UNIVERSIDADE DO ESTADO DE SANTA CATARINA – UDESC
CENTRO DE CIÊNCIAS TECNOLÓGICAS – CCT
MESTRADO PROFISSIONAL EM MATEMÁTICA EM REDE NACIONAL – PROFMAT

NICOLE CRISTINE RECH

PENSAMENTO COMPUTACIONAL: UMA PROPOSTA DE COMO INTRODUZI-LO NA EDUCAÇÃO BÁSICA RELACIONANDO COM A MATEMÁTICA

JOINVILLE
2024

NICOLE CRISTINE RECH

PENSAMENTO COMPUTACIONAL: UMA PROPOSTA DE COMO INTRODUZI-LO NA EDUCAÇÃO BÁSICA RELACIONANDO COM A MATEMÁTICA

Dissertação apresentada ao Curso de Pós-Graduação Mestrado Profissional em Matemática em Rede Nacional da Universidade do Estado de Santa Catarina no Centro de Ciências Tecnológicas como requisito parcial para obtenção do grau de Mestre em Matemática.

Orientador: Profa. Dra. Viviane Maria Beuter

JOINVILLE
2024

Ficha catalográfica elaborada pelo programa de geração automática da

Biblioteca Universitária Udesc,

com os dados fornecidos pelo(a) autor(a)

Rech, Nicole Cristine
Pensamento Computacional: uma proposta de como introduzi-lo na Educação Básica relacionando com a Matemática / Nicole Cristine Rech. -- 2024.
143 p.

Orientadora: Viviane Maria Beuter
Dissertação (mestrado) -- Universidade do Estado de Santa Catarina, Centro de Ciências Tecnológicas, Programa de Pós-Graduação Profissional em Matemática em Rede Nacional, Joinville, 2024.

1. Pensamento Computacional. 2. Programação. 3. Matemática. 4. Educação Básica. 5. Atividades desplugadas. I. Beuter, Viviane Maria. II. Universidade do Estado de Santa Catarina, Centro de Ciências Tecnológicas, Programa de Pós-Graduação Profissional em Matemática em Rede Nacional. III. Título.

NICOLE CRISTINE RECH

PENSAMENTO COMPUTACIONAL: UMA PROPOSTA DE COMO INTRODUZI-LO NA EDUCAÇÃO BÁSICA RELACIONANDO COM A MATEMÁTICA

Dissertação apresentada ao Curso de Pós-Graduação Mestrado Profissional em Matemática em Rede Nacional da Universidade do Estado de Santa Catarina no Centro de Ciências Tecnológicas como requisito parcial para obtenção do grau de Mestre em Matemática.

Orientador: Profa. Dra. Viviane Maria Beuter

BANCA EXAMINADORA:

Viviane Maria Beuter
UDESC

Membros:

Ivanete Zuchi Siple
UDESC

Felipe Delfini Caetano Fidalgo
UFSC

Joinville, 29 de agosto de 2024.

AGRADECIMENTOS

Agradeço a todos os professores que estiveram comigo durante o período em que estive no PROFMAT. Em especial, à minha orientadora, professora Viviane Maria Beuter, por conduzir minhas ideias, alertar sobre os cuidados a serem tomados e por sua extrema paciência em lidar com as minhas dificuldades.

Também agradeço a professora Elisandra Bar de Figueiredo por suas orientações e conselhos, especialmente em relação às dúvidas sobre continuar no programa do PROFMAT, assim como pelas conversas sobre a profissão.

E não posso de deixar de agradecer à minha mãe por me apoiar no que era possível, por suas orações e por sua paciência diante das minhas limitações. E também a todos os meus amigos que acreditavam e torciam por mim.

“Você acabou de conhecer a Ruby. Ela é uma garota e tanto, não é? Quer saber o segredo dela para não ter medo de experimentar coisas novas? É só lembrar que problemões são apenas probleminhas acumulados. É que, às vezes, o único jeito de aprender algo novo é cometer um monte de erros primeiro.” (Liukas, 2019, p. 70)

RESUMO

Este trabalho aborda o pensamento computacional (PC) na educação básica, buscando compreender as demandas estabelecidas pela Base Nacional Comum Curricular (BNCC). Muitos professores de Matemática enfrentam desafios em ensinar programação, principalmente devido à falta de formação específica na área. Com o objetivo de incentivar a introdução desses conceitos em sala de aula, desenvolveu-se um caderno de atividades desplugadas (em anexo), elaborado para ser acessível mesmo a estes professores. As atividades propostas visam não apenas a introdução de conceitos básicos de programação, mas também o fortalecimento de habilidades fundamentais do PC, conhecido como os quatro pilares do PC: decomposição, reconhecimento de padrões, abstração e algoritmos. Para que esse objetivo fosse alcançado, a pesquisa incluiu uma investigação sobre o PC e seus pilares, assim como uma análise preliminar do que a BNCC estabelece sobre programação e computação. Também é apresentada uma explicação simplificada das principais estruturas que compõem um algoritmo de computador, utilizando a notação de pseudocódigo chamada Portugol, com a intenção de contribuir para a compreensão dos fundamentos da lógica computacional por parte de professores de Matemática que talvez não tenham sido introduzidos às linguagens de programação.

Palavras-chave: Pensamento computacional. Programação. Matemática. Educação básica. Atividades desplugadas.

ABSTRACT

This paper addresses computational thinking (CT) in basic education, seeking to understand the demands established by the National Common Curricular Base (BNCC). Many Mathematics teachers face challenges in teaching programming, mainly due to the lack of specific training in the area. In order to encourage the introduction of these concepts in the classroom, a notebook of unplugged activities (attached) was developed, designed to be accessible even to these teachers. The proposed activities aim not only to introduce basic programming concepts, but also to strengthen fundamental CT skills, known as the four pillars of CT: decomposition, pattern recognition, abstraction and algorithms. To achieve this objective, the research included an investigation into CT and its pillars, as well as a preliminary analysis of what the BNCC establishes about programming and computing. A simplified explanation of the main structures that make up a computer algorithm is also presented, using the pseudocode notation called Portugol, with the intention of contributing to the understanding of the fundamentals of computational logic by Mathematics teachers who may not have been introduced to programming languages.

Keywords: Computational thinking. Programming. Mathematics. Basic education. Unplugged activities.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ALGORITMOS

SUMÁRIO

1 INTRODUÇÃO

Wing (2006) popularizou o termo “Pensamento Computacional” (que será denotado neste trabalho por PC) em um artigo publicado na revista Communications of the ACM. Ela argumenta que o PC deve ser integrado em várias disciplinas, pois essa abordagem facilita a resolução de problemas e a tomada de decisões. O PC vai além da programação, englobando a formulação de ideias, a solução de problemas, o pensamento recursivo, e o uso de abstração e decomposição para enfrentar tarefas. Ela ressalta a importância de incorporar o PC na educação para preparar os indivíduos para os desafios modernos.

Essa visão de preparar os estudantes para um mundo cada vez mais digital está alinhada com a Base Nacional Comum Curricular (BNCC):

> A dinamicidade e a fluidez das relações sociais – seja em nível interpessoal, seja em nível planetário – têm impactos na formação das novas gerações. É preciso garantir aos jovens aprendizagens para atuar em uma sociedade em constante mudança, prepará-los para profissões que ainda não existem, para usar tecnologias que ainda não foram inventadas e para resolver problemas que ainda não conhecemos. Certamente, grande parte das futuras profissões envolverá, direta ou indiretamente, computação e tecnologias digitais. (BNCC, 2018, p. 473)

A BNCC atribui, de maneira preferencial, ao professor de matemática da Educação Básica (EB) a responsabilidade de introduzir os alunos ao PC.

> Os processos matemáticos de resolução de problemas, de investigação, de desenvolvimento de projetos e da modelagem podem ser citados como formas privilegiadas da atividade matemática, motivo pelo qual são, ao mesmo tempo, objeto e estratégia para a aprendizagem ao longo de todo o Ensino Fundamental. Esses processos de aprendizagem são potencialmente ricos para o desenvolvimento de competências fundamentais para o letramento matemático (raciocínio, representação, comunicação e argumentação) e para o desenvolvimento do pensamento computacional. (BNCC, 2018, p. 267)

Para o ensino médio, o documento apresenta habilidades específicas que incluem a introdução de conceitos iniciais de programação e a implementação de algoritmos em linguagem corrente e/ou matemática. Além disso, recomenda-se investigar e registrar um algoritmo que resolva um problema, utilizando fluxogramas sempre que possível.

As referências ao PC na BNCC despertaram um interesse em explorar e entender mais profundamente esse tema.

Escolhi este tema, em parte, devido ao meu interesse por quebra-cabeças como o cubo mágico (tenho por volta de 60 modelos diferentes). A resolução de um cubo mágico envolve o

aprendizado de sequências específicas de movimentos, conhecidas como algoritmos, o que me levou a fazer uma associação direta com o campo da programação.

Meu primeiro contato com a programação aconteceu antes de ingressar na Licenciatura em Matemática, quando iniciei um curso a distância de Licenciatura em Informática, embora não tenha concluído. Uma das disciplinas que mais me interessou foi Algoritmos e Programação, onde fui introduzida às linguagens de programação Portugol e Pascal.

Durante o período em que cursei Licenciatura em Informática (entre 2013 e 2014), circulavam informações de que o MEC planejava integrar licenciados em informática ou computação nas escolas públicas, com a intenção de que atuassem nos laboratórios de informática e também pudessem avaliar os alunos em conhecimentos na área.

No entanto, diversos fatores me levaram a sair desse curso sem concluí-lo. Entre eles, destaco a dificuldade enfrentada por todos os matriculados para conseguir estágio nas escolas básicas, mesmo que o estágio fosse não remunerado. Os diretores das escolas frequentemente recusavam os estagiários por não saberem com qual professor associá-los, mesmo quando a instituição oferecia orientações.

Logo após esse período, observei que em algumas escolas estaduais de SC, no município de Joinville, os laboratórios de informática foram desativados. Nas quatro escolas estaduais onde atuei como estagiária e professora de matemática entre 2017 e 2023, não havia mais laboratório de informática. Essas escolas receberam outras demandas por lei, como a necessidade de salas de Atendimento Educacional Especializado (AEE), e, devido à falta de estrutura e espaço, optaram por substituir os laboratórios de informática pelas salas de AEE. A única tecnologia digital à disposição dos alunos era um conjunto de tablets que os professores podiam utilizar em suas aulas.

Com isso, podemos notar o quanto falta para que o ensino de programação básica ou de informática básica para os estudantes de escolas públicas possa se tornar realidade. Além disso, alguns professores de matemática, talvez até muitos, ainda têm pouca familiaridade com o uso de tecnologias, o que faz com que a proposta da BNCC de introduzir lógica de programação nas aulas de matemática seja vista como um grande desafio por parte deles.

Este trabalho tem como objetivo explorar alguns interesses e benefícios do ensino de programação nas escolas básicas, propondo uma abordagem simples de ensino-aprendizagem das noções básicas de programação. A ideia é que os professores possam se familiarizar com o conteúdo por meio de sua conexão com a matemática e que os alunos percebam como seus conhecimentos matemáticos podem contribuir para o desenvolvimento do PC e vice-versa.

Para alcançar esses objetivos, vamos aprofundar nosso conhecimento sobre o PC e seus pilares, explorando como esses conceitos podem ser aplicados no ensino básico, de acordo com as diretrizes da BNCC. Estudaremos os elementos básicos da programação para desenvolver atividades desplugadas e interdisciplinares que conectem matemática e PC, oferecendo suporte aos professores de matemática na implementação dessas práticas em sala de aula, mesmo em ambientes com limitações tecnológicas.

1.1 METODOLOGIA

A metodologia adotada para este trabalho tem característica de pesquisa qualitativa, pois o interesse era conhecer mais profundamente os significados dos termos que o PC aborda e as motivações para abordá-lo na EB. De acordo com Deslandes, Cruz Neto e Gomes (1994):

> A pesquisa qualitativa responde a questões muito particulares. Ela se preocupa, nas ciências sociais, com um nível de realidade que não pode ser quantificado. Ou seja, ela trabalha com o universo dos significados, motivos, aspirações, crenças, valores e atitudes, o que corresponde a um espaço mais profundo das relações, dos processos e dos fenômenos que não podem ser reduzidos à operacionalização de variáveis. (Deslandes, Cruz Neto e Gomes, 1994, p. 21)

E a pesquisa foi feita através de estudos de diversos materiais já desenvolvidas sobre o tema, o que é uma característica de uma pesquisa bibliográfica:

> A pesquisa bibliográfica, ou de fontes secundárias, abrange toda bibliografia já tornada pública em relação ao tema de estudo, desde publicações avulsas, boletins, jornais, revistas, livros, pesquisas, monografias, teses, material cartográfico etc., até meios de comunicação orais: rádio, gravações em fita magnética e audiovisuais: filmes e televisão. Sua finalidade é colocar o pesquisador em contato direto com tudo o que foi escrito, dito ou filmado sobre determinado assunto, inclusive conferencias seguidas de debates que tenham sido transcritos por alguma forma, quer publicadas, quer gravadas. (Marconi e Lakatos, 2003, p. 183)

A pesquisa envolveu o estudo de trabalhos correlatos ao tema deste trabalho, com o objetivo de analisar como abordavam o conceito de PC e as explicações sobre o que ele representa. Também foi realizada uma análise da BNCC para entender o interesse do documento em relação ao PC e ao ensino de lógica de programação nas escolas de EB, além de investigar como a BNCC conceitua o PC e de que forma o documento sugere que o PC deve ser integrado aos demais conhecimentos já presentes no currículo. Paralelamente, examinamos a estrutura curricular do PROFMAT para identificar se o programa prepara os profissionais nesse campo. Por fim, analisamos alguns Produtos Educacionais relacionados ao PC para entender como as atividades foram elaboradas e identificar possibilidades de inovação. Observou-se que muitos desses materiais utilizam a programação como ferramenta para ensinar conteúdos específicos de matemática, enquanto a proposta do Produto Educacional que será apresentado em anexo segue a abordagem oposta.

No capítulo sobre programação, analisamos alguns livros didáticos de matemática que abordam o ensino de programação e escolhemos o livro do Dante (2020) como referência para definir o que é adequado ensinar aos alunos da EB. O livro de Medina e Fertig (2006) foi utilizado

como base para explicar os comandos e a sintaxe. No entanto, enquanto o livro do Medina se destina a um estudo mais aprofundado de programação, o que não era nosso objetivo, o livro do Dante nos ajudou a estabelecer um limite para evitar que o conteúdo se tornasse excessivamente técnico.

Todos esses estudos buscavam fundamentar quais seriam as reais intenções de abordar o PC nas escolas básicas para auxiliar na elaboração de um caderno de atividades apresentado como produto educacional vinculado a este trabalho.

Para o desenvolvimento deste trabalho, apresentaremos a seguinte estrutura: o primeiro capítulo, já realizado, contendo a introdução e metodologia; o segundo capítulo explora as discussões sobre a definição do PC e apresenta um breve detalhamento dos quatro pilares do PC: Decomposição, Reconhecimento de Padrões, Abstração e Algoritmos. No terceiro capítulo abordamos como o PC está inserido na BNCC na disciplina de Matemática e listamos algumas contribuições do PROFMAT em relação ao desenvolvimento do PC. No quarto capítulo, apresentamos os conceitos básicos de programação, começando com um breve histórico e uma explicação sobre o funcionamento de um computador. Também abordamos fluxogramas, algoritmos e os principais elementos que compõem um programa de computador, detalhando como utilizar cada um desses componentes. Na etapa conclusiva deste trabalho, apresentam-se as considerações finais, nas quais são reunidos os aspectos relacionados à nossa exploração sobre a definição do PC, as propostas da BNCC para integrá-lo na EB e a realidade atual da Educação.

Em anexo, apresentamos o caderno de atividades, desenvolvido como produto educacional, composto por sugestões de atividades práticas destinadas à aplicação em sala de aula. A proposta contém atividades lúdicas utilizando materiais simples para facilitar sua aplicação. Essas atividades visam estimular os alunos a estruturar seus pensamentos, depurar falhas, identificar passos cruciais para chegar em um objetivo e, finalmente, a compreender e escrever estruturas básicas de algoritmos em uma pseudolinguagem de programação.

2 PENSAMENTO COMPUTACIONAL

As tecnologias conectadas estão cada vez mais presentes em nossas vidas e é evidente como elas estão integradas às nossas relações, tornando-nos até dependentes delas. Utilizamos essas tecnologias para nos comunicar, trabalhar, estudar, organizar nossa rotina, entreter-nos e receber informações e notícias sobre o que acontece ao nosso redor e no mundo. Diversas áreas da nossa vida são diretamente influenciadas pelo uso de dispositivos de computação pessoal conectados, como celulares, tablets, computadores etc.

Entendemos tecnologia como qualquer ferramenta ou utensílio criado pelo ser humano para suprir uma necessidade. Por exemplo, o lápis e a borracha são tecnologias que atendem à necessidade de fazer registros escritos e de apagá-los quando necessário. Desde a pré-história, o ser humano tem desenvolvido ferramentas, sendo a primeira delas a pedra lascada. Nesse sentido, conforme ressalta Lisboa (2023), "uma descoberta foi se vinculando a outra para criar uma árvore de tecnologia, cujas ramificações ocorrem até os dias atuais".

Da mesma forma que lápis e borracha são tecnologias, os computadores, enquanto tecnologias digitais, também o são, assim como os próprios softwares e, mais recentemente, a inteligência artificial, que representa uma forma de tecnologia mais avançada, ampliando ainda mais as possibilidades de inovação, visando necessidades humanas.

Com isso, é necessário o surgimento de novas tecnologias para aprimorar cada vez mais nossa vivência, trazendo ferramentas que facilitem nossa vida em diversos aspectos. Para tanto, precisamos de pessoas capacitadas para criar essas ferramentas, que saibam interagir com as máquinas e compreendam seu modo de funcionamento, isto é, que disponham de vivências com o PC.

Porém, o PC vai além do senso comum de que está restrito à tecnologia digital, especificamente à programação ou computadores, como o próprio nome sugere. Ele está relacionado ao simples fato de resolver problemas. Ora, os problemas que uma pessoa pode encontrar na vida não são todos diretamente relacionados ao uso de tecnologia digitais conectadas. Entenda como problema qualquer atividade que precise ser realizada com um determinado objetivo, desde fazer uma simples faxina em casa, até melhorar um procedimento no trabalho para obter melhor desempenho e agilidade, gerando maior satisfação dos clientes.

Neste capítulo, iremos abordar a difusão do Pensamento Computacional, algumas de suas definições, as discussões acerca do termo e os quatro pilares que o compõem.

2.1 DIFUSÃO DO PENSAMENTO COMPUTACIONAL

Em 2006, após Jeanette Wing, diretora de pesquisas computacionais do National Science Foundation, publicar o artigo *Computational Thinking* na revista Communications of the ACM (Wing, 2006), o tema pensamento computacional se popularizou. Desde então, esse conceito vem sendo redefinido e expandido, com muitos trabalhos sendo publicados e gradualmente incorporado ao currículo do ensino básico em muitos países, como veremos ao longo deste

capítulo.

Para Wing, o PC é um conjunto de atitudes e habilidades que combina a resolução de problemas com conceitos fundamentais da computação, tornando-se relevante para todos, não apenas para cientistas da computação. Além disso, deveríamos incluir o PC na habilidade analítica de todas as crianças, assim como a leitura, escrita e aritmética. (Wing, 2006)

Veremos como alguns autores buscam definir o PC e a relação que PC estabelece entre o ser humano e o computador.

Em 2011, Wing apresentou uma definição revisada do PC, que ela, Jan Cuny e Larry Snyder usaram e foi inspirada por uma troca de e-mail com Al Aho:

> Pensamento computacional consiste nos processos mentais envolvidos em formular problemas e suas soluções de modo que estas sejam representadas em uma forma que possa ser eficientemente executada por um agente processador de informações. (Wing, 2011, p. 20, tradução nossa)

De fato, essa definição se reflete com a de Aho (2012), que considera o PC como:

> os processos de pensamento envolvidos na formulação de problemas para que suas soluções possam ser representadas como etapas e algoritmos computacionais. Uma parte importante desse processo é encontrar modelos apropriados de computação com os quais formular o problema e derivar suas soluções. (Aho, 2012, p. 832, tradução nossa)

A ideia de introduzir técnicas da ciência da computação no ensino básico não começou com Wing, embora ela tenha desempenhado um papel importante na disseminação dessa ideia. Papert (1980), por exemplo, já discutia sobre o tema anteriormente:

> O pensamento computacional é uma ideia que é ao mesmo tempo nova e antiga. É nova no sentido de que o tema se tornou um tópico amplamente debatido em 2006, após o artigo (Wing, 2006). No entanto, muitos de seus conceitos centrais já vinham sendo discutidos há várias décadas [...]. Seymour Papert [...] foi um dos pioneiros de uma técnica que ele chamou de “pensamento procedimental” (Papert, 1980). Essa técnica compartilha muitas ideias com o que agora entendemos como PC. Usando o pensamento procedimental, Papert buscava fornecer aos estudantes um método para resolver problemas utilizando computadores como ferramentas. A ideia era que os estudantes aprendessem a criar soluções algorítmicas que um computador pudesse executar; para isso, ele utilizava a linguagem de programação LOGO. (Beecher, 2017, p. 8, tradução nossa)

Papert acreditava que os alunos aprenderiam bem matemática com a linguagem LOGO, pois estariam imersos em um mundo onde pudessem construir, brincar e usar matemática. Se cometessem um erro em seus programas LOGO, isso refletiria um erro no entendimento

matemático. Ao corrigirem o erro (depurarem o programa), estariam corrigindo seu próprio pensamento — depurando sua compreensão. (Guzdial, 2017).

A linguagem LOGO, desenvolvida por Papert e seus colaboradores, consiste em escrever comandos pré-definidos em um programa de computador para que um robô, representado por uma tartaruga, se movimente pela tela, podendo traçar seu caminho com uso de uma caneta ou usando a caneta apenas quando desejar (Bueno, 2023, p. 68; Papert, 1980). O objetivo principal para o desenvolvimento da linguagem LOGO foi compreender os modos de funcionamento da linguagem, partindo do interesse em estudar o desenvolvimento do pensamento da criança com relação a "como" e "o porquê" programar. (Navarro, 2021, p. 47)

Resnick (2013) reflete sobre por que aprender a programar, fazendo uma comparação com o ato de aprender a escrever:

> Vejo a codificação (programação de computador) como uma extensão da escrita. A capacidade de codificar permite que você "escreva" novos tipos de coisas – histórias interativas, jogos, animações e simulações. E, como na escrita tradicional, há razões poderosas para que todos aprendam a codificar. (Resnick, 2013, p. 1, tradução nossa)

Assim como a escrita amplia as possibilidades de aprendizado, a programação também abre portas para criar e explorar novos conceitos e projetos em diferentes áreas. Parafraseado Resnick (2013), "aprenda a programar, programe para aprender.´´

De acordo com Barr e Stephenson (2011), em 2009, a Computer Science Teachers Association (CSTA)[1] e a International Society for Technology in Education (ISTE)[2] iniciaram um projeto para definir e incorporar o PC em todo o currículo do ensino K-12 no EUA (equivalente à EB no Brasil), principalmente nas áreas de ciência, tecnologia, engenharia e matemática (CTEM). Os participantes definiram que:

> O PC é uma abordagem para resolver problemas de uma forma que pode ser implementada com um computador. Os alunos se tornam não apenas usuários de ferramentas, mas construtores de ferramentas. Eles usam um conjunto de conceitos, como abstração, recursão e iteração, para processar e analisar dados e criar artefatos reais e virtuais. PC é uma metodologia de resolução de problemas que pode ser automatizada, transferida e aplicada entre disciplinas. (Barr e Stephenson, 2011, p. 51, tradução nossa)

Em sua tese de doutorado, Brackmann (2017) traz uma definição, após estudar diversos autores em combinação com princípios básicos da computação, da seguinte forma:

---

[1] A Computer Science Teachers Association (CSTA) é uma organização que apoia e promove a educação em ciência da computação para estudantes e professores do ensino básico ao médio. <https://csteachers.org/>

[2] A International Society for Technology in Education (ISTE) é uma organização sem fins lucrativos que se concentra em acelerar a inovação na educação por meio do uso inteligente da tecnologia na educação. <https://iste.org/>

> 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, p. 29)

Linda Liukas, em seu livro infantojuvenil, na seção para os pais, descreve o PC como "a maneira de pensar nos problemas de forma que um computador possam resolvê-los. É praticado por humanos, não computadores. Inclui raciocínio lógico e a capacidade de reconhecer padrões, criar e ler algoritmos, e desconstruir e abstrair problemas." (Liukas, 2019, p. 111).

Alguns autores abordam o PC de forma mais sutil e cautelosa. Nem todos defendem que todas as pessoas devam pensar como cientistas da computação. Por exemplo, Beecher (2017) argumenta que "é necessário ficar claro que ensinar pensamento computacional é diferente de ensinar ciência da computação. O principal objetivo desta última é formar os alunos no estudo e aplicação dos princípios da computação matemática". Ele acrescenta que "o pensamento computacional utiliza um subconjunto relativamente pequeno de conceitos – que, coincidentemente, são importantes para a ciência da computação – e os usa para construir uma abordagem de resolução de problemas amplamente aplicável".

Hemmendinger (2010) publicou um artigo fazendo críticas ao artigo de Wing (2006) e o que ele chama de seguidores do PC. "Suas descrições muitas vezes carecem de exemplos apropriados e, talvez como resultado, é mal compreendido em textos mais casuais." Neste artigo, na visão de Barr e Stephenson (2011), Hemmendinger conclui que o

> objetivo final não deve ser ensinar todos a pensar como cientistas da computação, mas sim ensiná-los a aplicar esses elementos comuns para resolver problemas e descobrir novas questões que podem ser exploradas dentro e entre todas as disciplinas. (Barr e Stephenson, 2011, p. 50, nossa tradução)

Podemos perceber que, além da definição do PC estar em concordância com a maioria dos autores, ao longo dos anos, os componentes que abrangem o PC também têm sido delineados, como veremos em algumas citações a seguir.

Grover e Pea (2013) destacaram que a base dos currículos que buscam apoiar a aprendizagem do PC e avaliar seu desenvolvimento aceita como componentes desse conceito:

- Abstrações e generalizações de padrões (incluindo modelos e simulações)
- Processamento sistemático de informações
- Sistemas de símbolos e representações
- Noções algorítmicas de fluxo de controle
- Decomposição de problemas estruturados (modularização)
- Pensamento iterativo, recursivo e paralelo
- Lógica condicional
- Restrições de eficiência e desempenho
- Depuração e detecção sistemática de erros

(Grover e Pea, 2013, p. 39, tradução nossa)

No guia para professores *Computational Thinking - A guide for teachers* (Csizmadia et al., 2015), são estabelecidos conceitos para o PC, descrito como um processo cognitivo que envolve raciocínio lógico para resolver problemas e compreender melhor sistemas e procedimentos. Esse processo abrange diversas habilidades, como a capacidade de pensar:

- algoritmicamente;
- em termos de decomposição, generalização padrões, abstrações e avaliações;
- em generalizações, identificando e fazendo uso de padrões;
- em abstrações, escolhendo boas representações;
- em termos de avaliação.

Essas capacidades se alinham com as ideias apresentadas por Wing (2006), onde ela comenta que o PC é "usar abstração e decomposição ao atacar uma tarefa grande e complexa ou projetar um sistema complexo e grande." Wing enfatiza que "essas habilidades permitem a separação de interesses e a escolha de representações apropriadas para modelar os aspectos relevantes de um problema, tornando-o mais tratável."

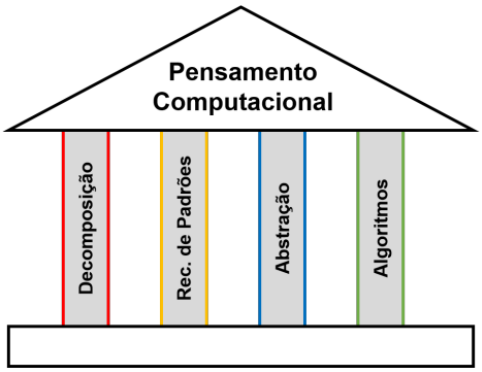
No site da BBC.Learning (BBC.Learning, 2024) encontramos uma seção dedicada ao PC[3], onde é apresentado de forma resumida o conceito de PC e são estabelecidas quatro técnicas principais (pilares) que o sustentam (Figura 1):

- Decomposição;
- Reconhecimento de padrões;
- Abstração;
- Algoritmos.

Ainda menciona que os pilares são como pernas em uma mesa - se uma perna estiver faltando, a mesa provavelmente irá desabar.

[3] https://www.bbc.co.uk/bitesize/guides/zp92mp3/revision/1

Figura 1 – Pilares do Pensamento Computacional



Fonte: Brackmann (2017)

> Pesquisas lideradas por Code.org (2016), Liukas (2019) e BBC.Learning (2015) mesclaram os elementos citados por Grover e Pea (2013) e o guia difundido por Computer at School Csizmadia et al. (2015) e resumiram nos chamados "Quatro Pilares do Pensamento Computacional", sendo eles: Decomposição, Reconhecimento de Padrões, Abstração e Algoritmos." (Brackmann, 2017, p. 32).

De modo bem resumido, a decomposição facilita a resolução de problemas ao dividi-los em partes menores; reconhecimento de padrões envolve identificar semelhanças com problemas já resolvidos; a abstração permite filtrar informações essenciais; e os algoritmos organizam passos para alcançar a solução desejada.

Segundo Dall Agnol, Gusberti e Bertagnolli (2020) esses pilares não precisam ser vistos como etapas do PC e que não é necessário que todos sejam empregados e nem mesmo em uma ordem específica.

Embora tenha sido feita uma analogia entre os quatro pilares do PC e os quatro pés de uma mesa, que são necessários para garantir estabilidade, esses pilares não precisam estar todos presentes nem seguir uma ordem específica para serem utilizados na resolução de um problema com o uso do PC. Com base no desenvolvimento da minha pesquisa e nos exercícios que estudei, criei e analisei, acredito estar em concordância com os últimos autores citados anteriormente. Existem problemas em que o desafio de identificar a presença de um determinado pilar pode ser mais difícil do que a própria resolução do problema. Assim, a presença dos pilares, bem como a ordem em que são aplicados, dependerá da natureza de cada problema.

2.1.1 Decomposição

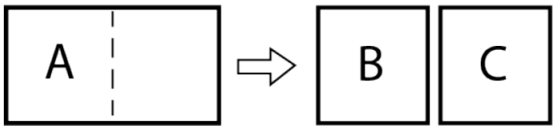
O ato de decompor um problema em partes menores é importante para simplificar sua solução. Encarar um problema como um todo pode ser muito mais desafiador do que resolver um passo mais simples que o compõe. Até mesmo em nosso dia a dia, quando temos uma tarefa a cumprir ou um trabalho a realizar, ouvimos recomendações para fragmentar a tarefa em pequenas partes, o que pode reduzir a ansiedade de realizar algo que inicialmente pareça muito difícil.

O mesmo se aplica a objetos em geral. Brackmann (2017) usa como exemplo uma bicicleta, destacando as partes importantes dela (quadro, rodas, correia, etc.) e explicando que, com essa decomposição, é possível compreender melhor a funcionalidade de cada parte. Isso também permite uma forma mais eficiente de corrigir algum problema na bicicleta, ao identificar qual parte apresenta anomalias. Se não houvesse essa decomposição e sempre considerássemos a bicicleta como uma unidade, a cada problema que surgisse, teríamos que substituir a bicicleta inteira, em vez de apenas corrigir a parte defeituosa.

Para compreender melhor o processo de decomposição, Rich, Egan e Ellsworth (2019) analisaram evidências desse processo em diversas disciplinas, entre elas as áreas CTEM, que têm uma conexão inerente com o PC. A partir das análises feitas, identificaram duas formas gerais de decomposição que emergem dos exemplos observados:

- *Decomposição substantiva*: Este tipo de decomposição envolve a divisão de um problema ou artefato em seus componentes ou características essenciais, que são atributos fundamentais do objeto ou ideia em questão.

Figura 2 – Processo de decomposição substantiva



Fonte: Rich, Egan e Ellsworth (2019)

A Figura 2 ilustra um exemplo em que um componente A é dividido em dois subcomponentes, B e C. Esses subcomponentes são então analisados separadamente, revelando informações adicionais que não eram evidentes quando os componentes estavam combinados.

Um exemplo matemático de decomposição substantiva ocorre quando se deseja determinar se um número inteiro é divisível por 6. O problema pode ser decomposto em duas partes menores:

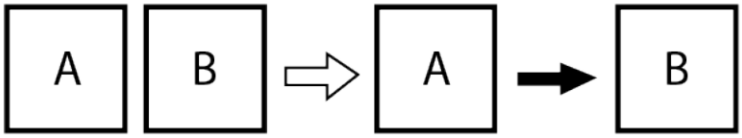
  - Verificar se o número é divisível por 2: Um número é divisível por 2 se o algarismo das unidades for 0, 2, 4, 6 ou 8.
  - Verificar se o número é divisível por 3: Um número é divisível por 3 se a soma dos seus algarismos for divisível por 3.

Se ambas as condições forem satisfeitas, o número é divisível por 6. Esse processo demonstra como a decomposição substantiva facilita a solução ao dividir o problema em subproblemas mais simples.

- *Decomposição relacional*: Este tipo de decomposição envolve a análise das relações entre componentes ou subcomponentes de um problema. A Figura 3 demonstra um caso de

decomposição relacional, onde os componentes A e B são conectados através de uma relação que não era evidente antes dessa análise.

Figura 3 – Processo de decomposição relacional



Fonte: Rich, Egan e Ellsworth (2019)

Um exemplo de decomposição relacional pode ser vista no cálculo do comprimento da diagonal de um paralelepípedo. Para encontrar a diagonal principal, o problema é decomposto em duas etapas:

- Calcular a diagonal da base: Usando o Teorema de Pitágoras, calculamos a diagonal da base do paralelepípedo, relacionando suas dimensões (comprimento e largura).
- Calcular a diagonal principal: Novamente utilizando o Teorema de Pitágoras, calculamos a diagonal principal do paralelepípedo, relacionando a diagonal da base obtida anteriormente com a altura do paralelepípedo.

