Bros, com a aprendizagem dos fundamentos da codificação. O uso de recursos visuais, presente nos jogos, é um facilitador para o aprendizado de conceitos abstratos e mais complexos da programação, como a repetição, recursividade e condicionantes. SpriteBox cobre conceitos básicos de algoritmos, como sequenciamento de instruções, instruções com parâmetros, loops simples e complexos.

Controles (Web):

Teclas de seta: correr para a esquerda / direita, pular, subir escadas

Mouse / trackpad: interagir / codificar

No decorrer do jogo, os estudantes descobrirão que algumas seções estão intransitáveis. Nesse ponto, eles devem procurar uma SpriteBox amarela, pular e bater nela por baixo. Isso chamará Sprite, que tentará ajudá-lo, como na figura.

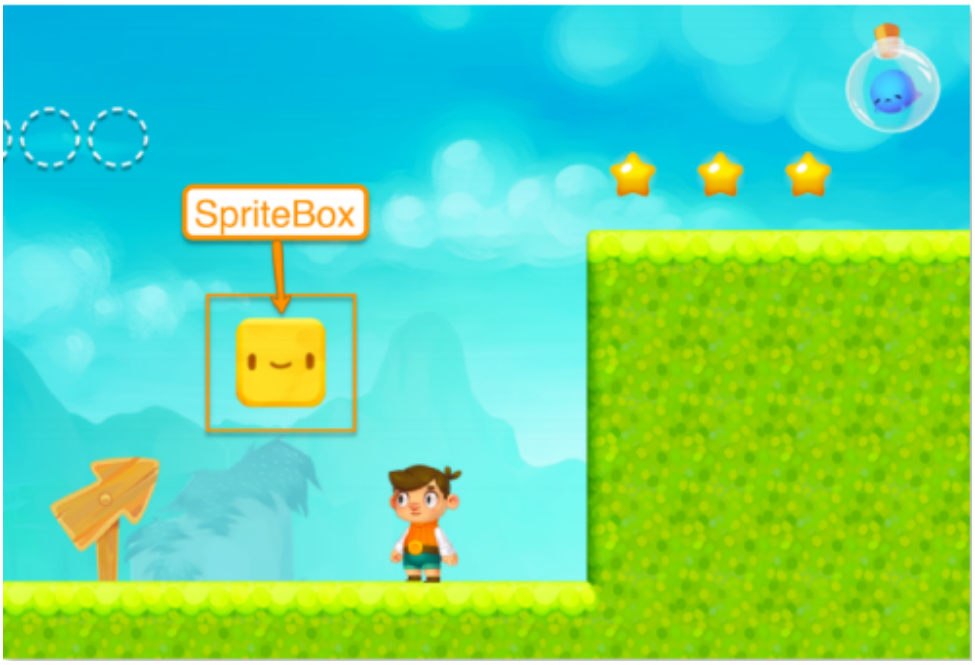


Figura 6.1: Tela do App SpriteBox. Fonte: Site SpriteBox - guia do professor.

Sprite pode então ajudar o jogador a progredir modificando o ambiente do jogo e tornando transitáveis as seções anteriormente intransponíveis. Para essa primeira missão, o Sprite irá pedir que o jogador reorganize as instruções para que o jogador possa assim avançar, como no exemplo da figura 6.2.

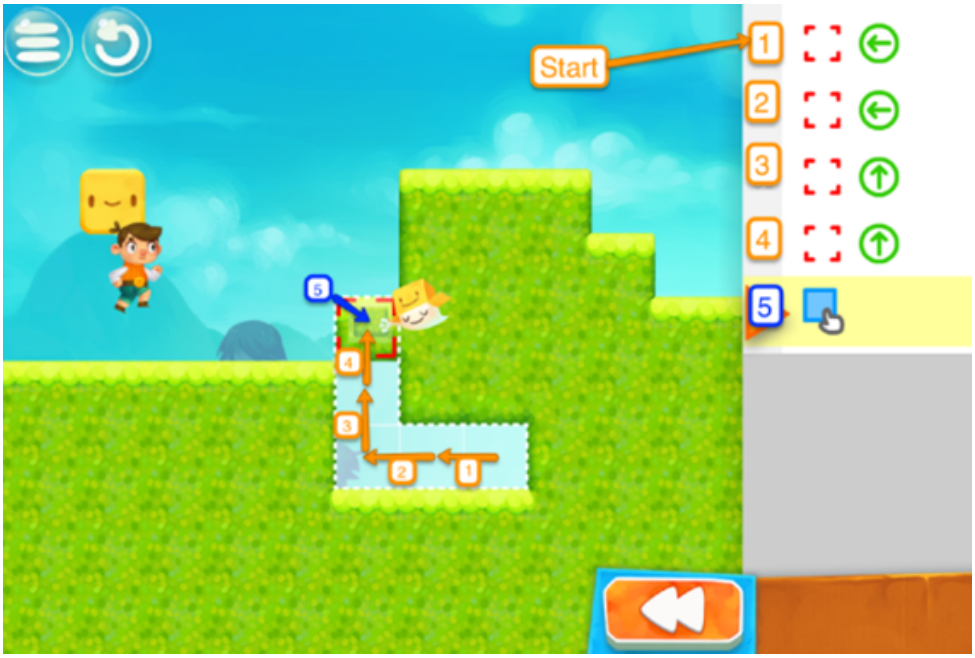


Figura 6.2: SpriteBox - sequência de instruções. Fonte: Site SpriteBox - guia do professor.

Ao pressionar Play, Sprite executará as instruções de cima para baixo. Aqui, Sprite irá mover para a esquerda, mover para a esquerda, mover para cima, mover para cima, set (box).

Aos poucos, as missões ficarão mais complicadas e difíceis de se transpor. Na segunda fase, o Sprite irá apresentar a construção loop. Um loop permite que os jogadores repitam as instruções que existem dentro do loop qualquer número de vezes, conforme figura 6.3.

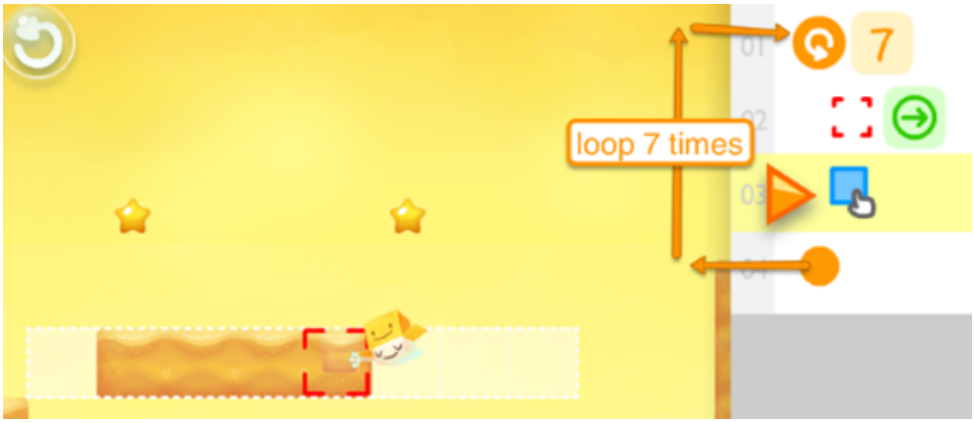


Figura 6.3: SpriteBox - estrutura em loop. Fonte: Site SpriteBox - guia do professor.

Sprite pedirá que o jogador copie os padrões mostrados em cada quebra-cabeça. Esses quebra-cabeças só serão desbloqueados quando todos os outros níveis forem resolvi-

dos e os sprites encontrados. Aqui, os jogadores terão uma quantidade limitada de espaço e, portanto, devem utilizar loops dentro de loops para resolver os quebra-cabeças.

6.2 Atividade App LightBot

Essa atividade, visa contribuir para a construção de um processo de ensino aprendizagem gamificado, no qual o desenvolvimento do raciocínio lógico, matemático e computacional é estimulado através de práticas que abordam a montagem de sequências lógicas de forma estruturada.

LightBot é um jogo de quebra-cabeça de programação; ele usa mecânicas de jogo que estão firmemente enraizadas em conceitos de programação. O LightBot permite que os jogadores obtenham uma compreensão prática de conceitos básicos como sequência de instruções, procedimentos e loops, apenas guiando um robô para iluminar as peças e resolver os níveis.

O jogo conta com um robô cuja função é realizar um percurso dentro de um labirinto com número limitado de passos. O jogador visualiza um bloco de ações principal **main** e outro ação secundária **proc1**, e dependendo do nível de dificuldade fica disponível mais um bloco de ação secundário **proc2**. O robô pode executar comandos como: andar para frente, acender a luz, virar à esquerda, virar à direita, pular e acionar procedimentos repetitivos (P1 e P2). Podemos observar na figura 6.4, a tela com um dos desafios que deve ser superado.

