PENSAMENTO COMPUTACIONAL

Um Guia Prático para Pensar Como um Computador



Gabrieli Jorge Barbosa
Marta Adriana Machado da Silva
Tatiana Nilson dos Santos
Olga Yevseyeva
Eliane Pozzebon

UFSC

LabTeC
Laboratório de Tecnologias Computacionais

PENSAMENTO COMPUTACIONAL

Um guia prático para pensar como um computador

Realização:

LabTeC - UFSC | Laboratório de Tecnologias Computacionais

Área de Conhecimento:

Ciência da Computação
Engenharias e Tecnologia
Linguagens e Comunicação

1ª Edição
Araranguá,
2026

Gabrieli Jorge Barbosa
Marta Adriana Machado da Silva
Tatiana Nilson dos Santos
Olga Yevseyeva
Eliane Pozzebon

PENSAMENTO COMPUTACIONAL

Um guia prático para pensar como um computador

Idealização e Conteúdo
Gabrieli Jorge Barbosa
Marta Adriana Machado da Silva
Tatiana Nilson dos Santos
Olga Yevseyeva
Eliane Pozzebon

Projeto Gráfico e Experiência Visual
Tatiana Nilson dos Santos
Ananda Muxfeldt Palma

Revisão Técnica
Olga Yevseyeva

Revisão Gramatical e Ortográfica
Karla Goularte da Silva Gründler

Apoio técnico
Débora Maria Russiano Pereira

Barbosa, Gabrieli Jorge.
Pensamento Computacional: um guia prático para pensar como um computador / Gabrieli Jorge Barbosa, Marta Adriana Machado da Silva, Tatiana Nilson do Santos / Olga Yevseyeva / Eliane Pozzebon. - Araranguá : LabTeC; UFSC, 2026.
125 p. : il.

ISBN: 978-65-02-14644-6

1. Pensamento Computacional. 2. Computadores. 3. Educação. I. LabTeC. II UFSC. III. Silva, Marta Adriana Machado da. IV. Santos, Tatiana Nilson dos. V. Yevseyeva, Olga. VI. Pozzebon, Eliane. VII. Título

Ficha catalográfica elaborada por: Debora Maria Russiano Pereira, CRB-14/1125

Apresentação

Em um mundo cada vez mais conectado, aprender a pensar bem tornou-se tão importante quanto aprender a usar tecnologias. O Pensamento Computacional nos ajuda exatamente nisso: organizar ideias, resolver problemas e encontrar soluções de forma lógica, criativa e eficiente.

Mais do que programação ou computadores, esse tipo de pensamento está presente no dia a dia para resolver um desafio.

Este e-book convida você a explorar o Pensamento Computacional de maneira acessível e prática, com exemplos claros e atividades que estimulam o raciocínio e a curiosidade. A proposta é aprender fazendo, refletindo e experimentando.

A proposta aqui não é formar programadores, mas sim pensadores: pessoas capazes de analisar problemas, propor soluções e compreender melhor o mundo digital e tecnológico em que estão inseridas.

Ao desenvolver o pensamento computacional, ampliamos não apenas competências técnicas, mas também cognitivas e sociais, fundamentais para a educação do século XXI.

Que esta leitura seja um convite à curiosidade, à experimentação e à descoberta. Que cada capítulo inspire novas formas de pensar, ensinar e aprender — com lógica, criatividade e propósito.

As autoras

Sumário



CAPÍTULO 01

Pensando como um Computador

1.1. Festa Algorítmica 8
1.2. Origem do Termo 10
1.3. Importância do Pensamento Computacional 13
1.4. Guia para o professor 15
1.5. Sugestão de atividades 17

CAPÍTULO 02

Pilar do Pensamento Computacional: decomposição

2.1. Dividir para conquistar 22
2.2. Plano de ação 23
2.3. Guia para o professor 26
2.5. Sugestão de atividades 29

CAPÍTULO 03

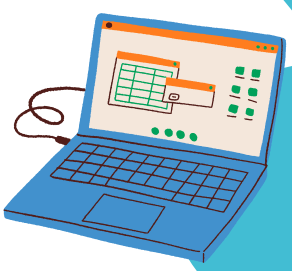
Pilar do Pensamento Computacional: reconhecimento de padrões

3.1. Eu já vi isso antes 34
3.2. Plano de ação 37
3.3. Guia para o professor 39
3.4. Sugestão de atividades 42

CAPÍTULO 04

Pilar do Pensamento Computacional: abstração

4.1. Focando no que realmente importa 47
4.2. Plano de ação 51
4.3. Guia para o professor 55
4.4. Sugestão de atividades 57



Pilar do Pensamento Computacional: pensamento algorítmico

CAPÍTULO 05

5.1. Criando a receita do sucesso 62
5.2. Plano de ação 66
5.3. Guia para o professor 69
5.4. Sugestão de atividades 71

Princípios da lógica computacional

CAPÍTULO 06

6.1. O enigma da porta misteriosa 77
6.2. Conceitos fundamentais da computação 78
6.3. Eixos da lógica computacional 78
6.3.1. Sequência 78
6.3.2. Decisão 79
6.3.3. Repetição 80
6.4. Conectando os três princípios 82
6.5. Guia para o professor 83
6.6. Sugestão de atividades 84

Representações da lógica computacional

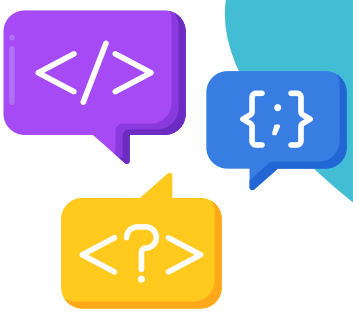
CAPÍTULO 07

7.1. Algoritmos narrativos 90
7.2. Pseudocódigo 91
7.3. Fluxograma 94
7.4. Lógica de blocos 95
7.5. Guia para o professor 97
7.6. Sugestão de atividades 99

Introdução ao Scratch

CAPÍTULO 08

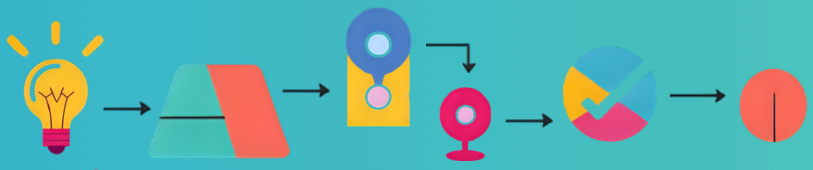
8.1. O que é Scratch? 102
8.2. Conhecendo a ferramenta 104
8.3. Casos reais na educação 107
8.4. Vamos praticar 108
8.5. Guia para professor 114
8.6. Sugestão de atividades 116