Neste exemplo, a decomposição relacional revela como diferentes partes do problema estão interconectadas, facilitando a compreensão e a solução do problema maior.

De modo geral, a decomposição ajuda a simplificar problemas complexos, tornando-os mais fáceis de entender e resolver, sendo uma habilidade para a resolução de problemas em diversas áreas. Da mesma forma, os sistemas mais complexos da computação são desenvolvidos utilizando esse princípio, dividindo-o em partes menores e mais gerenciáveis.

2.1.2 Reconhecimento de padrões

De acordo com Brackmann (2017) “padrões são similaridades ou características que alguns dos problemas compartilham e que podem ser explorados para que sejam solucionados de forma mais eficiente”.

Mesmo que um problema não tenha similaridade com outros, devemos nos exercitar na busca por padrões. Seja encontrando padrões entre diferentes problemas ou dentro de um mesmo problema, após sua decomposição em etapas-chave para resolvê-lo. (BBC.Learning, 2024)

O reconhecimento de padrões permite que a resolução de problemas com características semelhantes seja mais rápida. Quando não identificamos padrões, acabamos tendo que elaborar uma nova abordagem para cada problema. No entanto, ao registrar padrões recorrentes, basta sabermos resolver um deles e, a partir desse modelo, adaptar a solução para problemas futuros, ajustando apenas os detalhes específicos.

Na matemática, um exemplo clássico de reconhecimento de padrões é a aplicação da fórmula resolutiva de uma equação polinomial de segundo grau, conhecida como fórmula de Bhaskara. Temos um modelo para identificar uma equação de segundo grau, que é caracterizada pela forma $ax^2 + bx + c = 0$. Uma vez reconhecida essa forma, podemos aplicar a fórmula resolutiva

$$x = \frac{-b \pm \sqrt{\Delta}}{2a}, \quad \text{sendo } \Delta = b^2 - 4ac$$

para obter suas raízes. Por exemplo, ao identificar a equação $x^2 + 3x + 4 = 0$, reconhecemos os coeficientes $a = 1$, $b = 3$ e $c = 4$. Assim, reconhecemos a equação no padrão estabelecido pelo modelo, o que nos permite resolvê-la aplicando a fórmula resolutiva.

Se tivermos uma equação um pouco diferente, como $-2x^2 + x - 3 = 0$, enfrentaremos outro problema para resolver, mas, ao reconhecer as semelhanças com o modelo (ou padrão), conseguiremos resolvê-lo da mesma forma que a equação anterior.

2.1.3 Abstração

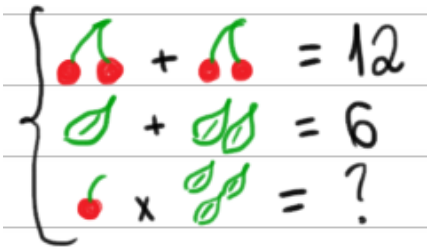
A abstração é o momento de focar nas informações que são importantes e ignorar os detalhes que não são relevantes para a solução do problema. Segundo Csizmadia et al. (2015), a habilidade em abstração está em escolher o detalhe certo para esconder, de modo que o problema se torne mais fácil, sem perder nada que seja importante. Uma parte fundamental disso está em escolher uma boa representação de um sistema.

Como podemos perceber, a abstração no PC é diferente do que frequentemente se entende por abstração, especialmente nos estudos de didática em cursos de Licenciatura em Matemática, onde abstração é muitas vezes vista como sinônimo de algo abstrato, em oposição ao concreto. Um exemplo disso é a introdução das equações algébricas na EB, onde são inseridas incógnitas — as temidas "letras na matemática". Discute-se que o uso de letras é uma abordagem mais abstrata (ou complexa) para resolver equações em comparação com o uso de um desenho que representasse o valor numérico a ser descoberto.

No PC, a abstração significa focar nas informações essenciais e eliminar o que é supérfluo, removendo elementos que sobrecarregam e dificultam a compreensão do problema. Esse processo nos permite lidar com o problema de maneira mais clara e objetiva. A abstração não serve para tornar a solução mais complexa ou rigorosa, mas sim para simplificá-la, proporcionando uma visão mais nítida do problema.

Desse ponto de vista, a abstração pode ser aplicada até mesmo ao exemplo das equações algébricas mencionado anteriormente. É comum encontrar na internet desafios matemáticos apresentados de forma lúdica, com desenhos atraentes e frases sensacionalistas ou desafiadoras "só para gênios" ou "95% das pessoas erram este desafio, será que você consegue?". Na Figura 4, aparece um desafio, que é um sistema de equações:

Figura 4 – Sistema de equações com desenhos



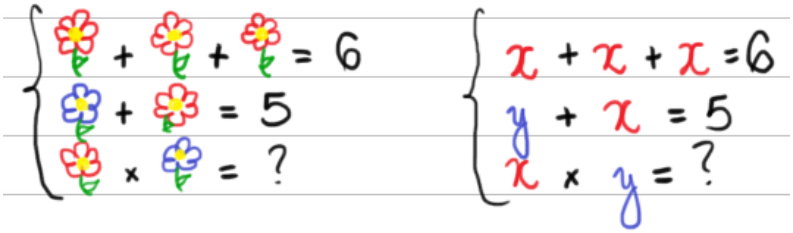
Fonte: Elaborado pela autora

É muito comum, nesses desafios, haver algum detalhe no desenho que, à primeira vista, parece idêntico, mas que, ao ser observado com atenção, revela uma diferença que impacta o resultado final, testando assim a percepção e atenção da pessoa que aceitou o desafio.

Na Figura 4, o desenho foi propositalmente feito à mão, partindo da suposição de que um professor de matemática deseja introduzir o conteúdo com esses desenhos e que precisará reproduzi-los manualmente na lousa. Os alunos poderiam questionar, por exemplo, se as folhas na última linha são as mesmas da segunda linha, dado que estão desenhadas em tamanho reduzido. Será que foram desenhadas menores apenas para caberem na mesma linha, ou porque representam uma espécie diferente de planta? Se forem de outra planta, não podemos assumir que possuam o mesmo valor das folhas da linha anterior.

Vejamos outro exemplo na Figura 5. Supondo que o professor tenha passado dos desenhos para o uso de incógnitas representadas por letras do nosso alfabeto.

Figura 5 – Sistema de equações com desenhos X incógnitas



Fonte: Elaborado pela autora

A abstração aqui permite evitar o excesso de informações que poderia confundir os alunos. Note que, na última linha com os desenhos, o professor acidentalmente desenhou a flor vermelha com cinco pétalas e as demais flores vermelhas com seis pétalas. Ora, o professor não estava preocupado com a quantidade de pétalas, mas sim com a cor das flores. Essa confusão se dissolve ao abstrair o problema para o uso de incógnitas.

Embora o professor possa ter habilidade para tentar fazer os desenhos de forma perfeitamente idêntica, ainda há muitos detalhes que podem não ser percebidos adequadamente. Além disso, o excesso de informações e detalhes nos desenhos pode levar os estudantes a pensar que devem considerar situações específicas nas quais isso não é necessário. Ou seja, a falta de

abstração no sentido do PC pode resultar em problemas de interpretação devido ao excesso de informações.

Rich e Yadav (2020) explicam como as representações na computação podem variar em níveis de abstração:

> Dentro da ciência da computação, as representações normalmente não são discutidas em termos simples de serem abstratas ou não. Em vez disso, as representações existem ao longo de um contínuo de abstrações e são frequentemente descritas como estando em um nível mais alto ou mais baixo de abstração em relação umas às outras (Hillis, 1998). As representações em altos níveis de abstração mostram imagens genéricas de um fenômeno, enquanto as representações em baixos níveis de abstração mostram uma parte menor do fenômeno com mais detalhes. Por exemplo, Hillis (1998) aponta que uma chamada de função dentro de um programa está em um nível mais alto de abstração do que as linhas de código na definição da função. A chamada de funções é uma representação que mostra o que uma função faz. A colocação de chamadas de função dentro do código requer relacionar a função ao problema mais amplo que o programa está abordando, portanto, colocar as chamadas requer que um programador trabalhe em um nível mais alto de abstração. Por outro lado, a informação mais detalhada sobre como a função funciona está na definição da função. Codificar a definição requer pensar em detalhes sobre como a função realizará sua tarefa, mas não requer pensar sobre como o uso da função contribuirá para resolver o problema mais amplo. Assim, os programadores que codificam definições de função, em vez de adicionar chamadas de função, estão trabalhando em níveis mais baixos de abstração. (Rich e Yadav, 2020, p. 396, tradução nossa)

Vamos ver outro exemplo. Suponha que você tenha uma lista finita de números e queira encontrar a mediana dessa lista. Intuitivamente, a *mediana* dessa lista é um valor que a separa em duas partes: Metade dos valores da lista estão acima da mediana, e a outra metade está abaixo. Para encontrar a mediana, aplicamos o seguinte processo:

1. Ordene a lista.
2. Identifique o número total de elementos na lista: Vamos chamar esse número de n.
3. Determine a posição da mediana:
   (a) Se n for ímpar, a mediana é o valor na posição $\frac{n+1}{2}$ da lista ordenada.
   (b) Se n for par, a mediana é a média aritmética dos valores nas posições $\frac{n}{2}$ e $\frac{n}{2}+1$ da lista ordenada.

Neste exemplo, o foco está em encontrar a mediana, enquanto a ordenação (no passo 1.) é tratada como um passo prévio que não é detalhado aqui (é abstraída). Já pensou num algoritmo para ordenar uma lista?

2.1.4 Algoritmos

De acordo com Beecher (2017), um "algoritmo é uma sequência de etapas claramente definidas que descrevem um processo a seguir um conjunto finito de instruções inequívocas, com pontos de início e fim bem definidos." Ele ainda acrescenta que "algoritmos são uma forma de especificar uma tarefa com várias etapas e são especialmente úteis quando desejamos explicar a uma terceira parte (seja humana ou máquina) como realizar passos com extrema precisão."

Se uma pessoa consegue se arrumar para ir ao trabalho, preparar o café da manhã e se vestir, ela certamente é capaz de seguir os passos de um algoritmo simples, pois entende que as tarefas precisam ser realizadas em uma ordem específica. Por exemplo, você nunca apertaria o botão da cafeteira para despejar o café sem antes colocar a xícara no local adequado para receber o líquido. Assim, ao escrever um algoritmo, é necessário identificar a ordem das instruções e garantir que elas estejam claramente descritas, evitando ambiguidades que possam surgir durante sua execução, seja por um computador ou por outra pessoa que leia o algoritmo.

Os algoritmos são usados para a criação de um programa de computador, pois é necessário estruturar um planejamento de como vai ser esse programa antes de começar diretamente nele.

> [...] os humanos já têm uma compreensão intuitiva de algoritmos. No entanto, ao mesmo tempo, uma ciência rica e precisa determina exatamente como os algoritmos funcionam. Adquirir uma compreensão mais profunda disso aprimorará seu raciocínio algorítmico. Isso é importante porque um algoritmo correto é a base fundamental de qualquer solução baseada em computador. (Beecher, 2017, p. 25, tradução nossa)

De acordo com Lopes e Garcia (2002), "atualmente, tem-se associado algoritmo à computação, mas este não é um termo restrito à computação ou que tenha nascido com ela. Na realidade, a palavra algoritmo vem do nome do matemático iraniano Abu Abdullah Mohammad Ibn Musa al-Khawarizmi."

Na matemática, desde os primeiros anos da EB, aprendemos a realizar operações como a soma utilizando procedimentos específicos, sem que tenhamos na época conhecimento do termo "algoritmo" ou do seu significado. Ao longo dos anos, continuamos a aprender e a aplicar outros algoritmos, como os da multiplicação, da divisão euclidiana, entre muitos outros.

Para criar programas de computador, os algoritmos podem ser representados de diferentes maneiras. Uma dessas formas é o fluxograma, um diagrama que ilustra o processo, mostrando as etapas e decisões a serem tomadas. Outra forma é o pseudocódigo, que consiste em escrever um conjunto de instruções em linguagem natural, facilitando a compreensão e a tradução para uma linguagem de programação.

No Capítulo 5, serão apresentados mais detalhes sobre esse pilar e fornecidas instruções básicas para criar algoritmos voltados à resolução de problemas simples.

2.1.5 Reconhecendo a presença dos 4 pilares do PC em um exemplo

Para resumir os quatro pilares do PC, vamos considerar um exemplo em que você precisa decidir se vai ou não a um evento na cidade, aplicando esses pilares para auxiliar na tomada de decisão.

Decomposição: Nesta etapa, você identifica os fatores que podem influenciar sua decisão. Esses fatores podem incluir a distância até o local, as condições climáticas, o orçamento disponível para gastar no evento, a roupa que irá usar, a presença de amigos, as características do local (aberto, fechado, climatizado, com assentos etc.), pendências de trabalho ou estudo, a existência de outros eventos importantes no mesmo mês, seu nível de energia social e a preferência por evitar grandes aglomerações, entre outros.

Reconhecimento de Padrões: Simultaneamente, você pode estar reconhecendo padrões ao refletir sobre experiências passadas, buscando na memória fatores que foram bem-sucedidos ou situações que foram desagradáveis em eventos anteriores (fatores positivos ou negativos). Esse processo ajuda a identificar quais aspectos são recorrentes e importantes em situações semelhantes.

Abstração Diante de tantos detalhes que acabam influenciando sua decisão de ir ou não ao evento, surge a necessidade de abstração. Aqui, você seleciona as informações mais importantes e ignora as demais, ou organiza os fatores em um ranking, priorizando três a cinco aspectos cruciais para sua decisão. Por exemplo, a escolha da roupa pode não ser um fator decisivo; se a roupa que você planejava usar estiver suja, você simplesmente escolhe outra sem desistir do evento. Por outro lado, pode pesar mais para você a necessidade de controlar suas finanças, e, portanto, a questão de quanto você pode gastar no evento pode ser o fator número 1 para decidir se vai ou não ao evento.

Algoritmos: A seguir são apresentamos uma ideia de algoritmo para esse problema.

Algoritmo 2.1 – Decidir se vai ao evento

```
1  Verificar quanto dinheiro pode ser gasto no evento. Se houver, verificar os próxi-
   mos itens, senão, não ir ao evento e não verificar os próximos itens;
   Pesquisar o local do evento:
2      O local é aberto?
3          Se sim então:
4              Ver a previsão do tempo. Vai chover? Se sim, ponto desfavorável, caso
               contrário, ponto favorável;
```

```
        Senão
5           O local é climatizado? Se sim, ponto favorável, caso contrário, ponto
            desfavorável;
6       O local tem lugar para sentar? Se sim, ponto favorável, caso contrário, ponto
        desfavorável;
7   Perguntar ao meu amigo se ele vai ao evento. Se sim ponto favorável, caso contrário,
    ponto desfavorável;
8   Analisar os pontos favoráveis e desfavoráveis. Se houver mais pontos favoráveis, ir
    ao evento; caso contrário, não ir.
```

Fonte: Elaborado pela autora

Essa lista não necessariamente precisa ser escrita, às vezes pode ser simplesmente uma ordem de afazeres que uma pessoa define apenas mentalmente ou um bloco de notas com apenas os pontos principais.

Ao final deste trabalho, apresentamos, em anexo, o produto educacional desenvolvido, que consiste em um caderno de atividades voltado para aplicação em sala de aula, com o objetivo de promover o desenvolvimento das habilidades do PC e introduzir comandos básicos de programação.

No que diz respeito aos pilares do PC, destaca-se a atividade intitulada *Labirintos Lógicos*. Além de trabalhar a compreensão dos operadores lógicos, que serão explicados mais adiante, essa atividade sugere como o professor pode orientar os alunos a seguirem os passos relacionados aos pilares do PC discutidos neste estudo, para resolverem os desafios dos labirintos lógicos. Após discutirmos a difusão do PC, suas definições e os pilares que sustentam este conceito, é essencial examinar como ele tem sido aplicado na EB. No próximo capítulo, ampliaremos essa análise, explorando as pesquisas já realizadas por pesquisadores nacionais e internacionais que discutem o impacto do PC nas escolas e o que a BNCC propõe sobre sua integração. Também refletiremos sobre como o PROFMAT pode contribuir para preparar os educadores para esse desafio.

3 O PENSAMENTO COMPUTACIONAL NA EDUCAÇÃO BÁSICA

Neste capítulo abordaremos o que alguns autores já pesquisaram e concluíram acerca do PC na EB. Em sequência apresentaremos o que a BNCC afirma sobre o PC e o que ela defende sobre a importância de trazer esse conceito para a sala de aula na EB. Finalizamos com uma busca sobre o que o PROFMAT tem a oferecer voltado a esse tema.

3.1 NA EDUCAÇÃO BÁSICA

A habilidade do PC é comparada por Brackmann (2017) como aprender a ler. O fato de aprender a ler não é um fim por si só, as pessoas aprendem a ler e a partir desse momento, usam essa habilidade para aprender outras coisas; de modo análogo deve acontecer com a programação: aprender a programar e passar a programar para aprender.

O autor ainda comenta que apesar de observar que muitos jovens possuem muita familiaridade na interação com novas tecnologias, eles possuem poucas experiências para criar novas tecnologias e se expressarem com elas. Não precisamos esperar que a maioria dos estudantes quando crescerem tornem-se programadores ou profissionais em Computação, mas precisam aprender a pensar de forma criativa e estruturar seus pensamentos, trabalhar de forma colaborativa em qualquer profissão que queiram seguir.

Rodrigues et al. (2015) realizaram uma pesquisa para analisar os efeitos do PC nas habilidades dos estudantes da EB. A pesquisa foi realizada com 103 alunos de cursos voltados à Computação, que forneceram suas pontuações obtidas no ENEM. Os pesquisadores apontam algumas relações entre as competências avaliadas no ENEM e as competências do PC. No formulário utilizado na pesquisa, foi perguntado se os estudantes sabiam programar antes de realizarem a prova do ENEM, dividindo-os em dois grupos: os que já sabiam programar e os que não sabiam. Partiu-se do pressuposto de que aqueles que sabem programar possuem habilidades de PC mais desenvolvidas. Ao comparar o desempenho em todas as áreas avaliadas no ENEM, os pesquisadores concluíram que existe uma correlação moderada, na qual os alunos que já sabiam programar obtiveram um desempenho superior em relação àqueles que não sabiam programar.

Segundo Vasconcelos, Vasconcelos e Costa (2022), dado que a educação tem, entre seus objetivos, preparar o indivíduo para o mercado de trabalho e a vida em sociedade, não faz sentido isolar o processo educacional das mudanças e avanços tecnológicos que ocorrem no mundo. Assim, saber programar e ter conhecimentos em informática podem ser vistos como habilidades básicas que um indivíduo precisa se quiser estar integralmente preparado. Além disso, ensinar o PC pode ajudar os alunos a desenvolverem diversas habilidades importantes para a vida em sociedade, como resolução de problemas, abstração, criatividade, autoexpressão e colaboração.

Raabe, Zorzo e Blikstein (2020) examinaram as diferentes abordagens dos autores sobre a computação na EB, analisando suas manifestações na realidade escolar e seu impacto na definição de currículos e políticas educacionais. Eles identificaram quatro abordagens principais:

- Construcionismo e letramento computacional: Trata a computação como um tema transversal, aplicável em várias áreas do conhecimento. O computador é visto como uma ferramenta de aprendizagem, e a programação é utilizada como um meio para aprender, focando na capacitação dos estudantes para criar inovação e tecnologia em qualquer disciplina, sem necessariamente ensinar os fundamentos de computação como conteúdo principal.

- Emergência do pensamento computacional: Enfatiza o ensino dos conceitos e práticas da ciência da computação como uma área de conhecimento independente, organizada como uma disciplina com conteúdos próprios. Defende a importância de ensinar programação e resolução de problemas computacionais para todos os estudantes.

- Demanda de mercado: Preocupa-se com a inserção dos jovens no mercado de trabalho, promovendo cursos técnicos e formação profissionalizante em programação, com menos foco nas discussões conceituais e mais em atrair os jovens para carreiras tecnológicas.

- Equidade e inclusão: Foca na criação de oportunidades iguais no acesso ao conhecimento de computação, com ênfase em inclusão social, cidadania e participação, especialmente para grupos historicamente excluídos, como meninas e minorias, e implementações em escolas públicas.

Raabe, Zorzo e Blikstein (2020) concluem que as abordagens compartilham algumas semelhanças e, ao examinar os objetivos da computação em cada uma delas, é possível estabelecer a seguinte classificação:

> - **Computação como meio**. Uso dos conhecimentos de computação para a produção de soluções:
>
>   - **Para construir softwares e artefatos enriquecidos por tecnologia:** compreende atividades mais práticas ligadas ao uso de ambientes de programação a fim de produzir sistemas de informação, artefatos enriquecidos por robótica, sistemas embarcados, histórias interativas, materiais educacionais, jogos, etc.
>
>   - **Para solucionar problemas complexos por meio da modelagem matemática:** utiliza-se mais fortemente de fundamentos de computação para modelagem de problemas, decompondo-os, criando algoritmos e estruturas de dados, avaliando as soluções existentes e propondo novas, empregando técnicas de aprendizagem de máquina (estatística) para inferir relações em dados, construindo simulações, etc.
>
> - **Computação como fim**. O estudo das classes de problemas, da eficiência dos algoritmos e dos limites da computação. (Raabe, Zorzo e Blikstein, 2020, p. 12)

3.2 NA MATEMÁTICA: BNCC

Conforme a BNCC (2018), a EB deve abordar diferentes dimensões da computação e das tecnologias digitais, englobando conhecimentos, habilidades, atitudes e valores:

> - 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;
>
> - mundo digital: envolve as aprendizagens relativas às formas de processar, transmitir e distribuir a informação de maneira segura e confiável em diferentes artefatos digitais – tanto físicos (computadores, celulares, tablets etc.) como virtuais (internet, redes sociais e nuvens de dados, entre outros) –, compreendendo a importância contemporânea de codificar, armazenar e proteger a informação;
>
> - cultura digital: envolve aprendizagens voltadas a uma participação mais consciente e democrática por meio das tecnologias digitais, o que supõe a compreensão dos impactos da revolução digital e dos avanços do mundo digital na sociedade contemporânea, a construção de uma atitude crítica, ética e responsável em relação à multiplicidade de ofertas midiáticas e digitais, aos usos possíveis das diferentes tecnologias e aos conteúdos por elas veiculados, e, também, à fluência no uso da tecnologia digital para expressão de soluções e manifestações culturais de forma contextualizada e crítica. (BNCC, 2018, p. 474)

Ainda na BNCC, a relação dos estudantes com a cultura digital é intrínseca, (especialmente os do Ensino Médio) é essencial aprofundar essas aprendizagens, tendo em vista que eles estão ativamente envolvidos na cultura digital, não apenas como consumidores, mas também como protagonistas cada vez mais engajados. De acordo com Moran (2017), “As competências digitais mais importantes hoje, além de programar, são: saber pesquisar, avaliar as múltiplas informações, comunicar-se, fazer sínteses, compartilhar online.”

A BNCC incorporou o PC à matemática, abordado como uma proposta de resolução de problemas:

> Os processos matemáticos de resolução de problemas, de investigação, de desenvolvimento de projetos e da modelagem podem ser citados como formas privilegiadas da atividade matemática, motivo pelo qual são, ao mesmo tempo, objeto e estratégia para a aprendizagem ao longo de todo o Ensino Fundamental. Esses processos de aprendizagem são potencialmente ricos para o desenvolvimento de competências fundamentais para o letramento matemático (raciocínio, representação, comunicação e argumentação) e para o desenvolvimento do pensamento computacional. (BNCC, 2018, p. 267)

Para o ensino médio, a BNCC não especifica explicitamente que o PC deve ser desenvolvido. No entanto, destaca que os alunos devem fortalecer os conhecimentos adquiridos no ensino fundamental.

> 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. Também devem construir uma visão mais integrada da Matemática, da Matemática com outras áreas do conhecimento e da aplicação da Matemática à realidade. (BNCC, 2018, p. 471)

Nota-se perante o trecho abaixo que a BNCC compreende que os conhecimentos matemáticos podem ser uma ferramenta para adquirir as competências do PC.

> Outro aspecto a ser considerado é que a aprendizagem de Álgebra, como também aquelas relacionadas a Números, Geometria e Probabilidade e estatística, podem contribuir para o desenvolvimento do pensamento computacional dos alunos, tendo em vista que eles precisam ser capazes de traduzir uma situação dada em outras linguagens, como transformar situações-problema, apresentadas em língua materna, em fórmulas, tabelas e gráficos e vice-versa. (BNCC, 2018, p. 271)

De modo superficial a BNCC dá uma ideia de como associar os 4 pilares do PC nas aulas de matemática:

> 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. A linguagem algorítmica tem pontos em comum com a linguagem algébrica, sobretudo em relação ao conceito de variável. Outra habilidade relativa à álgebra que mantém estreita relação com o pensamento computacional é a identificação de padrões para se estabelecer generalizações, propriedades e algoritmos. (BNCC, 2018, p. 271)

No currículo do Ensino Médio, destaca-se na área da Matemática e suas Tecnologias as seguintes habilidades que mencionam sobre programação:

- (EM13MAT315) Investigar e registrar, por meio de um fluxograma, quando possível, um algoritmo que resolve um problema. (BNCC, 2018, p. 537)
- (EM13MAT405) Utilizar conceitos iniciais de uma linguagem de programação na implementação de algoritmos escritos em linguagem corrente e/ou matemática. (BNCC, 2018, p. 539)

Embora a BNCC integre o PC à EB, associado à Matemática, não explicita claramente como essas habilidades devem ser desenvolvidas nas aulas. Em contraste, os participantes do projeto de incorporação do PC no currículo K-12 (Barr e Stephenson (2011)) oferecem uma abordagem mais detalhada, que pode ser empregada no EB brasileiro. Esse projeto estabelece as seguintes relações entre os conceitos e as capacidades básicas do PC e sua aplicação na Matemática:

- Coleção de dados: encontrar fontes de dados para um problema como, por exemplo, jogar uma moeda ou rolar um dado.
- Análise de dados: contar as ocorrências do caso da moeda e do dado e analisar os resultados.
- Representação dos dados: usar diversas formas de representação gráficas (histograma, gráfico de setores e gráfico de barras) para representar dados; uso de conjuntos, listas, gráficos, etc. para conter dados.
- Decomposição do Problema: aplicar uma ordem de operações em uma expressão.
- Abstração: usar variáveis em álgebra; identificar fatos essenciais em um problema verbal; estudar funções em álgebra comparadas com funções em programação; usar iteração para resolver problemas de palavras.

- Algoritmos e procedimentos: efetuar cálculos de divisão, fatoração; efetuar adição ou subtração com o procedimento do popularmente conhecido como “conta armada”.
- Automação: utilizar ferramentas como GeoGebra, Scratch e outros aplicativos que permitem uso de códigos criados pelo usuário.
- Paralelização: resolver sistemas lineares e multiplicação de matrizes.
- Simulação: representar graficamente uma função em um plano cartesiano e modificar os valores das variáveis.

Devido às experiências vivenciadas durante a pandemia de Covid-19 a partir de março de 2020, o fechamento de escolas e a necessidade urgente de usar intensivamente tecnologias digitais voltadas para a educação, a sociedade brasileira passou a reconhecer a importância crucial de ter recursos humanos capacitados para o ensino de Computação na EB. Políticas educacionais voltadas para a computação teriam certamente mitigado nossas dificuldades durante a pandemia (Parecer CNE/CEB nº 2/2022, 2022, p. 8).

Esse parecer determina diversas informações sobre o curso de Licenciatura em Computação, onde podemos observar um interesse de ter profissionais que sejam contratados especificamente para lecionar aos alunos de toda a EB para que eles aprendam habilidades como escrever algoritmos, programar, representar dados e usar equipamentos digitais. Porém as habilidades acima mencionadas estão inseridas na área da Matemática e suas Tecnologias, logo espera-se que o professor de matemática também desenvolva nos seus alunos tais habilidades. No momento não é comum encontrar professores de computação ou informática dentro das escolas básicas, ainda mais se formos observar as escolas públicas, mas existem profissionais sendo formados para tal. Apenas não foi dada alguma previsão de quando esses profissionais passarão a ser obrigatórios em todas as escolas.

Particularmente, acredito que é fundamental que haja um profissional exclusivo para o ensino de programação nas escolas. Delegar essa responsabilidade aos professores de matemática ou de outras disciplinas pode ser inadequado, considerando que muitos não possuem a formação necessária para ensinar programação. Incluir programação nas responsabilidades desses docentes resultaria, na maioria dos casos, em um conteúdo negligenciado. A alternativa viável seria capacitar os professores de matemática para abordar a programação de forma interdisciplinar, sem tratá-la como parte exclusiva da matemática. É essencial que o ensino de programação ou computação, incluindo aspectos históricos, conte com o suporte de um profissional formado em Licenciatura em Computação. Assim como um docente sem formação em matemática pode ensinar incorretamente essa disciplina, o mesmo risco ocorre com o ensino de computação por quem não possui a qualificação adequada.

Infelizmente a realidade que se encontra nas escolas nem sempre acompanha os avanços tecnológicos e muitas vezes isso dificulta a possibilidade de utilizar recursos tecnológicos

devido à falta de infraestrutura adequada, por isso as atividades desplugadas acabam sendo uma alternativa interessante para a abordagem (Dall Agnol, Gusberti e Bertagnolli, 2020).

A computação desplugada é uma abordagem que ensina os fundamentos da computação de maneira lúdica, sem o uso de computadores e sem distrações ou detalhes técnicos excessivos. Um de seus objetivos é remover as barreiras técnicas e corrigir equívocos sobre o que realmente é a computação (Vieira, Passos e Barreto, 2013).

É importante destacar que atividades apenas desplugadas não contemplam o conceito básicos da computação, ou seja, não são suficiente para desenvolver o PC dos alunos. Essas atividades, no entanto, podem ser usada para ensinar conceitos preliminares sobre o tema, indiferente a realidade da escola. Obviamente, para atingir o objetivo de aprender a programar é essencial o uso do computar, bem como para programar para aprender.

3.3 NO PROFMAT

Ao analisarmos o programa do PROFMAT, podemos observar que o PC não está inserido nas disciplinas obrigatórias. A grade curricular do PROFMAT é organizada em disciplinas obrigatórias básicas (Matemática Discreta, Aritmética, Geometria) e outras três disciplinas obrigatórias (Resolução de Problemas, Geometria Analítica e Fundamentos do Cálculo). Observamos que a disciplina de Resolução de Problemas trabalha com problemas/questões das quatro disciplinas básicas, mas não está relacionada à metodologia de Resolução de Problemas.

Essa grade está de acordo com o objetivo do programa: "O PROFMAT visa atender prioritariamente professores de Matemática em exercício na Educação Básica, especialmente de escolas públicas, que busquem aprimoramento em sua formação profissional, com ênfase no domínio aprofundado de conteúdo matemático relevante para sua docência."[1]

Obviamente, ser um bom professor de Matemática exige mais do que o domínio do conteúdo; é necessário também adotar boas práticas didáticas e metodologias de ensino diferenciadas, entre outros aspectos. No entanto, isso é pouco abordado nas disciplinas obrigatórias, ficando muitas vezes a depender do perfil do professor que leciona a disciplina.

Além das disciplinas obrigatórias, os alunos devem cursar mais duas disciplinas eletivas, entre 15 oferecidas. Assim como as disciplinas obrigatórias, todas as eletivas apresentam conteúdos de matemática e matemática aplicada. Entre essas, existem duas que apresentam alguns tópicos relacionados à computação.

> **MA 35 - Matemática e Atualidade I:** Posicionando na terra e no espaço. Frisos e mosaicos. Movimentos de robôs. Esqueletos e radiocirurgia com raios gama. Economias e empréstimos. Códigos corretores de erros. Criptografia de chave pública. Geradores de números aleatórios. Google e o algoritmo PageRank.

[1] <https://profmat-sbm.org.br/apresentacao/>

MA 36 - Recursos Computacionais no Ensino de Matemática: O uso da calculadora no ensino de matemática. Planilhas eletrônicas. Ambientes gráficos. Ambientes de geometria dinâmica. Sistemas de computação algébrica. Ensino a distância. Pesquisas eletrônicas, processadores de texto e hipertexto. Critérios para seleção de recursos computacionais no ensino de matemática.

Podemos perceber que, para compreender alguns tópicos dessas ementas, é necessário que os alunos já tenham conhecimentos de programação. No entanto, o desenvolvimento do PC do aluno não é abordado de forma clara, novamente, vai depender muito da didática do professor.

O PC aparece de forma explícita dentro do PROFMAT apenas em alguns trabalhos de conclusão de curso. Conforme descrito no *site* do Profmat, "o trabalho de conclusão final do PROFMAT versa sobre temas específicos pertinentes ao currículo de Matemática da Educação Básica, com impacto na sala de aula." Nesta etapa final do curso, o aluno tem a oportunidade de pesquisar e, consequentemente, aprender tendencias matemáticas, metodologias de ensino, práticas pedagógicas e outros temas que complementam as lacunas deixadas pelas oferecidas no programa.

Ao realizar uma busca no diretório de dissertações[2] do PROFMAT usando o termo "pensamento computacional", encontramos 23 resultados. A seguir, apresentamos uma breve descrição de alguns desses trabalhos. Foram consultados o resumo, o sumário e um trecho da introdução de cada trabalho.

- **Discente e ano:** Juliana Pelissaro Carboni (2023)
- **Instituição:** Universidade Federal de Mato Grosso do Sul - UFMS
- **Título:** O Ensino e aprendizagem do pensamento computacional na Educação Básica
- **Descrição breve:** Explica o PC de acordo com a BNCC e no Ensino Médio. Na fundamentação teórica, aborda os conceitos computacionais e trabalhos relacionados ao PC. Realiza a análise de livros didáticos (três livros). Apresenta uma sequência didática para o ensino do PC no Ensino Médio, utilizando sequências e progressões. A sequência didática apresentada não foi aplicada.