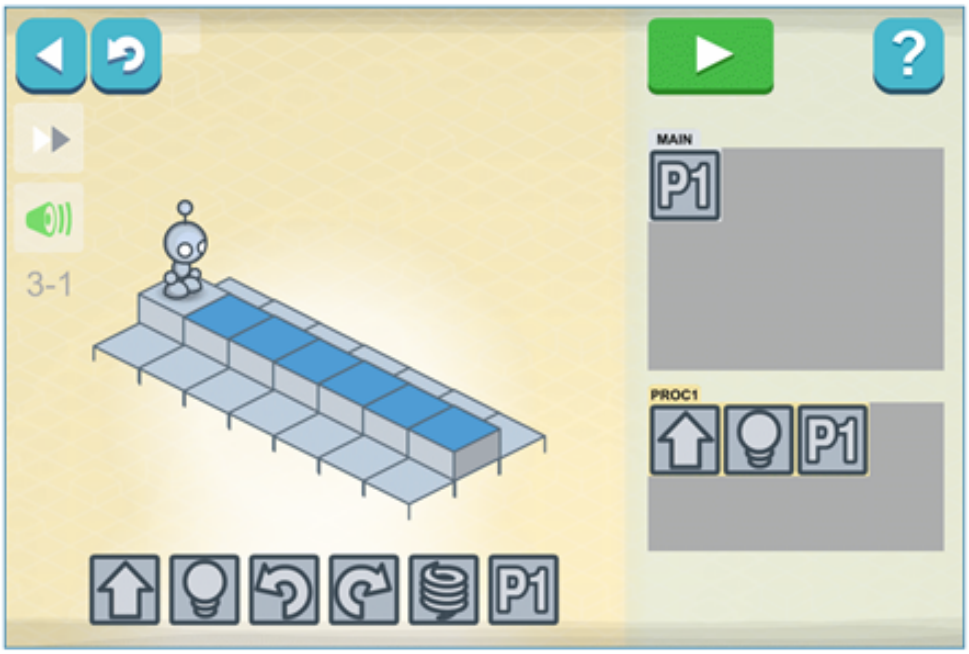


Figura 6.4: Tela do App LightBot. Fonte: Site LightBot.

Há também a opção do uso de estruturas de repetição, que se configuram em um conjunto de ações que podem acontecer mais de uma vez dentro de uma mesma rodada. Esse conjunto de ações fica pré-definido e só há a necessidade de escrevê-lo uma única vez para sempre que necessário, reutilizá-lo. No caso do LightBot, o jogo só permite a criação de dois procedimentos que se refere aos comandos P1 e P2. Na tabela 6.1, a interface do programa é detalhada.

Tabela 6.1: Interface do programa LightBot.

Nº	Nome	Descrição
1	MAIN	Bloco de ação onde os comandos são colocados. É o bloco principal, onde rodam as primeiras ações do robô.
2	PROC1	Caso seja utilizado, é um bloco de ações extras, que pode servir para estruturas de repetição loops.
3	PROC2	Idem PROC1.
4	Área principal	Labirinto onde o robô realizará as ações do MAIN ("tabuleiro").
5	Comandos	**Seta para cima**: o robô anda uma casa para frente. **Lâmpada**: o robô acende a luz da casa em que se encontra. **Seta no sentido anti-horário**: o robô vira para sua esquerda. **Seta no sentido horário**: o robô vira para sua direita. **Mola**: o robô pula para a casa a sua frente. **P1**: executa o bloco de ações que estão no PROC1. **P2**: executa o bloco de ações que estão no PROC2.

Fonte: Site LightBot.

O LightBot destina-se a apresentar as crianças e adolescentes que não têm nenhuma experiência em programação, sendo adequado para todas as idades. Isso significa que qualquer pessoa no Ensino Básico pode jogar, se divertir e aprender lógica de programação. Para realizar a instalação, basta acessar a loja de aplicativos do seu smartphone e baixar o programa, ou se preferir utilizar no computador, basta acessar o site https://lightbot.com.

O uso de um jogo lúdico como forma de ensinar um conceito básico é uma maneira interessante de trabalhar com um público que está muito conectado às tecnologias digitais. Dessa maneira, o jogo é recebido com entusiasmo por parte dos estudantes, que normalmente demonstram empolgação e motivação com a metodologia adotada.

Ressaltamos que as crianças devem ser estimuladas desde cedo no contato com a programação. E assim, a partir dessas atividades lúdicas, os alunos dos anos iniciais trabalhem conhecimentos na área de programação, algoritmos e resolução de problemas sem perceber, ou seja, o pensamento computacional é utilizado naturalmente para a solução do desafio proposto.

6.3 Atividade Matemática - 1º ano

Unidades temáticas: Grandezas e medidas

Objetos de conhecimento: Medidas de tempo: unidades de medida de tempo, suas relações e o uso do calendário

Habilidades: (EF01MA16) Relatar em linguagem verbal ou não verbal sequência de acontecimentos relativos a um dia, utilizando, quando possível, os horários dos eventos.

Atividade 1: Desenvolver um algoritmo verbal sobre os acontecimentos ao longo do dia. Por exemplo:

1º passo: acorda, vai tomar banho e escovar os dentes;

2º passo: sai de casa às 7h e vai para a escola;

3º passo: na escola, estuda as disciplinas diária;

4º passo: sai da escola às 12h, vai para a casa;

5º passo: toma banho e almoça;

6º passo: começa a fazer as tarefas e estudar às 15h;

7º passo: janta às 19h;

8º passo: assiste televisão;

9º passo: tomar banho, escovar os dentes e vai dormir às 22h.

Objetivo da atividade: Compreender o conceito de algoritmo como uma sequência de passo ou instruções, por meio de linguagem verbal ou imagens.

Observação: Pode ser utilizado imagens para ilustrar os acontecimentos, e solicitar que os alunos coloquem em ordem de execução.

6.4 Atividade Matemática - 2º ano

Unidades temáticas: Álgebra

Objetos de conhecimento: Problemas envolvendo significados de dobro, metade, triplo e terça parte

Habilidades: (EF02MA08) Resolver e elaborar problemas envolvendo dobro, metade, triplo e terça parte, com o suporte de imagens ou material manipulável, utilizando estratégias pessoais.

Atividade 1: Solicitar que os alunos descrevam os passos (escrevam um algoritmo) para calcular o dobro de um número qualquer. O professor deverá definir as palavras que serão

utilizados.

Por exemplo:

1º passo: Receber (Pegar) um "número qualquer"

2º passo: Multiplicar o número recebido por 2;

3º passo: Escreve o resultado.

Objetivo da atividade: Compreender o conceito de algoritmo como uma sequência de passo ou instruções, por meio de linguagem verbal ou imagens.

Unidades temáticas: Geometria

Objetos de conhecimento: Esboço de roteiros e de plantas simples

Habilidades: (EF02MA12) Identificar e registrar, em linguagem verbal ou não verbal, a localização e os deslocamentos de pessoas e de objetos no espaço, considerando mais de um ponto de referência, e indicar as mudanças de direção e de sentido.

Atividade 2: Escrever e executar um algoritmo simples em português estruturado, usando um vocabulário restrito para as instruções. Solicitar para que os alunos executem uma sequência de instruções para o deslocamento de uma pessoa, utilizando o próprio corpo, um objeto ou animal, diante de desenho com imagens: frente, traz, vire à direita, vire à esquerda.

Por exemplo:

1º passo: ande para frente;

2º passo: vire à direita;

3º passo: ande para frente;

4º passo: vire à direita;

5º passo: ande para frente;

6º passo: vire à direita;

7º passo: ande para frente.

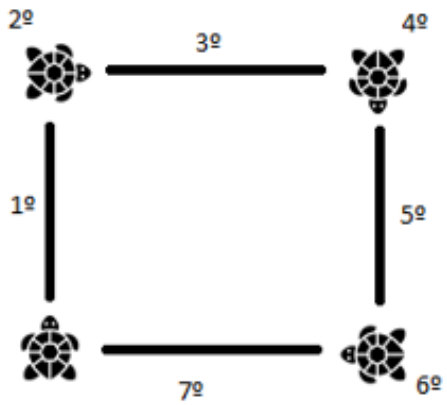


Figura 6.5: Deslocamento: algoritmo sequência de passos.

Objetivo da atividade: Compreender o conceito de algoritmo como uma sequência de passo ou instruções, por meio de linguagem verbal ou imagens. Compreender e executar uma sequência de instruções, por meio de símbolos ou palavras definidas pelo professor.

6.5 Atividade Matemática - 3º ano

Unidades temáticas: Números

Objetos de conhecimento: Procedimentos de cálculo (mental e escrito) com números naturais: adição e subtração

Habilidades: (EF03MA05) Utilizar diferentes procedimentos de cálculo mental e escrito, inclusive os convencionais, para resolver problemas significativos envolvendo adição e subtração com números naturais.

Atividade 1: Solicitar que os alunos descrevam os passos para realizar a soma entre dois números, ou seja, o algoritmo da conta armada. Vale lembrar que os passos devem ser precisos e assertivos. Detalhando as ações que devem ser realizadas, evitando passos como coloque um número embaixo do outro e some.

Objetivo da atividade: Compreender o conceito de algoritmo como uma sequência de passo ou instruções, por meio de linguagem verbal ou imagens.

Unidades temáticas: Geometria

Objetos de conhecimento: Localização e movimentação: representação de objetos e pontos de referência

Habilidades: (EF03MA12) Descrever e representar, por meio de esboços de trajetos ou utilizando croquis e maquetes, a movimentação de pessoas ou de objetos no espaço, incluindo mudanças de direção e sentido, com base em diferentes pontos de referência.

Atividade 2: Escrever um algoritmo simples em português estruturado, usando um vocabulário restrito para as instruções. Solicitar para que os alunos executem uma sequência de instruções para o deslocamento de um objeto em um tabuleiro. Por exemplo, para se descolar de uma posição A para uma posição B, utilizando comando/ação definidas pelo professor. Salientamos que o docente deverá ir aumentando a dificuldade conforme os alunos consigam realizar as atividades iniciais.