Reflexão final 120
Referências 120
Links uteis 121
Sobre as autoras 122



Capítulo 1:
Pensando como um Computador



TEMA 1

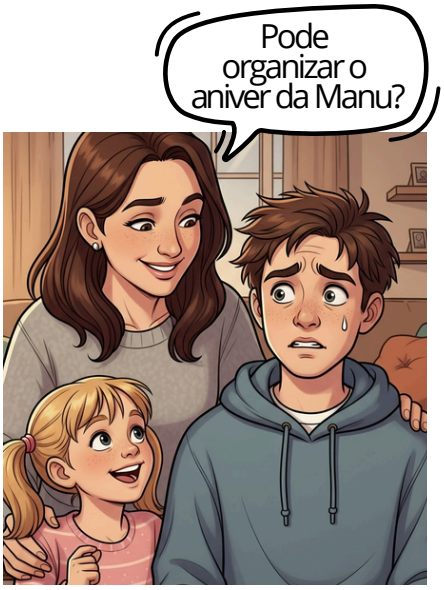
Por dentro da BNCC:

Habilidades:
EF01CO01
EF01CO02
EM13CO01

Antes de escrever uma linha de código ou usar qualquer tecnologia, existe algo ainda mais importante: a forma como pensamos. Pensar como um computador não significa agir como uma máquina, mas aprender a organizar ideias, analisar problemas e criar soluções de maneira lógica e estruturada.

Neste capítulo, você vai entender o que é o Pensamento Computacional, por que esse conceito é tão importante e como ele pode ser aplicado muito além do mundo da programação, ajudando a resolver desafios do dia a dia com mais clareza, eficiência e criatividade.

1. 1. Festa Algorítmica:



Imagine o seguinte:

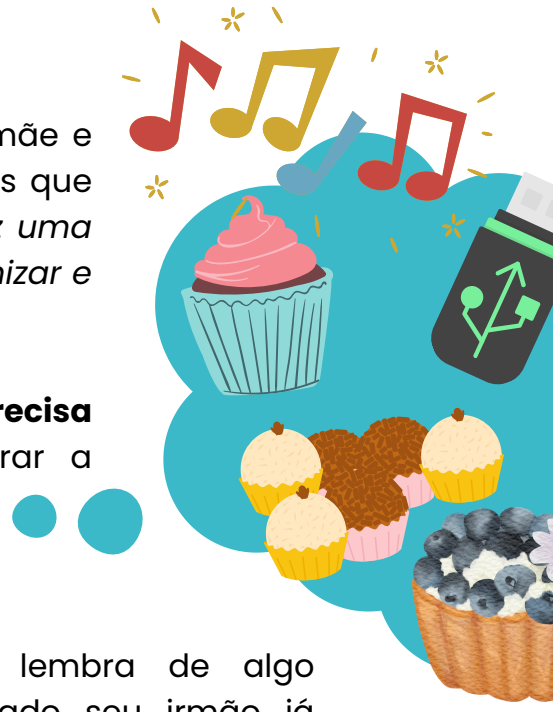
é o aniversário da sua irmãzinha mais nova, e sua mãe te deu a grande missão de planejar a festa. Não é só qualquer festa: ela quer que seja especial, cheia das coisas que sua irmã mais gosta — a decoração com as cores favoritas dela, as músicas que ela canta o dia inteiro e, claro, o famoso bolo que é tradição da família.

Mas aí surge a pergunta: **POR ONDE COMEÇAR?**



No primeiro momento, parece uma tarefa gigante, quase impossível para uma pessoa só. Você pensa em pedir ajuda: liga para sua avó, mas ela diz que está indo para a aula de ginástica; tenta falar com seu irmão, mas ele está no meio de uma partida de videogame e diz que *"é rapidinho"*... só que a gente sabe que não é.

Resultado: **parece que você vai ter que dar conta dessa missão sozinho.**



Bate aquele desespero. Você até considera ir até sua mãe e dizer: *"Não vai dar... é muita coisa pra fazer"*. Mas, antes que você termine a frase, ela sugere: *"Por que você não faz uma lista de tudo que precisa? Assim fica mais fácil de organizar e resolver um problema de cada vez."*

Bingo! Você pega papel e caneta e anota tudo o que precisa fazer: comprar o bolo, escolher as músicas, preparar a decoração, enviar os convites...



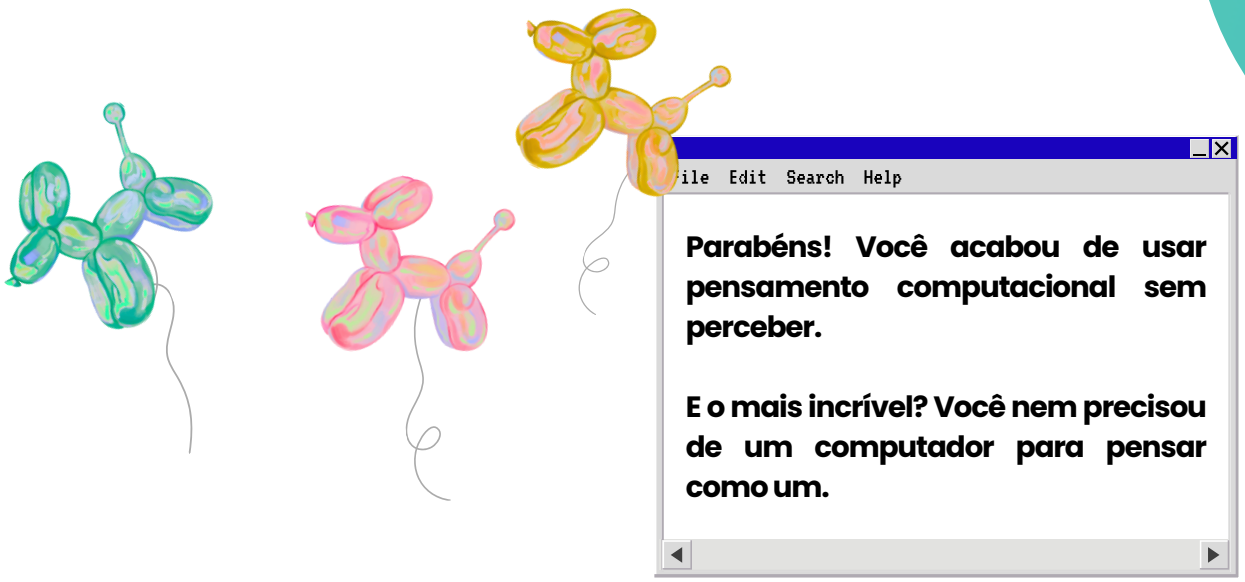
Enquanto escreve, você lembra de algo importante: no ano passado seu irmão já montou um *pen drive* com as músicas favoritas da família, então um problema a menos. Logo em seguida, recorda que seu avô sempre se encarrega das guloseimas. Mais um item que sai da sua lista.