- **Discente e ano:** Marcio Peters (2023)
- **Instituição:** Universidade Federal do Espírito Santo - UFES
- **Título:** Apropriação do pensamento computacional e da robótica educacional para um currículo alinhado às novas tendências em tecnologias educacionais

[2] <https://profmat-sbm.org.br/dissertacoes/>

- **Descrição breve:** Apresenta seu estudo sobre o PC: de acordo com autores; linha do tempo; os motivos para aprender; resolução de problemas e computação desplugada. Defende que o ensino-aprendizagem de computação pode ser mais fácil do que imaginamos. Explana sobre a (não) abordagem das tecnologias digitais na BNCC e no Currículo do Ensino Superior e sobre o letramento digital. Explica sobre fluxogramas, algoritmos e linguagens de programação utilizando diversos recursos e linguagens (Code.org, Scratch, Pictoblox, X-Logo, HTML, Pascal, Linguagem C). Aborda sobre robótica educacional, Arduino e componentes eletrônicos. Apresenta uma proposta didática com oficinas plugadas e desplugadas. Como o autor não apresenta resultados de aplicação, conclui-se que não foi possível aplicá-la antes da publicação do trabalho.

- **Discente e ano:** Kenderson Geane Corrêa (2022)
- **Instituição:** Universidade Federal de São João del-Rei - UFSJ
- **Título:** A inserção do pensamento computacional nas aulas de matemática no ensino básico.
- **Descrição breve:** Usa o PC e sistemas digitais para o Ensino de Matemática. Explica sobre fluxogramas em sua dissertação. Apresenta uma sequência didática contendo sete aulas. Aplicou em uma turma de 6º ano do ensino fundamental focando nos critérios de divisibilidade e usou fluxogramas e apresenta os resultados obtidos.

- **Discente e ano:** Vanessa Henriques Borges (2021)
- **Instituição:** Colégio Pedro II
- **Título:** Combinatória e pensamento computacional: conexões para a Educação Básica no século XXI.
- **Descrição breve:** Apresenta diversos problemas de análise combinatória nível de Ensino Médio comparando formas de resolver apenas pela resolução matemática e depois resolver o mesmo problema criando um algoritmo de programação que resolva o mesmo problema, comparando as resoluções, listando as diferenças, as interseções de ambos os métodos. Não apresenta um produto educacional, sua dissertação está mais voltada para o estudo em si onde notou que o pensamento matemático e o PC estão fortemente interligados e que é difícil separar ambos para poder identificar quando um pensamento termina e o outro começa.

- **Discente:** Wagner Figueiredo da Silveira (2021)
- **Instituição:** Universidade Estadual do Norte Fluminense Darcy Ribeiro – UENF
- **Título:** Pensamento computacional no ensino do cálculo da área de figuras planas na Educação Básica
- **Descrição breve:** Apresentou uma aplicação do uso do PC para o ensino do cálculo de áreas de polígonos; os pilares do PC para a resolução de um problema; as estruturas de um fluxograma. Como resultado da aplicação, verificou uma vantagem do uso dessa metodologia, permitindo que alunos que costumavam não saber como iniciar a resolução, conseguirem ter um ponto de partida. Disponibiliza a sequência didática e os materiais para serem impressos e entregue aos estudantes na aplicação da sequência.

- **Discente:** Juliano Thadeo Alves da Silva (2020)
- **Instituição:** Universidade Federal de Mato Grosso - UFMT
- **Título:** Pensamento computacional no Ensino da Matemática: desafios e possibilidades
- **Descrição breve:** Apresentou propostas de atividades para o ensino de PC durante as aulas de matemática para todas as séries de ensino fundamental, sendo em alguma delas com a utilização de jogos no computador. É mais focado nos algoritmos em representação textual. A dissertação não menciona sobre aplicação e seus possíveis resultados dessas atividades em sala de aula.

No Brasil, diversos trabalhos vêm sendo realizados sobre o tema na área de Ensino de Ciências e Matemática. As autoras Bueno e Santos (2023) apresentam um mapeamento sistemático das pesquisas conduzidas em programas de pós-graduação stricto sensu brasileiros, especificamente na área de Ensino de Ciências e Matemática, que abordam as relações entre PC e ensino de matemática. A pesquisa, realizada em janeiro de 2022, analisou teses e dissertações disponíveis nos sites dos programas, produzidas entre os anos de 2015 e 2021. No total, foram encontradas 4515 dissertações e 1120 teses nos programas selecionados.

4 PROGRAMAÇÃO

Neste capítulo, apresentaremos inicialmente um breve histórico da computação, seguido por uma descrição do funcionamento de um computador. Em sequência, abordaremos as linguagens de programação, algoritmos e fluxogramas. Também exploraremos os elementos essenciais de um programa, como operadores lógicos, aritméticos e relacionais. Por fim, explicaremos o uso de constantes e variáveis em algoritmos, além das principais estruturas condicionais e de repetição.

4.1 COMPUTADORES

O termo computador tem origem no latim *computatore*, que significa aquele que faz cálculos.

> Um computador é um dispositivo capaz de executar cálculos e tomar decisões lógicas em velocidades milhões (até bilhões) de vezes mais rápidas do que os seres humanos. Por exemplo, muitos dos computadores pessoais de hoje podem executar vários bilhões de cálculos em um segundo. Uma pessoa operando uma calculadora de mesa não conseguiria executar tantos cálculos em uma vida. [...] Os computadores processam dados sob o controle de conjuntos de instruções chamados programas de computador. Esses programas guiam o computador por conjuntos ordenados de ações especificadas por pessoas chamadas programadores de computador. (Deitel e Deitel, 2010)

Atualmente, computadores são fundamentais na automação de tarefas complexas, no processamento de grandes volumes de informações e na realização de análises que seriam impossíveis para o humano. O impacto dos computadores na sociedade é grande, abrangendo desde avanços científicos e tecnológicos até mudanças na forma como trabalhamos, nos comunicamos e vivemos.

4.1.1 Breve Histórico

De acordo com Dante (2020), a criação de computadores se iniciou na necessidade da sociedade de criar máquinas de calcular para auxiliar em cálculos mais complexos, conforme o desenvolvimento das civilizações foram avançando. O ábaco era a primeira fonte inspiradora para o desenvolvimento dessas máquinas, que seriam utilizadas principalmente em construções. Foram criadas calculadoras mecânicas, a fim de realizar operações básicas (primeiro apenas com soma e subtração, depois aparecendo a multiplicação, divisão e raiz quadrada) com números naturais e decimais. Inicialmente apenas com somas e subtrações e conforme os criadores iam aprimorando e inspirando-se no que já havia sido criado, possibilitando mais casas decimais e mais operações.

O computador eletrônico tem suas origens nas ideias desenvolvidas pelo cientista, matemático e filósofo Charles Babbage (1791-1871). Babbage procurou criar uma máquina com o objetivo de realizar cálculos.

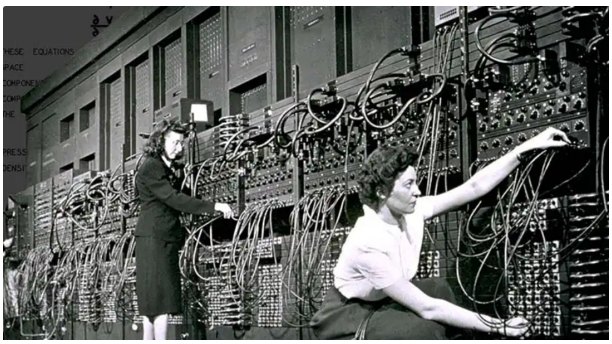
> [...] a máquina era constituída de unidade de controle de memória aritmética e de entrada e saída, e sua operação era governada por um conjunto de cartões perfurados, de modo que, de acordo com os resultados dos cálculos intermediários, a máquina poderia saltar os cartões, modificando, dessa forma o curso dos cálculos. [...] Babbage percebeu que, para criar as instruções responsáveis pela execução dos cálculos, precisaria de um tipo inteiramente novo de linguagem. (Fernandez e Cortes, 2015, p. 16)

Foi a criação de Charles Babbage (1791-1871) que inspirou a matemática e escritora inglesa Ada Augusta Byron (1815-1852), condessa de Lovelace, a criar o primeiro algoritmo de programação.

> Ada Augusta é considerada pioneira na programação de computadores, pois desenvolveu os algoritmos que permitiriam à máquina analítica computar os valores de funções matemáticas. Publicou também uma série de notas sobre a máquina analítica, que criou muitas bases utilizadas atualmente para programar computadores. (Manzano e Oliveira, 2016, p. 16)

Com a sequência das descobertas realizadas com o passar dos anos, os computadores mais semelhantes aos que conhecemos no nosso cotidiano passaram por quatro fases, separadas como gerações de computadores, segundo Fernandez e Cortes (2015). A primeira geração de computadores está compreendida entre os anos 1930 a 1955, os computadores eram gigantes em comparação ao que conhecemos atualmente, eles podiam ocupar cômodos inteiros e foram criados por interesses militares.

Figura 6 – ENIAC: Primeiro computador eletrônico da história.



Fonte: CNN Brasil
Foto: ARL Technical Library / U.S. Army

Deitel e Deitel (2010) explicam que os primeiros computadores só podiam realizar uma tarefa por vez, tal processo é conhecido como processamento em lote de usuário único:

> O computador executa um único programa por vez enquanto processa dados em grupos ou lotes. Nesses primeiros sistemas, os usuários geralmente enviavam seus trabalhos para um centro de computação em baralhos de cartões perfurados e muitas vezes tinham que esperar horas ou até dias antes que as impressões fossem devolvidas às suas mesas. (Deitel e Deitel, 2010)

A segunda geração de computadores ocorre entre 1955 a 1965, marcada pela invenção do transistor que permitiu a redução do tamanho das máquinas e aumento da capacidade de armazenamento.

Na terceira geração de computadores, entre 1965 a 1980, foram desenvolvidos computadores menores e mais velozes na análise de dados. Aqui iniciou a multiprogramação: envolve a operação simultânea de muitas tarefas que estão competindo para compartilhar os recursos do computador.

> Na década de 1960, vários grupos na indústria e nas universidades foram pioneiros em sistemas operacionais de compartilhamento de tempo. O compartilhamento de tempo é um caso especial de multiprogramação em que os usuários acessam o computador por meio de terminais, normalmente dispositivos com teclados e telas. Dezenas ou até centenas de usuários compartilham o computador ao mesmo tempo. O computador, na verdade, não os executa todos simultaneamente. Em vez disso, ele executa uma pequena parte da tarefa de um usuário e, em seguida, passa a atender o próximo usuário, talvez fornecendo serviço a cada usuário várias vezes por segundo.(Deitel e Deitel, 2010)

Por fim, a quarta geração de computadores que inicia em 1980 e permanece nos dias atuais é marcada pela redução significativa do tamanho dos computadores, produção em larga escala e a popularização das máquinas, onde a maioria das pessoas comuns acabam tendo acesso e possuindo um computador em casa (e, atualmente, no bolso: o celular) sendo que antes era um item que pensava-se ser de domínio apenas de militares ou grandes empresas.

Figura 7 – IBM Personal Computer

Fonte: Olhar Digital

> Em 1997, a Apple Computer tornou popular o fenômeno da computação pessoal. Inicialmente, isto era um sonho de quem a tinha como um hobby. Computadores tornaram-se suficientemente baratos para serem comprados para uso pessoal ou comercial. Em 1981, a IBM, a maior vendedora de computadores do mundo, criou o IBM PC (Personal Computer, computador pessoal). Do dia para a noite, literalmente, a computação pessoal se tornou comum no comércio, na indústria e em organizações governamentais. (Deitel e Deitel, 2010)

Segundo Wazlawick (2016), nos anos 1980, duas transformações significativas ocorreram na computação, moldando a sociedade tecnológica atual. A primeira foi a popularização do computador pessoal, impulsionada pelo desenvolvimento de circuitos integrados mais poderosos e acessíveis, tornando possível a criação de computadores capazes de realizar tarefas úteis a um custo acessível para famílias e pequenas empresas. A segunda transformação foi a interconexão desses computadores em rede. Durante essa década, a Internet, que já conectava várias redes ao redor do mundo, começou a se tornar acessível ao público geral, culminando com a criação da World Wide Web no final da década, o que transformou profundamente a dinâmica da sociedade.

4.1.2 Funcionamento de um computador

Um computador consiste em vários dispositivos chamados hardware, como teclado, tela, mouse, memória, discos, SSDs, portas USB, placa de vídeo e unidades de processamento. Os programas que rodam em um computador são chamados de software.

O computador é uma coleção de componentes interligados com o objetivo de efetuar operações aritméticas e lógicas de grandes quantidades de dados:

- Unidade de entrada - É responsável por receber dados no computador e colocá-los à disposição de outras unidades para processamento. Dispositivos como teclados, mouses e scanners são exemplos de unidades de entrada.
- Unidade de saída - É responsável por levar as informações processadas pelo computador e enviá-las aos dispositivos de saída, como monitores de vídeo, impressoras e alto-falantes.
- Unidade de memória (memória principal) - É a seção de armazenamento do computador com acesso rápido e capacidade relativamente limitada, usada para armazenar temporariamente os dados e instruções que o computador está utilizando. É composta principalmente pelas componentes de memórias RAM e ROM.
- Unidade aritmética e lógica (ALU) - Responsável por processar operações matemáticas e/ou lógicas, como somas, subtrações e comparações, sendo fundamental para a execução de tarefas computacionais.

- Unidade central de processamento (CPU) - Coordena o fluxo de informações entre as diferentes unidades do computador, controlando quando a unidade de entrada deve ler dados, quando a ALU deve processá-los, e quando a unidade de saída deve enviar os resultados.

- Unidade de memória secundária - Armazena dados de alta capacidade e longo prazo, permitindo que o computador mantenha informações mesmo após ser desligado. Exemplos incluem discos rígidos e SSDs.

A menor unidade de dado manipulada por um computador é o *bit*, que é a abreviação de *binary digit* (dígito binário). O bit é representado matematicamente pelos valores “1” (um) e “0” (zero). Os computadores processam e realizam operações usando uma linguagem própria, a qual é associada a valores lógicos como verdadeiro ou falso, ou ligado ou desligado, representados pelos dígitos 1 e 0. Uma sequência de bits pode codificar diversos tipos de dados. Por exemplo, bits podem representar números no sistema binário ou caracteres alfabéticos para formar palavras e frases. Esse conceito é expandido para o armazenamento e manipulação de imagens, sons, vídeos e outros tipos de mídia.

4.1.3 Linguagens de Programação

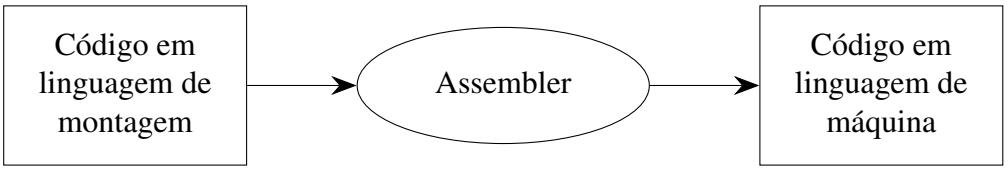
Os computadores dependem de códigos criados por programadores para executar suas tarefas. Para isso, utilizam-se linguagens de programação, que permitem aos computadores entender o que precisam fazer, já que a linguagem deles é diferente da nossa. Assim como aprendemos a linguagem das máquinas, elas possuem comandos que nos retornam valores, resultados ou apresentações compreensíveis.

De acordo com Medina e Fertig (2006), os softwares eram desenvolvidos com linguagens complexas baseadas em códigos binários, que são a base do entendimento das máquinas. Com o tempo, surgiram linguagens de alto nível, que aproximam os códigos da nossa escrita comum. As linguagens de programação podem ser divididas em três tipos gerais:

- Linguagens de máquina: códigos binários (0s e 1s) que o processador entende diretamente, executando operações básicas específicas do hardware.

- Linguagens de montagem (assembly): códigos com uma instrução alfanumérica para cada instrução numérica da linguagem de máquina. Para serem executados, precisam ser traduzidos por um programa chamado *assembler*. A linguagem de montagem é de baixo nível, próxima à linguagem de máquina, e exige conhecimento específico do processador.

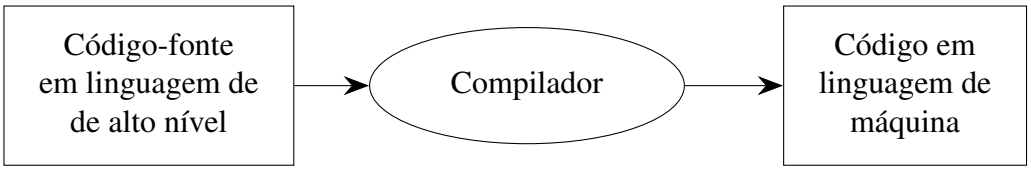
Figura 8 – Tradução para a linguagem de máquina



Fonte: Adaptado de Medina e Fertig (2006)

- Linguagens de alto nível: mais próximas da linguagem natural (como o inglês), usam notações matemáticas comuns e têm pouca similaridade com a linguagem da máquina. Existem duas maneiras de traduzir um programa de linguagem de alto nível. Um compilador converte o código-fonte em um arquivo de linguagem de máquina (código-objeto), que é carregado em memória durante a execução.

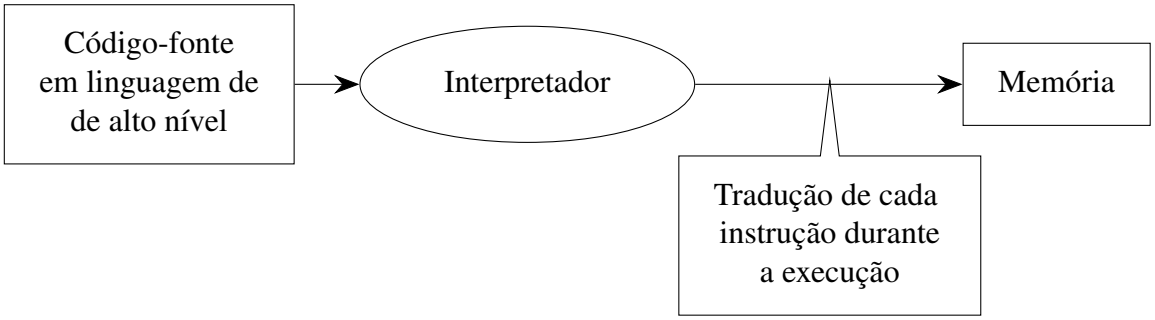
Figura 9 – Compilação de um programa



Fonte: Adaptado de Medina e Fertig (2006)

Já um interpretador traduz as instruções para a linguagem de máquina em tempo real, durante a execução, sem gerar um arquivo de código-objeto.

Figura 10 – Compilação de um programa



Fonte: Adaptado de Medina e Fertig (2006)

4.2 ALGORITMOS

Um algoritmo é uma sequência de etapas claramente definidas que descreve um processo para seguir um conjunto finito de instruções não ambíguas, com pontos de início e fim bem definidos.

Dante (2020) define um algoritmo como uma sequência finita e ordenada de ações para a execução de determinada tarefa ou para resolver um problema.

> Lógica e algoritmos são essenciais para o PC. Eles sustentam o assunto e aparecem repetidamente ao longo de sua aplicação. A boa notícia é: os humanos já têm uma compreensão inata e intuitiva tanto da lógica quanto dos algoritmos. A má notícia é: ambos são conceitos matemáticos por natureza. Consequentemente, cada um tem seu próprio conjunto de regras, procedimentos e definições, que são muito precisos e sistemáticos. Isso significa que você não pode confiar apenas na sua intuição ao lidar com esses tópicos, caso contrário, você cometerá erros. (Beecher, 2017, p. 14, tradução nossa)

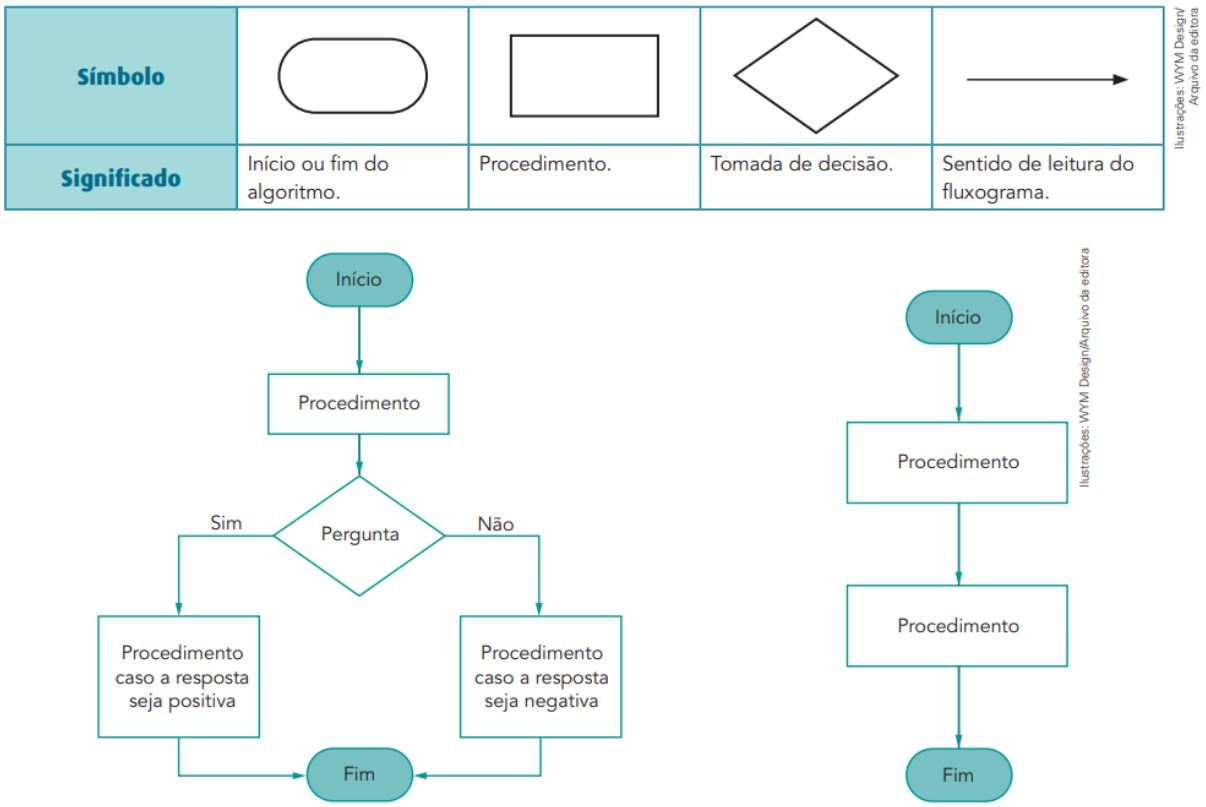
Receitas culinárias são exemplos simples de algoritmos, pois nelas encontramos os itens necessários (ingredientes e utensílios) e um passo a passo com orientações sobre os resultados esperados, por exemplo, o tempo de cozimento, a obtenção de uma massa homogênea, a manteiga derretida, a mistura transparente etc., que devem ser seguidos para avançar na receita. Um algoritmo deve apresentar características fundamentais, como a conclusão em um número finito de etapas, a definição clara e desambiguada das etapas, e a execução de ações simples.

Um algoritmo pode ser descrito basicamente de duas formas: uma forma gráfica a partir da utilização de fluxogramas, e outra forma textual a partir de uma linguagem de projeto de programação ou mesmo de uma linguagem de programação de computadores formal.

Fluxogramas são um tipo de representação gráfica para a visualização de algoritmos, regidos pela norma ISO 5807. Além de serem amplamente utilizados na computação, também são comuns em cursos de administração, onde são estudados para a criação de procedimentos a serem executados na resolução de problemas empresariais ou para otimização do trabalho.

A Figura 11 mostra os símbolos mais conhecidos para fluxogramas e uma ilustração de como são utilizados.

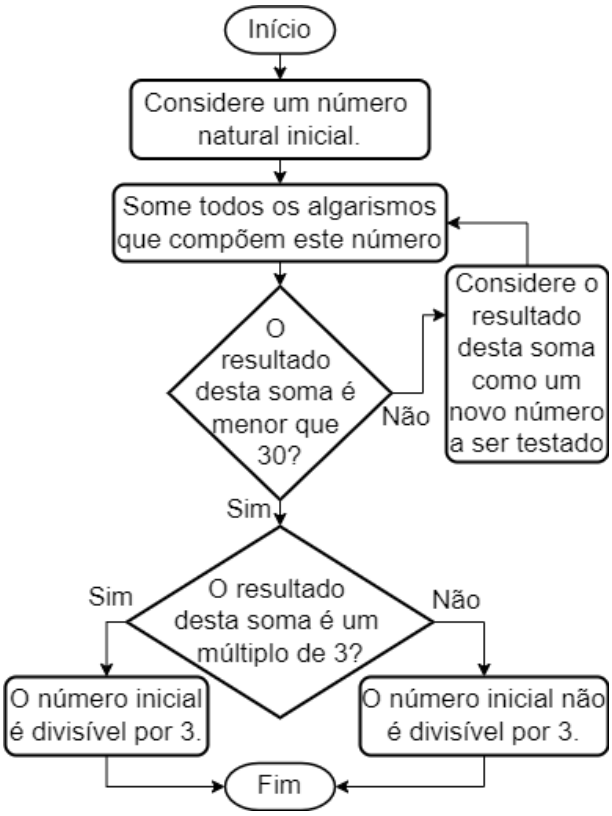
Figura 11 – Símbolos para fluxogramas



Fonte: Dante (2020)

Existem algoritmos usados na Matemática, dentre eles temos os critérios de divisibilidade. Na Figura 12, seguimos uma sequência de passos para definir se um número inteiro é divisível ou não por 3, assumindo que se sabe a tabuada de 3 (ou seja, os múltiplos de 3 entre 0 a 30).

Figura 12 – Fluxograma divisibilidade por 3



Fonte: Elaborado pela autora

Um algoritmo escrito em português (ou em qualquer outra linguagem natural) de forma estruturada é considerado uma pseudolinguagem de programação. Um exemplo disso é o Portugol, que é muito útil para introduzir as linguagens de programação mais utilizadas no mercado, pois utiliza comandos simples em português para executar algoritmos.

Segundo Manzano e Oliveira (2016, p. 29) o Portugol (ou português estruturado) é uma linguagem de projeto de programação e não uma linguagem de programação em si. É uma ferramenta textual também conhecida como pseudocódigo ou metalinguagem que permite descrever as etapas que um programa de computador deve executar, mas de forma simples e sem o rigor técnico de uma linguagem de programação formal, isto é, sem preocupar-se com pontuações, uso correto de parênteses entre outros detalhes de escrita de código que normalmente influenciam no bom funcionamento de uma linguagem de programação formal.

É importante destacar que diferentes compiladores da linguagem Portugol podem ter pequenas variações na sintaxe. Nos exemplos apresentados, seguiremos o padrão de Medina e Fertig (2006).

4.2.1 Elementos que compõem um programa

Segundo Medina e Fertig (2006) um programa de computador é um tipo de algoritmo, pois é composto por um conjunto de instruções que leva o computador a executar uma tarefa. Os elementos básicos que compõem um programa são:

- operadores lógicos,
- variáveis,
- estruturas condicionais,
- estruturas de repetição.

Um algoritmo pode conter todos eles ou apenas alguns deles. Abaixo será descrito um pouco sobre cada um deles.

4.2.1.1 Operadores lógicos

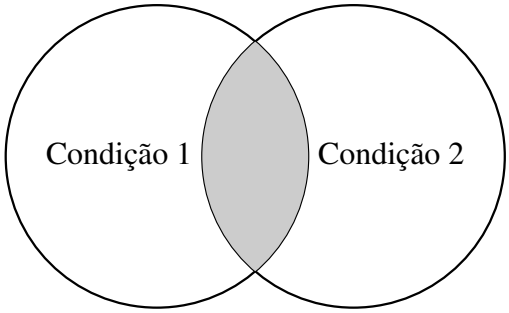
Para que computadores executem operações lógicas, precisamos de um sistema que reflita seu modo de operação. A lógica booleana é um desses métodos, trabalha com declarações que possuem apenas dois valores lógicos: verdadeiro ou falso. Esses valores são representados de outras formas, por praticidade e/ou conveniência, como 1 ou 0, ou ainda, ligado ou desligado.

Na lógica booleana, uma proposição é uma sentença declarativa – algo que será declarado por meio de termos, palavras ou símbolos – e cujo conteúdo poderá ser considerado verdadeiro ou falso. Além disso, elas devem ter significado claro e inequívoco.

É possível combinar proposições individuais para formar outras mais complexas. Isso é útil pois frequentemente precisamos avaliar várias declarações antes de chegar a uma conclusão. Proposições compostas são criadas ao conectar proposições simples usando operadores lógicos. Os operadores lógicos mais utilizados, do ponto de vista da programação, são quatro: operador lógico **e** (conjunção), operador lógico **ou** (disjunção inclusiva), operador lógico **xou** (disjunção exclusiva) e o operador lógico **não** (negação).

Conjunção: A lógica da conjunção (conectivo **e**) é a relação lógica entre duas ou mais proposições que resulta em um valor verdadeiro apenas se todas as proposições forem verdadeiras. Se pelo menos uma delas for falsa, o conectivo produzirá uma proposição com valor falso. Na programação, o termo "proposição" é frequentemente entendido como uma "condição". Observe o Diagrama de Venn para o operador **e** na Figura 13.

Figura 13 – Diagrama de Venn para o operador **e**



Imagine que será realizado um determinado evento na cidade. No entanto, para participar do evento, a organização determinou que é necessário pagar um valor de entrada e doar um quilo de alimento não perecível.

Nesse caso, temos o conectivo **e** relacionado à condição de poder participar do evento, pois a pessoa deve levar tanto o valor da entrada quanto um quilo de alimento. Se ela esquecer de levar qualquer um dos dois itens, não poderá entrar no evento

Observe na Tabela 1 a seguir cada situação possível com o resultado na terceira coluna.

Tabela 1 – Evento na cidade

Dinheiro da entrada	Alimento	Pode entrar?
Levou	Levou	Sim
Levou	Esqueceu	Não
Esqueceu	Levou	Não
Esqueceu	Esqueceu	Não

Aqui temos, essencialmente, uma tabela-verdade do conectivo **e**. A tabela-verdade é uma ferramenta usada no estudo da Lógica Matemática para determinar o valor lógico (verdadeiro ou falso) de uma proposição composta, definindo todas as possíveis combinações de valores lógicos para proposições simples.

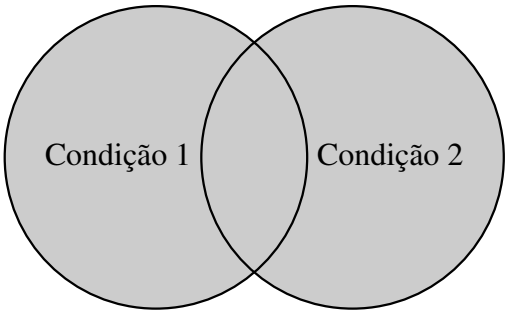
Em um programa de computador, ao trabalhar com duas variáveis que podem assumir apenas os valores lógicos verdadeiro ou falso, a tabela-verdade para essas variáveis com o conectivo **e** é apresentada da seguinte forma (Tabela 2):

Tabela 2 – Tabela-verdade conectivo **e**

var1	var2	var1 **e** var2
V	V	V
V	F	F
F	V	F
F	F	F

Disjunção: A lógica de disjunção inclusiva (conectivo **ou**) é a relação lógica entre duas ou mais proposições de tal modo que seu resultado lógico será verdadeiro quando pelo menos uma das proposições for verdadeira. Observe o Diagrama de Venn para o operador **ou** na Figura 14.

Figura 14 – Diagrama de Venn para o operador **ou**



Para exemplificar este conectivo, imagine uma pessoa se preparando para sair de casa e ficará fora de casa nos períodos da manhã e da tarde. Como ela mora em uma cidade onde costuma chover muito, ela tem o hábito de verificar a previsão do tempo antes de decidir se levará um guarda-chuva. A pergunta que ela faz é: Será que vai chover de manhã ou de tarde? Se chover em qualquer um desses períodos, ela precisará levar um guarda-chuva para não se molhar. As possíveis situações estão ilustradas na Tabela 3.

Tabela 3 – Levar guarda-chuva?

Manhã	Tarde	Levar guarda-chuva?
Chove	Chove	Sim
Chove	Não chove	Sim
Não chove	Chove	Sim
Não chove	Não chove	Não

De modo geral, a tabela-verdade do conectivo **ou** é apresentada na Tabela 4:

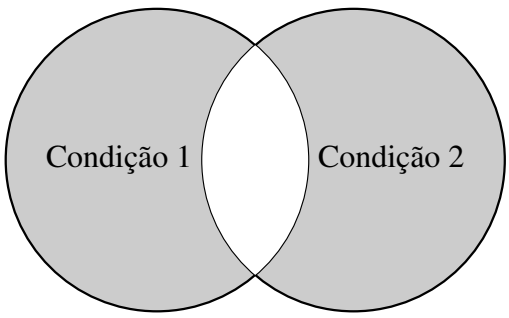
Tabela 4 – Tabela-verdade conectivo **ou**

var1	var2	var1 **ou** var2
V	V	V
V	F	V
F	V	V
F	F	F

Disjunção exclusiva: Às vezes, uma pessoa pode interpretar que o conectivo **ou** resultaria em um valor falso se ambas as condições fossem verdadeiras especialmente em situações de escolha, como na pergunta: Hoje você quer jantar pizza ou hambúrguer?. Naturalmente, não esperamos que alguém responda querer os dois. No entanto, em casos como esse, no campo da lógica de programação, estaríamos lidando com o **ou exclusivo**, que denotamos por **xou**.