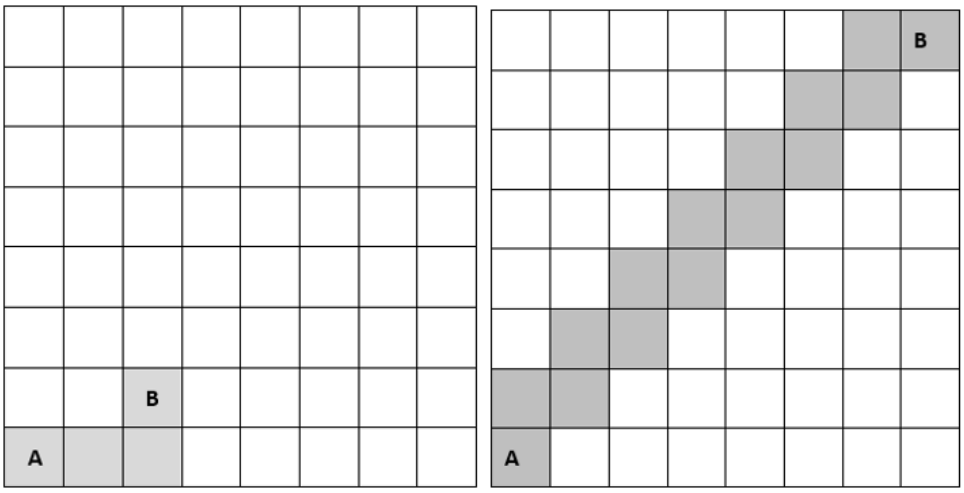


Figura 6.6: Tabuleiro: algoritmo para movimentação de objetos.

Objetivo da atividade: Compreender o conceito de algoritmo como uma sequência de passo ou instruções, por meio de linguagem verbal ou imagens. Reconhecer padrões de deslocamento e verificar que os passos são repetidos determinadas vezes.

6.6 Atividade Matemática - 4º ano

Unidades temáticas: Números

Objetos de conhecimento: Propriedades das operações para o desenvolvimento de diferentes estratégias de cálculo com números naturais

Habilidades: (EF04MA03) Resolver e elaborar problemas com números naturais envolvendo adição e subtração, utilizando estratégias diversas, como cálculo, cálculo mental e algoritmos, além de fazer estimativas do resultado.

Atividade 1: Solicitar que os alunos descrevam os passos para realizar a subtração entre dois números, ou seja, o algoritmo da conta armada.

Objetivo da atividade: Compreender o conceito de algoritmo como uma sequência de passo ou instruções, por meio de linguagem verbal.

Unidades temáticas: Álgebra

Objetos de conhecimento: Sequência numérica recursiva formada por múltiplos de um número natural

Habilidades: (EF04MA11) Identificar regularidades em sequências numéricas compostas por múltiplos de um número natural.

Atividade 2: Crie um algoritmo em portugol, que solicite o primeiro e o segundo número de uma sequência e retorne, o próximo número da sequência.

Por exemplo: Dados os números 3 e 6, determine o próximo número da sequência.

1^{o} passo: Entre com os valores para o primeiro e segundo números da sequência;

2^{o} passo: terceiro número é igual ao segundo número mais diferença entre os números;

3^{o} passo: Escreva o terceiro número.

Objetivo da atividade: Compreender o conceito de algoritmo como uma sequência de passo ou instruções, por meio de linguagem verbal.

Unidades temáticas: Geometria

Objetos de conhecimento: Localização e movimentação: pontos de referência, direção e sentido

Habilidades: (EF04MA16) Descrever deslocamentos e localização de pessoas e de objetos no espaço, por meio de malhas quadriculadas e representações como desenhos, mapas, planta baixa e croquis, empregando termos como direita e esquerda, mudanças de direção e sentido, intersecção, transversais, paralelas e perpendiculares.

Atividade 3: Utilizar os app´s SpriteBox e LightBot, onde os alunos podem definir a sequência de passos e depois executar, e se necessário pode corrigir o algoritmo.

Objetivo da atividade: Compreender o conceito de algoritmo como uma sequência de passo ou instruções, por meio de linguagem verbal. Trabalhar todos os pilares do pensamento computacional: abstração, decomposição, reconhecimento de padrão e algoritmo.

6.7 Atividade Matemática - 5^{o} ano

Unidades temáticas: Números

Objetos de conhecimento: Sistema de numeração decimal: leitura, escrita e ordenação de números naturais (de até seis ordens)

Habilidades: (EF05MA01) Ler, escrever e ordenar números naturais até a ordem das centenas de milhar com compreensão das principais características do sistema de numeração decimal.

Atividade 1: Escreva um algoritmo que leia dois números naturais (diferentes) e retorne,

os números ordenados do menor para o maior.

```
1 Algoritmo "Mostrar o maior e o menor número"
2 // Autor(a)     : Juliano Thadeo Alves da Silva
3
4 Var
5 // Seção de Declarações das variáveis
6 maior, menor, n1, n2: inteiro
7
8 Inicio
9 // Seção de Comandos, procedimento, funções, operadores, etc...
10 Escreva("Entre com o primeiro número:")
11 Leia(n1)
12 Escreva("Entre com o segundo número:")
13 Leia(n2)
14 Se n1 > n2 entao
15    maior <- n1
16    menor <- n2
17 senao
18    maior <- n2
19    menor <- n1
20 fimse
21 Escreval("O maior número é ", maior)
22 Escreval("O menor número é ", menor)
23
24 Fimalgoritmo
```

Figura 6.7: Exemplo: algoritmo retorna o menor e o maior número inteiro.

Unidades temáticas: Números

Objetos de conhecimento: Cálculo de porcentagens e representação fracionária

Habilidades: (EF05MA06) Associar as representações 10%, 25%, 50%, 75% e 100% respectivamente à décima parte, quarta parte, metade, três quartos e um inteiro, para calcular porcentagens, utilizando estratégias pessoais, cálculo mental e calculadora, em contextos de educação financeira, entre outros.

Atividade 2: Crie um algoritmo em portugol, que calcule porcentagens (10%, 25%, 50%, 75% e 100%) de um determinado valor informado.

```
1 Algoritmo "Porcentagens (10%, 25%, 50%, 75% e 100%) de um número"
2 // Autor(a)     : Juliano Thadeo Alves da Silva
3
4 Var
5 // Seção de Declarações das variáveis
6 numero: inteiro
7
8 Inicio
9 // Seção de Comandos, procedimento, funções, operadores, etc...
10 Escreva("Entre com o número:")
11 Leia(numero)
12 Escreval("Calculo de porcentagem do número", numero)
13 Escreval("10% do número é ", numero*10/100)
14 Escreval("25% do número é ", numero*25/100)
15 Escreval("50% do número é ", numero*50/100)
16 Escreval("75% do número é ", numero*75/100)
17 Escreval("100% do número é ", numero*100/100)
18
19 Fimalgoritmo
```

Figura 6.8: Exemplo: algoritmo cálculo de porcentagem.

Unidades temáticas: Números

Objetos de conhecimento: Problemas: multiplicação e divisão de números racionais cuja representação decimal é finita por números naturais

Habilidades: (EF05MA08) Resolver e elaborar problemas de multiplicação e divisão com números naturais e com números racionais cuja representação decimal é finita (com multiplicador natural e divisor natural e diferente de zero), utilizando estratégias diversas, como cálculo por estimativa, cálculo mental e algoritmos.

Atividade 3: Crie um algoritmo em portugol, que calcule solicite duas notas e retorne a média aritmética das notas.

```
1 Algoritmo "Calculo da Média Aritmética de duas notas quaisquer"
2 // Autor(a)     : Juliano Thadeo Alves da Silva
3
4 Var
5 // Seção de Declarações das variáveis
6 n1, n2, media: real
7
8 Inicio
9 // Seção de Comandos, procedimento, funções, operadores, etc...
10 Escreva("Entre com o valor da primeira nota:")
11 Leia(n1)
12 Escreva("Entre com o valor da segunda nota:")
13 Leia(n2)
14 media <- (n1+n2)/2
15 Escreval("A média aritmética das notas é ", media)
16
17 Fimalgoritmo
```

Figura 6.9: Exemplo: algoritmo cálculo da média aritmética.

Unidades temáticas: Álgebra

Objetos de conhecimento:Grandezas diretamente proporcionais

Habilidades: (EF05MA12) Resolver problemas que envolvam variação de proporcionalidade direta entre duas grandezas, para associar a quantidade de um produto ao valor a pagar, alterar as quantidades de ingredientes de receitas, ampliar ou reduzir escala em mapas, entre outros.

Atividade 4: Crie um algoritmo em portugol, que solicite o valor unitário de um produto e a quantidade desejada, e por último retorne, o valor total a pagar.

```
1  Algoritmo "Calcule o valor total a pagar, solicitando a qtd e valor unitário"
2  // Autor(a)     : Juliano Thadeo Alves da Silva
3
4  Var
5  // Seção de Declarações das variáveis
6  qtd, valor, total: real
7
8  Inicio
9  // Seção de Comandos, procedimento, funções, operadores, etc...
10 Escreva("Entre com o valor do produto comprado:")
11 Leia(valor)
12 Escreva("Entre com a quantidade:")
13 Leia(qtd)
14 total <- qtd * valor
15 Escreval("O valor total a pagar é ", total)
16
17 Fimalgoritmo
```

Figura 6.10: Exemplo: algoritmo grandeza direta.

Unidades temáticas: Geometria

Objetos de conhecimento: Plano cartesiano: coordenadas cartesianas (1^{o} quadrante) e representação de deslocamentos no plano cartesiano

Habilidades: (EF05MA15) Interpretar, descrever e representar a localização ou movimentação de objetos no plano cartesiano (1^{o} quadrante), utilizando coordenadas cartesianas, indicando mudanças de direção e de sentido e giros.