Essas lembranças te mostram um padrão: **nem tudo precisa ser feito do zero.** Você pode reaproveitar soluções já prontas e focar no que realmente é mais urgente — como escolher o tema da festa, definir o orçamento e decidir a lista final de convidados.

Com essa clareza, você define a ordem das tarefas: primeiro decidir o tema, depois calcular os gastos, em seguida comprar o bolo, e por aí vai. **Monta um passo a passo para que tudo aconteça no tempo certo**. E, claro, prepara planos alternativos caso algo dê errado — porque festa sem imprevisto não existe.

Ao final, o grande dia chega. A decoração está impecável, as músicas animam todo mundo, e o bolo... bom, o bolo acaba em poucos minutos. **A festa é um sucesso.**





Talvez você até já tenha agido desta forma antes sem sequer saber ou talvez você já se encontrou nessa mesma situação e acabou desistindo da tarefa por se frustrar com o peso da responsabilidade ou com possíveis problemas que surgiram no caminho. **É aí que entra o pensamento computacional.**

Essa forma estruturada e criativa de resolver problemas, diferentemente do que a maioria pensa, não é sobre decorar códigos de programação, aprender a programar, ser bom em matemática ou decorar comandos do computador, mas **sim sobre pensar passo a passo, identificar padrões, simplificar o que é complexo e criar soluções eficientes.**

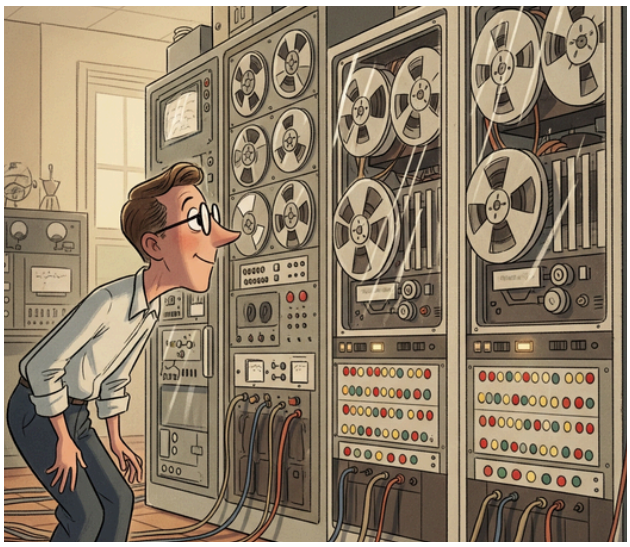
Definição simples: Pensar computacionalmente é organizar as ideias de um jeito que até um computador entenderia, mesmo que você não use um computador para resolver o problema.

1. 2. Origem do Termo:

Lá nos Alpes suíços, um biólogo muito sábio chamado ***Jean Piaget*** passava seus dias tentando desvendar um dos maiores mistérios do universo: **como as crianças aprendem?** Para espiar o universo dentro da cabeça de uma criança, suas ferramentas eram simples e poderosas: bloquinhos de madeira, massinha e a curiosidade por trás de cada pergunta.



Piaget descobriu algo incrível: **as crianças não são "adultos em miniatura"**. Elas aprendem de um jeito único, construindo o conhecimento com as próprias mãos, errando, tentando de novo e, aos pouquinhos, montando o quebra-cabeça do mundo.



Do outro lado do oceano, no agitado MIT - Instituto de Tecnologia de *Massachusetts* - nos Estados Unidos, havia um jovem matemático e cientista da computação chamado ***Seymour Papert***. Ele era fascinado por duas coisas: o poder dos computadores, que eram, no final dos anos 50, máquinas gigantes e misteriosas, e as ideias revolucionárias daquele sábio suíço.

Papert pensava assim: ***"Se as crianças aprendem construindo coisas no mundo real, o que aconteceria se a elas também pudessem construir coisas no mundo digital?"***

Movido por essa ideia, ele fez as malas e foi para a Suíça trabalhar com seu ídolo, *Jean Piaget*. E foi aí que muita coisa interessante começou a acontecer. Em meio a longas conversas, cafés e passeios, o psicólogo que entendia do desenvolvimento das crianças e o matemático interessado nas possibilidades das máquinas foram trocando ideias — e dessas trocas nasceram reflexões que ajudaram a transformar a maneira de pensar a aprendizagem.

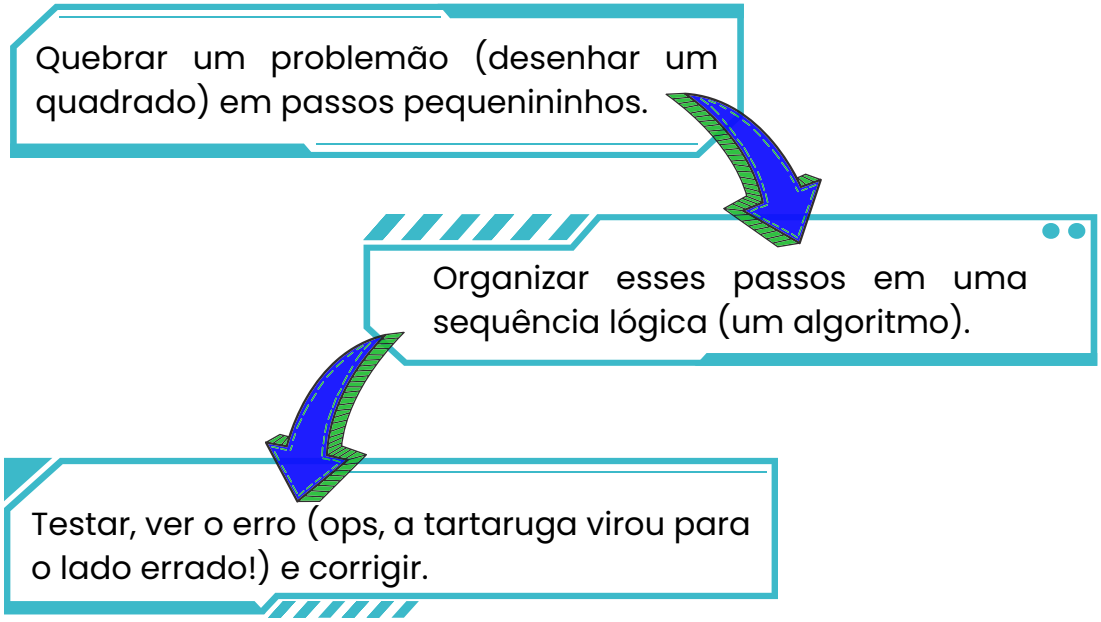
Piaget falava sobre como uma criança organiza blocos para fazer um castelo, e ***Papert*** via nisso uma sequência de passos, uma lógica, quase um programa de computador.