Observe o Diagrama de Venn para o operador **xou** na Figura 15:

Figura 15 – Diagrama de Venn para o operador **xou**



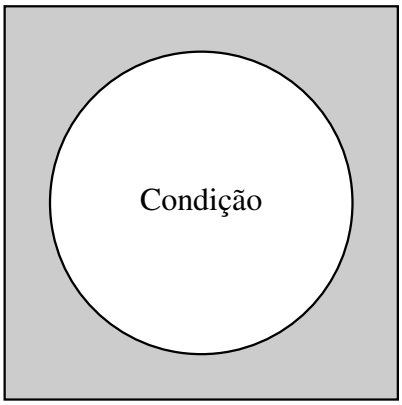
A tabela-verdade do conectivo **xou** pode ser vista na Tabela 5):

Tabela 5 – Tabela-verdade conectivo **xou**

var1	var2	var1 **xou** var2
V	V	F
V	F	V
F	V	V
F	F	F

Negação: O operador **não** apenas inverte o valor lógico da proposição (verdadeiro para falso e vice-versa). Observe o Diagrama de Venn para o operador **não** na Figura 16:

Figura 16 – Diagrama de Venn para o operador **não**



A tabela-verdade do operador **não** pode ser representada com o uso de apenas uma variável, como mostrado da Tabela 6.

Tabela 6 – Tabela-verdade conectivo **não**

var1	**não** var1
V	F
F	V

Esse operador é utilizado quando se deseja que o algoritmo tome um caminho de execução apenas quando uma condição for falsa. Por exemplo, ao tentarmos acender a luz de um cômodo, se, ao pressionarmos o interruptor, ela **não** acender, ela deve ser trocada.

4.2.1.2 Operadores ariméticos e relacionais

Além dos operadores lógicos apresentados previamente, os programas de computador possuem operadores aritméticos (cujos símbolos e funções podem variar de acordo com a linguagem). Apresentamos aqui os símbolos e comandos que utilizaremos (Tabela 7).

Tabela 7 – Operadores Aritméticos

Operador	Função	Exemplos
+	Adição	$2+3=5$
-	Subtração	$8-5=3$
*	Multiplicação	$4*2=8$
/	Divisão	$5/2=2,5$
pot	Exponenciação	3 pot $2=9$
raiz	Radiciação	9 raiz $2=3$
div	Resultado inteiro da divisão	21 div $4=5$
resto	Resto da divisão inteira	21 resto $4=1$

Temos também os operadores relacionais que nos permitem fazer a comparação de uma variável com um número fixo, ou de uma variável com outras variáveis (Tabela 8):

Tabela 8 – Operadores relacionais

Operador	Função	Exemplos
=	Igual a	2=2, x=y
>	Maior que	4>3, x>y
<	Menor que	2<6, x<y
>=	Maior ou igual a	2>=1, x>=y
<=	Menor ou igual a	3<=8, x<=y
<>	Diferente de	5<>4, x<>y

Quando uma sentença contém operadores relacionais, o resultado obtido será sempre um valor lógico (verdadeiro ou falso).

Quanto à precedência dos operadores, os operadores aritméticos seguem a ordem da esquerda para a direita:

1. parênteses mais internos;
2. expoentes e radicais, na ordem em que aparecerem;
3. multiplicações e divisões, na ordem em que aparecerem;
4. adições e subtrações, na ordem em que aparecerem.

Um aspecto a considerar é que, na notação computacional para expressões numéricas, são utilizados apenas parênteses, em vez de empregar também colchetes e chaves.

A precedência entre os operadores lógicos, da esquerda para a direita, segundo Forbellone e Eberspächer (2005, p. 24) e Manzano e Oliveira (2016, p. 86), é:

1. não;

2. e;
3. ou, xou.

A Precedência entre todos os operadores, da esquerda para a direita, de acordo com Forbellone e Eberspächer (2005) é:

1. parênteses mais internos;
2. operadores aritméticos;
3. operadores relacionais;
4. operadores lógicos.

4.2.1.3 Constantes, variáveis, entrada e saída

Parafraseando (Manzano e Oliveira, 2016, p. 43), programas de computador permitem a entrada de dados (em geral numéricos, textuais ou lógicos), que são processados utilizando os operadores descritos anteriormente, e resultam em valores que são devolvidos ao usuário.

Em programas de computador, é possível definir constantes e variáveis conforme necessário, permitindo o armazenamento de informações. Embora o uso de constantes possa variar entre diferentes linguagens de programação, de maneira geral, seu valor permanece inalterado durante a execução do programa, enquanto as variáveis podem ter seu valor modificado ao longo do tempo.

Constantes são úteis para definir valores utilizados frequentemente na execução de um programa. Por exemplo, em um programa que realize operações trigonométricas, pode-se definir uma constante `PI` com uma aproximação apropriada[1] de π para que seja utilizada sempre que necessário. No Algoritmo 4.1 mostraremos um exemplo disso. Por outro lado, variáveis são espaços reservados para armazenar informações que podem ou não ser alteradas.

Tanto as constantes quanto as variáveis devem ser nomeadas de forma única e sem repetições, dadas restrições que variam entre as linguagens de programação. Neste trabalho, permitiremos que esses nomes sejam compostos por letras romanas sem acento (maiúsculas ou minúsculas), números (exceto na primeira posição) e o caractere "_" (sublinhado ou *underline*). Caracteres especiais não são permitidos, nem serão considerados espaços nos nomes de variáveis ou constantes. Também não devem ser definidos nomes que já são reservados aos comandos do programa (escreva, leia, se, enquanto, etc.), que serão apresentados adiante. Vale ressaltar que letras maiúsculas e minúsculas diferenciam os identificadores; isto é, "numero" e "Numero" são considerados distintos.

Os valores constantes são definidos na declaração de constantes e variáveis. Por outro lado, as variáveis não têm seus valores definidos na declaração (antes do *Início*), elas apenas precisam ser especificadas quanto ao tipo de dado que irão receber (inteiro, real, caractere

[1] Nas linguagens de programação comuns com valores em *ponto flutuante*, não é possível armazenar um número irracional, tal como π, de modo exato.

ou booleano). Duas ou mais variáveis que forem receber o mesmo tipo de dado, podem ser declaradas em mesma linha, separadas por vírgula.

Algoritmo 4.1 – Exemplo de uso de constantes e variáveis

```
const
    pi = 3.141592;
var
    raio, geratriz, area: real;
Início
1   escreva("Este programa calcula a área da superfície lateral de um cone");
2   escreva("Entre com o valor do raio do círculo da base");
3   leia(raio);
4   escreva("Entre com o valor da geratriz do cone");
5   leia(geratriz);
6   area ← pi * raio * geratriz;
7   escreva("A área da superfície lateral desse cone é ", area, "cm²");
Fim
```

Fonte: Elaborado pela autora

No exemplo do Algoritmo 4.1 também apresentamos os comandos de entrada, saída e atribuição. São eles:

- **Escreva:** comando de saída, onde o computador irá escrever algo, podendo ser instruções ou resultados para o usuário. Para escrever um texto diretamente do comando, usa-se aspas e para escrever um valor armazenado em alguma variável ou constante apenas se indica o nome (sem aspas).
- **Leia:** comando de entrada que faz com que aquilo que o usuário digitou antes desse comando será armazenado na variável que foi indicada dentro dos parênteses;
- **Atribuição:** simbolizado pela flecha apontando para a esquerda (←) que faz com que a variável indicada à esquerda da flecha receba o valor resultante da expressão escrita à direita. No caso do exemplo mencionado, o programa irá calcular o resultado de

$$\pi \times \text{raio} \times \text{geratriz}$$

e depois atribuir este valor na variável "area".

4.2.1.4 Estruturas condicionais

As estruturas condicionais são os comandos **se...então** e **senão**. Esses comandos permitem que o programa execute uma tarefa somente se uma condição estiver ocorrendo no momento.

Vamos ilustrar esse conceito por meio do algoritmo abaixo. O objetivo deste algoritmo é informar ao usuário se ele deve ou não encher sua garrafa de água, com base no valor que ele insere no programa. Esse valor é determinado pelo próprio usuário, conforme sua percepção sobre o nível de água em sua garrafa.

Algoritmo 4.2 – Garrafa de água: encher ou não?

```
var
    agua: real;
Início
1   escreva("De 0 a 10, quão cheio de água está sua garrafa? Obs: 0 para vazio, 10
    para cheio");
2   leia(agua);
3   se (agua <= 3) então
4       escreva("encha a garrafa");
    fim se
5   escreva("Não esqueça de tomar água!");
Fim
```

Fonte: Elaborado pela autora

Um ponto importante a ser observado é que o argumento que fica entre **se** e **então** deve sempre ser uma proposição (ou premissa) que tenha valor booleano (verdadeiro ou falso). Supondo que o usuário tenha digitado o valor 2 em resposta à pergunta sobre a quantidade de água em sua garrafa; o programa atribuiu esse valor à variável "agua" (na linha 2 do algoritmo). Nesse caso, a proposição "agua <= 3" torna-se $2 \leq 3$ o que é verdadeiro. Assim, tendo um resultado verdadeiro dentro da condicional **se...então**, o programa irá executar as linhas 4 e 5.

Porém, se o usuário inserir um valor maior que 3, por exemplo, o valor 5, a proposição "agua <= 3" passa a ser $5 \leq 3$, o que é falso. Neste caso, o programa não irá executar a linha 4 e passará diretamente da linha 3 (onde verifica o valor booleano da condicional) para a linha 5. Note que essa última linha não pertence ao bloco de instruções da condicional **se...então**; portanto, essa instrução será executada independente do valor inserido pelo usuário.

Caso seja necessário, é possível também usar a forma composta do comando **se...então**, que é usando a cláusula **senão**, a qual permite executar um bloco de instruções caso a condição (definida entre o **se** e **então**) for verdadeira ou outro bloco de instruções apenas se a condição for falsa. Observe no exemplo a seguir (Algoritmo 4.3):

Algoritmo 4.3 – Beba água

```
var
    bebeu: booleano;
Início
1   escreva("Você está bebendo água hoje? Responda V para sim, F para não");
```

```
2       leia(bebeu);
3       se (bebeu) então
4           escreva("Parabéns! Continue assim!");
        senão
5           escreva("Você conhece os malefícios de não beber água? Se não, sugiro que
            pesquise. Beba água!");
        fim se
    Fim
```

Fonte: Elaborado pela autora

Note que aqui o argumento de controle do comando **se...então** foi definido apenas pela variável "bebeu" que aceita apenas valores booleanos e é o suficiente para definir se, a partir da linha 3, o programa irá executar apenas a linha 4 (caso verdadeiro) ou apenas a linha 5 (caso falso).

É possível também usar estruturas condicionais aninhadas, isto é, utilizar um comando **se...então** dentro de outro comando **se...então**. No Algoritmo 4.4 o usuário insere dois valores numéricos. Neste exemplo, usamos duas variáveis x e y para armazenar esses valores e solicitamos que o programa defina a relação de ordem entre esses valores, isto é, se o primeiro número digitado (x) é maior, menor ou igual ao segundo número (y).

Algoritmo 4.4 – Comparar dois números

```
    var
        x, y: inteiro;
    Início
1       escreva("Digite o primeiro número: ");
2       leia(x);
3       escreva("Digite o segundo número: ");
4       leia(y);
5       se (x = y) então
6           escreva("O primeiro número é igual ao segundo número.");
        senão
7           se (x > y) então
8               escreva("O primeiro número é maior do que o segundo número.");
            senão
9               escreva("O primeiro número é menor do que o segundo número.");
            fim se
        fim se
    Fim
```

Fonte: Elaborado pela autora

4.2.1.5 Estruturas de repetição

As estruturas de repetição, também conhecidas como laços ou *loop*, permitem que determinadas ações sejam repetidas com base em uma condição. Essas estruturas são úteis para evitar repetição manual de comandos e são especialmente práticas quando o número de repetições é determinado pelo usuário por meio de uma entrada de dados. Assim, os comandos são escritos apenas uma vez, simplificando o código.

No Algoritmo 4.5 apresentamos um uso do comando **enquanto...faça**. Este comando permite repetir a execução de um bloco de instruções enquanto uma determinada condição for verdadeira. No exemplo, o algoritmo solicita ao usuário que digite uma senha para abrir um documento. A senha está definida como uma constante (embora, em sistemas mais complexos onde a senha possa ser alterada, ela deveria ser definida como uma variável). Se o usuário acertar a senha, o programa envia a mensagem de que está abrindo o documento; caso contrário, o programa informará que a senha está incorreta e solicitará que o usuário digite a senha novamente.

Algoritmo 4.5 – Senha

```
const
    senha = 1234ab;
var
    tentativa : caractere;
Início
1   escreva("Digite a senha para abrir o documento: ");
2   leia( tentativa );
3   enquanto ( tentativa <> senha) faça
    início
4       escreva("Senha incorreta .");
5       escreva("Digite a senha: ");
6       leia( tentativa );
    fim
7   escreva("Abrindo o documento.");
Fim
```

Fonte: Elaborado pela autora

Observe que o comando **enquanto...faça** pode, eventualmente, não ser executado nenhuma vez, como no exemplo acima, onde o usuário poderia acertar a senha na primeira tentativa. O problema desse algoritmo é que, caso o usuário nunca venha a acertar a senha, o programa nunca será encerrado. Vamos aprimorar esse exemplo para que o programa seja capaz de sair do *loop* em algum momento.

Um recurso muito utilizado nas estruturas de repetição são os contadores. Contadores nada mais são que variáveis que irão receber um incremento a cada vez que o bloco de instruções

de uma estrutura de repetição for executado. Eles podem ser usados para determinar quando um comando de repetição deve parar de ser executado.

No Algoritmo 4.6, apresentaremos uma adaptação do algoritmo anterior, utilizando um contador para limitar a quantidade de tentativas de senhas incorretas

Algoritmo 4.6 – Senha com limite

```
const
    senha = 1234ab;
var
     tentativa : caractere;
    num_tent: inteiro;
Início
1       num_tent ← 0
2       escreva("Digite a senha para abrir o documento: ");
3       leia( tentativa );
4       enquanto ( tentativa <> senha e num_tent <= 5) faça
        início
5           escreva("Senha incorreta .");
6           num_tent ← num_tent + 1;
7           escreva("Você tem mais ", 5 – num_tent, " tentativa (s).");
8           escreva("Digite a senha: ");
9           leia( tentativa );
        fim
10      se ( tentativa = senha) então
11          escreva("Abrindo o documento.");
        senão
12          escreva("Você não pode mais abrir o documento.");
        fim se
Fim
```

Fonte: Elaborado pela autora

É importante ressaltar a diferença entre a última instrução do Algoritmo 4.5 (na linha 7) e as últimas instruções do Algoritmo 4.6 (linhas 10, 11 e 12). A implementação do comando **se...então** foi necessária, pois não faria sentido o usuário sair do laço **enquanto...faça** após exceder o número de tentativas de inserção da senha e, em seguida, o programa exibir a mensagem "Abrindo o documento", sendo que o usuário não digitou a senha correta em nenhum momento.

Caso seja necessário, assim como no comando **se...então**, é possível aninhar estruturas de repetição.

Existe também o comando **repita**, o qual permite que o bloco de instruções seja executado pelo menos uma vez e o teste de condição para a saída do laço ocorre apenas no final do bloco.

No Algoritmo 4.7, a situação é somar o total a pagar de uma lista de compras, onde o usuário adiciona valores produto a produto.

Algoritmo 4.7 – Compras

```
   var
        precoprod, total : real;
        adicionar : booleano;
   Início
1       total ← 0;
        repita
        início
2           escreva("Entre com o preço do produto a ser somado: ");
3           leia(precoprod);
4           total ← total + precoprod;
5           escreva("Adicionar mais valores? V para sim, F para não.");
6           leia( adicionar );
        fim
7       até (não adicionar);
8       escreva("O valor total da sua compra é: ", total );
   Fim
```

Fonte: Elaborado pela autora

Observamos que, na linha 7 do algoritmo acima, temos o comando: "**até** (**não** adicionar)". No comando **repita**, o programa de computador executará o bloco de instruções repetidamente, parando apenas quando a condição do comando **até** for verdadeira. Neste caso, se o usuário deseja adicionar mais valores e, portanto, insere o valor "V" na variável *adicionar*, o argumento do comando **até** será "**não** V", o que resulta em "F". Como a condição é falsa, o programa entende que ainda não é o momento de encerrar o comando **repita**. Da mesma forma, se o usuário não tiver mais valores a adicionar, ele inserirá o valor "F" na variável *adicionar*, fazendo com que "**não** F" resulte em verdadeiro. Nesse momento, o programa interromperá a execução da estrutura de repetição.

Voltado para os conhecimentos apresentados acima, nosso produto educacional inclui uma atividade destinada aos alunos, permitindo que eles apliquem os conhecimentos básicos de programação e a escrita de algoritmos. Essas atividades encontram-se no último capítulo do caderno, que contém questões fechadas para que os alunos sigam os algoritmos e respondam ao resultado final que o algoritmo apresenta, com base nos valores de entrada, além de questões que exigem o preenchimento de lacunas no algoritmo.

Os professores que desejam implementar atividades abertas e avaliar os algoritmos elaborados pelos alunos, sugerimos a utilização de um compilador da linguagem Portugol, como,

por exemplo, o site *Portugol Webstudio*, para a transcrição dos algoritmos e a verificação de seu funcionamento. É importante, no entanto, estudar as sintaxes exigidas por esse compilador.

5 CONSIDERAÇÕES FINAIS

Este trabalho focou nos princípios básicos da programação e do Pensamento Computacional, em resposta à demanda da BNCC para a inclusão desses conceitos na EB, particularmente na área do conhecimento de Matemática e suas Tecnologias.

Diante do desafio que muitos professores enfrentam, especialmente aqueles com mais tempo na profissão, em ensinar programação sem possuir formação específica, os estudos deste trabalho tiveram como objetivo o desenvolvimento de um caderno de atividades (em anexo) com conceitos básicos e simplificados. A proposta é que, mesmo com pouca familiaridade com tecnologia ou sem acesso a ela, os professores possam se sentir confortáveis ao introduzir conceitos de programação em suas aulas. As atividades foram elaboradas com base nas pesquisas realizadas para a elaboração dessa dissertação e na minha experiência adquirida ao longo dos últimos três anos como docente de EB na rede pública, refletindo sobre as necessidades e realidades observadas em sala de aula.

Para viabilizar esse objetivo, foram investigados o conceito do PC, seus pilares e o que a BNCC aborda sobre PC, programação e computação, a fim de compreender as intenções de inserir esses conceitos na EB.

O conceito de PC e seus pilares: decomposição, reconhecimento de padrões, abstração e algoritmos, foi um aspecto que chamou atenção durante a pesquisa. Entretanto, foi percebida uma lacuna na literatura sobre a definição e aplicação desses pilares.

As fontes que mencionavam sobre os pilares do PC apresentavam definições muito breves e com poucos ou nenhum exemplo prático. Isso limitava o entendimento e não sanava a curiosidade de aprofundar esses conceitos. Neste trabalho, procurei refletir e descrever com mais detalhes e exemplos matemáticos sobre esses pilares. Acredito que, se houvesse mais estudos e materiais especificamente sobre os pilares, seria possível desenvolver materiais didáticos mais completos para a EB, voltados ao exercício mental necessário para resolver problemas.

Exercitar a decomposição com os alunos, por exemplo, poderia ser benéfico para o raciocínio até mesmo em questões de matemática. Muitas vezes os alunos da EB se deparam com o enunciado de uma questão e, por apenas focarem no problema como um todo, não conseguem identificar um ponto de partida, o que frequentemente os leva a desistir de tentar resolver.

Por outro lado, entendo que o desejo de que existam mais materiais e estudos especificamente sobre os pilares do PC não deveriam idealizar uma tendência de criação de aulas e atividades onde os alunos apenas memorizem os conceitos dos pilares. Tampouco seria produtivo forçar atividades em que os alunos precisem identificar todos os pilares em qualquer situação cotidiana. O foco deve ser o desenvolvimento das habilidades descritas por esses pilares, não apenas o aprendizado teórico sobre eles. Por essa razão, o aprendizado teórico sobre os pilares em si deveria ser reservado preferencialmente aos professores que irão aplicar esses conceitos.

Além disso, é importante destacar nosso conhecimento sobre o interesse do MEC em capacitar profissionais formados em Licenciatura em Computação (ou áreas afins) para atuarem

nas escolas básicas. No entanto, ainda não há previsões ou diretrizes claras sobre como essa implementação ocorrerá. Por enquanto, a BNCC apenas delega o ensino dos conceitos básicos de programação aos professores de matemática. Entretanto, acredito que seria de grande valor que o professor de Matemática utilizasse o conhecimento em programação e do PC como uma abordagem interdisciplinar para conduzir suas aulas.

Uma limitação deste trabalho foi a impossibilidade de aplicar as atividades elaboradas no produto educacional, o que me impede de apresentar dados e resultados dessa aplicação. O programa do PROFMAT inclui um extenso número de disciplinas — 9 ao todo — além do Exame Nacional de Qualificação (ENQ), da elaboração da dissertação e do produto educacional. Devido à estrutura curricular do curso, os alunos começam a considerar o tema da dissertação e do produto educacional, bem como a buscar um orientador, apenas após serem aprovados no ENQ, que ocorre na metade do curso. No entanto, mesmo após essa aprovação, as exigências das disciplinas, somadas à carga de trabalho docente, tornam extremamente desgastante a realização da pesquisa em paralelo ao cumprimento das demais demandas acadêmicas.

Com apenas um semestre disponível para a escrita da dissertação e do produto educacional, o tempo foi insuficiente para, além de realizar a pesquisa, compreender os conceitos envolvidos, investigar as reais intenções do ensino desse tema, definir o que seria apropriado ensinar e o que seria excessivamente avançado, desenvolver as atividades, solicitar e obter a autorização do comitê de ética e, finalmente, aplicar as atividades e analisar os resultados.

No entanto, espero que os estudos apresentados nesta dissertação possam servir como inspiração para futuras pesquisas, para a aplicação das atividades propostas ou, ainda, para a criação de novos materiais relacionados a este tema.

Uma ideia que me marcou pessoalmente durante o estudo, encontrado no livro de Liukas (2019, p. 102), é que ela enfatiza que não tem problema em errar. Embora os códigos frequentemente apresentem falhas quando executados pela primeira vez, é justamente por meio dessas falhas que o programador pode depurar e identificar a raiz do problema. Os erros, nesse contexto, podem se tornar valiosas fontes de aprendizado. Particularmente, como professora de Matemática, sempre me entristeceu observar a maneira como os alunos lidam com seus erros. Com frequência, ao cometerem um erro, eles não buscam compreender a causa, simplesmente apagam tudo e, por esse motivo, podem acabar repetindo o mesmo caminho e cometendo o mesmo erro novamente.

Enfatizo isso ao lembrar que grandes descobertas matemáticas, provavelmente, passaram por inúmeras tentativas e erros. No entanto, até mesmo os erros são analisados para compreender o que precisa ser ajustado. Considero que uma das maneiras mais eficazes de aprender Matemática é não apagar os erros, mas mantê-los registrados, para que se saiba quais caminhos já foram percorridos. Aprender uma linguagem de programação segue a mesma lógica. Muitas vezes, ao escrever um código, ele pode apresentar erros no início e não compilar corretamente; isso, porém, não significa que devemos apagar tudo e recomeçar do zero. Pelo contrário, são necessários pequenos ajustes e complementações até que o código funcione adequadamente.

REFERÊNCIAS

AHO, A. V. Computation and computationalthinking. **The Computer Journal**, v. 55, n. 7, p. 832–835, 2012. Disponível em: <https://academic.oup.com/comjnl/article-abstract/55/7/832/339564>. Citado na página 17.

BARR, V.; STEPHENSON, C. Bringing computational thinking to k-12: What is involved and what is the role of the computer science education community? **ACM Inroads**, v. 2, n. 1, p. 48–54, 2011. Citado 3 vezes nas páginas 18, 19 e 34.

BBC.LEARNING. **Computational thinking**. 2024. Disponível em: <https://www.bbc.co.uk/bitesize/topics/z7tp34j>. Acesso em: 23 jun. 2024. Citado 2 vezes nas páginas 20 e 23.

BEECHER, K. **Computational Thinking: A beginner's guide to problemsolving and programming**. Swindon: BCS Learning Development Ltda, 2017. Citado 4 vezes nas páginas 17, 19, 27 e 46.

BNCC. **Base Nacional Comum Curricular**. Brasília: Ministério da Educação, 2018. Disponível em: <http://basenacionalcomum.mec.gov.br/>. Acesso em: 11 fev. 2023. Citado 4 vezes nas páginas 12, 32, 33 e 34.

BORGES, V. H. **Combinatória e pensamento computacional: conexões para a Educação Básica no século XXI**. 260 p. Dissertação (Dissertação de Mestrado) — Pró-Reitoria de Pós-Graduação, Pesquisa, Extensão e Cultura do Colégio Pedro II, Rio de Janeiro, 2021. Mestrado Profissional em Matemática em Rede Nacional. Citado na página 38.

BRACKMANN, C. P. **Desenvolvimento do Pensamento Computacional através de atividades desplugadas na Educação Básica**. Tese (Doutorado) — Universidade Federal do Rio Grande do Sul, 2017. Citado 6 vezes nas páginas 18, 19, 21, 22, 23 e 30.

BUENO, C. S. **Um panorama das discussões sobre o Pensamento Computacional e sua inserção na Educação Básica**. Tese (Doutorado) — Universidade do Estado de Santa Catarina, Florianópolis, 2023. Disponível em: <https://www.udesc.br/arquivos/faed/id_cpmenu/8673/Carolina_Soares_Bueno_16976411069598_8673.pdf>. Acesso em: ago. 2024. Citado na página 18.

BUENO, C. S.; SANTOS, L. M. dos. Mapeamento sistemático de pesquisas realizadas em programas de pós-graduação stricto sensu brasileiros da área de ensino de ciências e matemática que tratam de relações entre pensamento computacional e ensino de matemática. **Revista Tempos e Espaços em Educação**, v. 16, p. 1–18, 2023. Disponível em: <https://periodicos.ufs.br/revtee/article/view/18287>. Citado na página 39.

CARBONI, J. P. **O Ensino e aprendizagem do pensamento computacional na Educação Básica**. 73 p. Dissertação (Dissertação de Mestrado) — Universidade Federal de Mato Grosso do Sul, Campo Grande, 2023. Mestrado Profissional em Matemática em Rede Nacional. Citado na página 37.

CORRêA, K. G. **A inserção do pensamento computacional nas aulas de matemática no ensino básico**. 132 p. Dissertação (Dissertação de Mestrado) — Universidade Federal de São João del Rei, São João del Rei, 2022. Mestrado Profissional em Matemática em Rede Nacional. Citado na página 38.

CSIZMADIA, A. et al. **Computational thinking – A guide for teachers**. Computing At School (CAS), 2015. Disponível em: <https://www.researchgate.net/publication/327302966_Computational_thinking_-_a_guide_for_teachers>. Acesso em: 31 jul. 2024. Citado 3 vezes nas páginas 20, 21 e 24.

DALL AGNOL, A.; GUSBERTI, C.; BERTAGNOLLI, S. C. O ensino de pensamento computacional através de um jogo de tabuleiro em ambiente desplugado: relato de experiência de formação docente. **Revista Novas Tecnologias na Educação**, Porto Alegre, v. 18, n. 1, 2020. Disponível em: <https://seer.ufrgs.br/index.php/renote/article/view/106036>. Acesso em: 25 jun. 2024. Citado 2 vezes nas páginas 21 e 36.

DANTE, L. R. **Matemática em contextos: análise combinatória, probabilidade e computação**. 1. ed. São Paulo: Editora Ática, 2020. Citado 3 vezes nas páginas 14, 40 e 45.

DEITEL, P. J.; DEITEL, H. M. **Java: Como Programar**. 8. ed. São Paulo: Pearson, 2010. Título original: Java: How to Program. ISBN 978-85-7605-563-1. Tradução: Edson Furmankiewicz. Citado 4 vezes nas páginas 40, 41, 42 e 43.

DESLANDES, S. F.; CRUZ NETO, O.; GOMES, R. **Pesquisa Social: Teoria, Método e Criatividade**. Petrópolis, RJ: Vozes, 1994. Citado na página 14.

FERNANDEZ, M. P.; CORTES, M. I. **Introdução à Computação**. 3. ed. Fortaleza: EdUECE, 2015. Citado na página 41.

FORBELLONE, A. L. V.; EBERSPäCHER, H. F. **Lógica de Programação: A Construção de Algoritmos e Estrutura de Dados**. 3. ed. São Paulo: Pearson Prentice Hall, 2005. Citado 2 vezes nas páginas 53 e 54.

GROVER, S.; PEA, R. Computational thinking in k-12: A review of the state of the field. **Educational Researcher**, v. 42, n. 1, p. 3843, 2013. Disponível em: <https://journals.sagepub.com/doi/10.3102/0013189X12463051>. Citado 3 vezes nas páginas 19, 20 e 21.

GUZDIAL, M. **Learner-Centered Design of Computing Education: Research on Computing for Everyone**. Williston: Morgan & Claypool Publishers, 2017. Citado na página 18.

HEMMENDINGER, D. A plea for modesty. **ACM Inroads**, v. 1, n. 2, p. 4, 2010. Disponível em: <https://dl.acm.org/doi/10.1145/1805724.1805725>. Citado na página 19.

LISBOA, A. **O que é tecnologia?** 2023. Editado por Douglas Ciriaco. Disponível em: <https://canaltech.com.br/internet/o-que-e-tecnologia/>. Acesso em: 23 out. 2024. Citado na página 16.

LIUKAS, L. **Olá, Ruby: Uma aventura pela programação**. São Paulo: Companhia das letrinhas, 2019. Citado 4 vezes nas páginas 5, 19, 21 e 63.

LOPES, A.; GARCIA, G. **Introdução à programação**. Rio de Janeiro: Elsevie, 2002. Citado na página 27.

MANZANO, J. A. N. G.; OLIVEIRA, J. F. d. **Algoritmos: lógica para desenvolvimento de programação de computadores**. 28. ed. São Paulo: Érica, 2016. Citado 4 vezes nas páginas 41, 48, 53 e 54.

MARCONI, M. de A.; LAKATOS, E. M. **Fundamentos de Metodologia Científica**. 5. ed. São Paulo: Atlas, 2003. Citado na página 14.

MEDINA, M.; FERTIG, C. **Algoritmos e Programação: Teoria e Prática**. 2. ed. São Paulo: Novatec, 2006. Citado 4 vezes nas páginas 14, 44, 45 e 48.

MORAN, J. **Tecnologias digitais para uma aprendizagem inovadora**. 2017. Disponível em: <https://moran10.blogspot.com/2017/07/tecnologias-digitais-para-uma.html>. Acesso em: 16 mai. 2024. Citado na página 32.

NAVARRO, E. R. **O desenvolvimento do conceito de Pensamento Computacional na educação matemática segundo contribuições da teoria histórico-cultural**. Tese (Doutorado) — Universidade Federal de São Carlos, São Carlos, 2021. Disponível em: <https://repositorio.ufscar.br/handle/ufscar/15112>. Acesso em: out. 2024. Citado na página 18.

PAPERT, S. **Mindstorms: Children, computers, and powerful ideas.** 1. ed. New York: Basic Books, 1980. Citado 2 vezes nas páginas 17 e 18.

Parecer CNE/CEB nº 2/2022. **Normas sobre Computação na Educação Básica – Complemento à Base Nacional Comum Curricular (BNCC)**. Brasília, DF: [s.n.], 2022. Parecer homologado. Despacho do Ministro, publicado no D.O.U. de 3/10/2022, Seção 1, Pág. 55. Processo nº 23001.001050/2019-18. Citado na página 35.

PETERS, M. **Apropriação do pensamento computacional e da robótica educacional para um currículo alinhado às novas tendências em tecnologias educacionais**. 206 p. Dissertação (Dissertação de Mestrado) — Universidade Federal do Espírito Santo, Vitória, 2023. Mestrado Profissional em Matemática em Rede Nacional. Citado na página 37.

RAABE, A.; ZORZO, A. F.; BLIKSTEIN, P. **Computação na Educação Básica: Fundamentos e Experiências**. Porto Alegre: Penso, 2020. Citado 2 vezes nas páginas 30 e 31.

RESNICK, M. **Learn to Code, Code to Learn**. 2013. Disponível em: <https://www.edsurge.com/news/2013-05-08-learn-to-code-code-to-learn>. Acesso em: 24 out. 2024. Citado na página 18.

RICH, K. M.; YADAV, A. Applying levels of abstraction to mathematics word problems. **395–403**, v. 64, p. 395–403, 2020. Citado na página 26.

RICH, P. J.; EGAN, G.; ELLSWORTH, J. A framework for decomposition in computational thinking. In: **ITiCSE '19: Innovation and Technology in Computer Science Education**. ACM Digital Library, 2019. p. 416–421. ISBN 978-1-4503-6895-7. Disponível em: <https://doi.org/10.1145/3304221.331979>. Citado 2 vezes nas páginas 22 e 23.

RODRIGUES, R. d. S. et al. Análise dos efeitos do pensamento computacional nas habilidades de estudantes no ensino básico: um estudo sob a perspectiva da programação de computadores. **Brazilian Symposium on Computers in Education (Simpósio Brasileiro de Informática na Educação - SBIE)**, Maceió, v. 26, n. 1, p. 121–130, 2015. Disponível em: <http://milanesa.ime.usp.br/rbie/index.php/sbie/article/view/5125>. Acesso em: 24 jun. 2024. Citado na página 30.

SILVA, J. T. A. d. **Pensamento computacional no Ensino da Matemática: desafios e possibilidades**. 78 p. Dissertação (Dissertação de Mestrado) — Universidade Federal do Mato

Grosso, Cuiabá, 2020. Mestrado Profissional em Matemática em Rede Nacional. Citado na página 39.

SILVEIRA, W. F. d. **Pensamento computacional no ensino do cálculo da área de figuras planas na Educação Básica**. 133 p. Dissertação (Dissertação de Mestrado) — Universidade Estadual do Norte Fluminense Darcy Ribeiro, Campos dos Goytacazes, 2021. Mestrado Profissional em Matemática em Rede Nacional. Citado na página 39.