Atividade 5: Escrever um algoritmo simples em português estruturado. Por exemplo, Quais passos são necessário para se descolar de um ponto A para um ponto B. O docente deverá ir aumentando a dificuldade a mediada que os alunos consigam realizar as atividades iniciais, é importante observar que existem vários passos/caminhos que podem ser utilizados para o deslocamento.

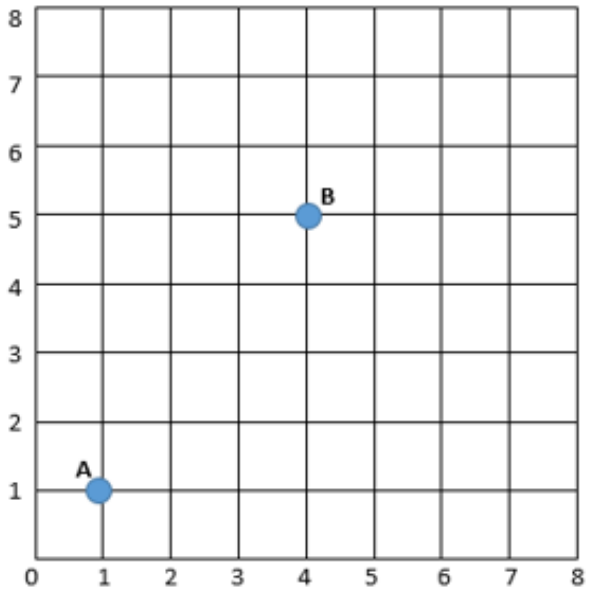


Figura 6.11: Exemplo: deslocamento no plano cartesiano.

Atividade 6: Utilizar os app´s SpriteBox e LightBot, onde os alunos podem definir a sequência de passos e depois executar, e se necessário pode corrigir o algoritmo.

Objetivo das atividades: Compreender o conceito de algoritmo como uma sequência de passo ou instruções, por meio de linguagem verbal. Trabalhar todos os pilares do pensamento computacional, e elementos de programação como estrutura condicional e repetição.

6.8 Atividade Matemática - 6º ano

Unidades temáticas: Números

Objetos de conhecimento: Fluxograma para determinar a paridade de um número natural

Habilidades: (EF06MA04) Construir algoritmo em linguagem natural e representá-lo por fluxograma que indique a resolução de um problema simples (por exemplo, se um número natural qualquer é par).

Atividade 1: Escreva um algoritmo que leia um número inteiro, e determine a paridade do mesmo.

```
Algoritmo "Leia um número e determine sua paridade"
// Autor(a)     : Juliano Thadeo Alves da Silva

Var
// Seção de Declarações das variáveis
num: inteiro

Inicio
// Seção de Comandos, procedimento, funções, operadores, etc...
Escreva("Entre o número que deseja saber a paridade:")
Leia(num)
Se num % 2 = 0 entao
   Escreval("O número digitado é par")
senao
   Escreval("O número digitado é ímpar")
fimse

Fimalgoritmo
```

Figura 6.12: Exemplo: algoritmo retorna paridade de um número inteiro.

Unidades temáticas: Números

Objetos de conhecimento: Cálculo de porcentagens por meio de estratégias diversas, sem fazer uso da "regra de três"

Habilidades: (EF06MA13) Resolver e elaborar problemas que envolvam porcentagens, com base na ideia de proporcionalidade, sem fazer uso da ?regra de três?, utilizando estratégias pessoais, cálculo mental e calculadora, em contextos de educação financeira, entre outros.

Atividade 2: Uma família deseja guardar um percentual de sua renda para investimentos futuros. Crie um algoritmo que leia a renda familiar e o percentual que desejam poupar, e retorne o valor que deverá ser guardado para investimentos futuros.

```
Algoritmo "Leia a renda familiar e o percentual que desejam poupar, e retorne o valor"
// Autor(a)     : Juliano Thadeo Alves da Silva

Var
// Seção de Declarações das variáveis
renda, percentual, poupado: real

Inicio
// Seção de Comandos, procedimento, funções, operadores, etc...
Escreva("Entre com o valor da renda familiar:")
Leia(renda)
Escreva("Entre com o percentual de deseja poupar:")
Leia(percentual)
poupado <- renda * percentual/100
Escreva("O valor que deve ser poupado é ", poupado)

Fimalgoritmo
```

Figura 6.13: Exemplo: algoritmo retorna valor poupado.

Unidades temáticas: Geometria

Objetos de conhecimento: Prismas e pirâmides: planificações e relações entre seus elementos (vértices, faces e arestas)

Habilidades: (EF06MA17) Quantificar e estabelecer relações entre o número de vértices, faces e arestas de prismas e pirâmides, em função do seu polígono da base, para resolver problemas e desenvolver a percepção espacial.

Atividade 3: Crie um algoritmo que solicite a quantidade de lados do polígono da base de uma pirâmide, e retorne à quantidade de vértices, faces e arestas desse polígono.

```
1  Algoritmo "Quantidade de vértices, faces e arestas polígono base de uma pirâmide"
2  // Autor(a)     : Juliano Thadeo Alves da Silva
3
4  Var
5  // Seção de Declarações das variáveis
6  vertice, face, aresta, poligono, lado: inteiro
7
8  Inicio
9  // Seção de Comandos, procedimento, funções, operadores, etc...
10 Escreva("Quantos lados tem o polígono na base da pirâmide:")
11 Leia(lado)
12 Escreval("A pirâmide com base poligonal de ", lado, " lados tem: ")
13 Escreval("- vértices: ", lado+1)
14 Escreval("- faces: ", lado+1)
15 Escreval("- arestas: ", lado*2)
16
17 Fimalgoritmo
```

Figura 6.14: Exemplo: algoritmo retorna número de vértice, faces e arestas.

Unidades temáticas: Geometria

Objetos de conhecimento: Construção de retas paralelas e perpendiculares, fazendo uso de réguas, esquadros e softwares

Habilidades: (EF06MA23) Construir algoritmo para resolver situações passo a passo (como na construção de dobraduras ou na indicação de deslocamento de um objeto no plano segundo pontos de referência e distâncias fornecidas etc.).

Atividade 4: Utilizar os app´s SpriteBox e LightBot, onde os alunos podem definir a sequência de passos e depois executar, e se necessário pode corrigir o algoritmo.

Objetivo da atividade: Compreender o conceito de algoritmo como uma sequência de passo ou instruções, por meio de linguagem verbal. Trabalhar todos os pilares do pensamento computacional, e elementos de programação como loop (estrutura de repetição) e estrutura de condição.

6.9 Atividade Matemática - 7º ano

Unidades temáticas: Números

Objetos de conhecimento: Cálculo de porcentagens e de acréscimos e decréscimos simples

Habilidades: (EF07MA02) Resolver e elaborar problemas que envolvam porcentagens, como os que lidam com acréscimos e decréscimos simples, utilizando estratégias pessoais, cálculo mental e calculadora, no contexto de educação financeira, entre outros.

Atividade 1: Crie um algoritmo que solicite o valor de título, e calcule o seu valor atualizado, sabendo que após o vencimento é cobrado o valor inicial da dívida mais 5% da mesma.

```
1 Algoritmo "Calcule o valor atualizado"
2 // Autor(a)     : Juliano Thadeo Alves da Silva
3
4 Var
5 // Seção de Declarações das variáveis
6 valor: real
7
8 Inicio
9 // Seção de Comandos, procedimento, funções, operadores, etc...
10 Escreva("Digite o valor original do título vencido:")
11 Leia(valor)
12 Escreval("O valor à vista do produto é ", valor + valor * 5/100)
13
14 Fimalgoritmo
```

Figura 6.15: Exemplo: algoritmo retorna valor atualizado.

Atividade 2: Dado o valor de um produto e um desconto à vista, crie um algoritmo que retorne o valor do produto com o desconto informado.

```
1 Algoritmo "Calcule o valor de um produto com desconto"
2 // Autor(a)     : Juliano Thadeo Alves da Silva
3
4 Var
5 // Seção de Declarações das variáveis
6 valor, desconto: real
7
8 Inicio
9 // Seção de Comandos, procedimento, funções, operadores, etc...
10 Escreva("Qual o valor do produto:")
11 Leia(valor)
12 Escreva("Qual o percentual de desconto à vista:")
13 Leia(desconto)
14 Escreval("O valor à vista do produto é ", valor - valor * desconto/100)
15
16 Fimalgoritmo
```

Figura 6.16: Exemplo: algoritmo retorna valor do produto com desconto.

Unidades temáticas: Números

Objetos de conhecimento: Fração e seus significados: como parte de inteiros, resultado da divisão, razão e operador

Habilidades: (EF07MA05) Resolver um mesmo problema utilizando diferentes algoritmos.

(EF07MA06) Reconhecer que as resoluções de um grupo de problemas que têm a mesma estrutura podem ser obtidas utilizando os mesmos procedimentos.

(EF07MA07) Representar por meio de um fluxograma os passos utilizados para resolver um grupo de problemas.

Atividade 3: Elaborar um algoritmo que efetue a conversão de um valor em real (R$) para dólar (US$). O algoritmo deverá solicitar o valor da cotação do dólar e também a quantidade de reais disponíveis com o usuário.

```
 1 Algoritmo "Conversao monetaria: Real para Dolar"
 2 // Autor(a)     : Juliano Thadeo Alves da Silva
 3
 4 Var
 5 // Seção de Declarações das variáveis
 6 cotacao, v_real: real
 7
 8 Inicio
 9 // Seção de Comandos, procedimento, funções, operadores, etc...
10 Escreval("*** Realizando a conversão de Real para Dólar ***")
11 Escreva("Digite a cotação atual do dólar:")
12 Leia(cotacao)
13 Escreva("Digite o valor em real (R$):")
14 Leia(v_real)
15 Escreval("R$",v_real, " é equivalente a US$",v_real/cotacao)
16
17 Fimalgoritmo
```

Figura 6.17: Exemplo: algoritmo conversão real para dólar.