A sacada genial de *Papert*, inspirada por *Piaget*, foi de usar programação como ferramenta de construção, como os blocos que seu instrutor utilizava, para montar o mundo que elas ainda descobriam ao seu redor.

Ele criou então uma linguagem de programação super simples chamada LOGO. Com ela, qualquer criança podia dar comandos para uma tartaruguinha na tela desenhar o que quisesse. Para desenhar um quadrado, por exemplo, a criança não precisava ser um gênio da matemática. Bastava "ensinar" a tartaruga: *"ande para frente, vire 90 graus, ande para frente, vire 90 graus..."*.

Ao fazer isso, a criança era estimulada a pensar de um jeito diferente:



Papert percebeu que o mais importante não era o desenho na tela, mas o que acontecia na cabeça da criança. **Ela estava aprendendo a resolver problemas de forma estruturada, criativa e lógica.**

Essa amizade entre *Piaget* e *Papert* plantou uma sementinha. Eles mostraram que a lógica dos computadores não precisava ser algo frio e distante, mas sim uma ferramenta poderosa para a mente humana.

Muitos anos depois, uma outra cientista, chamada ***Jeannette Wing*,** olhou para tudo isso e deu o nome perfeito para essa habilidade: **Pensamento Computacional.**

E é isso que ele é até hoje: não se trata de ser um computador, mas de ter a capacidade humana de resolver problemas de forma inteligente, um superpoder que nasceu da amizade de um psicólogo e um matemático.

1.3. Importância do pensamento computacional

"Se eu não quero trabalhar como programador, por que eu preciso saber como um computador 'pensa'?"

Essa é a pergunta do milhão. E a resposta vai muito além de códigos e conexões eletrônicas. O Pensamento Computacional não está na ideia de transformar você em uma máquina, muito pelo contrário, é sobre garantir que você não seja substituído por uma.

Há um prejulgamento perigoso e limitante de que esse assunto é exclusivo para quem trabalha com tecnologia, cria jogos ou hackeia sistemas. Isso é como dizer que aprender a escrever só serve para quem quer ser escritor renomado. A escrita é uma ferramenta de comunicação. O Pensamento Computacional é uma ferramenta de **resolução de problemas**.

Estamos em um período de mudanças radicais que permeiam todas as áreas da vida humana. Muito em breve, funções que dependem de tarefas repetitivas serão feitas por máquinas. Até mesmo a antiga missão de orientar caminhos — onde um viajante decifra um mapa e recorre a necessidade de pedir informações — foi, há muito, substituída pela voz automatizada dos aplicativos de navegação.

Você sabia que...



Até meados do século XX, o termo 'computador' descrevia uma profissão— exercida majoritariamente por mulheres —, cuja função era fazer cálculos matemáticos complexos à mão para a NASA e grandes projetos de engenharia.

A revolução digital não substituiu a inteligência delas, mas automatizou a exaustão: o que uma sala de "computadoras humanas" levava dias para processar com lápis e papel, os primeiros *mainframes* passaram a resolver em segundos, sem precisar de pausa para o café.

São nesses momentos que, para nós humanos, sobram a capacidade de criar, de resolver problemas e de pensar criticamente!

Se há 100 anos, ser alfabetizado significava saber ler e escrever. Hoje, isso não basta. Em um mundo onde a tecnologia nos rodeia em um mar de dados e algoritmos, se você não entende como esse mundo digital é construído, você se torna um náufrago passivo dele.

O Pensamento Computacional é o “ler e escrever” da nossa era e dominá-lo é uma questão de sobrevivência e cidadania. Ele permite que você deixe de ser apenas um consumidor de tecnologia e se torne um cidadão crítico, capaz de compreendê-la e, possivelmente, também de criá-la e usá-la a seu favor.

Facilitar o acesso a esse conhecimento é uma forma poderosa de reduzir desigualdades. Afinal, ao ser dominado pela tecnologia, o indivíduo passa a ser visto pelo mercado de trabalho como desatualizado e acaba perdendo grandes oportunidades.

Quando um estudante aprende a decompor problemas e criar algoritmos, ele ganha a mesma caixa de ferramentas mentais que um engenheiro da Nasa. Isso nivela o jogo. Como aponta a ONU, essa é uma habilidade essencial para garantir uma educação de qualidade.

Ao desenvolver o Pensamento Computacional, você está, essencialmente, fazendo um *upgrade* no sistema operacional do seu cérebro. Ao aprender a decompor um problema, identificar padrões, abstrair detalhes, e traçar um plano, você ganha o poder de olhar para uma dificuldade da sua realidade — seja o lixo na rua, a falta de organização no grêmio estudantil, a gestão do seu próprio dinheiro, ou como vimos, uma festa de aniversário — e gerar uma solução eficiente. Você deixa de ser quem só reclama do problema e passa a ser quem projeta a melhoria.

Neste e-book, o foco não é decorar comandos de computador. Vamos focar nas ferramentas que você pode usar para operar a máquina mais complexa e poderosa que existe, a sua mente, a todo o vapor!



A tecnologia é "operária".
Você é o "arquiteto".

1.4. Guia para o professor

Neste capítulo fizemos a introdução ao Pensamento Computacional. Ainda não vamos programar nem decorar termos técnicos, mas apenas compreender a mudança de mentalidade: deixar de ser um usuário passivo da tecnologia para se tornar um criador ativo de soluções.