VASCONCELOS, M. C. D. S.; VASCONCELOS, M. V. dos S.; COSTA, M. A. da S. A introdução aos conceitos de lógica de programação com a linguagem de programação scratch. **Anais VIII CONEDU**, Realize Editora, Campina Grande, 2022. Disponível em: <https://editorarealize.com.br/artigo/visualizar/90589>. Acesso em: 2024-06-25. Citado na página 30.

VIEIRA, A.; PASSOS, O.; BARRETO, R. Um relato de experiência do uso da técnica computação desplugada. In: **Anais do XXI Workshop sobre Educação em Computação**. Porto Alegre, RS, Brasil: SBC, 2013. p. 671–680. ISSN 2595-6175. Disponível em: <https://sol.sbc.org.br/index.php/wei/article/view/27763>. Citado na página 36.

WAZLAWICK, R. **História da Computação**. Rio de Janeiro: Grupo GEN, 2016. E-book. ISBN 9788595156180. Disponível em: <https://app.minhabiblioteca.com.br/books/9788595156180/>. Acesso em: ago. 2024. Citado na página 43.

WING, J. M. Computational thinking. **Communications of the ACM**, v. 49, n. 3, p. 33–35, 2006. Disponível em: <https://www.researchgate.net/publication/274309848_Computational_Thinking>. Citado 5 vezes nas páginas 12, 16, 17, 19 e 20.

WING, J. M. Computacional thinking – what and why? **Research notebook: The Link Magazine**, v. 49, n. 3, p. 20–23, 2011. Disponível em: <https://www.cs.cmu.edu/link/research-notebook-computational-thinking-what-and-why>. Citado na página 17.

Apêndice A - Produto Educacional

UNIVERSIDADE DO ESTADO DE SANTA CATARINA – UDESC
CENTRO DE CIÊNCIAS TECNOLÓGICAS – CCT
PROGRAMA DE PÓS-GRADUAÇÃO MESTRADO PROFISSIONAL EM MATEMÁTICA EM REDE NACIONAL

PRODUTO EDUCACIONAL

PENSAMENTO COMPUTACIONAL COM MATEMÁTICA: Atividades desplugadas introduzindo conceitos básicos da programação.

NICOLE CRISTINE RECH

JOINVILLE, SC
2024

Instituição de Ensino: UNIVERSIDADE DO ESTADO DE SANTA CATARINA
Programa: MATEMÁTICA EM REDE NACIONAL
Nível: MESTRADO PROFISSIONAL
Área de Concentração: Matemática na Educação Básica.
Linha de Pesquisa: Matemática na Educação Básica e suas Tecnologias

Título: PENSAMENTO COMPUTACIONAL COM MATEMÁTICA
Autora: Nicole Cristine Rech
Orientadora: Profa. Dra. Viviane Maria Beuter
Data: 29/08/2024

Produto Educacional: Caderno Pedagógico
Nível de ensino: Ensino Fundamental II e Ensino Médio.
Área de Conhecimento: Matemática
Tema: Pensamento Computacional

Descrição do Produto Educacional:
Resultado da pesquisa sobre Pensamento Computacional, este caderno pedagógico apresenta atividades para o professor de Matemática que deseja introduzir conceitos de programação por meio de fluxogramas e pseudocódigo em sala de aula. O Pensamento Computacional torna-se um conceito relevante, especialmente no desenvolvimento de habilidades para a resolução de problemas, que também dialoga com a Matemática. O caderno inclui quatro atividades desplugadas, dentre elas alguns desafios lógicos, permitindo ao professor aprimorar as competências de programação junto a seus alunos.

Biblioteca Universitária UDESC: https://www.udesc.br/bu

Publicação Associada: Pensamento Computacional: uma proposta de como introduzi-lo na Educação Básica relacionando com a Matemática

URL: http://www.udesc.br/cct/profmat

Arquivo	*Descrição	Formato
4.932KG	**Texto completo**	**Adobe PDF**

SUMÁRIO

1. LABIRINTOS LÓGICOS

1.1 Objetivos

Objetivo Geral:

Aprimorar o raciocínio lógico utilizando estruturas condicionais em sentenças lógicas simples e compostas.

Objetivos Específicos:

- Compreender operadores lógicos em contextos;
- Identificar passos importantes para chegar ao objetivo (decomposição e reconhecimento de padrões);
- Criar e construir uma estratégia para resolver o labirinto (decomposição e abstração);
- Desenvolver e aplicar uma notação para abstrair os movimentos possíveis no labirinto, simplificando o processo de identificar o caminho que leva à solução (abstração).

Pré-requisitos

- Esta atividade pode ser aplicada em turmas do ensino médio e também para turmas do 8º e 9º ano.
- Conhecimentos matemáticos prévios: identificar números pares e ímpares; multiplicação; desigualdade (comparar números).

1.2 Apresentação dos Labirintos Lógicos

Para andar pelos labirintos lógicos, você usará dois marcadores, que chamaremos de pinos. Esta atividade não é competitiva, ou seja, ambos os pinos pertencem ao mesmo participante, que os utiliza para avançar pelo labirinto. Apenas um pino pode ser movido por vez, mas não é necessário alterná-los — você pode mover o mesmo pino quantas vezes quiser.

Objetivo

O objetivo é realizar os movimentos corretos para que um dos pinos alcance o GOL. Não é necessário que ambos os pinos cheguem ao GOL; basta que um deles o faça para resolver o labirinto.

Para que isso aconteça, você deve seguir o caminho conforme a instrução escrita na caixa. A maioria das caixas contém uma pergunta, e o caminho que o pino seguirá dependerá da resposta a essa pergunta, referente à posição atual do pino.

Se a resposta for "Sim", mova o pino escolhido pelo caminho rotulado como **Sim**; se for "Não", siga o caminho rotulado como **Não**.

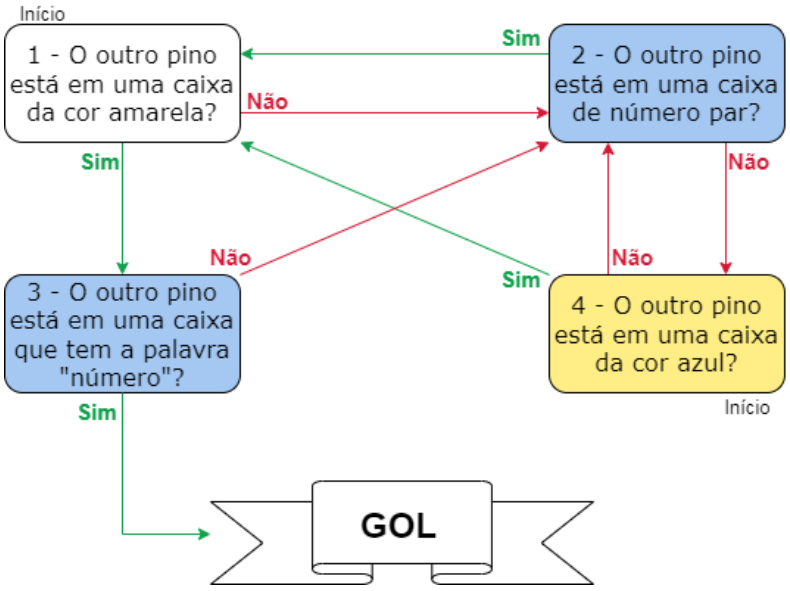
Além disso, a maioria das perguntas nas caixas se refere ao outro pino, ou seja, aquele que você não pretende mover no momento.

Vamos com calma! Este labirinto pode parecer confuso no início, pois é diferente dos convencionais. A regra para fazer um movimento é a seguinte:

- Primeiro, escolha o pino que deseja movimentar. Uma dica é segurá-lo sobre a caixa onde ele está;
- Em seguida, leia a pergunta (ou instrução) que está na caixa deste pino escolhido e siga as instruções.

Vamos aprender usando o labirinto feito como tutorial para esta atividade (Figura 1.1). A posição inicial nesse labirinto (e nos demais também) é com um pino sobre a caixa número 1 e outro pino sobre a caixa número 4.

Figure 1.1: Labirinto - Tutorial



Fonte: Elaborado pela autora

Suponha que você escolheu movimentar o pino que está sobre a **Caixa 4**. A pergunta dessa caixa é: *O outro pino está em uma caixa da cor azul?* Neste caso, o outro pino é o que está sobre a **Caixa**

1, que é branca. Logo, a resposta para a pergunta da Caixa 4 é **Não**. Lembre-se de que o pino que você escolheu movimentar é o que está na Caixa 4, e ele vai seguir pelo caminho **Não**, chegando, assim, à **Caixa 2**.

Agora você tem um pino na Caixa 1 e outro na Caixa 2. Suponha que agora você deseja movimentar o pino que está na **Caixa 1**. A pergunta da caixa é: *O outro pino está em uma caixa da cor amarela?* Agora o outro pino está na Caixa 2, que é azul, então a resposta para a pergunta da caixa 1 é **Não**. Assim, o pino na Caixa 1 vai seguir pelo caminho **Não**, chegando também à **Caixa 2**. Agora ambos os pinos estão sobre a **Caixa 2**.

Pode haver dois pinos na mesma caixa? Sim! E continua sendo possível responder à pergunta, afinal, você primeiro escolhe um pino e só então observa o outro para responder. Escolhendo um dos pinos da **Caixa 2**, a pergunta é: *O outro pino está em uma caixa de número par?* O outro pino também está na Caixa 2, que é par, então a resposta é **Sim**. Um dos pinos seguirá pelo caminho marcado como **Sim**, enquanto o outro permanece. O pino chegará à **Caixa 1**, com o outro pino ainda na **Caixa 2**.

Assim, você seguirá se movimentando no labirinto até que um dos pinos alcance o GOL. Sempre que quiser, você pode retornar à posição inicial (Caixas 1 e 4) para recomeçar, caso sinta que se perdeu no labirinto. Tenho certeza de que, neste labirinto tutorial, você conseguirá encontrar o caminho!

Você já percebeu que, nesta atividade, analisará uma situação e responderá perguntas com respostas apenas de **Sim** ou **Não**. Algumas perguntas terão sentenças simples, como Essa caixa é da cor azul? ou *O outro pino está em uma caixa de número par?*, enquanto outras terão sentenças compostas, como *O outro pino está em uma caixa de número par OU de cor azul?*. Por isso, iniciaremos um breve estudo sobre os operadores lógicos: **e**, **ou** e **não** envolvendo contextos lógicos.

1.3 Operadores lógicos

Os valores **booleanos** são um tipo de dado lógico que pode assumir apenas dois valores: **Verdadeiro** ou **Falso** — frequentemente simplificados para suas iniciais, **V** ou **F**. Nos labirintos lógicos, os valores booleanos são **Sim** ou **Não**, correspondendo, respectivamente, a **V** e **F**.

Operador lógico e

Uma sentença formada com o operador "**e**" resulta em um valor **V** apenas se ambas as condições associadas a ele forem simultaneamente **V**.

Como exemplo, imagine que será realizado um evento na cidade. No entanto, a organização determinou que para participar do evento é necessário pagar uma taxa de entrada "**e**" doar um quilo de alimento não perecível. Observe na tabela a seguir:

Table 1.1: Levou dinheiro **e** alimento?

Levou o dinheiro da entrada?	Levou o alimento?	Pode entrar?
Sim	Sim	Sim
Sim	Não	Não
Não	Sim	Não
Não	Não	Não

Neste caso, temos o conectivo "**e**" relacionado à condição de entrada no evento, pois deve-se levar tanto o valor da entrada quanto um quilo de alimento. Se esquecer de levar qualquer um dos dois itens, não poderá entrar no evento.

Operador lógico ou

Para que uma sentença formada com operador "**ou**" seja **V**, basta que pelo menos uma das condições seja **V**. Se ambas forem **V**, o resultado permanece **V**. O resultado será **F** apenas se ambas as condições forem **F**.

Por exemplo, para entrar na sala de prova do ENEM, é necessário apresentar um documento com foto. No edital, há uma lista de documentos aceitos, entre os quais estão a Carteira de Identidade Nacional (CIN) e a Carteira Nacional de Habilitação (CNH).

Table 1.2: Levou a CIN **ou** a CNH?

Levou a CIN?	Levou a CNH?	Pode entrar?
Sim	Sim	Sim
Sim	Não	Sim
Não	Sim	Sim
Não	Não	Não

Neste caso, temos o conectivo "**ou**", pois basta apresentar um documento para entrar na sala. Não há problema em levar ambos.

Operador lógico não

O operador **não** inverte o valor booleano da condição original. Se era **V**, passa a ser **F**; se era **F**, passa a ser **V**.

Nos labirintos, essa operação é representada pela característica das caixas. Por exemplo: *O outro pino está em uma caixa de cor que **não seja** amarela?*. Neste caso, temos a tabela da seguinte forma:

Table 1.3: A cor da caixa é **não** amarela?

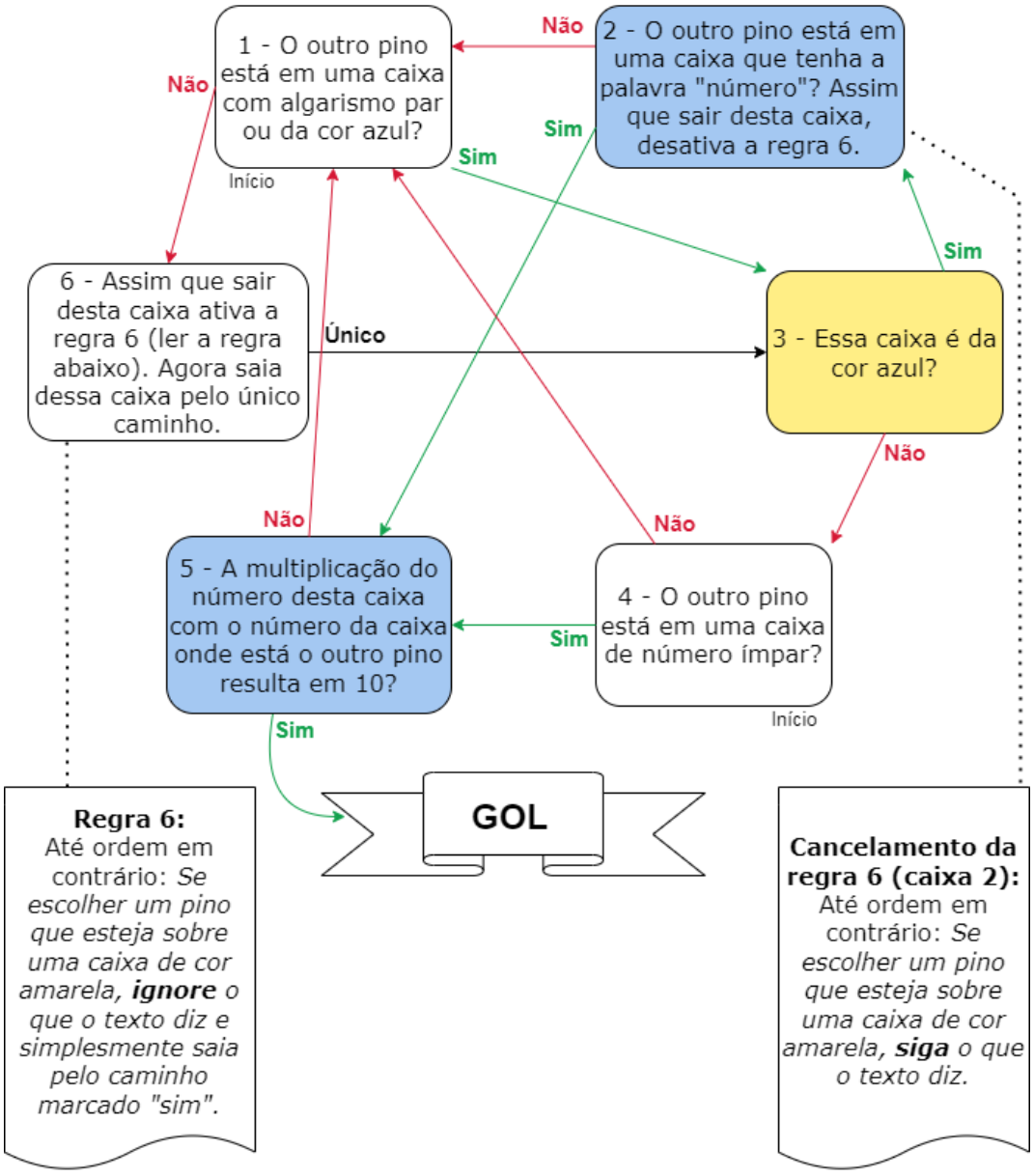
É amarela?	É **não** amarela?
Sim	Não
Não	Sim

1.4 Atividades

Agora, você se aventurará na resolução dos próximos labirintos lógicos. Aborde essa tarefa com atenção e dedicação, faça anotações cuidadosas e analise os resultados obtidos ao longo do caminho para garantir que você chegue ao GOL.

Labirinto 1

Figure 1.2: Labirinto 1



Fonte: Elaborado pela autora

Perguntas sobre o Labirinto 1

Aluno(s): _____

Responda as seguintes questões:

1 - Você usou alguma estratégia? Qual?

2 - Escreveu ou rascunhou alguma coisa para ajudar a organizar o raciocínio? Como foi feito?

3 - Sabe dizer se existe alguma caixa ou posição dos dois pinos que nunca te leva a solução?

4 - Há apenas um caminho (desde o início) que leva a solução?

5 - Sabe dizer se existe alguma posição dos pinos que nunca irá ocorrer neste labirinto?

6 - No caminho que você encontrou, em algum momento os dois pinos ficam na mesma caixa?

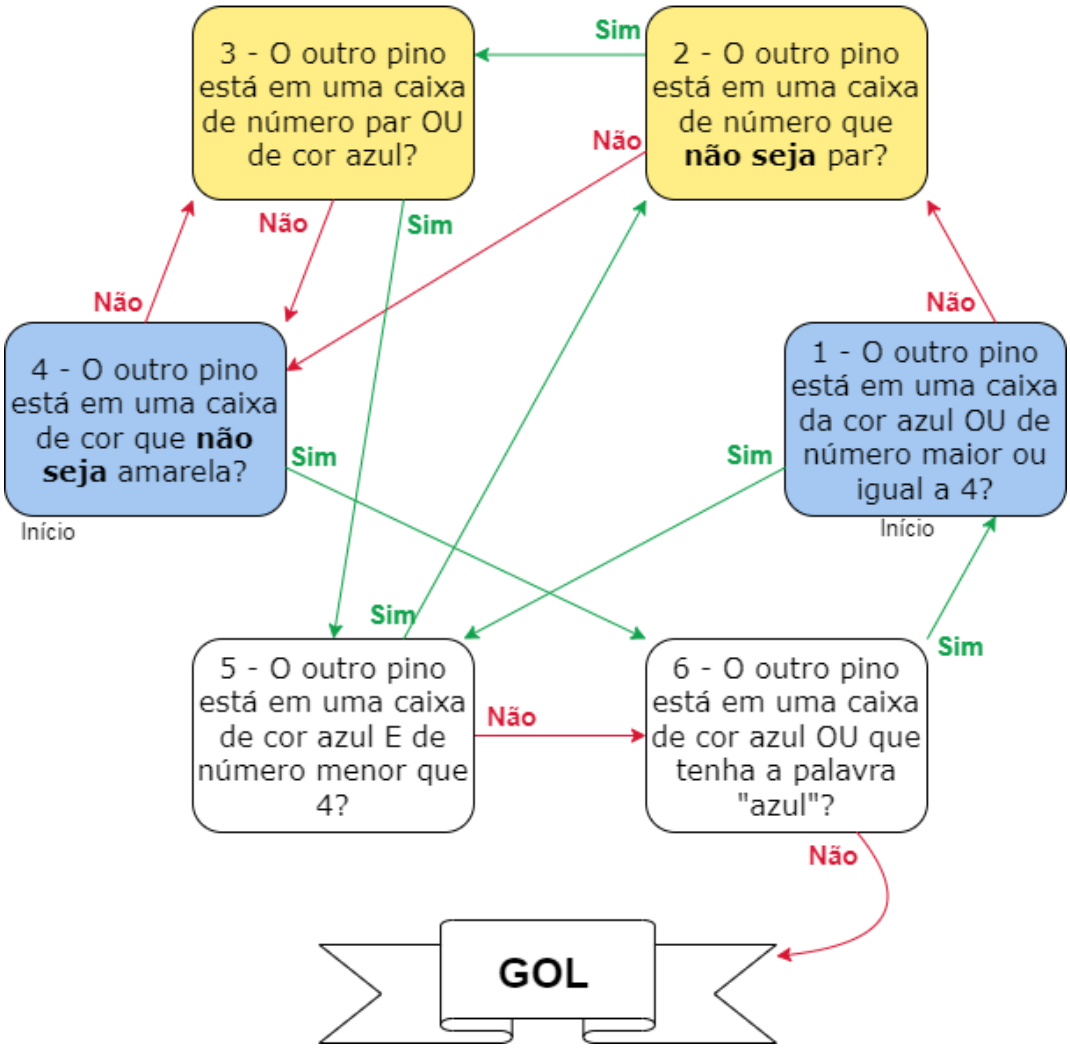
7 - Se a resposta anterior for sim, você consegue encontrar um caminho em que isso nunca ocorra, ou seja, em nenhum momento os dois pinos fiquem juntos na mesma caixa?

8 - Se a resposta à Questão 6 for não, você consegue encontrar um caminho em que, em algum momento, os dois pinos fiquem juntos na mesma caixa?

9 - Você já criou, ou consegue criar, uma notação para que outra pessoa consiga seguir o caminho para chegar à solução do labirinto? Descreva abaixo:

Labirinto 2

Figure 1.3: Labirinto 2



Fonte: Elaborado pela autora

Perguntas sobre o Labirinto 2

Aluno(s): _____

Responda as seguintes questões:

10 - Sabe dizer se existe alguma caixa ou posição dos dois pinos que nunca te leva a solução?

11 - Há apenas um caminho (desde o início) que leva a solução?

12 - Sabe dizer se existe alguma posição dos pinos que nunca irá ocorrer neste labirinto?

13 - No caminho que você encontrou, em algum momento os dois pinos ficam na mesma caixa?

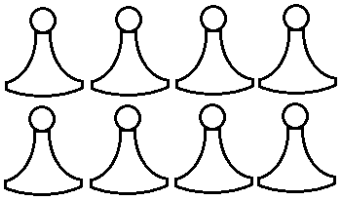
14 - Se a resposta anterior for sim, você consegue encontrar um caminho em que isso nunca ocorra, ou seja, em nenhum momento os dois pinos fiquem juntos na mesma caixa?

15 - Se a resposta à questão 13 for não, você consegue encontrar um caminho em que, em algum momento, os dois pinos fiquem juntos na mesma caixa?

Figure 1.4: Placa indicadora Regra 6

Fonte: Elaborado pela autora

Figure 1.5: Pinos



Fonte: Elaborado pela autora

1.5 Orientações sobre a atividade para o Professor

A proposta que apresentamos, em resumo, é a seguinte:

1. Labirinto Tutorial: para que os alunos aprendam os movimentos básicos.
2. Labirinto 1: um labirinto um pouco mais elaborado, onde os alunos deverão desenvolver suas próprias estratégias para resolvê-lo.
3. Perguntas sobre o Labirinto 1: o professor entrega questões relacionadas ao labirinto para que os alunos respondam.
4. Estratégia de solução do Labirinto 1: o professor apresenta as estratégias apresentadas na Seção 1.7 para ajudar os alunos a resolver o labirinto e responder às perguntas propostas.
5. Labirinto 2: os alunos terão a oportunidade de resolver um novo labirinto utilizando as estratégias que criaram anteriormente ou as apresentadas pelo professor.

Caso o professor prefira apresentar uma estratégia ou notação diferente da que foi apresentada aqui, pode fazê-lo. Assim como se perceber que algum aluno tenha criado uma estratégia mais interessante, pode ser motivador.

Para realizar esta atividade, nossa sugestão é que seja realizada individualmente ou, no máximo, em duplas.

Labirinto Tutorial

Sugerimos que, no início da atividade, o professor entregue aos alunos uma impressão da Seção Objetivo (1.1 até (inclusive) a Seção Operadores Lógicos (1.3), incluindo dois pinos. As atividades dos labirintos podem ser deixadas para depois.

O professor deve explicar como funciona o labirinto utilizando o Labirinto Tutorial (1.1) e as regras/movimentos descritos na Seção Apresentação dos Labirintos Lógicos (1.2).

É importante permitir que os alunos pensem por si mesmos para encontrar a solução, garantindo que tenham compreendido os movimentos no labirinto.

Quando todos tiverem encontrado a solução, descreva-a oralmente, evitando mostrar a notação apresentada na seção sobre as soluções.

Labirinto 1

Entregue aos alunos o Labirinto 1 (Figura 1.2) e a placa indicadora da Regra 6, para que possam lembrar se a regra está ativa ou não. Os pinos podem ser os mesmos usados no tutorial.

Novamente, incentive os alunos a pensarem por si mesmos sobre a solução e a escreverem qualquer rascunho que considerem útil para a resolução.

Peça aos alunos para anotarem de alguma forma o caminho que percorreram, permitindo que criem seu próprio tipo de notação.

Quando a maioria dos alunos encontrar a solução, entregue as perguntas sobre o Labirinto 1.

Estratégia de solução do Labirinto 1

Neste momento, o professor irá explicar a estratégia para a solução do labirinto, conforme apresentado na Seção Estratégias para solução (1.7). Se preferir, pode apresentar uma estratégia ou notação diferente.

Comece utilizando o Labirinto Tutorial, para que os alunos tenham tempo de aplicar a estratégia no Labirinto 1. Em seguida, apresente os caminhos possíveis do Labirinto 1.

Labirinto 2

Entregue o Labirinto 2 aos alunos. A placa indicadora da regra 6 não será mais necessária.

Permita que os alunos escolham entre utilizar a estratégia apresentada no passo anterior da atividade ou aquela que desenvolveram por conta própria. Além disso, forneça as perguntas sobre o Labirinto 2.

1.6 Discussões e reflexões sobre a atividade para o Professor

Esta atividade trabalha, principalmente, o raciocínio lógico, fazendo com que os alunos exercitem a análise de proposições lógicas que são compostas, indiretamente, de uma estrutura condicional da programação: **se...então**.

Ao responder cada uma das perguntas contidas nos labirintos, os alunos estarão seguindo a instrução: se sim, vou pelo caminho marcado **Sim**; se não, vou pelo caminho marcado **Não**. Além disso, o fato de que a referência estará mudando constantemente faz com que a análise precise ser feita com maior cautela.

O ponto mais marcante na resolução dos labirintos é notar qual é a resposta desejada para que o pino posicionado próximo ao GOL possa alcançá-lo. Por exemplo, no Labirinto 2 (Figura 1.3), na Caixa 6, existe a pergunta: *O outro pino está em uma caixa de cor azul OU que tenha a palavra azul?*. Para que o pino chegue até o GOL, a resposta deverá ser **Não**. Aqui, além de pensar qual a resposta desejada, o aluno deverá analisar uma sentença composta (pois tem o operador lógico **ou**), perceber que o outro pino deverá estar em uma caixa que não seja azul e nem tenha a palavra *azul* (onde a única caixa que se enquadra nesses requisitos é a Caixa 2) e, por fim, planejar um caminho para que consiga formar essa posição no labirinto.

Há diversos aspectos a considerar nesta atividade. Para solucionar um labirinto, podemos relacioná-la ao Pensamento Computacional e seus pilares: decomposição, reconhecimento de padrões, abstração e algoritmos.

Decomposição

A decomposição é o ato de separar um problema geral em partes menores e, quando possível, buscar formas de resolver cada uma dessas partes, além de observar elementos que podem ser importantes ou irrelevantes na solução do problema.

Nesta atividade dos labirintos lógicos, a decomposição pode ocorrer ao identificar passos essenciais para resolver o labirinto (será descrito mais detalhadamente no pilar seguinte) e também quando se cria uma estratégia para encontrar a solução, que pode ser vista como um exercício de abstração.

Nesse último caso, a decomposição acontece quando os alunos anotam separadamente quais pares de caixas numeradas resultam em respostas **Sim** e quais resultam em **Não**. Por exemplo, ao fixar uma caixa, como a Caixa 1, e supondo que um pino esteja nessa posição, o aluno deve imaginar onde pode estar o outro pino e qual resposta o pino da Caixa 1 seguirá em função desse outro pino.

Para exemplificar:

- Se ambos os pinos estiverem na Caixa 1, o pino que eu escolhi vai pelo caminho **Sim** ou pelo **Não**?

- Se o outro pino estiver na Caixa 2, o pino na Caixa 1 vai pelo caminho **Sim** ou pelo **Não**?
- Se o outro pino estiver na Caixa 3, o pino na Caixa 1 vai pelo caminho **Sim** ou pelo **Não**?
- Se o outro pino estiver na Caixa 4, o pino na Caixa 1 vai pelo caminho **Sim** ou pelo **Não**?
- e assim por diante.

Ou seja, estaria analisando a resposta para todos os pares de caixas possíveis.

Reconhecimento de Padrões

O reconhecimento de padrões consiste em buscar semelhanças entre o problema atual e problemas já solucionados anteriormente. Na atividade dos labirintos lógicos, podemos identificar o reconhecimento de padrões em diferentes aspectos.

Primeiro, pode-se buscar semelhanças entre os labirintos propostos aqui e os labirintos tradicionais, que podemos chamar de *labirintos visuais*. Muitas pessoas, ao ver uma imagem de labirinto e querer solucioná-lo, costumam começar pelo final em busca do início, pois essa abordagem frequentemente ajuda a encontrar o caminho correto mais facilmente do que se começassem pelo início. É claro que essa estratégia só é possível em labirintos desenhados em papel; em labirintos físicos, onde a visão não alcança o final (e nem a maioria dos caminhos), essa tática falharia.

Da mesma forma, ao tentar resolver um labirinto lógico, a pessoa pode ter a ideia de começar analisando o final. Embora não seja tão simples fazer os pinos voltarem à posição inicial do labirinto, essa abordagem permite identificar quais pares devem ser alcançados para chegar ao GOL. Por exemplo, no Labirinto 1, os pinos precisam estar posicionados de modo que um esteja na Caixa 5 e o outro na Caixa 2 (para que a multiplicação resulte em 10). Entretanto, a única maneira de um pino alcançar a Caixa 2 é vindo pela Caixa 3, seguindo o caminho marcado **Sim**. No entanto, a pergunta na Caixa 3 é um absurdo, pois a resposta é sempre *Não*, tornando impossível mover pelo **Sim** (da Caixa 3) a menos que a Regra 6 esteja ativa. Assim, um bom plano foi traçado.

Outra forma de reconhecer padrões, comparando com labirintos visuais, é por meio da estratégia mencionada no livro de Ian Stewart, *Aventuras Matemáticas: Vacas no labirinto e outros enigmas lógicos*, que inspirou esta atividade. Nessa estratégia, qualquer ponto do labirinto em que uma escolha deva ser feita é definido como um **nó**. O procedimento é chamado de *busca em profundidade primeiro* e funciona da seguinte forma:

1. Comece no nó de início.
2. Enquanto for possível, vá até qualquer nó adjacente que ainda não tenha sido visitado.
3. Quando estiver em um beco sem saída, volte até encontrar o último nó próximo a um caminho ainda não visitado. Visite-o e, em seguida, retorne à instrução nº 2.
4. Se você refez algum caminho, nunca mais o utilize.

Isso garante que, pelo menos enquanto não encontrar a saída, todos os caminhos possíveis tenham sido visitados.

Essa estratégia pode parecer inútil no caso dos labirintos lógicos, porém se você criar uma notação para ligar as posições dos pinos (tornando cada posição um nó do labirinto) você verá que ele passará a ter um caminho visual, que parecerá mais como uma árvore genealógica. Neste caso, convém analisar quando os nós se repetem (ou forem invertidos), então você poderá eliminar os que se repetiram, deixando apenas um deles.

Para mais instruções de como fazer isso, veja na Seção Solução (1.7).

Apesar de tudo isso, é possível que alguns alunos não utilizem nenhuma dessas estratégias ao

tentarem resolver os labirintos por conta própria. Nesse caso, eles podem reconhecer padrões ao perceber quais tentativas não deram certo e precisarem experimentar de outra forma, ou quando você, professor, apresentar a estratégia mencionada na seção das soluções.

Abstração

A abstração está presente na possibilidade de os alunos criarem uma forma de registrar no papel a solução do labirinto ou na hora de responder às perguntas propostas na atividade.

Caso os alunos não façam nenhum tipo de registro por conta própria durante a atividade com o Labirinto 1, talvez precisem de um exemplo para se basear. Como a proposta da atividade descrita na seção anterior sugere que o professor apresente a estratégia de solução, os alunos poderão exercitar a abstração a partir desse momento.

Para um maior detalhamento sobre a abstração, consulte a seção a seguir, que aborda as estratégias de solução.

Algoritmo

Acontece no momento que os alunos criam uma notação para descrever os movimentos que levam à solução.

1.7 Estratégias para solução

Labirinto Tutorial (e explicações de notação)

Decomposição:

Conforme explicado na Seção 1.6, os alunos podem criar um quadro apenas para anotações, registrando, para cada número, para qual caixa o pina vai quando a resposta for **Sim** e para qual caixa o pina vai quando a resposta for **Não**. Também é importante anotar quais números do outro pino resultam em uma resposta **Sim** e quais números levam a uma resposta **Não**.

Por exemplo:

1	2	3	4
$S \to 3; N \to 2$ S: 4 N: 1,2,3	$S \to 1; N \to 4$ S: 2,4 N: 1,3	$S \to GOL; N \to 2$ S: 2,3 N: 1,4	$S \to 1; N \to 2$ S: 2,3 N: 1,4

Observe a primeira coluna. Temos fixo o número 1 (um pino sobre a Caixa 1). Na linha abaixo está simbolizando o seguinte: Quando a resposta for **Sim**, o pino sobre a Caixa 1 irá para a Caixa 3. Quando a resposta for **Não**, o pino sobre a Caixa 1 irá para a Caixa 2.