Atividade 4: Elaborar um algoritmo que efetue a conversão de um valor em real (R$) para euro. O algoritmo deverá solicitar o valor da cotação do euro e também a quantidade de reais disponíveis com o usuário.

```
1  Algoritmo "Conversao monetaria: Real para Euro"
2  // Autor(a)     : Juliano Thadeo Alves da Silva
3
4  Var
5  // Seção de Declarações das variáveis
6  cotacao, v_real: real
7
8  Inicio
9  // Seção de Comandos, procedimento, funções, operadores, etc...
10 Escreval("*** Realizando a conversão de Real para Euro ***")
11 Escreva("Digite a cotação atual do euro:")
12 Leia(cotacao)
13 Escreva("Digite o valor em real (R$):")
14 Leia(v_real)
15 Escreval("R$",v_real, " é equivalente a €",v_real/cotacao)
16
17 Fimalgoritmo
```

Figura 6.18: Exemplo: algoritmo conversão real para euro.

Atividade 5: Escrever um algoritmo para determinar o consumo médio de um automóvel, sendo fornecida a distância total percorrida pelo automóvel e o total de combustível gasto.

```
1  Algoritmo "Consumo médio de um automóvel"
2  // Autor(a)     : Juliano Thadeo Alves da Silva
3
4  Var
5  // Seção de Declarações das variáveis
6  distancia, qtd_combustivel: real
7
8  Inicio
9  // Seção de Comandos, procedimento, funções, operadores, etc...
10 Escreval("*** Calculando o consumo médio de um automóvel ***")
11 Escreva("Digite a distância percorrida:")
12 Leia(distancia)
13 Escreva("Digite a quantidade de combutível gasta:")
14 Leia(qtd_combustivel)
15 Escreval("O consumo médio é de ", distancia/qtd_combustivel)
16
17 Fimalgoritmo
```

Figura 6.19: Exemplo: algoritmo consumo médio de combustível

Atividade 6: A Loja MT está vendendo seus produtos em até 6 (seis) prestações sem juros. Faça um algoritmo que receba um valor de uma compra e o número de parcela, e mostre o valor das prestações.

```
1 Algoritmo "Parcelando as compras em até 6 x sem juros"
2 // Autor(a)     : Juliano Thadeo Alves da Silva
3
4 Var
5 // Seção de Declarações das variáveis
6 valor, parcela: real
7
8 Inicio
9 // Seção de Comandos, procedimento, funções, operadores, etc...
10 Escreval("*** Calculando valor das prestações ***")
11 Escreva("Digite o valor da compra:")
12 Leia(valor)
13 Escreva("Digite a quantidade de parcelas:")
14 Leia(parcela)
15 Escreval("A compra pode ser parcela em ", parcela, " de ", valor/parcela)
16
17 Fimalgoritmo
```

Figura 6.20: Exemplo: algoritmo compra parcelada

Unidades temáticas: Álgebra

Objetos de conhecimento: Linguagem algébrica: variável e incógnita

Habilidades: (EF07MA13) Compreender a ideia de variável, representada por letra ou símbolo, para expressar relação entre duas grandezas, diferenciando-a da ideia de incógnita.

Atividade 7: Crie um algoritmo que solicite um número inteiro e retorne o seu fatorial.

```
1 Algoritmo "Fatorial de um número"
2 // Autor(a)     : Juliano Thadeo Alves da Silva
3
4 Var
5 // Seção de Declarações das variáveis
6 fatorial, x, num: inteiro
7
8 Inicio
9 // Seção de Comandos, procedimento, funções, operadores, etc...
10 Escreval("----- Calculando o Fatorial de um número -------")
11 Escreva("Digite um número:")
12 Leia(num)
13 fatorial <- 1
14 Para x de 1 até num faca
15    fatorial <- fatorial * x
16 fimpara
17 Escreval("O fatorial do número", num, " é", fatorial)
18
19 Fimalgoritmo
```

Figura 6.21: Exemplo: algoritmo retorna valor atualizado.

Unidades temáticas: Geometria

Objetos de conhecimento: Triângulos: construção, condição de existência e soma das medidas dos ângulos internos

Habilidades: (EF07MA24) Construir triângulos, usando régua e compasso, reconhecer a condição de existência do triângulo quanto à medida dos lados e verificar que a soma das medidas dos ângulos internos de um triângulo é 180°.

(EF07MA26) Descrever, por escrito e por meio de um fluxograma, um algoritmo para a construção de um triângulo qualquer, conhecidas as medidas dos três lados.

(EF07MA31) Estabelecer expressões de cálculo de área de triângulos e de quadriláteros.

Atividade 8: Crie um algoritmo que calcule a área de retângulo, dados a medida dos seus lados.

```
1 Algoritmo "Calculo da área retângulo"
2 // Autor(a)     : Juliano Thadeo Alves da Silva
3
4 Var
5 // Seção de Declarações das variáveis
6 lado1, lado2, area: real
7
8 Inicio
9 // Seção de Comandos, procedimento, funções, operadores, etc...
10 Escreval("----- Calculando a área de retângulo -------")
11 Escreva("Digite a medida em metros de um dos lados:")
12 Leia(lado1)
13 Escreva("Digite a medida em metros do outro lado:")
14 Leia(lado2)
15 area <- lado1 * lado2
16 Escreval("A área do retângulo é", area)
17
18 Fimalgoritmo
```

Figura 6.22: Exemplo: algoritmo retorna área de um retângulo.

Atividade 9: Crie um algoritmo que calcule a área de um triângulo, dados as suas medidas da base e altura.

```
1 Algoritmo "Calculo da área de um triângulo"
2 // Autor(a)     : Juliano Thadeo Alves da Silva
3
4 Var
5 // Seção de Declarações das variáveis
6 base, altura, area: real
7
8 Inicio
9 // Seção de Comandos, procedimento, funções, operadores, etc...
10 Escreval("----- Calculando a área de um triângulo -------")
11 Escreva("Digite a medida da base:")
12 Leia(base)
13 Escreva("Digite a medida da altura:")
14 Leia(altura)
15 area <- base*altura/2
16 Escreval("A área do terreno dado é", area)
17
18 Fimalgoritmo
```

Figura 6.23: Exemplo: algoritmo retorna área de um triângulo.

Atividade 10: Crie um algoritmo que verifique a condição de existência de um triângulo, dadas as medidas de seus lados.

```
1 Algoritmo "Condição de existência de um triângulo"
2 // Autor(a)     : Juliano Thadeo Alves da Silva
3
4 Var
5 // Seção de Declarações das variáveis
6 x, y, z : real
7
8 Inicio
9 // Seção de Comandos, procedimento, funções, operadores, etc...
10 Escreval("Verificando a existência de um triângulo")
11 Escreva("Digite a medida do 1° lado do triângulo:")
12 Leia(x)
13 Escreva("Digite a medida do 2° lado do triângulo:")
14 Leia(y)
15 Escreva("Digite a medida do 3° lado do triângulo:")
16 Leia(z)
17 se (x < y + x) e (y < x + z) e (z < x + y) entao
18    Escreval("As medidas informadas formam um triângulo")
19 senao
20    Escreval("As medidas informadas NÃO formam um triângulo")
21 fimse
22 Fimalgoritmo
```

Figura 6.24: Exemplo: algoritmo verifica a condição de existência de um triângulo.

Atividade 11: Escrever um algoritmo que leia três valores e verifique se eles podem ser os lados de um triângulo. Se forem informar qual o tipo de triângulo que eles formam: equilátero, isósceles ou escaleno.

```
Algoritmo "Condição de existência de um triângulo e classificação"
// Autor(a)     : Juliano Thadeo Alves da Silva

Var
// Seção de Declarações das variáveis
x, y, z : real

Inicio
// Seção de Comandos, procedimento, funções, operadores, etc...
Escreval("Verificando a existência e classificação de um triângulo")
Escreva("Digite a medida do 1° lado do triângulo:")
Leia(x)
Escreva("Digite a medida do 2° lado do triângulo:")
Leia(y)
Escreva("Digite a medida do 3° lado do triângulo:")
Leia(z)
se (x < y + x) e (y < x + z) e (z < x + y) entao
   se (x = y) e (x = z) entao
      Escreval("O triângulo é equilátero")
   senao
      se (x = y) ou (x = z) entao
         Escreval("O triângulo é isóscele")
      senao
         Escreval("O triângulo é escaleno")
      fimse
   fimse
senao
   Escreval("As medidas informadas NÃO formam um triângulo")
fimse
Fimalgoritmo
```

Figura 6.25: Exemplo: algoritmo verifica a existência e classificação de um triângulo.

6.10 Atividade Matemática - 8º ano

Unidades temáticas: Números

Objetos de conhecimento: Potenciação e radiciação

Habilidades: (EF08MA02) Resolver e elaborar problemas usando a relação entre potenciação e radiciação, para representar uma raiz como potência de expoente fracionário.