Objetivos de aprendizagem

- Compreender a definição de Pensamento Computacional e sua origem.
- Identificar resolução de problemas como uma habilidade humana, e não uma habilidade exclusiva de máquinas.
- Diferenciar "usar o computador" de "pensar computacionalmente".
- Relacionar a história lúdica do capítulo com a ideia de ferramentas para a resolução de problemas.

Roteiro sugerido de aula (aproximadamente 50 min)

- **Abertura:** apresente a história da "Festa Algorítmica", em que um jovem precisa planejar a festa de aniversário da irmã, mas não sabe por onde começar.

- **Pergunte:**

  "Qual foi o momento em que as coisas começaram a dar certo na história?"

- **Explique:** ao se limitar, pensando que não conseguiria, o personagem trava. Mas, quando começa a listar as tarefas e a pensar computacionalmente sobre elas, tudo passa a fazer sentido, mesmo sem conhecer ainda todas as ferramentas.

- **Exploração:** é comum se encontrar em uma situação dessas e não saber o que fazer? Apresente brevemente os "Pais da Ideia".

  - Conte que um cientista chamado Seymour Papert queria que crianças ensinassem computadores, não o contrário.
  - Fale sobre Jean Piaget e seu interesse pelo método de aprender fazendo.
  - Cite Jeannette Wing, explicando que ela disse que essa habilidade serve não só para programadores.

- **Desenvolvimento:** a dinâmica da tartaruga humana
  - Crie um trajeto no chão com fita crepe ou obstáculos. Coloque um "tesouro" no final.
  - Divida a turma em duplas.
  - Um aluno é o Programador.
  - O outro aluno é a Tartaruga, que deve estar vendada.
  - A Tartaruga deve obedecer aos comandos exatos do Programador. *"Passo para frente"*, *"Gire 90 graus para a direita"*. E não pode desviar de um obstáculo sem instrução.

- **Visualização:** após a atividade, destaque que a experiência realizada reflete exatamente a proposta de Piaget e Papert: aprender colocando a mão na massa.

"Notem como o pensamento veio da cabeça do Programador, mas a execução acontece no corpo da Tartaruga. Isso é Pensamento Computacional: ver os problemas como coisas que podem ser resolvidas, até de olhos vendados, e que pode ser projetado."

- **Socialização:** peça que os grupos relatem como foi dar ordens tão precisas e seguir a instruções cegamente. Pergunte como eles acham que poderiam melhorar as suas instruções.

- **Reflexão final:** conduza uma conversa baseada na visão de Jeannette Wing: *"se vocês usarem essa clareza de pensamento para organizar os estudos ou planejar uma festa, vocês estão usando Pensamento Computacional, mesmo sem nenhum computador ligado na tomada."*

Avaliação

- Observe se os alunos entenderam a separação de funções: o programador planeja, a máquina executa.
- Verifique se, durante a brincadeira, os alunos perceberam a necessidade de serem literais, lógicos e precisos.
- Promova uma autoavaliação reflexiva, perguntando:

"Vocês conseguiam perceber a diferença entre 'pedir para um amigo pegar água' (que tem inteligência) e 'programar um robô para pegar água' (que precisa de instruções exatas)?"

1.5. Sugestão de Atividades



Atividade 1 - o idioma da tartaruga (20 a 25 minutos)

- **Objetivo:**
  - O aluno deve vivenciar a dificuldade de se comunicar com uma máquina e traduzir um pensamento humano para um sistema com regras rígidas, visualizando a necessidade de estruturar o pensamento.

- **Recursos necessários:**
  - Folhas de papel quadriculado
  - Canetas/lápis

- **Passo a passo**
  - Separar a turma em duplas e distribuir folhas quadriculadas para todos.
  - O professor explica a grande limitação: a tartaruga não fala português, ela entende **apenas 3 comandos exatos**. Ela não tem bom senso e levará tudo ao pé da letra.
    - **"FRENTE"** (Avança e desenha uma linha reta do tamanho de 1 quadrado, na direção para a qual está olhando).
    - **"DIREITA"** (A Tartaruga apenas gira o seu corpo 90 graus para a direita, mas não sai do lugar nem desenha nada).
    - **"PULA"** (A Tartaruga avança 1 quadrado para frente, mas sem desenhar nada — deixa um espaço em branco).
  - A tartaruga **NÃO entende a palavra "ESQUERDA"**, nem a palavra "VOLTA".
  - O professor entrega ao arquiteto (escondido da tartaruga) o objetivo: ele deve dar os comandos para que a tartaruga desenhe a **letra "S"** , uma **escada** ou qualquer outro desenho na folha quadriculada.
  - Se o arquiteto usar qualquer palavra fora do dicionário a tartaruga deve cruzar os braços e dizer "Erro no Sistema", obrigando o arquiteto a recomeçar a instrução daquele ponto.

- **Debate em sala**

  - Após concluírem a atividade, os alunos são convidados a partilhar as suas frustrações e sucessos. O(A) professor(a) deverá questionar a turma:
    - Para o arquiteto foi difícil controlar alguém que não preenche as lacunas por você? Percebeu que precisava "quebrar" o caminho em passos minúsculos? Como vocês resolveram o problema de fazer a tartaruga virar para a *esquerda*, já que esse comando não existia?
    - Para a tartaruga, como foi depender totalmente de instruções claras? Vocês sabiam o que o arquiteto queria desenhar desde o começo ou só descobriram no final?
    - O texto de Seymour Papert diz que a criança *"testa, vê o erro e corrige"*. Quando vocês mandaram a tartaruga virar para o lado errado, precisaram apagar e refazer o pensamento? Como esse exercício prova que nós somos a inteligência criativa e a máquina é apenas a força de execução?



Atividade 2 - O sistema operacional da turma (20 a 25 minutos)

- **Objetivo:**

  - O aluno é convidado a refletir sobre como problemas complexos do seu dia a dia já são resolvidos usando o Pensamento Computacional sem que eles se apercebam.

- **Recursos necessários:**

  - Lousa
  - Canetas ou giz
  - Folhas A4

- **Passo a passo**

  - Organizar a turma em grupos de 3 ou 4 integrantes.
  - O professor lança o desafio: "no capítulo, vimos que organizar uma festa de aniversário exige pensamento computacional. Vamos fazer a analisar um problema da nossa realidade: organizar uma feira de ciências na escola."