Na linha seguinte, **S:4** indica que a resposta é **Sim** quando o outro pino está na Caixa 4. Já na última linha, **N: 1,2,3** indica que a resposta é **Não** quando o outro pino está nas Caixas 1, 2 ou 3.

Note que a Caixa 1 no Labirinto Tutorial pergunta: *O outro pino está em uma caixa da cor amarela?*. Como a única caixa amarela é a Caixa 4, a única forma do pino sobre a Caixa 1 ir pelo caminho **Sim** (chegando à Caixa 3) é se o outro pino estiver na Caixa 4. Logo, se o outro pino estiver nas Caixas 1, 2 ou 3, a resposta será **Não**, e o pino chegará na Caixa 2 nesses casos.

Esse método evita a necessidade de analisar constantemente as perguntas sobre cores, números ou palavras nas caixas, permitindo usar essa anotação para saber para qual caixa o pino irá em determinada posição.

Por exemplo, suponha que temos um pino na Caixa 3 e outro na Caixa 4. Se decidirmos mover o pino sobre a Caixa 4, olhamos a coluna do número 4 e verificamos em qual resposta está o número 3. O número 3 está na linha do sim (S: 2,3). Observando a linha de sim, vemos que o caminho **Sim** leva à Caixa 1. Portanto, sem analisar a pergunta, sabemos que, estando na posição (3,4) (isto é, Caixas 3 e 4). se movermos o pino da Caixa 4, iremos para a posição (3,1) isto é, Caixas 3 e 1).

Embora seguir essa tabela possa parecer complicado no início, em labirintos mais complexos, onde seria necessário analisar repetidamente características como cor, paridade, tamanho e palavras nas caixas, essa abordagem torna o processo muito mais direto e evita passos inválidos.

Abstração:

Como já fizemos loga acima, iremos denotar as posições dos pinos como se fossem pares ordenados (porém a ordem aqui é irrelevante). Por exemplo, a posição inicial dos labirintos é um pino na Caixa 1 e o outro pino na Caixa 4, teremos então a notação $(1,4)$ para esta posição.

Como já indicamos anteriormente, denotaremos as posições dos pinos em pares ordenados (embora, neste caso, a ordem não seja relevante). Por exemplo, a posição inicial dos labirintos, com um pino na Caixa 1 e outro na Caixa 4, será representada por $(1,4)$.

Para os próximos passos, exploraremos todas as possibilidades. Primeiramente, mostraremos o resultado ao mover o pino que está à esquerda na nossa notação (partindo da posição $(1,4)$, seria o número 1) e, em seguida, o resultado ao mover o número que está à direita (neste caso, o número 4).

Vamos começar com o Labirinto Tutorial. Abaixo, temos a primeira linha representando a posição inicial $(1,4)$. Em seguida, a segunda linha mostra as duas possibilidades de escolha para o primeiro passo: podemos optar por mover o pino que está na Caixa 1 ou o que está na Caixa 4.

$(1,4)$

$(3,4)(1,2)$

Lendo o que diz a segunda linha, temos que, se escolhêssemos o pino sobre a Caixa 1, ele iria para a Caixa 3 enquanto o outro pino permaneceria na Caixa 4. Por outro lado, se escolhêssemos o pino sobre a Caixa 4, este iria para a Caixa 2 enquanto o outro pino permaneceria na Caixa 1.

Note que, nesta notação, não estaremos mais preocupados com o que está escrito nas caixas, porque já definimos anteriormente para qual caixa o pino 1 vai quando o outro pino está na Caixa 4 (pois vai pelo caminho marcado **Sim**), e o mesmo para o caso em que escolhêssemos o pino sobre a Caixa 4.

Assim, as informações sobre se a caixa é amarela, azul, se o número é par ou ímpar não precisam mais ser analisadas, pois fizemos a *abstração* dessas informações. A partir de agora, basta analisar o par que temos e para qual número o pino escolhido irá.

Prosseguindo com a solução, note que, nesta segunda linha, já temos duas possibilidades de posição para fazer duas escolhas em cada uma delas. Para definir uma divisão entre as escolhas da primeira posição com as escolhas da segunda posição, iremos usar o símbolo ponto e vírgula (;).

$(1,4)$

$(3,4)(1,2)$

$(2,4)(3,1)$; $(2,2)\cancel{(1,4)}$

Os dois primeiros pares se referem ao par $(3,4)$ e os dois últimos se referem ao par $(1,2)$. Note que o par $(1,4)$ foi cancelado, pois ele já existe na primeira linha. Significa que essa escolha fará retornar ao início do labirinto. Seguiremos cancelando os pares que se repetem, analisando todas as linhas anteriores (e caso se repitam na mesma linha, deixando apenas um deles). Lembre que pares com números apenas invertidos, por exemplo $(1,4)$ e $(4,1)$, são considerados como a mesma posição.

A partir do momento que um par é cancelado, não iremos mais abrir suas possibilidades, assim poupa uma quantidade considerável de escrita e espaço no papel.

Quando um par está formado por números repetidos, como é o caso do par $(2,2)$ iremos escrever apenas uma saída.

Seguindo para o próximo passo, teremos:
$(1,4)$
$(3,4)(1,2)$
$(2,4)(3,1)$; $(2,2)\cancel{(1,4)}$
$\cancel{(1,4)(2,1)}$; $\cancel{(2,1)}(3,2)$; $\cancel{(2,1)}$
GOL

Assim, o caminho solução (mais curto) do Labirinto Tutorial é:

$$(1,4)(3,4)(3,1)(3,2)\text{GOL}.$$

Labirinto 1

1	2	3
$S \to 3; N \to 6$ S: 2,4,5,6 N: 1,3	$S \to 5; N \to 1$ S: 2,4,5 N: 1,3,6	$S \to 2; N \to 4$ S: nunca, exceto Regra 6 N: 1,2,3,4,5,6; exceto Regra 6
4	5	6
$S \to 5; N \to 6$ S: 1,3,5 N: 2,4,6	$S \to GOL; N \to 1$ S: 2 N: 1,3,4,5,6	Único$\to 3$ todos caminho único

Como temos a Regra 6, que permite mudar as regras, especificamente da Caixa 3, iremos denotar com um asterisco (*) para quando a Regra 6 estiver ativa. Assim, um par $(1,2)$ é diferente do par $(1,2)^*$, pois eventualmente pode chegar à Caixa 3 sem ter desativado a Regra 6, e isso fará diferença.

Veja a seguir os caminhos (possibilidades):

$(1,4)$
$(3,4)(1,5)$
$(4,4)(3,5)$; $\cancel{(3,5)}(1,1)$
$\cancel{(1,4)}$; $(4,5)(3,1)$; $(6,1)$
$(5,5)\cancel{(4,1)}$; $\cancel{(4,1)}(3,6)$; $(3,1)^*\cancel{(6,3)}$
$\cancel{(1,5)}$; $(4,6)(3,3)^*$; $(2,1)^*\cancel{(3,6)^*}$
$\cancel{(1,6)}(4,3)^*$; $(2,3)^*$; $\cancel{(1,1)(2,3)^*}$; $(2,6)^*\cancel{(3,3)^*}$
$(5,3)^*(4,2)^*$; $\cancel{(1,3)}(2,2)^*$; $\cancel{(1,6)(2,3)^*}$
$\cancel{(1,3)^*}(5,2)^*$; $\cancel{(1,2)^*(4,5)}$; $\cancel{(5,2)^*}$
GOL$\cancel{(5,5)}$

Há diversos caminhos igualmente curtos, com 10 passos, para a solução:

$(1,4)(3,4)(3,5)(3,1)(3,6)(4,6)(4,3)^*(5,3)^*(5,2)^*\text{GOL}$
$(1,4)(3,4)(3,5)(3,1)(3,6)(3,3)^*(2,3)^*(2,2)^*(5,2)\text{GOL}$
$(1,4)(1,5)(3,5)(3,1)(3,6)(4,6)(4,3)^*(5,3)^*(5,2)^*\text{GOL}$

$(1,4)(1,5)(1,1)(1,6)(3,6)(4,6)(4,3)^*(5,3)^*(5,2)^*$GOL
$(1,4)(1,5)(3,5)(3,1)(3,6)(3,3)^*(2,3)^*(2,2)^*(5,2)$GOL
$(1,4)(1,5)(1,1)(1,6)(3,6)(3,3)^*(2,3)^*(2,2)^*(5,2)$GOL
$(1,4)(1,5)(1,1)(1,6)(1,3)^*(3,3)^*(2,3)^*(2,2)^*(5,2)$GOL
$(1,4)(1,5)(1,1)(1,6)(1,3)^*(1,2)^*(3,2)^*(2,2)^*(5,2)$GOL

Os demais caminhos seriam possíveis de chegar à solução, mas provavelmente, com mais passos.

Labirinto 2

1	2	3
$S \rightarrow 5; N \rightarrow 2$ S: 1,4,5,6 N: 2,3	$S \rightarrow 3; N \rightarrow 4$ S: 1,3,5 N: 2,4,6	$S \rightarrow 5; N \rightarrow 4$ S: 1,2,4,6 N: 3,5
4	5	6
$S \rightarrow 6; N \rightarrow 3$ S: 1,4,5,6 N: 2,3	$S \rightarrow 2; N \rightarrow 6$ S: 1 N: 2,3,4,5,6	$S \rightarrow 1; N \rightarrow GOL$ S: 1,3,4,5,6 N: 2

Veja a seguir os caminhos (possibilidades):

$(1,4)$
$(5,4)(1,6)$
$(6,4)(5,6)$; ~~$(5,6)(1,1)$~~
~~$(1,4)(6,6)$~~; ~~$(6,6)(5,1)$~~; ~~$(5,1)$~~
~~$(1,6)$~~; $(2,1)(5,5)$
$(3,1)(2,2)$; ~~$(6,5)$~~
~~$(5,1)(3,2)$~~; $(4,2)$
$(5,2)(3,3)$; ~~$(3,2)(4,4)$~~
$(6,2)(5,3)$; $(4,3)$; ~~$(6,4)$~~
GOL~~$(6,4)$~~; $(6,3)$~~$(5,4)$~~; ~~$(3,3)(4,5)$~~

Há alguns caminhos igualmente curtos, com 10 passos, para a solução:

$(1,4)(5,4)(5,6)(5,1)(2,1)(3,1)(3,2)(5,2)(6,2)$GOL
$(1,4)(1,6)(5,6)(5,1)(2,1)(3,1)(3,2)(5,2)(6,2)$GOL
$(1,4)(1,6)(1,1)(5,1)(2,1)(3,1)(3,2)(5,2)(6,2)$GOL

Os demais caminhos seriam possíveis de chegar à solução, mas provavelmente, com mais passos.

Observamos que, para todos os labirintos aqui apresentados, não há uma posição em que, a partir dela, seja impossível solucionar o labirinto. Ou seja, em algum momento, será possível retornar a uma posição que faça parte da solução.

1.8 Respostas às perguntas do questionário

Questionário do Labirinto 1:

1. Você usou alguma estratégia? Qual?
   R: Pessoal
2. Escreveu ou rascunhou alguma coisa para ajudar a organizar o raciocínio? Como foi feito?
   R: Pessoal
3. Sabe dizer se existe alguma caixa ou posição dos dois pinos que nunca te leva a solução?
   R: Todas as caixas ou posições possíveis partindo da posição inicial permitem chegar à solução.
4. Há apenas um caminho (desde o início) que leva a solução?
   R: Não
5. Sabe dizer se existe alguma posição dos pinos que nunca irá ocorrer neste labirinto?
   R: Existem algumas, basta observar se não aparece na descrição de caminhos - possibilidades. Por exemplo, os pares (6,6) e (5,6) (ou (6,5)) não aparecem em nenhum momento.
6. No caminho que você encontrou, em algum momento os dois pinos ficam na mesma caixa?
   R: Pessoal. Note que é possível as duas soluções, conforme o que foi apresentado.
   $(1,4)(3,4)(3,5)(3,1)(3,6)(4,6)(4,3)^*(5,3)^*(5,2)^*$GOL
   é uma das soluções que não ocorre de ter os dois pinos na mesma caixa.
7. Se a resposta anterior for sim, você consegue encontrar um caminho em que isso nunca ocorra, ou seja, em nenhum momento os dois pinos fiquem juntos na mesma caixa?
   R: Pessoal
8. Se a resposta à questão 6 for não, você consegue encontrar um caminho em que, em algum momento, os dois pinos fiquem juntos na mesma caixa?
   R: Pessoal
9. Você já criou, ou consegue criar, uma notação para que outra pessoa consiga seguir o caminho para chegar à solução do labirinto?
   R: Pessoal

   Questionário do Labirinto 2:
10. Sabe dizer se existe alguma caixa ou posição dos dois pinos que nunca te leva a solução?
    R: Todas as caixas ou posições possíveis partindo da posição inicial permitem chegar à solução.
11. Há apenas um caminho (desde o início) que leva a solução?
    R: Não
12. Sabe dizer se existe alguma posição dos pinos que nunca irá ocorrer neste labirinto?
    R: Pela análise dos caminhos possíveis do Labirinto 2, todas as posições são possíveis, partindo do início.
13. No caminho que você encontrou, em algum momento os dois pinos ficam na mesma caixa?
    R: Pessoal. Note que é possível as duas soluções, conforme o que foi apresentado.

    $(1,4)(5,4)(5,6)(5,1)(2,1)(3,1)(3,2)(5,2)(6,2)$GOL é uma das soluções que não ocorre os dois pinos na mesma caixa.

    $(1,4)(1,6)(1,1)(5,1)(2,1)(3,1)(3,2)(5,2)(6,2)$GOL é a única solução (mais curta) que ocorre dois pinos na mesma caixa.
14. Se a resposta anterior for sim, você consegue encontrar um caminho em que isso nunca ocorra, ou seja, em nenhum momento os dois pinos fiquem juntos na mesma caixa?
    R: Pessoal
15. Se a resposta à questão 13 for não, você consegue encontrar um caminho em que, em algum momento, os dois pinos fiquem juntos na mesma caixa?
    R: Pessoal

1.9 Sugestões e bibliografia

Se o professor ou algum aluno tiver interesse em conhecer um labirinto lógico mais elaborado (e complicado), consulte o livro de inspiração para esta atividade:

STEWART, Ian. **Aventuras matemáticas: vacas no labirinto e outros enigmas lógicos.** Tradução Maria Luiza X. de A. Borges. Tradução de: *Cows in the maze (And other mathematical explorations)*. Rio de Janeiro: Zahar, 2012.

2. OPERADORES LÓGICOS

2.1 Objetivos

Objetivo Geral:

Desenvolver o domínio do uso dos operadores lógicos: **e**, **ou**, **não** e **xou** com o auxílio dos conhecimentos sobre intervalos reais.

Objetivos Específicos:

- Revisar e consolidar o conhecimento sobre operações com intervalos reais;
- Desenvolver o raciocínio lógico, visando chegar em um objetivo.

Pré-requisitos

- Essa atividade pode ser aplicada em qualquer turma do ensino médio.
- Conhecimentos matemáticos prévios: operações com intervalos (união, interseção, complementar, diferença).

2.2 Intervalos $\times$ desigualdades

Antes de explicarmos o jogo, vamos fazer uma rápida revisão sobre intervalos reais. Relembre que todo intervalo pode ser escrito em termos de desigualdades. Por exemplo,

- $(-3,0] = \{x \in \mathbb{R} \mid -3 < x \leq 0\}$;
- $\left(\frac{1}{2},5\right] = \left\{x \in \mathbb{R} \mid \frac{1}{2} < x \leq 5\right\}$;
- $(-\infty,2) = \{x \in \mathbb{R} \mid x < 2\}$;
- $[-7,\infty) = \{x \in \mathbb{R} \mid x \geq -7\}$

Para obter sucesso no jogo, o uso das representações gráficas dos intervalos pode ser útil. A Tabela 2.1 apresenta os diferentes tipos de intervalos, sua representação em termos de desigualdades e sua representação gráfica.

Table 2.1: Tipos de intervalos em que $a < b$.

Notação	Descrição do intervalo	Representação gráfica
(a,b)	$\{x \in \mathbb{R} \mid a < x < b\}$	
$[a,b]$	$\{x \in \mathbb{R} \mid a \leq x \leq b\}$	
$[a,b)$	$\{x \in \mathbb{R} \mid a \leq x < b\}$	
$(a,b]$	$\{x \in \mathbb{R} \mid a < x \leq b\}$	
(a,∞)	$\{x \in \mathbb{R} \mid x > a\}$	
$[a,\infty)$	$\{x \in \mathbb{R} \mid x \geq a\}$	
$(-\infty,b)$	$\{x \in \mathbb{R} \mid x < b\}$	
$(-\infty,b]$	$\{x \in \mathbb{R} \mid x \leq b\}$	
$(-\infty,\infty)$	$\mathbb{R}$ (conjunto de todos os reais)	

2.3 Apresentação do jogo

Nesta atividade você irá brincar com os operadores lógicos **e**, **ou**, **xou** ou **não**. Se você, estudante, já domina bem as operações de união e interseção de intervalos reais irá compreender facilmente essa parte.

Neste jogo você irá formar sentenças com desigualdades e operadores lógicos que correspondem a um intervalo, ou união de intervalos, que contenha um número o qual você sorteou, que chamaremos de *número-alvo*.

E como vou formar essas sentenças?

Você irá formar utilizando as cartas. Por exemplo, vamos supor que seu número-alvo fosse $x = 3$ e você tivesse as seguintes cartas:

$\boxed{x > 4}$ $\boxed{2 < x < 5}$ $\boxed{x \leq 2}$ $\boxed{\text{e}}$ $\boxed{\text{ou}}$ $\boxed{\text{não}}$

Se você organizar elas da seguinte forma:

$\boxed{2 < x < 5}$ $\boxed{\text{e}}$ ($\boxed{\text{não}}$ $\boxed{x > 4}$ $\boxed{\text{ou}}$ $\boxed{x \leq 2}$),

que é equivalente a

$\boxed{2 < x < 5}$ $\boxed{\text{e}}$ ($\boxed{x \leq 4}$ $\boxed{\text{ou}}$ $\boxed{x \leq 2}$),

o intervalo correspondente dessa sentença será $(2,4]$. Logo o número 3 estará contido neste intervalo.

Atenção! Apenas tome cuidado para não se confundir quando os extremos estiverem aberto! Supondo que seu número-alvo fosse $x = 2$, o intervalo resultante do exemplo acima não o contém. Então você não teria atingido seu objetivo!

Como funciona o operador xou?
Esse operador é parecido com o **ou**, porém ele subtrai o(s) intervalo(s) comum(ns) entre os intervalos que estão sendo operados. Por exemplo,

$$(2 < x < 5 \textbf{ xou } x > 4),$$

corresponde a seguinte operação de intervalos:

$$((2,5) \cup (4,+\infty)) - ((2,5) \cap (4,+\infty)).$$

Assim, teremos

$$((2,5) \cup (4,+\infty)) - ((2,5) \cap (4,+\infty)) = (2,+\infty) - (4,5) = (2,4] \cup [5,+\infty).$$

Isso pode ser muito útil para diminuir o total do comprimento dos intervalos durante o jogo! (Mais adiante você verá por quê).

Comprimento dos intervalos? O que é isso?
Para esse jogo, define-se o **comprimento** de cada um dos intervalos $[a,b]$, (a,b), $(a,b]$ e $[a,b)$ como sendo a diferença de

$$b - a.$$

No caso da união, o comprimento da união de dois ou mais intervalos disjuntos define-se como a soma do comprimento de cada intervalo.
Vejamos alguns exemplos:

- $(-2,1]$ tem comprimento $1-(-2) = 3$.
- $(-10,10)$ tem comprimento $10-(-10) = 20$;
- $[-4,-1) \cup [3,15]$ tem comprimento $(-1-(-4)) + (15-3) = 3+12 = 15$;
- $[-3,0] \cup (4,5]$ tem comprimento $3+1 = 4$;
- $(-4,2) \cup [3,5)$ tem comprimento $6+2 = 8$.

Observação: Se tiver um único "número avulso" na união de intervalos, por exemplo, $[-1,3] \cup \{4\} \cup (5,7]$, como o número 4 está sem valores próximos, note que ele é apenas um ponto e não tem comprimento, isto é, o conjunto $\{4\}$ tem comprimento 0 (zero), então ele será desconsiderado. Portanto, o comprimento da união de intervalos $[-2,3] \cup \{4\} \cup (5,7]$ é $5+0+2 = 7$

2.4 Regras do Jogo

Número de jogadores:

De 2 a 4 jogadores.

Pontuação e Eliminação:

- **Pontuação Inicial:** Todos os jogadores iniciam com 30 pontos.
- **Redução de Pontos:** A cada partida, a pontuação dos jogadores é reduzida pelo valor definido por aquela partida.
- **Eliminação e vencedor:** Quando a pontuação de um jogador chega a zero ou fica negativa, ele é eliminado do jogo. O último jogador com pontos positivos é o vencedor.

Componentes do jogo:

- **Dado dos operadores lógicos**: **e**, **ou**, **xou** ou **não**;
- **Cartas de número-alvo:** São escritas da forma $\boxed{x = \_\_}$, com algum número real na lacuna;
- **Cartas de desigualdades:** São cartas que apresentam desigualdades, como $\boxed{x < \_\_}$ ou $\boxed{\_\_ < x \leq \_\_}$;
- **Cartas de operadores lógicos:** Estão escritas com um dos operadores lógicos: **e**, **ou**, **xou** ou **não**;
- **Cartas de parênteses:** São apenas parênteses ( );

Objetivo de cada partida

Seu objetivo é obter desigualdades combinadas com operadores lógicos que correspondam a um intervalo, ou união de intervalos, que contenha o seu *número-alvo* definido no início da partida. Além disso, o intervalo deve possuir o menor comprimento possível.
Que tal um exemplo? Suponha que o número-alvo sorteado foi $x = 2$ e que, durante a partida, você obteve $x \geq -5$ **e** $x < 10$. Se você finalizar sua partida assim, o intervalo correspondente é $[-5, 10)$, que contém o número 2, e seu comprimento é 15. Você alcançou seu objetivo, porém o comprimento ainda poderia ser reduzido
Continue lendo as regras a seguir para aprender como melhorar esse intervalo.

Como jogar uma partida

O jogo segue no sentido anti-horário (o próximo jogador é o que está a sua direita). Definam quem começa a jogar.

Sorteando as cartas iniciais

Cada jogador sorteia um *número-alvo* e depois que todos tiverem sorteado, cada jogador irá escolher, sem olhar o que está escrito, uma carta de desigualdade. Essa escolha pode ser feita pegando qualquer carta do monte (não precisa ser a que estiver mais em cima), para isso pode-se

abrir o monte de cartas de desigualdades como um leque deixando a parte escrita para baixo e o jogador, em sua vez, escolhe uma.

A partir deste momento, os jogadores analisam o intervalo definido pela carta de desigualdade e seu número-alvo. Seguindo o mesmo sentido, os jogadores podem escolher se desejam pegar mais uma carta de desigualdade ou finalizar sua jogada. Isto é, se o jogador preferir finalizar sua jogada com apenas uma carta de desigualdade e sem usar nenhum operador lógico, ele pode, desde que o intervalo contenha o seu número-alvo.

Não é necessário esconder suas cartas dos outros jogadores, pode colocá-las sobre a mesa.

Pegando mais cartas de desigualdades:
Para cada carta de desigualdade nova que o jogador quiser pegar, ele deverá jogar o dado dos operadores lógicos.
Após jogar o dado, o jogador pega uma carta de operador lógico com o escrito de acordo com o que definido pelo dado e mantém essa carta consigo.
Caso o dado caia no operador **não**, o jogador pode pegar um operador **não** e então jogar o dado novamente até que caia em outro operador, diferente deste. (Se cair em **não** novamente, não poderá pegar mais um operador **não**, apenas joga o dado novamente para definir outro operador que não seja o **não**).

Parênteses são de graça: você pode usar quantos forem necessários, não precisa esperar sua vez de jogar para pegar as cartas de parênteses.

Finalizando a partida:
Com essas cartas, cada jogador deverá formar desigualdades com os operadores lógicos que correspondam a um intervalo, ou união de intervalos, que contenha o seu número-alvo.
Para realizar as operações, os jogadores podem fazer representações gráficas dos intervalos em um papel se necessário, isso certamente ajudará a encontrar a melhor solução.
Os jogadores precisam usar o máximo possível de cartas dentre as que eles pegaram, pois, apesar de não ser obrigatório usar todas as cartas em sua mão, a quantidade de cartas não utilizadas fará com que você perca mais pontos, então pense com sabedoria se vale a pena arriscar pegar uma nova carta!
As únicas cartas que não dão pontuação são os parênteses (lembra que são de graça?) e o operador **não**, este é considerado um curinga no jogo.
Se um jogador decidir parar de pegar cartas novas, então apenas pula sua vez nas rodadas seguintes até que todos tenham terminado de formar suas desigualdades nesta partida.
Assim que todos decidirem parar de definir suas sentenças, encerra-se a partida e inicia a contagem dos pontos.

Contagem dos pontos

A pontuação total da partida de cada jogador é definida pela soma do **comprimento do intervalo (ou união de intervalos) resultante** com o **número de cartas não utilizadas** em sua mão.

Vamos apresentar abaixo um exemplo para ficar mais claro:

Suponha que seu número alvo fosse $x = 3$ e você pegou cartas na seguinte ordem:

- $\boxed{x > 4}$: teu alvo não está aqui;
- $\boxed{x > 2}$ $\boxed{\text{ou}}$: agora você tem o seu alvo, porém não consegue limitar (eliminar o infinito) ainda;
- $\boxed{x \leq 7}$ $\boxed{\text{não}}$ $\boxed{\text{e}}$: aqui você consegue limitar o intervalo (eliminando o infinito).

Se você organizasse as cartas da seguinte forma:

$$\boxed{\text{não}}\ \boxed{x > 4}\ \boxed{\text{e}}\ \boxed{x > 2}$$

O intervalo resultante desta sentença será $(2,4]$ e o número 3 estará contido no intervalo.
O comprimento do intervalo resultante é 2, essa seria a pontuação da partida, porém note que não foram usadas as cartas $\boxed{\text{ou}}$ e $\boxed{x \leq 7}$. Portanto acrescenta 1 ponto para cada carta não utilizada na pontuação da partida.
Então a pontuação dessa partida é 4. E esse número será reduzido dos pontos deste jogador, então note que quanto menor a pontuação em cada partida, melhor é para o jogador!
Lembre-se que caso não utiliza-se a carta do operador **não**, esta não iria contar pontos.

Observação: Caso algum jogador não consiga mais definir uma sentença que resulte um intervalo, contendo seu número-alvo, sem limitar os intervalos (ou seja, não conseguiu "cancelar o infinito") então a pontuação dele será 25 pontos (caso o jogo tenha começado com 25 pontos cada jogador, ele já estaria eliminado).

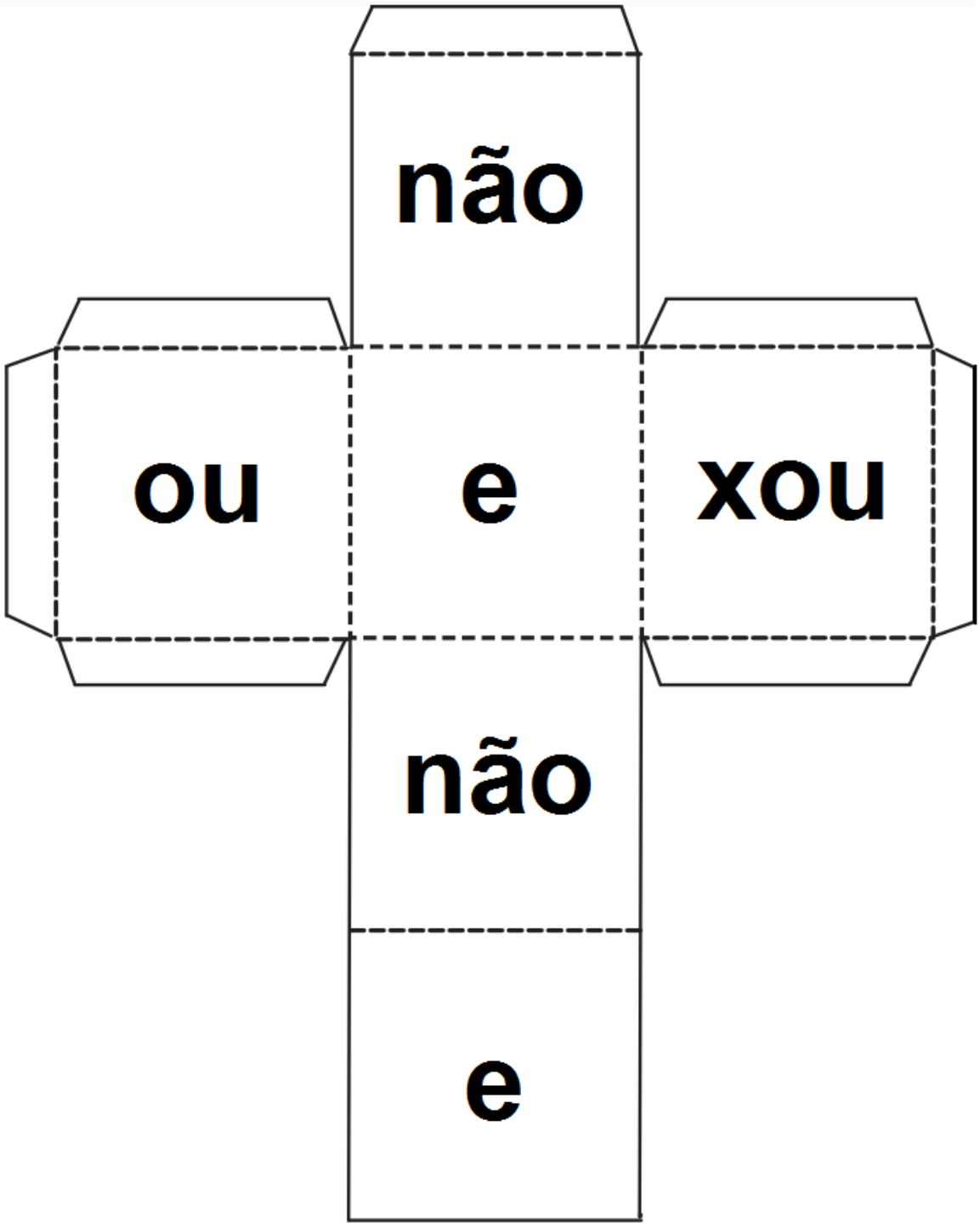
Para registrar as pontuações dos jogadores, anotem em um quadro da seguinte forma:

Partida	Jogador 1	Jogador 2	Jogador 3
Início	25	25	25
1ª	$-4 = 21$	$-6 = 19$	$-8 = 17$
2ª			
3ª			
4ª			
5ª			
...			

(Os valores acima são apenas exemplos).

2.5 Confecção

Figure 2.1: Dado dos operadores lógicos



Fonte: Elaborado pela autora

Cartas de número-alvo

$x=-7$	$x=-6$	$x=-5$	$x=-4$	$x=-3$
$x=-2$	$x=-1$	$x=0$	$x=1$	$x=2$
$x=3$	$x=4$	$x=5$	$x=6$	$x=7$
$x=3$	$x=2$	$x=-2$	$x=-1$	$x=-1$

Parênteses

(	(	(	(	(	(	(	(	(	(	(	(	(	(
(	(	(	(	(	(	(	(	(	(	(	(	(	(
(	(	(	(	(	(	(	(	(	(	(	(	(	(
(	(	(	(	(	(	(	(	(	(	(	(	(	(
(	(	(	(	(	(	(	(	(	(	(	(	(	(
(	(	(	(	(	(	(	(	(	(	(	(	(	(
(	(	(	(	(	(	(	(	(	(	(	(	(	(

Cartas de desigualdades

$-10 < x < 1$	$-9 \leq x < 2$	$-8 < x \leq 3$	$-7 \leq x \leq 4$
$-6 < x < 5$	$-5 \leq x < 6$	$-4 < x \leq 7$	$-3 \leq x \leq 8$
$-2 < x < 9$	$-1 \leq x < 10$	$-10 < x \leq 0$	$0 \leq x \leq 10$
$-10 < x < 2$	$-9 \leq x < 3$	$-8 < x \leq 4$	$-7 \leq x \leq 5$
$-6 < x < 6$	$-5 \leq x < 7$	$-4 < x \leq 8$	$-3 \leq x \leq 9$
$-2 < x < 10$	$-1 \leq x < 9$	$-9 < x \leq 0$	$0 \leq x \leq 9$
$0 < x < 8$	$-7 \leq x < 0$	$-9 < x \leq -1$	$1 \leq x \leq 9$
$x < 3$	$x \leq 4$	$x > 2$	$x \geq 7$
$x < 6$	$x \leq 5$	$x > 4$	$x \geq 3$
$x < 2$	$x \leq 1$	$x > 0$	$x \geq -1$
$x < -2$	$x \leq -3$	$x > -4$	$x \geq -5$
$x < -6$	$x \leq -7$	$x > 5$	$x \geq 4$

Cartas de operadores lógicos

não	**não**	**não**	**não**	**não**
não	**não**	**não**	**não**	**não**
não	**não**	**não**	**não**	**não**
e	**e**	**e**	**e**	**e**
e	**e**	**e**	**e**	**e**
e	**e**	**e**	**e**	**e**
e	**e**	**e**	**e**	**e**
ou	**ou**	**ou**	**ou**	**ou**
ou	**ou**	**ou**	**ou**	**ou**
ou	**ou**	**ou**	**ou**	**ou**
xou	**xou**	**xou**	**xou**	**xou**
xou	**xou**	**xou**	**xou**	**xou**

2.6 Orientações sobre a atividade para o Professor

Professor, ao aplicar esta atividade, é importante revisar com os alunos sobre as operações com intervalos reais.