Atividade 1: Faça um algoritmo que solicite a altura e peso de uma pessoa, e retorne o seu IMC.

```
1 Algoritmo "Calculo do IMC"
2 // Autor(a)     : Juliano Thadeo Alves da Silva
3
4 Var
5 // Seção de Declarações das variáveis
6 imc, altura, peso : real
7
8 Inicio
9 // Seção de Comandos, procedimento, funções, operadores, etc...
10 Escreval("Verificando o Índice de Massa Corporal")
11 Escreva("Digite a sua altura:")
12 Leia(altura)
13 Escreva("Digite o seu peso:")
14 Leia(peso)
15 imc <- peso/altura^2
16 Escreval("O seu IMC é", imc)
17 Fimalgoritmo
```

Figura 6.26: Exemplo: algoritmo retorna valor atualizado.

Atividade 2: Faça um algoritmo que receba um valor que foi depositado e exiba o valor com rendimento após um mês. Considere fixa a taxa de juros da poupança em 0,70% a.m.

```
1 Algoritmo "Rendimentos da poupança"
2 // Autor(a)     : Juliano Thadeo Alves da Silva
3
4 Var
5 // Seção de Declarações das variáveis
6 depositado, rendimento, atualizado : real
7
8 Inicio
9 // Seção de Comandos, procedimento, funções, operadores, etc...
10 Escreval("Verificando o valor corrigido ápos um mês na poupança")
11 Escreva("Digite o valor depositado:")
12 Leia(depositado)
13 rendimento <- depositado*0.7/100
14 atualizado <- depositado + rendimento
15 Escreval("O rendimento da poupança em 1 mês é de:", rendimento)
16 Escreval("Dessa forma, o saldo atualizado é", atualizado)
17 Fimalgoritmo
```

Figura 6.27: Exemplo: algoritmo rendimento de um investimento.

Atividade 3: Faça um algoritmo que receba um valor que foi depositado e a quantidade de mês que será reajustado, e exiba o valor com rendimento após o período. Considere fixa a taxa de juros da poupança em 0,70% a.m.

```
Algoritmo "Rendimentos da poupança"
// Autor(a)     : Juliano Thadeo Alves da Silva

Var
// Seção de Declarações das variáveis
depositado, periodo, rendimento, atualizado : real

Inicio
// Seção de Comandos, procedimento, funções, operadores, etc...
Escreval("Verificando o valor corrigido ápos um mês na poupança")
Escreva("Digite o valor depositado:")
Leia(depositado)
Escreva("Digite a quantidade meses:")
Leia(periodo)
atualizado <- depositado * (1+0.7/100)^periodo
rendimento <- atualizado - depositado
Escreval("O rendimento da poupança no período foi de:", rendimento)
Escreval("Dessa forma, o saldo atualizado é", atualizado)
Fimalgoritmo
```

Figura 6.28: Exemplo: algoritmo rendimento de um investimento por um período.

Unidades temáticas: Números

Objetos de conhecimento: Porcentagens

Habilidades: (EF08MA04) Resolver e elaborar problemas, envolvendo cálculo de porcentagens, incluindo o uso de tecnologias digitais.

Atividade 4: Escrever um algoritmo que leia o salário fixo de um vendedor e o total de vendas efetuadas por ele no mês. Sabendo que este vendedor ganha 15% de comissão sobre suas vendas efetuadas, retorne o valor da comissão do mês e o seu salário final.

```
Algoritmo "Calculando salário final de um vendedor"
// Autor(a)     : Juliano Thadeo Alves da Silva

Var
// Seção de Declarações das variáveis
sal_fixo, sal_final, comissao, venda : real

Inicio
// Seção de Comandos, procedimento, funções, operadores, etc...
Escreval("Calculando o salário final de um vendedor")
Escreva("Digite o seu salário fixo:")
Leia(sal_fixo)
Escreva("Digite o valor total das vendas no mês:")
Leia(venda)
comissao <- venda*15/100
sal_final <- sal_fixo + comissao
Escreval("O valor da sua comissão é", comissao)
Escreval("O seu salário final é", sal_final)
Fimalgoritmo
```

Figura 6.29: Exemplo: algoritmo salário fixo com comissão.

Unidades temáticas: Álgebra

Objetos de conhecimento: Valor numérico de expressões algébricas

Habilidades: (EF08MA06) Resolver e elaborar problemas que envolvam cálculo do valor numérico de expressões algébricas, utilizando as propriedades das operações.

Atividade 5: Faça um algoritmo que calcule o valor atualizado de um título, sabendo que deve ser cobrado mora, e juros diário. Solicite que o usuário informe o valor do título, o percentual de mora, valor dos juros diário e a quantidade de dias em atraso, e retorne o valor atualizado do título.

```
 1 Algoritmo "Calculando o valor atualizado de um titulo"
 2 // Autor(a)     : Juliano Thadeo Alves da Silva
 3
 4 Var
 5 // Seção de Declarações das variáveis
 6 valor, mora, juros, dia_atrasado, valor_atual : real
 7
 8 Inicio
 9 // Seção de Comandos, procedimento, funções, operadores, etc...
10 Escreval("Calculando o valor atualizado de um título vencido")
11 Escreva("Digite o valor do documento:")
12 Leia(valor)
13 Escreva("Digite o percentual de mora:")
14 Leia(mora)
15 Escreva("Digite o valor do juros diário:")
16 Leia(juros)
17 Escreva("Digite a quantidade de dias em atraso:")
18 Leia(dia_atrasado)
19 valor_atual <- valor + (valor*mora/100) + (juros*dia_atrasado)
20 Escreval("O valor atualizado do documento é", valor_atual)
21 Fimalgoritmo
```

Figura 6.30: Exemplo: algoritmo valor atualizado de um título.

Unidades temáticas: Álgebra

Objetos de conhecimento: Sequências recursivas e não recursivas

Habilidades: (EF08MA11) Identificar a regularidade de uma sequência numérica recursiva e construir um algoritmo por meio de um fluxograma que permita indicar os números seguintes.

Atividade 6: Crie um algoritmo e retorne os 20 primeiros números da sequência de Fibonacci.

```
Algoritmo "Sequência de Fibonacci"
// Autor(a)     : Juliano Thadeo Alves da Silva

Var
// Seção de Declarações das variáveis
v1, v2, v, x: inteiro

Inicio
// Seção de Comandos, procedimento, funções, operadores, etc...
Escreval("Mostrando os vinte primeiros números da sequência de Fibonacci")
v1 <- 1
v2 <- 1
Escreval (v1)
Escreval (v2)
Para x de 1 ate 18 faca
      v <- v1 + v2
      Escreval(v)
      v1<-v2
      v2<-v
fimpara
Fimalgoritmo
```

Figura 6.31: Exemplo: algoritmo sequência de Fibonacci.

Unidades temáticas: Geometria

Objetos de conhecimento: Área de figuras planas

Habilidades: (EF08MA19) Resolver e elaborar problemas que envolvam medidas de área de figuras geométricas, utilizando expressões de cálculo de área (quadriláteros, triângulos e círculos), em situações como determinar medida de terrenos.

Atividade 7: Crie um algoritmo que calcule a área de um terreno retangular, dados a medida dos seus lados.

```
Algoritmo "Calculo da área de um terreno retangular"
// Autor(a)     : Juliano Thadeo Alves da Silva

Var
// Seção de Declarações das variáveis
lado1, lado2, area: real

Inicio
// Seção de Comandos, procedimento, funções, operadores, etc...
Escreval("----- Calculando a área de um terreno retangular -------")
Escreva("Digite a medida em metros de um dos lados:")
Leia(lado1)
Escreva("Digite a medida em metros do outro lado:")
Leia(lado2)
area <- lado1 * lado2
Escreval("A área do terreno dado é", area)

Fimalgoritmo
```

Figura 6.32: Exemplo: algoritmo área de um terreno retangular.

Atividade 8: Crie um algoritmo que calcule a área de um terreno circular, dada a medida do seu raio.

```
1 Algoritmo "Calculo da área de um terreno circular"
2 // Autor(a)     : Juliano Thadeo Alves da Silva
3
4 Var
5 // Seção de Declarações das variáveis
6 raio, area: real
7
8 Inicio
9 // Seção de Comandos, procedimento, funções, operadores, etc...
10 Escreval("----- Calculando a área de um terreno circular -------")
11 Escreva("Digite a medida em metros do raio do terreno:")
12 Leia(raio)
13 area <- pi*raio^2
14 Escreval("A área do terreno dado é", area)
15
16 Fimalgoritmo
```

Figura 6.33: Exemplo: algoritmo área de um terreno circular.

Unidades temáticas: Probabilidade e estatística

Objetos de conhecimento: Medidas de tendência central e de dispersão

Habilidades: (EF08MA25) Obter os valores de medidas de tendência central de uma pesquisa estatística (média, moda e mediana) com a compreensão de seus significados e relacioná-los com a dispersão de dados, indicada pela amplitude.

Atividade 9: Crie um algoritmo que leia 10 números e retorne a média aritmética dos valores.

```
1 Algoritmo "Média aritmética"
2 // Autor(a)     : Juliano Thadeo Alves da Silva
3
4 Var
5 // Seção de Declarações das variáveis
6 media, nota, soma: real
7 x: inteiro
8
9 Inicio
10 // Seção de Comandos, procedimento, funções, operadores, etc...
11 Escreval("Calculando a média aritmética de dez números")
12 Para x de 1 ate 10 faca
13      Escreva("Digite a nota",x,":")
14      Leia(nota)
15      soma <- soma + nota
16 fimpara
17 media <- soma/x
18 Escreval("A média aritmética dos números digitados é:",media)
19 Fimalgoritmo
```

Figura 6.34: Exemplo: algoritmo média aritmética.