- Os grupos têm 10 minutos para agir como computadores humanos e fazer a engenharia reversa deste problema, escrevendo na folha como aplicariam as ferramentas mentais mencionadas no texto:
    1. **Decompor**: quais as grandes tarefas?
    2. **Identificar Padrões:** o que correu mal na feira do ano passado que não queremos repetir? O que podemos reaproveitar?
    3. **Abstrair:** o que é urgente e o que é perda de tempo agora?
    4. **O Algoritmo:** qual é a ordem exata em que as coisas têm de acontecer no dia da feira?
- Os grupos trocam os seus planejamentos para ver se outra equipe se esqueceu de um passo fundamental (ex.: *"esqueceram-se de pedir autorização ao diretor!"*).

- **Debate em sala**
    - Ao colocar o "problemão" no papel e dividí-lo em pequenas partes, o desespero de organizar a feira diminuiu?
    - Conseguem perceber que acabaram de fazer um *upgrade* no seu "sistema operacional mental"?
    - De que forma fazer isto os transforma em criadores de soluções (arquitetos) em vez de pessoas que apenas reclamam dos problemas da escola?



Atividade 3 - Consumidor vs. criador (20 a 25 minutos)

- **Objetivo:**
    - O aluno é convidado a refletir criticamente sobre o impacto social da tecnologia, debatendo que dominar a tecnologia é uma questão de cidadania, sobrevivência e redução de desigualdades.

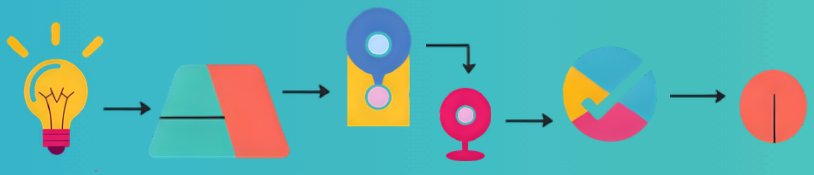
- **Recursos necessários:**
    - Quadro
    - Marcador ou giz

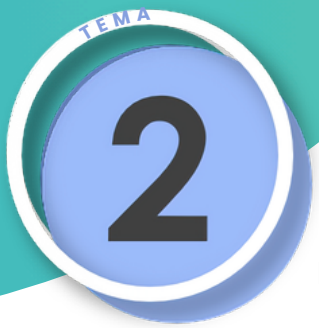
- **Passo a passo**
  - Organizar a turma em dois grandes grupos: "os otimistas da tecnologia" e "os críticos da automação".
  - Apresentação do problema: o texto do tópico “Importância do Pensamento Computacional” diz que as máquinas vão fazer todos os trabalhos repetitivos. Imaginem que, daqui a 10 anos, uma máquina consiga fazer hambúrgueres, conduzir automóveis e até limpar as ruas perfeitamente.
  - Cada um dos grupos terá 5 minutos para defender a sua ideia com argumentos.
  - **Os otimistas:** liberta o ser humano da exaustão, as pessoas terão tempo para usar a mente para criar arte, ciência e novas tecnologias.
  - **Os críticos:** vai gerar desemprego para quem não tem o "pensamento computacional" desenvolvido e tem trabalhos repetitivos; as pessoas que apenas sabem consumir tecnologia ficarão para trás; aumento da desigualdade social.

- **Debate em sala**
  - O(A) professor(a) deverá atuar como mediador, lançando provocações para a turma.
  - O texto diz que se vocês não entenderem o mundo digital, serão 'náufragos passivos'. Os críticos tem razão em ter medo?
  - O que significa "reduzir as desigualdades" ao ensinar a pensar como um computador nas escolas? Como é que isso nivela o jogo para todos?
  - A revolução digital pode ser dominada? As máquinas dependem do ser humano? O cérebro humano é incapaz de alcançar quaisquer performances de um computador?



Capítulo 2:
Pilar do Pensamento Computacional: Decomposição





Por dentro da BNCC:

Habilidades:
EF01CO01
EF01CO02
EM13CO01

No capítulo anterior, você organizou uma festa incrível e usou um monte de truques inteligentes para que tudo desse certo, quando parecia uma missão impossível.

Pois saiba que esses "truques" têm nome, fórmula e podem ser utilizadas por todos. Eles são: os **Quatro Pilares do Pensamento Computacional.**

Definidos por *Wing* (2006), esses quatro pilares usados na resolução de problemas complexos de forma sistemática são: **Decomposição, Reconhecimento de Padrões, Abstração e Algoritmos.**

Eles são a engrenagem fundamental por trás da precisão impressionante exibida por elementos eletrônicos e digitais. Porém, como já vimos, essas habilidades não são limitadas a resolução de problemas estritamente computacionais, podendo ser implementadas na mais vasta gama de áreas de atuação profissional ou de competências estudantis.

Se o Pensamento Computacional fosse uma **caixa de ferramentas** para o seu cérebro, os quatro pilares seriam as suas ferramentas principais: **uma para desmontar, uma para encontrar atalhos, uma para focar e outra para criar o mapa.**



Neste capítulo nós vamos abrir essa caixa, pegar a ferramenta de decomposição e entender para que ela serve.

2.1. Dividir para conquistar

No Pensamento Computacional um dos primeiros passos para resolução de qualquer barreira é o ato de dividi-la em componentes menores para facilitar a gestão de possíveis soluções. Esse procedimento tem a função de simplificar a complexidade de algo que parece quase impossível em um primeiro momento.



Sabe quando você se reúne entre amigos e decide montar um quebra-cabeças?

Olhar para aquela montanha de pecinhas de formatos diferentes e sem nenhuma conexão aparente pode ser assustador.

Mas o que você faz? Você separa as peças em grupos por afinidade:

- os cantos que montam as bordas em um montinho
- as peças que têm a cor do fundo da imagem em outro
- depois as peças com os traços do desenho...

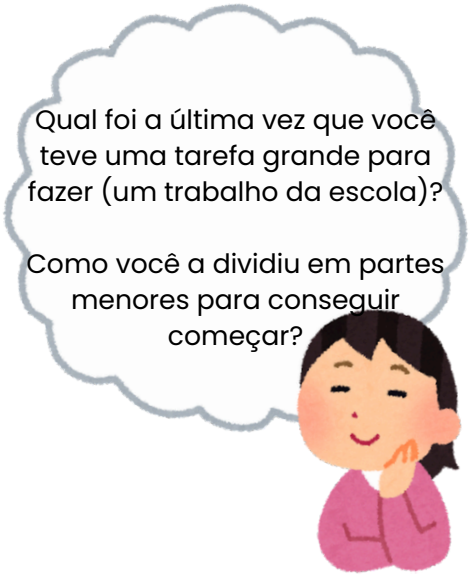
No final, você tem vários grupos de peças que te ajudarão a montar uma pré-imagem do seu objetivo final.