Na aplicação do jogo é preferível que seja feito em grupos de 4 alunos. Caso os alunos tenham dificuldades, é possível que joguem em duplas, de forma colaborativa, contra outras duplas, permitindo também a inclusão de mais alunos.

Explique o operador **xou** usando a representação gráfica de intervalos. Caso você perceba que os alunos não estejam preparados para usarem este operador, pode solicitar que jogem algumas partidas tratando o **xou** como se fosse apenas **ou** e depois solicitar para que os alunos joguem da forma original, considerando o **xou**.

Se o tempo de aula permitir, pode ser interessante que os grupos que estão jogando anotem quantas partidas foram realizadas até que um vencedor seja definido. Em seguida, compare entre os diferentes grupos qual deles precisou de mais partidas para definir um vencedor, o que indicaria que o grupo conseguiu formar sentenças de forma otimizada, perdendo a menor quantidade de pontos a cada partida em comparação aos outros grupos.

Sugestão extra: Se você tiver grupos de alunos que não se sentem à vontade com jogos competitivos, uma alternativa é ajustar as regras para transformar o jogo em uma *atividade colaborativa*. Nesse formato, em vez de os alunos competirem entre si, eles podem jogar juntos, começando com uma pontuação menor (por exemplo, 15 pontos). Eles continuam jogando até zerar ou ficarem com pontuação negativa, e então recomeçam com a mesma pontuação inicial, mas com o objetivo de fazer o jogo durar mais partidas do que na fase anterior. Assim, de dois a quatro alunos podem jogar com as mesmas cartas e o mesmo número-alvo, colaborando para superar o recorde anterior, ou seja, prolongar o número de partidas.

2.7 Discussões e reflexões sobre a atividade para o Professor

Essa atividade focou na compreensão dos operadores lógicos. Diferentemente da atividade dos *Labirintos Lógicos*, que utiliza os operadores com valores booleanos, aqui os operadores são aplicados a valores numéricos.

Pode ser interessante posteriormente à esta atividade, solicitar aos alunos que escrevam um pseudocódigo utilizando esses conceitos. Por exemplo: um pseudocódigo sobre como um programa de planilha iria fazer para preencher as células da planilha com cores de acordo com os valores das notas de uma turma.

Para mais instruções sobre como escrever pseudocódigo confira o quarto capítulo da dissertação associada a este caderno de atividades.

2.8 Orientações de confecção

A confecção das cartas se dá apenas imprimindo as cartas contidas nas páginas a seguir e destacá-las.

Pode ser interessante fazer com que cada tipo de carta seja impresso em uma cor de folha sulfite diferente, para facilitar a distinção entre cartas de número-alvo, cartas de desigualdades, cartas de operadores lógicos e cartas de parênteses. Isso facilitaria na hora de guardar o material para usar em outra turma, ou usar outro dia, pois as cores diferentes facilitariam em separar os tipos de cartas, caso os alunos venham a misturá-las.

Em relação ao dado, caso considere uma opção melhor, tanto pela praticidade quanto pelo fato de que o dado de papel pode ficar viciado devido ao peso que deve influenciar das abas e cola para

montá-lo, é possível também utilizar um dado comum (de 6 faces) e anotar no quadro ou imprimir as seguintes definições para os valor do dado:

1. não, 2. e, 3. ou, 4. e, 5. xou, 6. não.

3. ALGORITMOS

3.1 Objetivos

Objetivo Geral

Escrever um passo a passo (um algoritmo) para resolver um quebra-cabeça dado.

Objetivos Específicos:

- Compreender o conceito de algoritmo;
- Identificar e dividir uma tarefa em etapas menores e essenciais;
- Descrever os passos necessários para completar uma atividade de forma clara e sequencial;
- Antecipar e resolver possíveis ambiguidades na linguagem algorítmica.

Pré-requisitos

- Essa atividade é voltada para alunos do 9º ano.
- Pré-requisito: Os alunos devem conhecer os nomes de determinadas formas geométricas e de termos como: hipotenusa, cateto, ângulo reto, setor circular, etc.

3.2 Apresentação

O que é um algoritmo?
Algoritmo é uma sequência de passos a serem realizados com objetivo de realizar alguma tarefa.

Receitas culinárias são um exemplo prático de algoritmos. Nelas podemos encontrar os ingredientes e as etapas necessárias para fazer uma comida. Veja a seguir dois exemplos de algoritmos:

Algoritmo 3.1: Como ligar um computador

1 Conecte o cabo de energia na tomada;
2 No torre, pressione o botão de ligar;
3 Se o monitor estiver desligado, ligue–o;
4 Selecione o usuário desejado na tela de login;
5 Digite a senha correspondente ao usuário selecionado;
6 Pressione a tecla ENTER.

Fonte: Elaborado pela autora

O seguinte exemplo apresenta um algoritmo para determinar se um número natural é divisível por 3, assumindo que se sabe a tabuada de 3 (ou seja, sabe-se quais são os múltiplos de 3 entre 0 a 30):

Algoritmo 3.2: Verificar se um número é divisível por 3

1 Escolha um número natural inicial.
2 Some todos os algarismos que compõem este número.
3 Verifique se o resultado dessa soma é um número entre 0 e 30:
4 Se não for, retorne ao passo 2 usando o número obtido como o novo número.
5 Se for, siga para o próximo passo.
6 Determine se o resultado é um múltiplo de 3:
7 Se sim, então o número inicial é divisível por 3.
8 Se não, então o número inicial não é divisível por 3.

Fonte: Elaborado pela autora

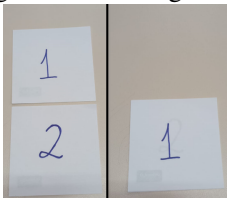
É importante garantir que as instruções sejam claras e precisas, evitando qualquer ambiguidade, para que a interpretação da máquina ou por outra pessoa não ocasione um resulte não esperado. As instruções a seguir, demonstram um exemplo de ambiguidade:

Algoritmo 3.3: Algoritmo com ambiguidade

1 Pegue dois pedaços de papel de mesmo tamanho;
2 Escreva o número 1 em um papel e o número 2 no outro papel;
3 Posicione o papel com o número 1 em cima do papel com o número 2.

Fonte: Elaborado pela autora

Figure 3.1: Ambiguidade



Fonte: Elaborado pela autora

Ora, aqui temos uma ambiguidade, pois a instrução de colocar o papel 1 “em cima” do papel 2 pode ser interpretada de duas maneiras, observe essas interpretações na Figura 3.1).

Observe que ambos os casos podem ser interpretados como o papel número 1 está em cima do papel número 2, pois não foi definido nenhuma referência para descartar uma das possibilidades. Para descartar, por exemplo, a segunda opção, poderia-se adicionar a instrução: *“Considere as disposições dos papéis apenas sobre o plano da mesa, ou seja, não podendo haver sobreposição dos papéis;”* assim, só poderá existir a primeira opção.

3.3 Atividade

A atividade pode ser realizada em grupos e dividida em 4 etapas, na qual cada grupo receberá um tangram.

Etapa 1: Cada grupo resolve o seu quebra-cabeça.

Etapa 2: Em seguida o grupo deve escrever um algoritmo (as instruções) para a solução do quebra-cabeça. Não pode desenhar.

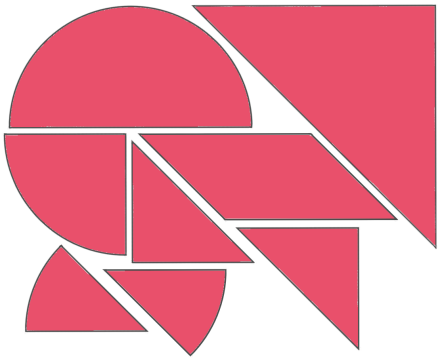
Etapa 3: Os quebra-cabeças devem ser desmontados e trocados entre os grupos com diferentes quebra-cabeças, entregando junto o algoritmo escrito. Cada grupo deve usar o algoritmo elaborado pelo outro grupo para resolver o quebra-cabeça recebido.

Etapa 4: Por fim, dever ser realizada uma discussão com a turma sobre as falhas nos algoritmos, ambiguidades e como melhorar o algoritmo.

Tangram coração - Etapa 1

Este quebra-cabeça é composto pelas seguintes peças: um semicírculo, um setor circular de 90°, dois setores circulares de 45°, um triângulo retângulo grande, dois triângulos retângulos pequenos de mesmo tamanho e um paralelogramo. Conforme a figura a seguir.

Figure 3.2: Tangram coração de 8 peças



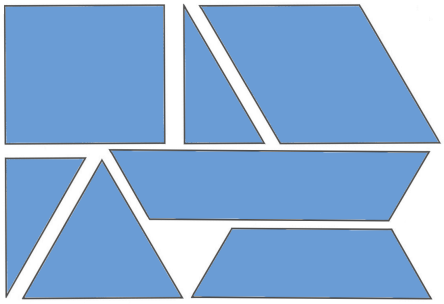
Fonte: Elaborado pela autora

O objetivo é formar um coração com essas peças. Devem ser usadas todas as peças, sem sobreposição das peças e sem sobrar espaços entre elas.

Tangram triângulo - Etapa 1

Este quebra-cabeça é composto pelas seguintes peças: um quadrado, dois triângulos retângulos de mesmo tamanho, um paralelogramo, um triângulo equilátero, dois trapézios sendo um mais comprido que o outro.

Figure 3.3: Tangram triângulo equilátero de 7 peças



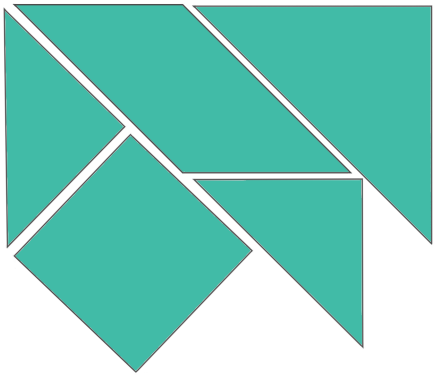
Fonte: Elaborado pela autora

O objetivo é formar um triângulo equilátero com essas peças. Devem ser usadas todas as peças, sem sobreposição das peças e sem sobrar espaços entre elas.

Tangram quadrado - Etapa 1

Este quebra-cabeça é composto pelas seguintes peças: um quadrado, um paralelogramo, um triângulo retângulo grande e dois triângulos retângulos pequenos.

Figure 3.4: Tangram quadrado de 5 peças



Fonte: Elaborado pela autora

O objetivo é formar um quadrado com essas peças. Devem ser usadas todas as peças, sem sobreposição das peças e sem sobrar espaços entre elas.

Escrever o algoritmo - Etapa 2

Alunos: _____

Pré-definições do algoritmo:

- Considere as disposições das peças apenas sobre o plano da mesa, ou seja, não podendo haver sobreposição das peças;
- Devem ser usadas todas as peças;
- Não deve sobrar espaços entre as peças.

Em equipe, resolvam o quebra-cabeça que foi entregue a vocês. Após completá-lo, elaborem um passo a passo (um algoritmo) que descreva as etapas necessárias para montá-lo. Esse algoritmo deve ser claro o suficiente para que outra equipe possa utilizá-lo para resolver o quebra-cabeça seguindo as instruções fornecidas.

Depois de terminar de escrever o algoritmo, desmonte o quebra-cabeça e entregue as peças junto com o algoritmo para outra equipe (que tenha recebido um quebra-cabeça diferente da sua equipe).

Montar o quebra-cabeça seguindo o algoritmo - Etapa 3

Alunos: _____

Equipe de quem vocês receberam o quebra-cabeça e o algoritmo:

Tentem resolver o quebra-cabeça seguindo exatamente s instruções do algoritmo elaborado pela equipe de quem vocês o receberam.

Discussão com a turma - Etapa 4

Respondam as seguintes questões:

1 - No quebra-cabeça que sua equipe resolveu e precisou escrever o algoritmo, quais foram as dificuldades que vocês encontraram? (Etapas 1 e 2)

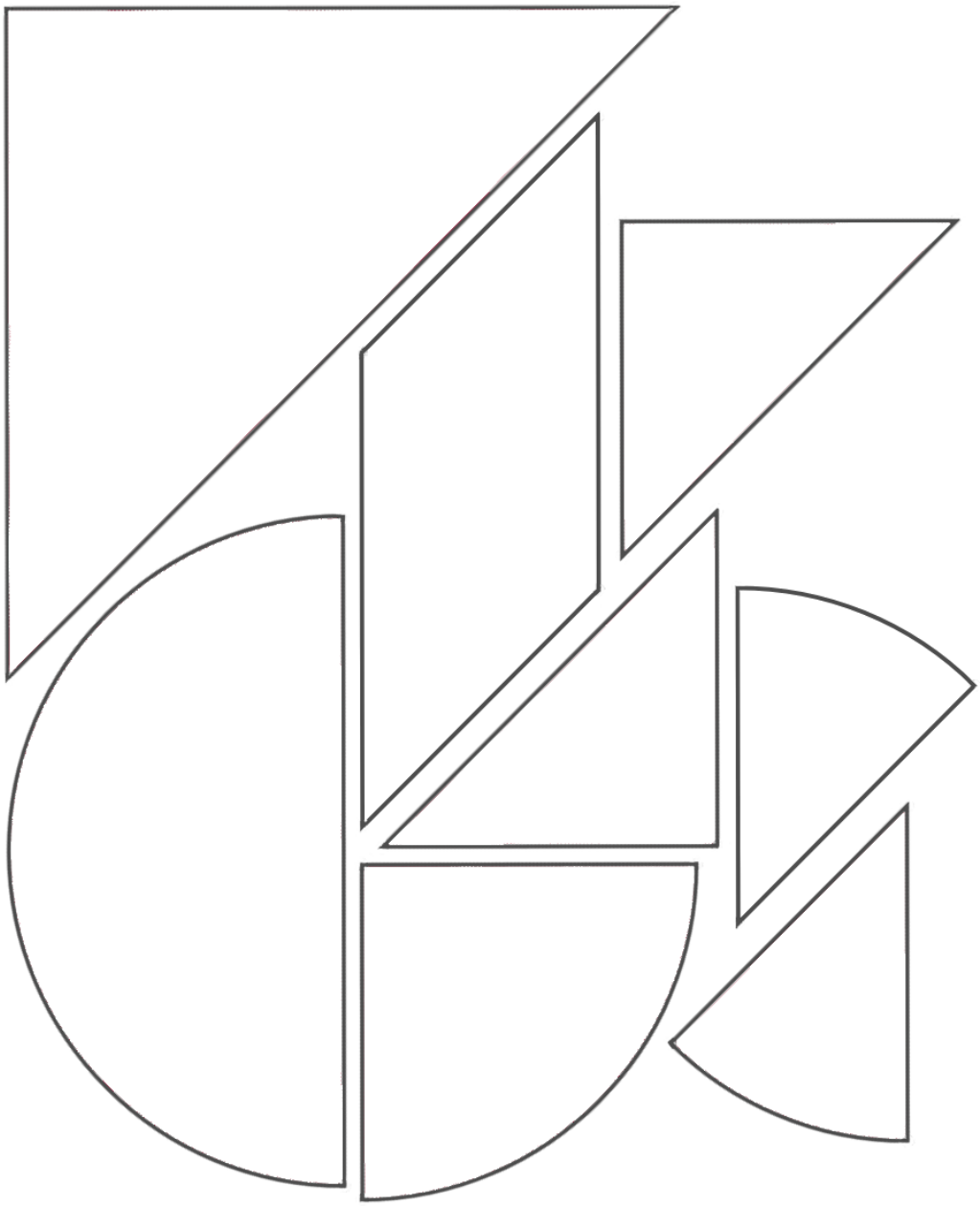
2 - Quando sua equipe recebeu o novo quebra-cabeça com o algoritmo escrito pela outra equipe, vocês conseguiram resolver facilmente ao seguir as instruções do algoritmo? (Etapa 3)

3 - Vocês tiveram dúvida sobre alguma etapa do algoritmo (escrito pela outra equipe) que poderia causar interpretações diferentes? (Etapa 3)

4 - Em alguma parte da solução com o algoritmo, vocês precisaram adivinhar a posição de algumas peças por si mesmos? (Etapa 3)

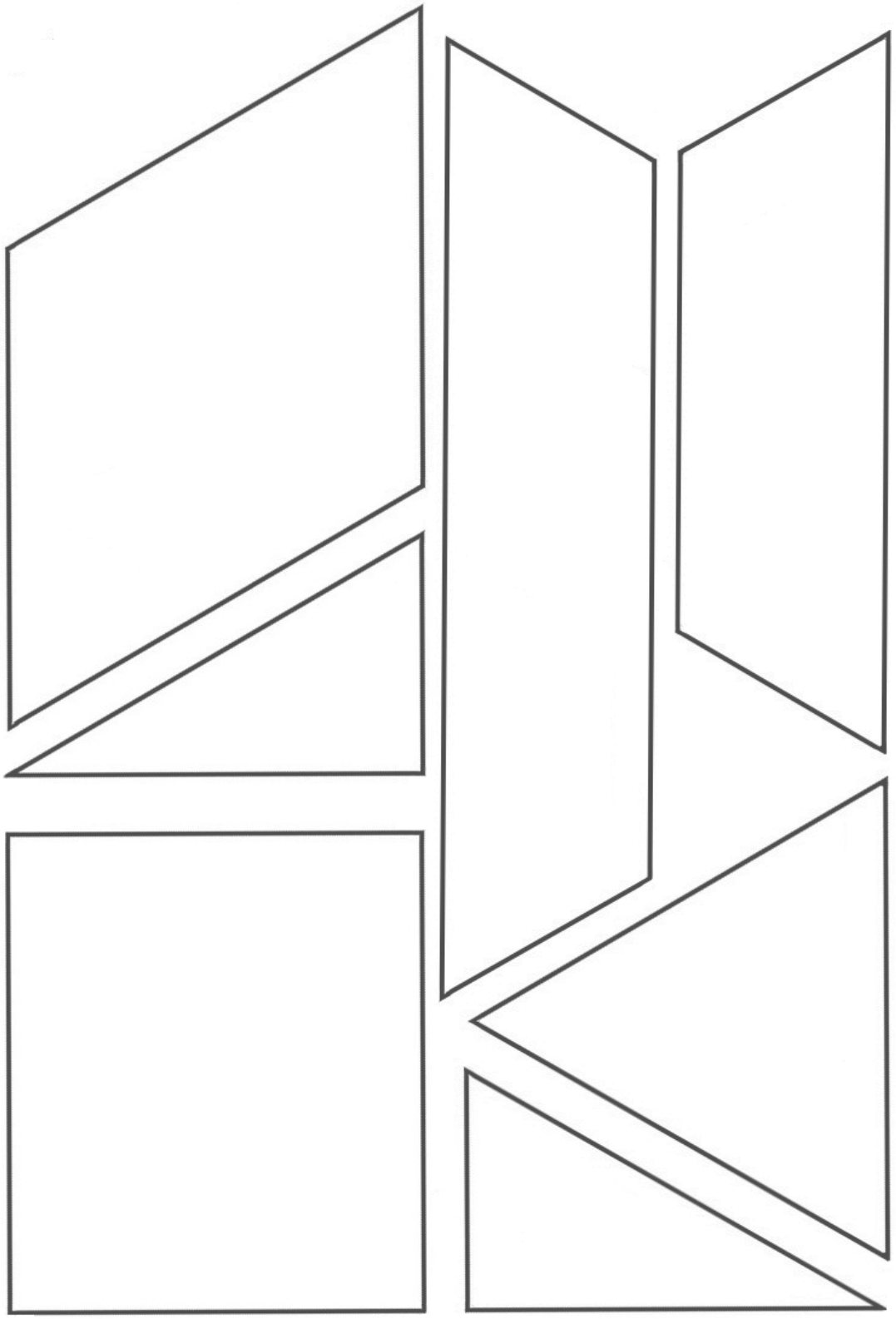
3.4 Confecção

Figure 3.5: Confecção Tangram Coração



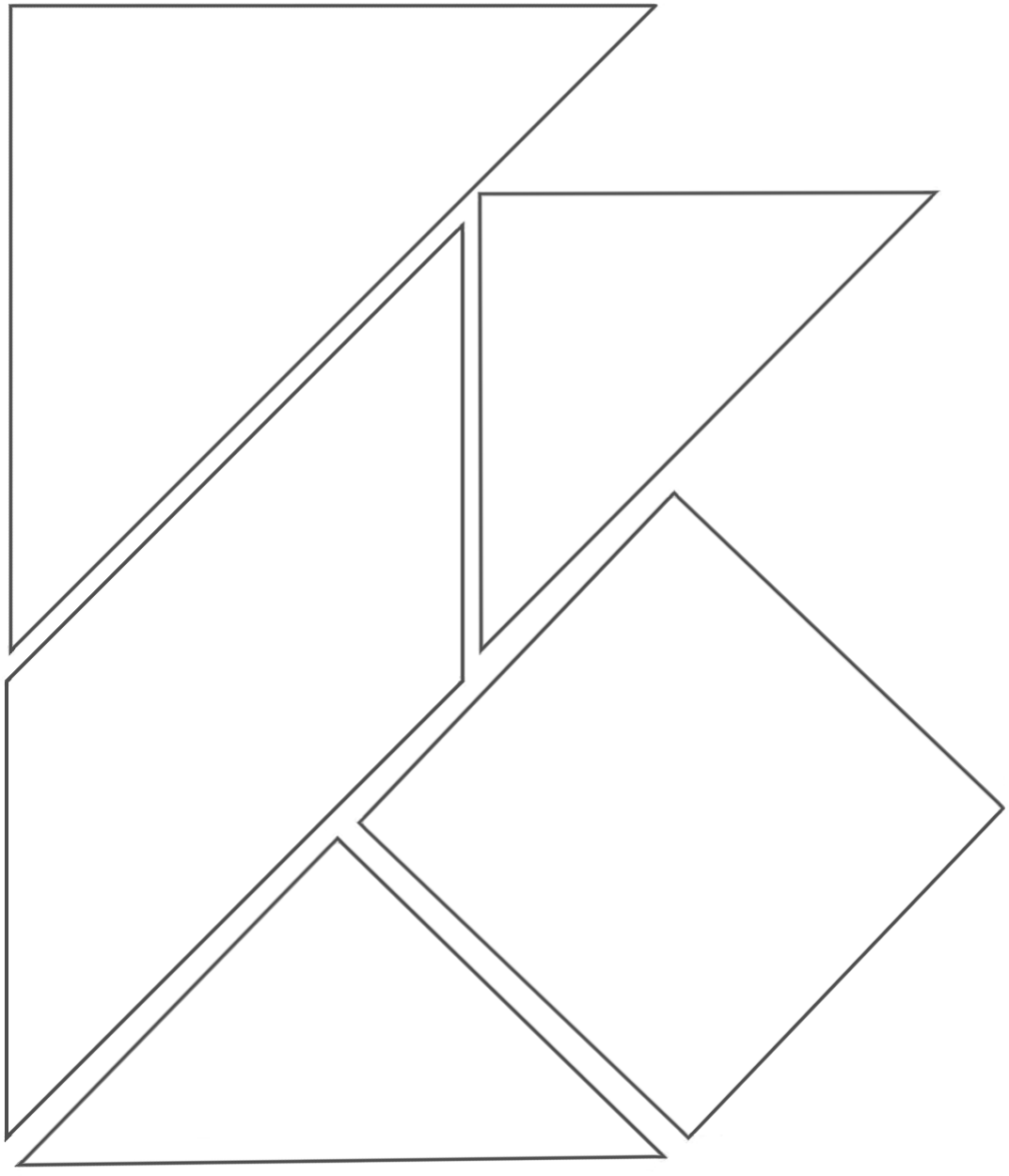
Fonte: Elaborado pela autora

Figure 3.6: Confecção Tangram Triângulo Equilátero



Fonte: Elaborado pela autora

Figure 3.7: Confecção Tangram Quadrado - 5 peças



Fonte: Elaborado pela autora

3.5 Orientações sobre a atividade para o Professor

Apresentação

Na seção da apresentação, já foi instruído aos alunos sobre o que é um algoritmo, mas o professor pode explicar o conteúdo, se preferir, ao invés de solicitar que eles aprendam sozinhos através da leitura.

As equipes podem ser divididas em 3 ou 4 alunos, mas é possível também que trabalhem individualmente ou duplas. Tenha em mente que em equipes muito grandes podem resultar em alguns alunos pouco participativos.

Para complementar a ideia principal desta atividade, pode apresentar o seguinte vídeo com título: "Como ensinar linguagem de programação para uma criança" (é possível clicar no título, pois ele está com hiperlink e será direcionado ao vídeo no YouTube), que apresenta um pai solicitando as filhos para descreverem os passos de como fazer um sanduíche de manteiga de amendoim com geleia, onde o pai segue apenas os passos descritos, tentando encontrar brechas nas instruções que possam resultar em interpretações erradas.

Etapa 1

Imprima os quebra-cabeças apresentados na seção de Confecção. Você pode levar as peças já recortadas ou pedir que os próprios alunos façam o recorte, pois, nessa seção, os quebra-cabeças não estão resolvidos.

Na distribuição dos quebra-cabeças, procure evitar que equipes diferentes que estejam próximas recebam o mesmo quebra-cabeça. Se quiser encontrar outros quebra-cabeças na internet, apenas cuide para que todas as peças sejam figuras geométricas com nomes conhecidos pelos alunos.

No entanto, não é necessário que todas as equipes tenham quebra-cabeças diferentes, inclusive, pode ser interessante na discussão da atividade (última etapa), que equipes diferentes tenham elaborado algoritmos completamente distintos, seja na ordem das peças, na clareza da descrição das etapas, entre outros fatores.

Ao iniciar a execução da atividade, na primeira etapa, onde os alunos resolvem os quebra-cabeças pela primeira vez, é importante que você esteja atento ao tempo disponível para realizar dessa etapa.

Conforme as equipes progridem na solução do quebra-cabeça, se houver algumas com maior dificuldade, o professor pode oferecer pequenas dicas sobre como determinadas peças devem ser posicionadas para facilitar a solução. Por exemplo: "essas peças devem ficar juntas desta forma", "essa parte desta peça é um dos lados do quadrado", ou "esse ângulo reto não pertence a nenhum dos vértices do quadrado que vocês querem montar". Essas dicas podem agilizar a finalização desta etapa, pois a parte principal da atividade é a etapa em que os estudantes irão escrever os algoritmos, e eles precisarão de mais tempo para isso.

Eventualmente, pode acontecer de algumas equipes não conseguirem resolver o quebra-cabeça no tempo determinado. Nessa situação, é preferível que o professor disponibilize apenas a solução visual para essas equipes (sem apresentar o algoritmo que se encontra abaixo da solução visual), solicitando apenas que a equipe elabore o algoritmo para a solução do quebra-cabeça. Portanto, se for perceptível que muitas equipes já terminaram de resolver o quebra-cabeça e começaram a escrever os algoritmos, é aconselhável que o professor intervenha nas equipes que ainda não encontraram a solução.

Etapa 2

Na elaboração dos algoritmos realizada pelos alunos (etapa 2) o professor evita fazer intervenções na escrita. As ambiguidades, caso existam, provavelmente surgirão quando a outra equipe for resolver o quebra-cabeça seguindo apenas o algoritmo escrito. Se for necessário fazer alguma sugestão, isso deve ser direcionado apenas às equipes que estejam mais estagnadas na escrita ou quando os alunos precisarem se lembrar dos nomes dos elementos das figuras geométricas (cateto, ângulo reto, trapézio, base maior, etc.).

Etapa 3

Na execução da etapa 3 da atividade, onde os quebra-cabeças são trocados entre as equipes, pode ser interessante solicitar que os alunos anotem as possíveis ambiguidades que surgirem ao resolver o quebra-cabeça seguindo as orientações do algoritmo elaborado pela outra equipe. Isso ajudará a garantir que eles não esqueçam das dúvidas que tiveram durante a execução do algoritmo, para que possam relatar essas dificuldades na discussão (etapa 4).

Etapa 4

Na etapa final, o professor conduzirá a turma a compartilhar suas experiências tanto na elaboração do algoritmo quanto na execução do algoritmo escrito por outra equipe.

É possível que algumas equipes encontrem trechos do algoritmo que deixem dúvidas sobre a disposição das peças. Instrua os alunos a discutir as ambiguidades existentes, apontando as diferentes possibilidades de interpretação dos comandos escritos e mostrando, com as peças, o que aconteceria em cada caso. Incentive-os a pensar em como poderiam ajustar a escrita do comando para eliminar essas ambiguidades.

3.6 Discussões e reflexões sobre a atividade para o Professor

Observe, professor, que a intenção principal desta atividade é que os alunos elaborem um algoritmo com instruções claras e sem ambiguidades. A necessidade de desenvolver um texto com instruções de forma bem estruturada é justificada pela prática profissional da programação, onde é essencial prever todas as possíveis interpretações de um código por uma máquina, que pode executá-lo de maneira imprevista se houver qualquer ambiguidade. Além disso, programadores muitas vezes criam sistemas que serão usados por clientes e o desafio é torná-los o mais intuitivos possível. Essa tarefa de simplificar e antecipar as ações do usuário nem sempre é fácil!

Nas redes sociais, é muito comum profissionais da programação criarem publicações descontraídas (os famosos *memes*) expressando suas frustrações quando um sistema que parecia intuitivo para eles acaba confundindo um usuário. O programador acredita que o uso do sistema é óbvio, mas encontra usuários que, sem compreender o funcionamento, acabam tentando ações absurdas (do ponto de vista do programador) para realizar uma tarefa simples.

Embora tenhamos comentado sobre a área de Tecnologia da Informação (TI), não concorda que, em qualquer profissão, é fundamental saber se comunicar bem, instruir com clareza e se fazer entender? A capacidade de transmitir instruções claras e precisas é essencial em várias áreas de nossa vida!

Quanto à matemática nesta atividade, observe que espera-se que os alunos utilizem seu conhecimento sobre formas geométricas para construir instruções claras. Sugerimos, no entanto, que os

alunos já possuam esses conceitos de geometria para que possam realizar a atividade sem grandes dificuldades. Embora a atividade também possa funcionar como uma revisão de conteúdos geométricos, o foco principal é desenvolver um algoritmo claro e objetivo.

3.7 Orientações de confecção

O professor pode confeccionar esse material imprimindo as imagens apresentadas na seção Confecção. As imagens estão com as peças misturadas, isto é, sem a solução, então o docente pode solicitar que os próprios alunos recortem para destacar as peças, pois eles não terão acesso à solução do quebra-cabeça sem antes tentar montar por si mesmos.

Pode ser interessante fazer a impressão com folhas sulfites coloridas para deixar mais bonito o material. Apenas lembre-se que **não é recomendado trocar as cores de um mesmo quebra-cabeça** (por exemplo, imprimir o quebra-cabeça do coração em várias cores diferentes para que, depois de recortar, seja trocado peças iguais, mas de cores diferentes, com a intenção de deixar o quebra cabeça todo colorido), para que os alunos não usem a identificação das peças através das cores, o interesse aqui é que os alunos definam as peças pelas suas formas geométricas. Se as cores das peças de um mesmo quebra-cabeça forem diferentes, os alunos certamente irão escrever no algoritmo, por exemplo, “Use a peça amarela” ao invés de escrever “Use a peça em forma de setor circular de 90°”.

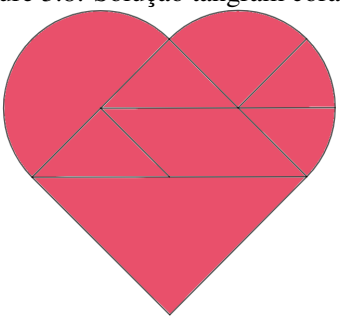
Sugerimos que quebra-cabeças iguais tenham a mesma cor. Por exemplo, todos os quebra-cabeças em forma de coração podem ser impressos em papel sulfite rosa, os quebra-cabeças em forma de triângulo em papel sulfite azul e os quebra-cabeças quadrados de 5 peças em papel sulfite verde. Isso ajuda a identificar rapidamente qual quebra-cabeça cada equipe está montando, evitando que o mesmo quebra-cabeça seja repetido no momento de trocar com outra equipe.

Se preferir, antes de recortar, cole a folha impressa dos quebra-cabeças em um pedaço de papelão (mas que ainda seja fácil de cortar) para deixar as peças mais resistentes, porém isso é opcional.

3.8 Soluções

Nesta seção apresentamos as soluções visuais dos quebra-cabeças e um possível algoritmo que resolve. Note que a escrita do algoritmo não é única, ela serve apenas como base para verificar os detalhes que precisam ser mencionados para não ter ambiguidades de interpretação.

Figure 3.8: Solução tangram coração



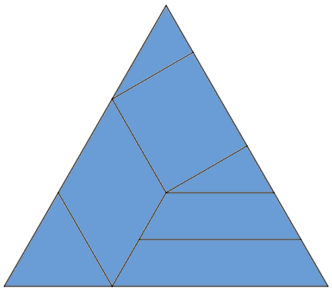
Fonte: Elaborado pela autora

Algoritmo 3.4: Solução do tangram coração

1 A resolução se dá separando o quebra−cabeça em três partes: um quadrado e dois semicírculos.
2 Deixe o semicírculo formado por uma única peça de lado.
3 Forme outro semicírculo com as peças em formato de setor circular: uma peça de setor de 90° e dois setores de 45°.
4 Junte os setores de 45°, formando um setor de 90°.
5 Junte a peça de setor circular de 90° com as peças do passo anterior, formando um semicírculo.
6 Forme um quadrado com as seguintes peças: o triângulo grande, os dois triângulos pequenos e o paralelogramo.
7 Considere quadrado resolvido de forma que suas diagonais sejam paralelas aos lados de uma mesa retangular.
8 Posicione o triângulo maior na parte inferior do quadrado, supostamente resolvido, de forma que o ângulo reto seja coincidente com o vértice do quadrado. E que os catetos desse triângulo sejam os lados adjacentes deste quadrado. Note que a hipotenusa éuma das diagonais do quadrado desejado.
9 Com um dos triângulos menores, mantendo a hipotenusa voltada para baixo, una−o triângulo maior de forma que a hipotenusa do triângulo menor esteja contida na hipotenusa do triângulo maior, ainda que os vértices de ambos os triângulos se coincidam no lado esquerdo.
10 Com a peça em formato de paralelogramo, encaixe de forma que o lado menor coincida com o cateto (direito) do triângulo menor e o lado maior do paralelogramo esteja contido na hipotenusa do triângulo maior.
11 Com o último triângulo pequeno, também de forma que a hipotenusa esteja posicionada para baixo, junte a peça coincidindo a hipotenusa com o lado superior do paralelogramo.
12 Com o quadrado concluído, segue os próximos passos.
13 Pegue o semicírculo de uma peça e encaixe−o no lado superior esquerdo do quadrado, coincidindo o lado do quadrado com o diâmetro do semicírculo.
14 Repita o mesmo processo o outro semicírculo montado com o lado direito superior do quadrado.