6.11 Atividade Matemática - 9º ano

Unidades temáticas: Números

Objetos de conhecimento: Potências com expoentes negativos e fracionários

Habilidades: (EF09MA03) Efetuar cálculos com números reais, inclusive potências com expoentes fracionários.

Atividade 1: Crie um algoritmo que calcule as raízes para equação do segundo grau.

```
1  Algoritmo "Equação do segundo grau"
2  // Autor(a)     : Juliano Thadeo Alves da Silva
3
4  Var
5  // Seção de Declarações das variáveis
6  delta, x1, x2: real
7  a, b, c: inteiro
8
9  Inicio
10 // Seção de Comandos, procedimento, funções, operadores, etc...
11 Escreval("Calculando as raízes da equação do segundo grau")
12 Escreva("Digite o valor de [a]: ")
13 leia(a)
14 Escreva("Digite o valor de [b]: ")
15 leia(b)
16 Escreva("Digite o valor de [c]: ")
17 leia(c)
18 delta <- b^2 -(4*a*c)
19 Escreval("Delta = ", delta)
20 se (delta < 0) entao
21    escreval("Não existem raízes")
22 senao
23    x1 <- (-b + delta^(1/2)) / (2*a)
24    x2 <- (-b - delta^(1/2)) / (2*a)
25    Escreval("----------------------")
26    Escreval("-    X1 = ", x1:6:2,"    -")
27    Escreval("-    X2 = ", x2:6:2,"    -")
28    Escreval("----------------------")
29 fimse
30 Fimalgoritmo
```

Figura 6.35: Exemplo: algoritmo calcula raízes da equação do 2º grau.

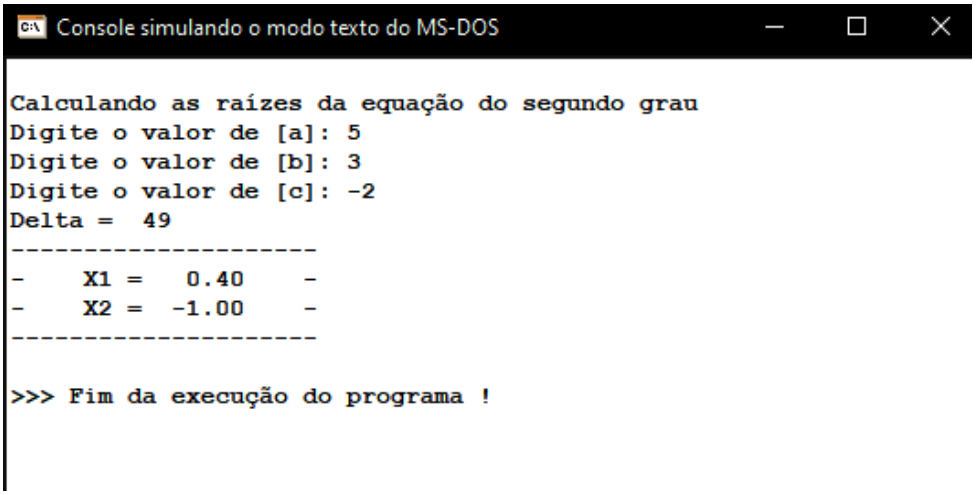


Figura 6.36: Exemplo: console da execução do algoritmo.

Unidades temáticas: Números

Objetos de conhecimento: Números reais: notação científica e problemas

Habilidades: (EF09MA04) Resolver e elaborar problemas com números reais, inclusive em notação científica, envolvendo diferentes operações.

Atividade 2: Crie um algoritmo que mostre os trinta primeiros números da sequência de Finobacci e mostre o valor do número áureo (a partir da divisão dos dois últimos termos) com 8 casas decimais.

```
Algoritmo "Sequência de Fibonacci"
// Autor(a)     : Juliano Thadeo Alves da Silva

Var
// Seção de Declarações das variáveis
v1, v2, v, x: inteiro

Inicio
// Seção de Comandos, procedimento, funções, operadores, etc...
Escreval("Mostrando os trinta primeiros números da sequência de Fibonacci")
v1 <- 1
v2 <- 1
Escreval (" 1°:",v1)
Escreval (" 2°:",v2)
Para x de 3 ate 30 faca
      v <- v1 + v2
      Escreval(x,"°:",v)
      v1<-v2
      v2<-v
fimpara
Escreval("*********************************")
Escreval("*   Número áureo:",v2/v1:1:8,"   *")
Escreval("*********************************")
Fimalgoritmo
```

Figura 6.37: Exemplo: algoritmo Fibonacci e número áureo.

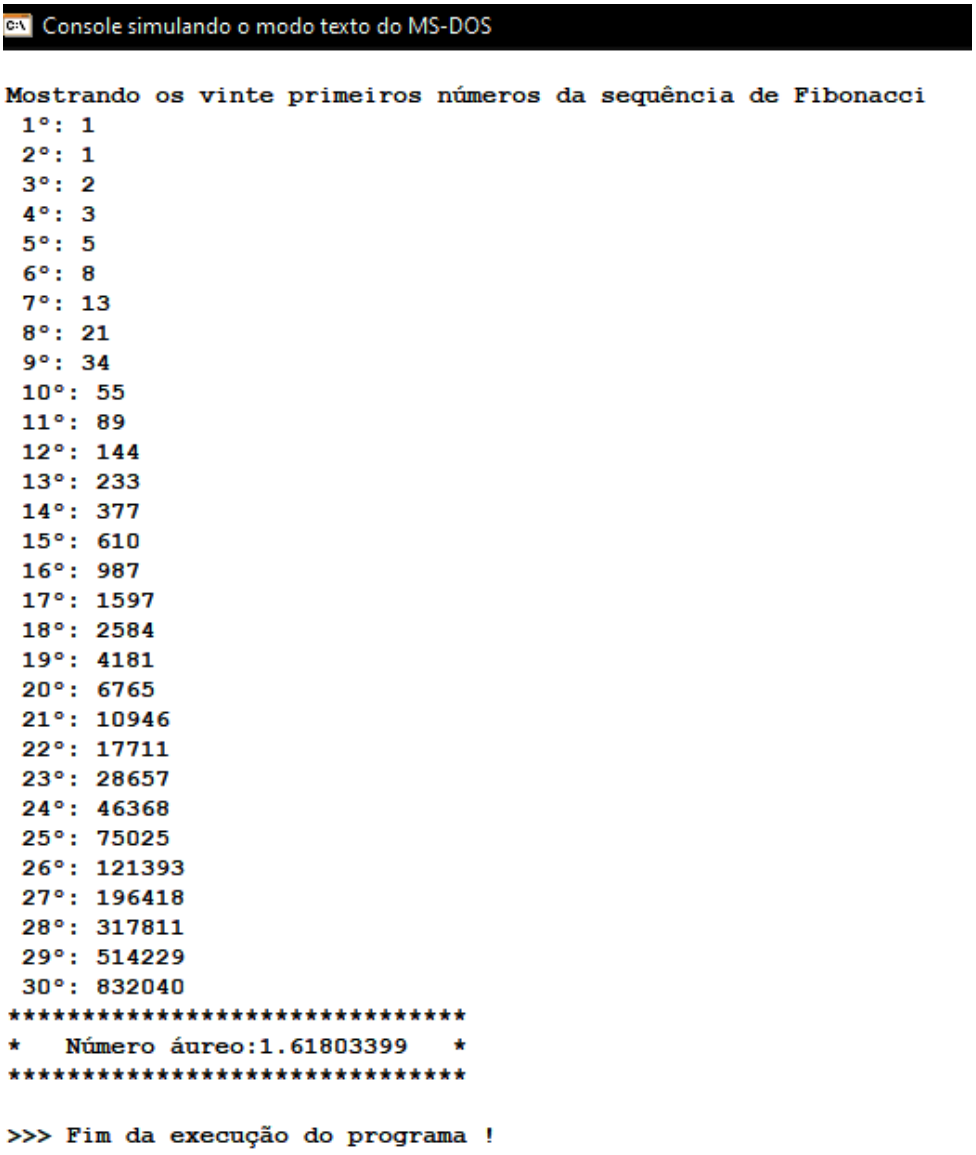


Figura 6.38: Exemplo: console da execução do algoritmo.

Unidades temáticas: Números

Objetos de conhecimento: Números reais: notação científica e problemas

Habilidades: (EF09MA05) Resolver e elaborar problemas que envolvam porcentagens, com a ideia de aplicação de percentuais sucessivos e a determinação das taxas percentuais, referencialmente com o uso de tecnologias digitais, no contexto da educação financeira.

Atividade 3: Crie um algoritmo que leia o salário bruto de uma pessoa, e retorne o valor que será descontado de Imposto de Renda.

Tabela 6.2: Tabela base para cálculo do imposto de renda em 2020.

Base de cálculo	**Alíquota**	**Dedução do IRPF**
Até R$ 1.903,98	Isento	R$ 0,00
De R$ 1.903,99 até R$ 2.826,65	7,5%	R$ 142,80
De R$ 2.826,66 até R$ 3.751,05	15%	R$ 354,80
De R$ 3.751,06 até R$ 4.664,68	22,5%	R$ 636,13
Acima R$ 4.664,68	27,5%	R$ 869,36

Fonte: Receita Federal - imposto de renda pessoa física.

```
Algoritmo "Leia salário bruto e retorne o valor dos descontos de IR"
// Autor(a)     : Juliano Thadeo Alves da Silva

Var
// Seção de Declarações das variáveis
salario, aliquota, deduzir, liquido: real

Inicio
// Seção de Comandos, procedimento, funções, operadores, etc...
Escreva("Entre o salário bruto:")
Leia(salario)
Se salario < 1903.98 entao
   aliquota <- 0
   deduzir <- 0
senao
   Se salario < 28626.65 entao
      aliquota <- 7.5
      deduzir <- 142.8
   senao
      Se salario < 3751.05 entao
         aliquota <- 15
         deduzir <- 354.8
      senao
         Se salario < 4664.68 entao
            aliquota <- 22.5
            deduzir <- 636.13
         senao
            aliquota <- 27.5
            deduzir <- 869.36
         fimse
      fimse
   fimse
fimse
liquido <- salario - (salario * aliquota/100 - deduzir)
Escreva("O salario bruto é ", salario, " e o líquido é ", liquido)
Fimalgoritmo
```

Figura 6.39: Exemplo: algoritmo cálculo do imposto de renda.