Isso é Decomposição! É a arte de olhar para um desafio gigante e quebrá-lo em tarefas menores e mais fáceis de resolver.

Na Festa da Manu: Foi exatamente o que você fez! Em vez de entrar em pânico com a missão "organizar a festa", sua mãe deu a dica de ouro: fazer uma lista.

Você pegou papel e caneta e **dividiu o "problemão" em pedacinhos:** comprar o bolo, escolher as músicas, preparar a decoração, enviar os convites. Assim, cada item da lista se tornava uma tarefa menor e muito mais simples de executar.

Para pensar!



2.2. Plano de ação

A ideia de “quebrar algo em pedacinhos” pode ser assustadora para muitos, afinal, atribuímos essa imagem a algo que está incompleto que não pode ter utilidade.

Da mesma forma, imagina-se que ao fazer o mesmo com um problema - para solucioná-lo em partezinhas -, estaria sendo desprezado o todo e, portanto, seriam perdidas partes importantes para sua resolução final.

Porém, essas ideias não necessariamente representam um fato. Algo incompleto ou separado em várias partes pode ser unido novamente com cuidado e voltar à sua utilidade anterior.

A chave para o sucesso deste pilar do Pensamento Computacional está na organização e no método. Mas, na prática, como fazer isso? Por onde começar a "fatiar" o problemão? A seguir, apresentamos um roteiro passo a passo para aplicar a decomposição em qualquer desafio.



Vamos usar uma nova missão como exemplo:
"Planejar uma tarde de estudos pré-prova."

Passo 1:
Defina o "problemão"

Primeiro, você precisa saber quem é o "inimigo" que você quer vencer. Escreva o problema principal em uma frase clara.

"Organizar minhas quatro horas de estudos antes da prova de História".

Por que isso é importante?

Se você não definir o problema, pode acabar resolvendo a coisa errada (como estudar para a matéria da prova da semana que vem) ou até mesmo resolver o problema certo, porém nas condições erradas (como estudar por tempo demais e perder a hora de ir dormir, ou esquecer de se alimentar).

Passo 2:
Defina o "problemão"

Todo problema grande tem "juntas" ou categorias principais.

Pense nas grandes áreas que formam o seu problema. Não se preocupe com os detalhes ainda, apenas com os "blocos" maiores.

A melhor pergunta a se fazer é:

"Quais são as partes principais?"

Uma tarde de estudos necessariamente precisa de...

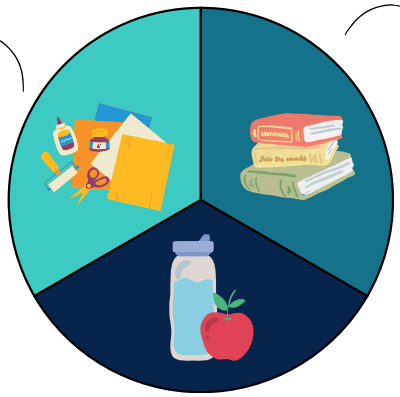
- Livros didáticos
- Caderno e materiais de escrever
- Lanche entre sessões de estudo

Passo 3:
Continue dividindo

Agora, pegue cada um desses blocos grandes e "fatie" de novo, pensando em quais são as **subdivisões** desse tópico:

Materiais de escrever:

- Canetas para fazer um resumo colorido e lápis para resolver os exercícios
- Caderno da matéria e folhas avulsas.
- Marca-textos para grifar partes importantes.



Livros Didáticos

- Livros que melhor abrangem a matéria.
- Listas de exercícios para fixação.
- Pensar se existem outras fontes de estudo sugeridas (videoaulas).

- **Lanchinho:**
  - Fruta e um sanduíche para matar a fome.
  - Garrafa de água em cima da escrivaninha.
  - Xícara de café para dar uma animada quando o sono bater.

Porém, como saber que você "fatiou" o suficiente? Quando o pedacinho se tornar uma **tarefa única e clara**, que você sabe exatamente como fazer (ou que pode entregar para uma pessoa resolver).

Exemplo: *"Deixar a garrafinha na escrivaninha"* ainda é grande. Vamos "fatiar" mais:

- Encontrar onde a garrafa está.
- Enchê-la de água.
- Colocá-la sobre a mesa.



Pare! Todas as instruções acima são tarefas únicas.

Você chegou ao nível certo!

Quando terminar, você não terá mais um inimigo que não sabe como derrotar. Você terá um **mapa de tarefas**, um verdadeiro *checklist*. Agora, basta começar a marcar os itens, um por um.

Momento da dica...



Use ferramentas visuais! Escreva cada pequena tarefa em um *post-it* e cole na parede.

A cada parte do monstro que você vencer, pode amassar o papel e jogar fora.

Isso ajuda seu cérebro a ver o problema de forma muito mais clara, e de quebra ainda gera um sistema de recompensa ao sentir que o monstro foi esmagado!

2.3. Guia para o professor

Neste capítulo conhecemos o primeiro pilar do Pensamento computacional - Decomposição. Como vimos, o foco não está na tecnologia, mas na habilidade de dividir um problema e partes menores para conseguir resolvê-lo.

Espera-se com este capítulo que o aluno compreenda que grandes desafios podem ser resolvidos de forma mais fácil quando são divididos em etapas mais simples. É importante destacar que a decomposição não fragmenta um problema desordenadamente, mas cria um mapa de ações a serem executadas.

Objetivos de aprendizagem

- Compreender o conceito de Decomposição como um dos quatro pilares do Pensamento Computacional.
- Identificar situações do cotidiano que exigem divisão de tarefas.
- Diferenciar um problema genérico de um problema organizado em partes menores.
- Aplicar o processo de decomposição em desafios escolares reais.
- Reconhecer quando uma tarefa foi “dividida o suficiente”.

Roteiro sugerido de aula (aproximadamente 50 min)

- **Abertura:** Retome a história da "Festa Algorítmica. Conduza os alunos à percepção de que a clareza surge quando o problema é dividido e apresente o termo Decomposição.