Fonte: Elaborado pela autora

Figure 3.9: Solução tangram triângulo



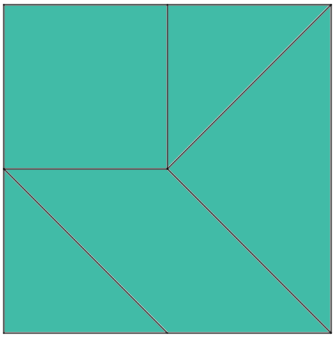
Fonte: Elaborado pela autora

Algoritmo 3.5: Solução - Triângulo

A resolução se dá formando três partes principais do quebra−cabeça, essas partes são trapézios que denominaremos de **trapézios principais**.
Forme o **Trapézio principal 1** usando as duas peças em forma de trapézio:
    Coloque o trapézio menor acima do trapézio maior, de forma que as bases de tamanhos iguais coincidam.
Forme o **Trapézio principal 2** usando as peças em forma de losango e de triângulo equilátero:
    Una um lado do triângulo com um dos lados do losango.
Forme o **Trapézio principal 3** usando as peças restantes, um retângulo e dois triângulos retângulos:
    Una o cateto maior de um dos triângulos com um dos lados menores do retângulo.
    Repita o processo com o outro triângulo, unindo o cateto maior com o outro lado menor do retângulo, de forma que essas três peças formem um trapézio.
Etapa final: Considere o quebra−cabeça resolvido de forma que o triângulo tenha o lado inferior paralelo ao lado inferior de uma mesa retangular.
    Posicione o **Trapézio principal 1** no canto inferior direito da mesa de forma que a base maior esteja voltada para baixo. O segmento desta base será a parte da base do do triângulo resolvido, e o vértice inferior direito dese trapézeio será um vértice do triângulo desejado.
    Posicione o **Trapézio Principal 2** de forma que a base menor coincida com o lado esquerdo (não paralelo) do Trapézio principal 1. Um lado não paralelo do Trapézio principal 2, que se encontra na parte inferior do quebra−cabeça, forma um prolongado da base maior do Trapézio principal 1.
    Posicione **Trapézio Principal 3** de forma que a base menor coincida com o lado não paralelo (superior) do Trapézio Principal 2, enquanto o lado não paralelo (inferior) do Trapézio Principal 3 coincide com a base menor do Trapézio Principal 1.

Fonte: Elaborado pela autora

Figure 3.10: Solução tangram quadrado



Fonte: Elaborado pela autora

Algoritmo 3.6: Solução - Quadrado

1 Considere o quadrado resolvido de forma que seus lados sejam paralelos aos lados de uma mesa retangular.
2 Posicione a peça quadrada no canto superior esquerdo da mesa. Um dos vértices deste quadrado também será o vértice do quadrado desejado, e os lados adjacentes a este vértice farão parte dos lados adjacentes do quadrado desejado.
3 Pegue um dos triângulos menores e junte um dos catetos com o lado direito do quadrado posicionado no passo anterior, de forma que o outro cateto forme um prolongado do lado superior do quadrado.
4 Pegue o triângulo maior e junte um dos catetos com a hipotenusa do triângulo posicionado anteriomente. O vértice que contém o ângulo reto do triângulo maior deve se unir ao vértice inferior direito da primeira peça (o quadrado) e também com um dos vértices do triângulo menor.
5 Pegue o paralelogramo e posicione−o de forma que o lado menor coincida com o lado inferior da peça quadrada, e o lado maior coincida com o cateto do triângulo maior.
6 Posicione o último triângulo pequno no canto inferior esquerdo do quebra−cabeça, de forma que a hipotenusa desse triângulo coincida com o maior lado do paralelogramo.

Fonte: Elaborado pela autora

4. FLUXOGRAMA E PSEUDOCÓDIGO

4.1 Objetivos

Objetivo Geral:

Desenvolver a capacidade de compreender e organizar soluções lógicas usando fluxogramas e pseudocódigos de forma clara e compreensível.

Objetivos específicos

- Compreender fluxogramas;
- Usar estruturas condicionais e de repetição;
- Depurar falhas em algoritmos

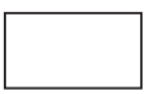
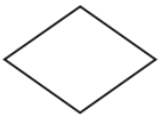
Pré-requisitos

- Essa atividade é voltada para turmas do 9º ano do ensino fundamental em diante.
- Conhecimentos matemáticos abordados nas questões: critérios de divisibilidade; fórmula resolutiva da equação de segundo grau.

4.2 Fluxogramas

Fluxogramas são representações gráficas usadas para visualizar algoritmos. Amplamente utilizados na computação, eles também são comuns em áreas como a administração. Abaixo, veja os símbolos mais utilizados em fluxogramas, que serão aplicados nas próximas atividades:

Figure 4.1: Símbolos para fluxogramas

Símbolo				
Significado	Início ou fim do algoritmo.	Procedimento.	Tomada de decisão.	Sentido de leitura do fluxograma.

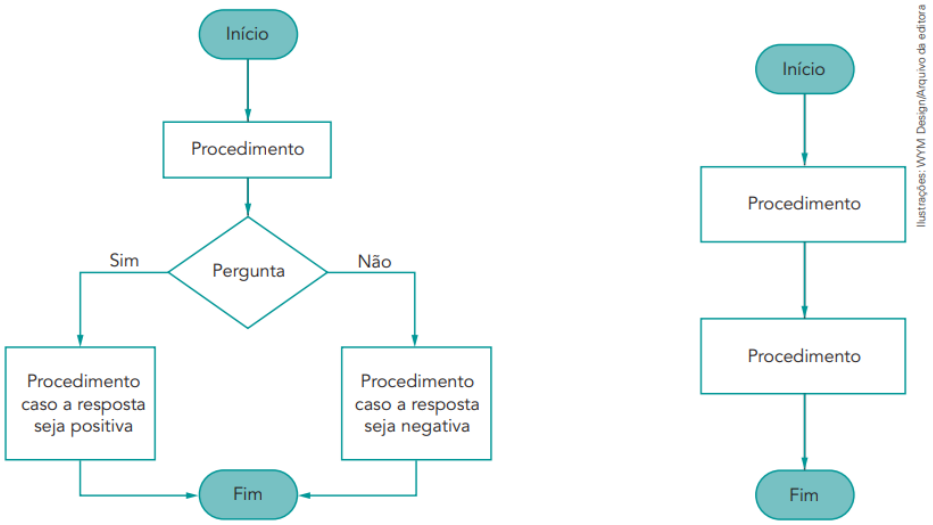
Ilustrações: WYM Design/Arquivo da editora

Fonte: Dante (2020)

A figura a seguir apresenta dois exemplos básicos de estrutura de fluxogramas. Note que o fluxograma apresentado à esquerda possui o símbolo de tomada de decisão, que será definido por uma pergunta. Nos códigos de programas de computador essa tomada de decisão costuma ser denotada pelo comando **se...então**. O procedimento caso a resposta seja negativa é um elemento opcional quando nos referimos às estruturas do comando **se...então**, caso tenha um procedimento para a resposta negativa (como mostra na figura abaixo), ele será acompanhado da cláusula **senão** em seu algoritmo.

A figura a seguir apresenta dois exemplos básicos de estruturas de fluxogramas. Observe que o fluxograma à esquerda inclui o símbolo de tomada de decisão, que é definido por uma pergunta. Nos códigos de programas de computador, essa tomada de decisão é geralmente representada pelo comando se... então. O procedimento para a resposta negativa é um elemento opcional nas estruturas do comando se... então; se houver um procedimento para essa resposta (como mostrado na figura abaixo), ele será acompanhado pela cláusula senão no algoritmo.

Figure 4.2: Estrutura básica de fluxogramas

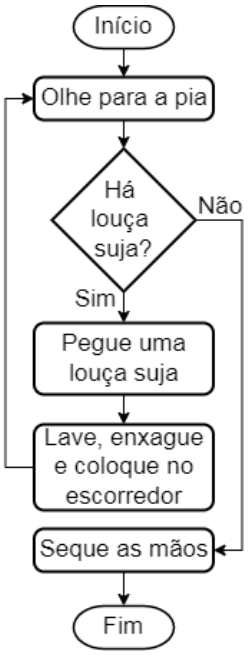


Fonte: Dante (2020)

Questão 1

Veja o fluxograma a seguir que descreve a rotina de lavar a louça.

Figure 4.3: Fluxograma: Lavando louça



Fonte: Elaborado pela autora

Para compreender o fluxograma suponhamos que existem 3 louças na pia para serem lavadas. Veja a sequência de fluxo:

1. Olho a pia;
2. Sim;
3. Pego uma louça;
4. Lavo;
5. Olho a pia;
6. Sim;
7. Pego uma louça;
8. Lavo;
9. Olho a pia;
10. Sim;
11. Pego uma louça;
12. Lavo;
13. Olho a pia;
14. Não;
15. Seco as mãos.

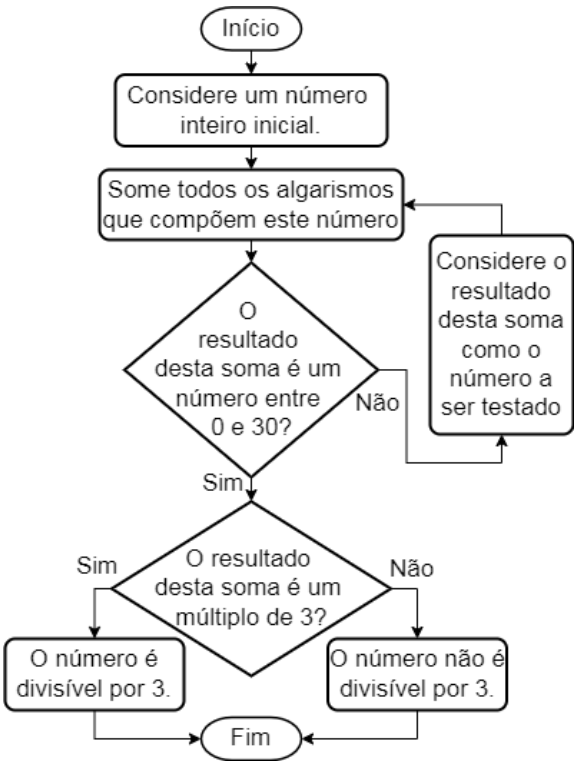
Exercícios: Faça a sequência de fluxo nas seguintes situações:

a) Existe apenas um louça na pia para ser lavada;
b) Existem 4 louças na pia para serem lavadas

Questão 2

Veja o fluxograma a seguir que descreve uma sequência de como determinar se um número natural é divisível por 3 sabendo a tábua do 3 (ou seja, sabendo quais são os múltiplos de 3 entre 0 e 30).

Figure 4.4: Fluxograma: Divisibilidade por 3



Fonte: Elaborado pela autora

Por exemplo, se o número é 2578469 a sequência é:

- Número: 2578469;
- $2+5+7+8+4+6+9=41$;
- Não;
- Número: 41;
- $4+1=5$;
- Sim;
- Não;
- O número 2578469 não é divisível por 3;

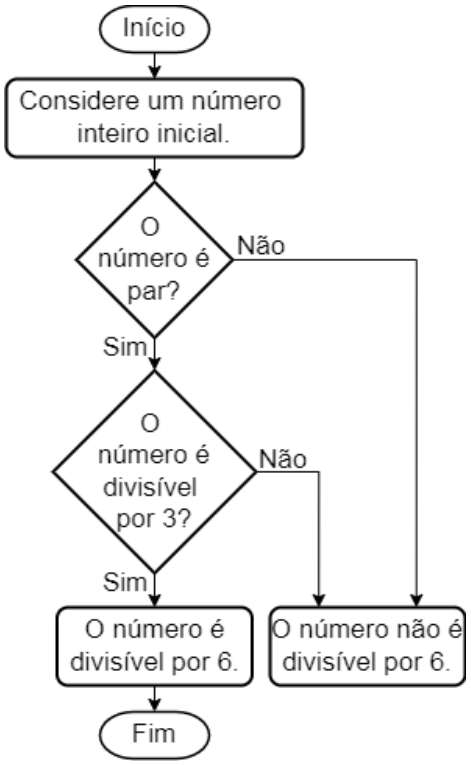
Exercícios: Descreva a sequência gerada a partir do fluxograma para determinar se o número dado é divisível por 3 (como realizado no exemplo acima):

a) 98573469.
b) 2869406.
c) 5768592.

Questão 3

O próximo fluxograma descreve o algoritmo para determinar se um número inteiro é divisível por 6.

Figure 4.5: Fluxograma: Divisibilidade por 6



Fonte: Elaborado pela autora

Exemplo:
O número é 642
Algoritmo:

- Número: 642;
- Sim;
- $6+4+2=12$, Sim;
- O número 642 é divisível por 6;

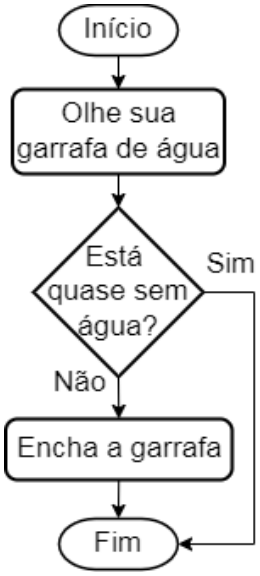
Exercícios
a) O número é 784.
b) O número é 46725.
c) O número é 5768592.
d) O número é 54628.

Questão 4

O fluxograma a seguir apresenta alguns erros. Na programação chamamos de *depuração* o ato de corrigir erros em um código, ou ainda, informalmente, conhecido como *desbugar*, que vem do inglês *bug* popularmente utilizado para informar que há um problema no código, sistema, *hardware*, etc.

Descubra o erro, indicando o que pode acontecer se as instruções forem seguidas conforme é apresentado nos fluxogramas a seguir, e apresente uma sugestão para corrigir.

Figure 4.6: Fluxograma: Decidir se irá encher sua garrafa de água



Fonte: Elaborado pela autora

Há um erro nesse fluxograma. Você consegue descobrir qual?

Responda:

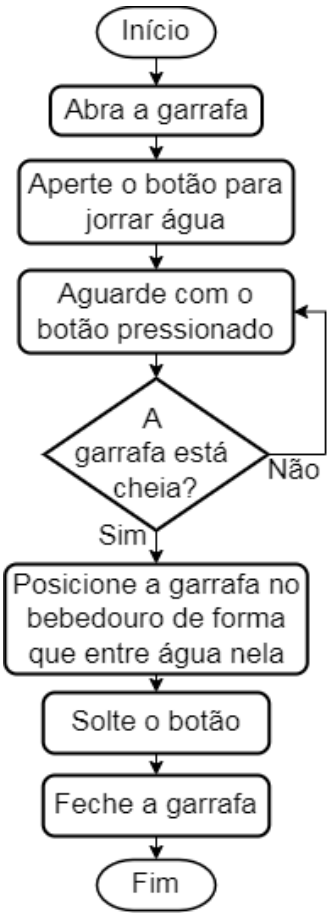
a) O que pode acontecer se alguém seguir as instruções exatamente como se apresentam nesse fluxograma?

b) Apresente uma sugestão para corrigir esse fluxograma.

Questão 5

Na situação do fluxograma abaixo, uma pessoa já está em frente ao bebedouro, com a garrafa em suas mãos.

Figure 4.7: Fluxograma: Enchendo a garrafa de água



Fonte: Elaborado pela autora

Há um erro nesse fluxograma. Você consegue descobrir qual?

Responda:

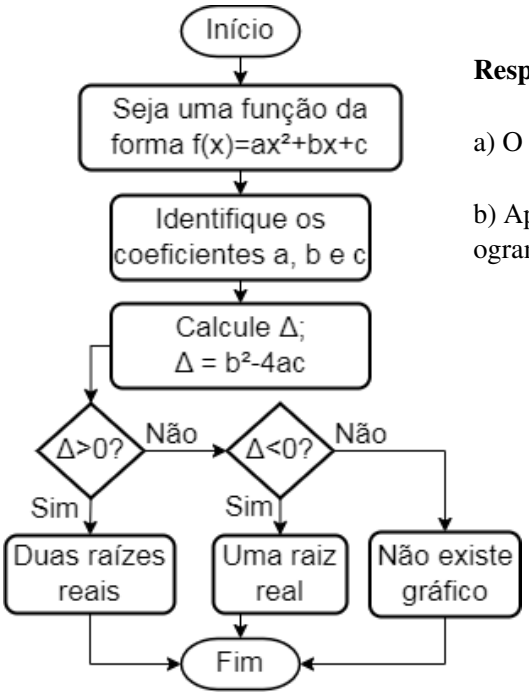
a) O que pode acontecer se alguém seguir as instruções exatamente como se apresentam nesse fluxograma?

b) Apresente uma sugestão para corrigir esse fluxograma.

Questão 6

O fluxograma abaixo representa as instruções para identificar quantas raízes possui uma função de 2º grau.

Figure 4.8: Fluxograma: Quantas raízes tem a função do 2º grau



Fonte: Elaborado pela autora

Há um erro nesse fluxograma. Você consegue descobrir qual?

Responda:

a) O que há de errado nesse fluxograma?

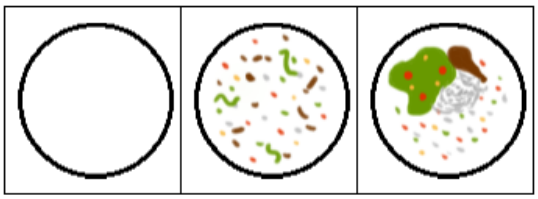
b) Apresente uma sugestão para corrigir esse fluxograma.

4.3 Pseudocódigos

Questão 7

Tina trabalha em um restaurante e sua função é recolher os pratos das mesas para lavá-los. Em determinado momento ela pode encontrar pratos limpos ou sujos. Os pratos limpos, ela apenas higieniza com um pano e um pouco de álcool e depois os guarda nos armários. Os pratos sujos, ela coloca na pia para lavar depois. Porém, há alguns pratos que ainda possuem restos de comida. Esses ela não pode simplesmente colocar na pia para lavar, pois os restos de comida podem entupir o ralo. Para estes pratos, ela deve primeiro esvaziá-los para depois colocá-los na pia.

Apresentamos abaixo a representação desses pratos:



Fonte: Elaborado pela autora

Para essas representações definimos as seguintes identificações:

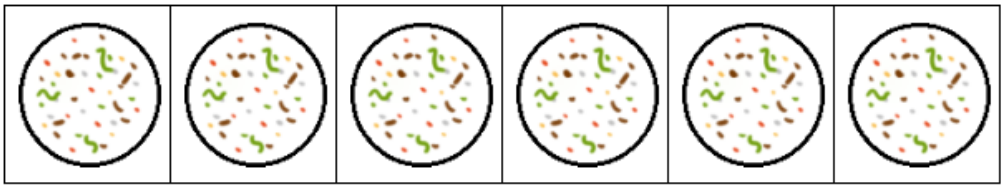
- **Prato limpo:** será denotado por ***limpo***;
- **Prato sujo:** será denotado por ***sujo***;
- **Prato com restos de comida:** será denotado por ***restocomida***.

E teremos os seguintes comandos:

- **guarde**: Higieniza e guarda o prato no armário. Após, segue para o próximo prato à direita;
- **paralavar**: Coloca o prato na pia. Após, segue para o próximo prato à direita;
- **esvazie**: Joga no lixo os restos de comida que estão no prato.

Observe as seguintes sequências, da esquerda para a direita, em que os pratos aparecem e preencha as lacunas dos algoritmos que definem as instruções que a Tina deverá seguir.

a) Para a sequência abaixo



Fonte: Elaborado pela autora

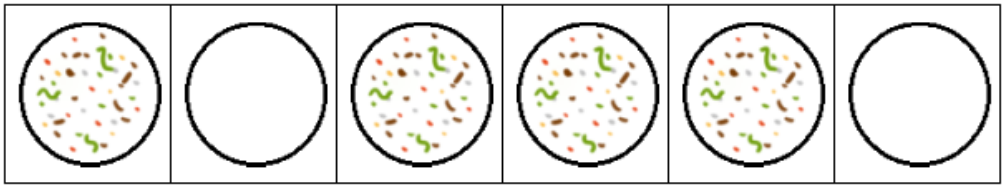
i) Tina deverá seguir o seguinte algoritmo:

```
Início
1     paralavar;
2     paralavar;
3     _____;
4     _____;
5     _____;
6     _____;
Fim
```

ii) Que tal um atalho? O que falta nesse algoritmo para que tenha o mesmo resultado do algoritmo anterior?

```
Início
1     repita __ vezes
2         _____;
Fim
```

b)

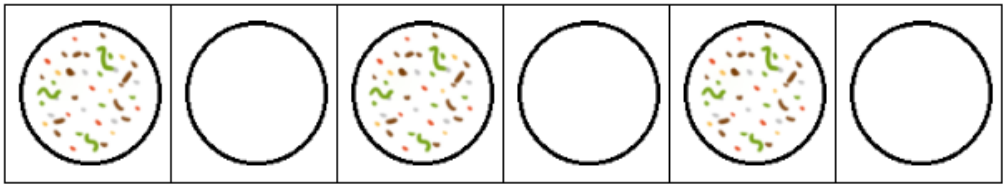


Fonte: Elaborado pela autora

Tina deverá seguir o seguinte algoritmo:

```
Início
1     paralavar;
2     guarde;
3     _____;
4     _____;
5     _____;
6     _____;
Fim
```

c)

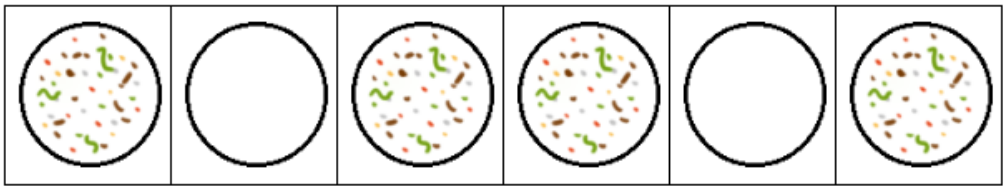


Fonte: Elaborado pela autora

Tina deverá seguir o seguinte algoritmo:

```
Início
1       repita __ vezes
        início
2           ____________;
3           ____________;
        fim
Fim
```

d)

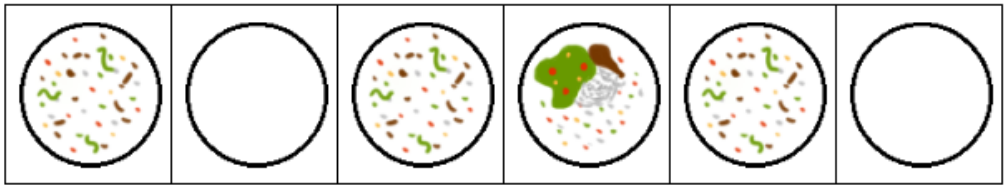


Fonte: Elaborado pela autora

Tina deverá seguir o seguinte algoritmo:

```
Início
1       repita __ vezes
        início
2           _____________;
3           _____________;
4           _____________;
        fim
Fim
```

e)



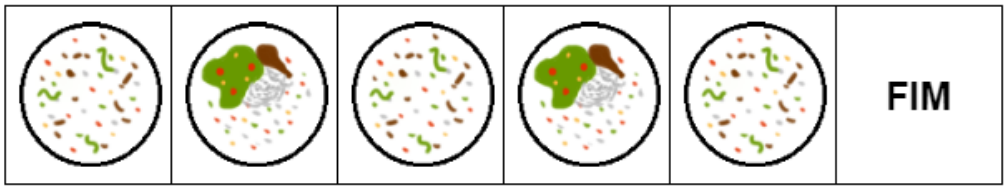
Fonte: Elaborado pela autora

Tina deverá seguir o seguinte algoritmo:

```
Início
1       _____________;
2       _____________;
3       _____________;
4       _____________;
5       _____________;
6       _____________;
7       _____________;
Fim
```

Dica: Lembre que o comando **esvazie** não foi definido que após esvaziar irá colocar o prato na pia e depois seguir para o próximo prato à direita. Então para o mesmo prato deverá instruir dois comandos!

f) Vamos tentar automatizar a decisão do comando?

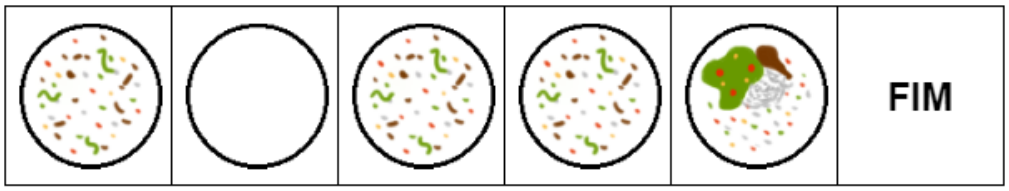


Fonte: Elaborado pela autora

Tina deverá seguir o seguinte algoritmo:

```
Início
1     repita até FIM
      início
2         se sujo então
3             _____;
          senão
4             _____;
5             _____;
          fim se
      fim
Fim
```

g)



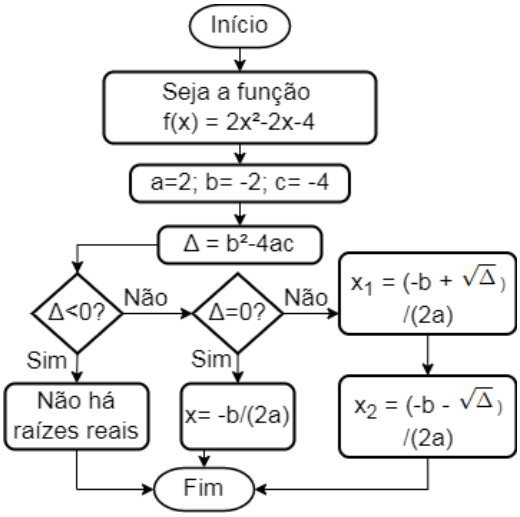
Fonte: Elaborado pela autora

Tina deverá seguir o seguinte algoritmo:

```
Início
1     repita até FIM
      início
2         se restocomida então
3             _____;
4             _____;
          senão
5             se _____ então
6                 _____;
              senão
7                 paralavar;
              fim se
          fim se
      fim
Fim
```

Questão 8

De acordo com o fluxograma a seguir, transcreva-o em algoritmo da forma textual. Para isso preencha as lacunas que faltam do algoritmo.



Fonte: Elaborado pela autora

Algoritmo:

```
var
    a, b, c, delta, x1, x2: real;
Início
1   escreva("Seja a função f(x) = 2x^2 - 2x - 4.");
2   a ← 2;
3   b ← __;
4   c ← −4;
5   delta ← b pot 2 −4∗a∗c;
6   se (delta____) então
7       escreva("Não há raízes reais.");
    senão
8       ____ (delta=0) _________
9           x1 ← −b/(2a);
10          escreva("A única raiz real é x=",x1);

            _______
11          x1 ← (−b+delta raiz 2)/(2a);
12          ____ ← (−b−delta raiz 2)/(2a);
13          ____________("As raízes são x1 =",x1," e x2 =",x2);
        fim se
    fim se
Fim
```

Observação: O comando **escreva** faz com que o computador apresente na tela, o que está escrito

dentro dos parênteses. É uma instrução para que o *computador escreva* para você, usuário, ler. Ou seja, a instrução é para o computador seguir, pois é ele que irá escrever.

4.4 Orientações sobre a atividade para o Professor

Essa atividade pode ser realizada individualmente ou em duplas.

Os alunos podem fazer anotações diretamente na folha impressa (a não ser que o professor solicite o contrário), pois não é necessário que os alunos desenhem os símbolos dos fluxogramas, nem escrevam um código de programa de computador à mão.

As questões são fechadas, para fácil correção a ser realizada pelo professor. Grande parte das questões constituem em preencher lacunas.

A seção Pseudocódigos desta atividade tem a intenção de ensinar os alunos a identificar as estruturas de condição e de repetição em um algoritmo.

Todos os exercícios possuem respostas apresentadas na seção Soluções apresentada a seguir.

4.5 Soluções

Nesta seção apresentamos as respostas dos exercícios propostos acima

Seção - Fluxogramas

Questão 1

Fluxograma: Lavando louça

a)
1. Olho a pia;
2. Sim;
3. Pego uma louça;
4. Lavo;
5. Olho a pia;
6. Não;
7. Seco as mãos.

b)
1. Olho a pia;
2. Sim;
3. Pego uma louça;
4. Lavo;
5. Olho a pia;
6. Sim;
7. Pego uma louça;
8. Lavo;
9. Olho a pia;
10. Sim;
11. Pego uma louça;
12. Lavo;
13. Olho a pia;
14. Sim;
15. Pego uma louça;
16. Lavo;
17. Olho a pia;
18. Não;
19. Seco as mãos.

Questão 2

Fluxograma: Divisibilidade por 3

a) O número é 98573469.

- Número: 98573469;
- $9+8+5+7+3+4+6+9=51$;
- Não;
- Número: 51;
- $5+1=6$;
- Sim;
- Sim;
- O número 98573469 é divisível por 3;

b) O número é 2869406.

- Número: 2869406;
- $2+8+6+9+4+0+6=35$;
- Não;
- Número: 35;
- $3+5=8$;
- Sim;
- Sim;
- O número 2869406 não é divisível por 3;

c) O número é 5768592.

- Número: 5768592;
- $5+7+6+8+5+9+2=42$;
- Não;
- Número: 42;
- $4+2=6$;

- Sim;
- Sim;
- O número 5768592 é divisível por 3;

Questão 3

Fluxograma: Divisibilidade por 6

a) O número é 784.
- Número: 784;
- Sim;
- $7+8+4=19$, Não;
- O número 784 não é divisível por 6;

b) O número é 46725.
- Número: 46725;
- Não;
- O número 46725 não é divisível por 6;

c) O número é 5768592.
- Número: 5768592;
- Sim;
- $5+7+6+8+5+9+2=42$, $4+2=6$, Sim;
- O número 5768592 é divisível por 6;

d) O número é 54628.
- Número: 54628;
- Sim;
- $5+4+6+2+8=25$, Não;
- O número 54628 não é divisível por 6;

Questão 4

Fluxograma: Decidir se irá encher sua garrafa de água

a) A pessoa irá encher a garrafa, mesmo ela já estando cheia. A garrafa irá transbordar. E se a garrafa estiver quase sem água, a pessoa não irá encher a garrafa e ficará sem água e com sede.

b) Trocar de lugar a palavra *Sim* com a palavra *Não* e vice-versa.

Questão 5

Fluxograma: Enchendo a garrafa de água

a) A pessoa apenas irá apertar o botão pra jorrar água sem colocar a garrafa no local adequado. Como a garrafa nunca irá encher, a pessoa continuará apertando o botão eternamente, desperdiçando água.

b) Colocar a caixa que diz "Posicione a garrafa no bebedouro de forma que entre água nela" entre as caixas "Abra a garrafa" e "Aperte o botão para jorrar água".

Questão 6

Fluxograma: Quantas raízes tem a função do 2º grau

a) A última caixa que diz "Não existe gráfico" está com um erro teórico sobre funções de 2º grau. Quando Δ for negativo, existe gráfico, sim.

b) Substituir o texto "Não existe gráfico" por "Não existem raízes reais".

Questão 7

a) i) Algoritmo:

```
Início
1     paralavar;
2     paralavar;
3     paralavar;
4     paralavar;
5     paralavar;
6     paralavar;
Fim
```

ii) Algoritmo:

```
Início
1     repita 6 vezes
2         paralavar;
Fim
```

b) Algoritmo:

```
Início
1     paralavar;
2     guarde;
3     paralavar;
4     paralavar;
5     paralavar;
6     guarde;
Fim
```

c) Algoritmo:

```
Início
1     repita 3 vezes
      início
2         paralavar;
3         guarde;
      fim
Fim
```

d) Algoritmo:

```
Início
1     repita 2 vezes
      início
2         paralavar;
3         guarde;
4         paralavar;
      fim
Fim
```

e) Algoritmo:

```
Início
1     paralavar;
2     guarde;
3     paralavar;
4     esvazie;
5     paralavar;
6     paralavar;
7     guarde;
Fim
```

f) Algoritmo:

```
Início
1     repita até FIM
      início
2         se sujo então
3             paralavar;
          senão
4             esvazie;
5             paralavar;
          fim se
      fim
Fim
```

g) Algoritmo:

```
Início
1     repita até FIM
      início
2         se restocomida então
3             esvazie;
4             paralavar;
          senão
5             se limpo então
6                 guarde;
              senão
7                 paralavar;
              fim se
          fim se
      fim
Fim
```

Questão 8

```
var
    a, b, c, delta, x1, x2: real;
Início
1   escreva("Seja a função f(x) = 2x^2 − 2x − 4.");
2   a ← 2;
3   b ← -2;
4   c ← −4;
5   delta ← b pot 2 −4∗a∗c;
6   se (delta<0) então
7       escreva("Não há raízes reais.");
    senão
8       se (delta=0) então
9           x1 ← −b/(2a);
10          escreva("A única raiz real é x=",x1);
        senão
11          x1 ← (−b+delta raiz 2)/(2a);
12          x2 ← (−b−delta raiz 2)/(2a);
13          escreva("As raízes são x_1 =",x1," e x_2 =",x2);
        fim se
    fim se
Fim
```