Unidades temáticas: Geometria

Objetos de conhecimento: Polígonos regulares

Habilidades: (EF09MA15) Descrever, por escrito e por meio de um fluxograma, um algoritmo para a construção de um polígono regular cuja medida do lado é conhecida, utilizando régua e compasso, como também softwares.

Atividade 4: Crie um algoritmo com os passos necessário para criar um triângulo equilátero de 4cm, utilizando apenas régua e compasso.

1. Marque um ponto A;
2. Traçar segmento de 4 cm, ponto B;
3. Trace a circunferência de raio 4cm, com centro em A;
4. Trace outra circunferência de raio 4cm, com centro em B;
5. Na interseção das circunferências, marque o ponto C;
6. Trace mais dois segmentos AC e BC.

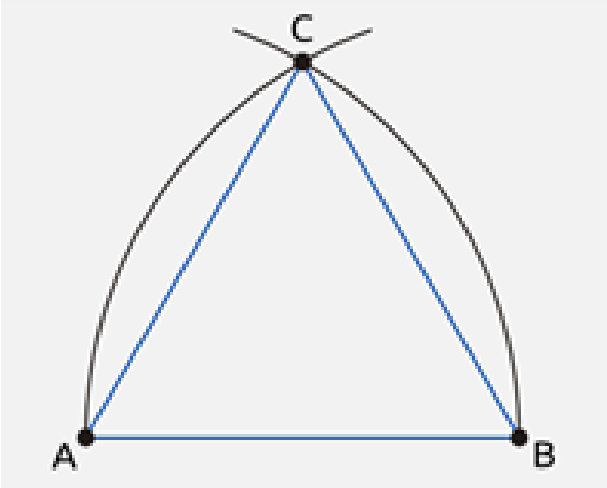


Figura 6.40: Exemplo: algoritmo triângulo equilátero.

Atividade 5: Crie um algoritmo com os passos necessário para criar um quadrado de 5cm, utilizando apenas régua e compasso.

1. Marcar o ponto A;
2. Traçar segmento de 5 cm;
3. Traçar a perpendicular ao lado que passa por A;
4. Traçar a circunferência de raio 5 e centro A;
5. Marcar o ponto C de interseção da perpendicular com a circunferência;
6. Traçar duas circunferências de raio 5cm e centros em B e C;
7. Marcar o ponto D, interseção das circunferências.

Considerações finais

Esta pesquisa procurou apresentar uma estratégia para auxiliar no processo de ensino aprendizagem do pensamento computacional, sugerindo uma abordagem construcionista e a utilização de metodologias ativas. Observando os últimos anos de ensino de computação nas escolas, existem dois segmentos bem distintos: aprender sobre computadores e aprender com computadores. Embora seja importante o primeiro segmento, o último tem papel crucial no processo de ensino e aprendizagem.

Com a elaboração da BNCC (2018), o pensamento computacional passou a fazer parte do nosso currículo nacional, logo, não podemos permitir que seja esquecido ou tratado superficialmente. Nós, professores e escolas, temos o dever de oferecer um currículo amplo e equilibrado que aparelhe os alunos com técnicas utilizadas no pensamento computacional e dessa forma, possam compreender e mudar o mundo.

Assim, o objetivo do pensamento computacional se move para a utilização do algoritmo na resolução de problemas do cotidiano. O pensamento computacional é uma habilidade que encoraja, fortalece e empodera, e sendo assim, todos os alunos devem compreender e desenvolver essa competência. Os alunos que podem pensar computacionalmente são mais capazes de resolver problemas, compreender e usar a tecnologia baseada em computador e, portanto, estão melhor preparados para o presente e o futuro.

Almejamos que as atividades aqui apresentadas possam inspirar pesquisas relacionadas ao tema e incentivar o uso de metodologias ativas que coloquem o aluno no centro do processo de ensino. Esperamos que esse material possa contribuir e apoiar os professores de Matemática a desenvolver com excelência o ensino do pensamento computacional em suas salas de aula, promovendo o uso apropriado da tecnologia para apoiar o processo de ensino. Fornecendo amplo escopo de atividades didáticas para que os alunos possam desenvolver conhecimento, habilidades e atitudes para solucionar problemas.

Referências Bibliográficas

Barba, L. A. e Solomon, C. (2016). Computational thinking: I do not think it means what you think it means. Disponível em https://lorenabarba.com/blog/computational-thinking-i-do-not-think-it-means-what-you-think-it-means/ Acesso em: 03/05/2020.

Blikstein, P. (2008). O pensamento computacional e a reinvenção do computador na educação. Disponível em http://www.blikstein.com/paulo/documents/online/ol_pensamento_computacional.html Acesso em: 07/05/2020.

Brackmann, C. P. (2017). *Desenvolvimento do pensamento computacional através de atividades desplugadas na educação básica Porto Alegre/RS – Brazil: Paradigmas para a Pesquisa sobre o Ensino Científico e Tecnológico.* Tese de Doutorado, CINTED–UFGRS, Porto Alegre/RS.

Brasil – MEC (2018). Ministério da Educação. Base Nacional Comum Curricular. Disponível em http://basenacionalcomum.mec.gov.br/a-base Acesso em: 20/07/2020.

CAS (2014). Developing computational thinking. teaching London computing. Disponível em http://teachinglondoncomputing.org/resources/developing-computational-thinking/ Acesso em: 10/05/2020.

CIEB (2018). Currículo de referência Centro de Inovação para a Educação Brasileira. Disponível em https://curriculo.cieb.net.br/ Acesso em: 09/04/2020.

CSTA (2011). K-12 Computer Science standards. Disponível em http://scratch.ttu.ee/failid/CSTA_K-12_CSS.pdf Acesso em: 16/04/2020.

Cuny, J., Snyder, L., e Wing, J. M. (2010). Demystifying computational thinking for non-

computer scientists. Disponível em https://www.cs.cmu.edu/link/research-notebook-computational-thinking-what-and-why Acesso em: 20/04/2020.

Dante, L. R. (2003). *Didática da resolução de problemas de matemática, 1a. a 5a. series: para estudantes do curso de Magisterio e professores do 1o. grau.* Ed. Ática, S. Paulo.

FGVcia (2020). 31a Pesquisa Anual do Uso de TI. Disponível em https://eaesp.fgv.br/producao-intelectual/pesquisa-anual-uso-ti Acesso em: 06/08/2020.

Freire, P. (2019). *Pedagogia da autonomia: Saberes necessários à prática educativa.* Ed. Paz e Terra, S. Paulo.

Grover, S. e Pea, R. (2013). Computational thinking in k–12: A review of the state of the field. *Educational researcher*, 42(1):38–43.

ISTE (2016). Iste standards for students. Disponível em http://www.iste.org/standards/standards/for-students#startstandards Acesso em: 10/07/2020.

Kapp, K. M. (2012). *The gamification of learning and instruction: game-based methods and strategies for training and education.* Ed. Pfeiffer, Pennsylvania, USA.

Liukas, L. (2016). *Hello Ruby: adventures in coding.* Ed. Penguin Books Australia, Hawthorn, Austrália.

Manzano, J. A., Lourenço, A. E., e Matos, E. (2018). *Algoritmos Técnicas de Programação.* Ed. Érica, S. Paulo.

Manzano, J. A. e Oliveira, J. F. (2019). *Algoritmos lógica para desenvolvimento de programação de computadores.* Ed. Érica, S. Paulo.

Mckinsey e Company (2020). Future of work. Disponível em https://www.mckinsey.com/featured-insights/future-of-work Acesso em: 23/04/2020.

Moran, J. e Bacich, L. (2018). Metodologias ativas para uma educação inovadora. In *Série desafios da educação.* Ed. Penso, P. Alegre.

Mueller, J. P. e Massaron, L. (2018). *Algoritmos para leigos. Os primeiros passos para o sucesso.* Ed. Alta Books, R. Janeiro.

OCDE (2020). Pisa – perguntas frequentes. Disponível em https://www.oecd.org/pisa/pisafaq/ Acesso em: 15/05/2020.

Papert, S. A. (1972). A computer laboratory for elementary schools. Artificial Intelligence Lab. Massachusetts, USA.

Papert, S. A. (1980). *Mindstorms: Children, Computers, and Powerful Ideas*. Ed. Basic books, New York.

Raabe, A., Zorzo, A. F., e Blikstein, P. (2020). Computação na educação básica: Fundamentos e experiências. In *Série Tecnologia e inovação na educação brasileira*. Ed. Penso, P. Alegre.

SENAC – Departamento Nacional (2018). Metodologias Ativas de Aprendizagem. Disponível http://www.extranet.senac.br/modelopedagogicosenac/ em Acesso em: 15/05/2020.

Wing, J. M. (2006). Computational thinking. *Communications of the ACM*, 49(3):33–35.

Wing, J. M. (2011). Research notebook: Computational thinking what and why. The link magazine. Pittsburgh, USA.