- **Pergunte:**

"Se eu disser apenas: 'Organizem uma festa', o que acontece?"; "E se eu dividir em: decoração, música, convidados, comida?".

- **Explique:** A Decomposição é a habilidade de pegar um problema grande e dividi-lo em partes menores deixando-o mais simples de resolver. Mostre que:

  - Um problema grande gera ansiedade.
  - Um conjunto de tarefas pequenas gera ação.
  - Resolver partes não significa ignorar o todo.
  - Dividir é uma estratégia de organização, não de divisão descontrolada.

- **Exploração:** Utilize o exemplo do quebra-cabeça.

  - Pergunte:
    - **"Vocês montariam começando por qualquer peça?"**
    - **"Por que separar bordas ajuda?"**
  - Mostre que:
    - Separar bordas = criar categorias.
    - Agrupar por cor = identificar subconjuntos.
    - Resolver uma parte = reduzir a complexidade total.

Explique que isso é exatamente o que fazemos ao estudar para uma prova, planejar uma viagem ou organizar um trabalho.

- **Desenvolvimento: A Dinâmica "Desmontando o Monstro"**

  - Escreva no quadro um problema grande, por exemplo: *"Organizar uma feira cultural na escola."*
  - Divida a turma em grupos.
  - Cada grupo deve:
    - Definir o problema em uma frase clara.
    - Identificar as partes principais.
    - Continuar "dividindo" até que as tarefas sejam únicas e executáveis.

- Distribua post-its:
  - Cada tarefa em um papel.
  - Organizar na parede por categorias.

- **Visualização:** Após a atividade, explique que o critério de parada é quando a tarefa for tão simples que não gere dúvida sobre como executá-la.

- **Socialização:** Peça que os grupos apresentem respondendo as seguintes perguntas:

  - ***Qual foi o problema?***
  - ***Quais foram as grandes categorias?***
  - ***Qual tarefa foi mais difícil de dividir?***

Discuta se houve grupos que:

  - ***Pararam cedo demais (tarefas ainda amplas).***
  - ***Dividiram demais (fragmentação desnecessária).***

- **Reflexão final:** Conduza uma conversa mostrando que a decomposição: (1) Não elimina o desafio; (2) Torna o caminho visível; (3) Reduz ansiedade; e (4) Aumenta senso de controle.

Avaliação

Observe se os alunos compreenderam como dividir o problema. Verifique se os alunos:

- Conseguem diferenciar problema geral de subtarefa.
- Entendem que decomposição não é bagunça, é organização.
- Percebem o momento adequado de parar de dividir.
- Demonstram clareza ao explicar as etapas criadas.

Promova uma autoavaliação dos alunos com perguntas como:

"Eu consigo começar agora essa tarefa sem dúvida?"
"Se eu entregasse essa tarefa para outra pessoa, ela conseguiria executá-la?"

2.4. Sugestão de Atividades



Atividade 1 - Do problemão à resolução (20 a 25 minutos)

- **Objetivo:**
  - O aluno deve ser capaz de visualizar que a forma mais rápida de solucionar os problemas é pela decomposição.

- **Recursos necessários:**
  - Folhas A4
  - Lápis de cor
  - Mesa grande e redonda

- **Passo a passo**
  - Cada aluno fará um desenho secreto em uma folha A4 de um objeto de uma cor só e que ocupe toda a folha. Por exemplo: um gato laranja, um carro azul, um avião branco, uma estrela amarela, etc.
  - Os alunos devem cortar o seu desenho em 10 pedaços.
  - O professor(a) coleta todos os pedaços e os mistura sobre a mesa.
  - Juntos todos devem se reunir ao redor da mesa e tentar montar uma imagem (além da que desenhou) usando o método da decomposição.

- **Debate em sala**

Após concluírem a atividade, os alunos são convidados a compartilhar quais desenhos foram capazes de montar. O(A) professor(a) deverá questionar a turma:

- Foi possível identificar os desenhos antes da decomposição do monte? Da mesma forma é possível identificar a resolução de um problema imediatamente quando ele se apresenta?
- Você utilizou a técnica de separar os pedaços da mesma cor do resto das peças?
- Havia outra forma melhor de remontar as imagens? Qual?

Atividade 2 - Engenharia Reversa (20 a 25 minutos)

- **Objetivo:**
  - O aluno é convidado a refletir sobre o funcionamento de um aplicativo conhecido, usando a técnica de engenharia reversa para discernir componentes de um todo.

- **Recursos necessários:**
  - Folhas A4
  - Lápis ou caneta

- **Passo a passo**
  - Organizar a turma em trios.
  - Cada grupo deve escolher um aplicativo de sua preferência para fazer uma avaliação técnica de funções. Por exemplo: *WhatsApp*, *Instagram*, *TikTok*, *Uber*, etc. (Cada grupo deve escolher um aplicativo diferente.)

"Se vocês fossem criar um aplicativo, precisariam decompor o aplicativo em pequenas ferramentas independentes."

  - Os alunos devem ser instruídos a analisar 3 principais funções do aplicativo, funções base que definem sua utilização.
  - Depois de listar, devem debater: se cada uma das ferramentas individualmente parasse de funcionar por causa de um bug, o usuário ainda consegue acessar o aplicativo?
  - Em seguida, os alunos irão escrever as funções listadas em uma folha sem explicitar qual é o aplicativo escolhido. Ao lado de cada função devem escrever as conclusões do debate e quais funções, caso ausentes, afetariam essencialmente o desempenho do aplicativo.
  - Por fim, as folhas dos relatórios devem ser trocadas entre os grupos para que tentem adivinhar quais os aplicativos estão sendo descritos.

- **Debate em sala**
  - Se todo o problema é uma máquina velha e complicada que só funciona quando entendemos cada pecinha, você concorda que deve conhecer também o que faz as peças se encaixarem?

- Separando o aplicativo em tarefas, você foi capaz de apontar por que cada função precisa uma da outra?
- Usando a decomposição de situações você consegue exemplificar um fator ou fatores na sua vida ou em um jogo/história que estão inseparavelmente ligados?



Atividade 3 - Debate sobre prioridades (15 a 25 minutos)

- **Objetivo:**
  - O aluno é convidado a refletir sobre como na decomposição de problemas, nem todas as partes têm a mesma importância. Perceber a decomposição como ferramenta de tomada de decisões estratégicas

- **Recursos necessários:**
  - Lousa
  - Canetão ou giz

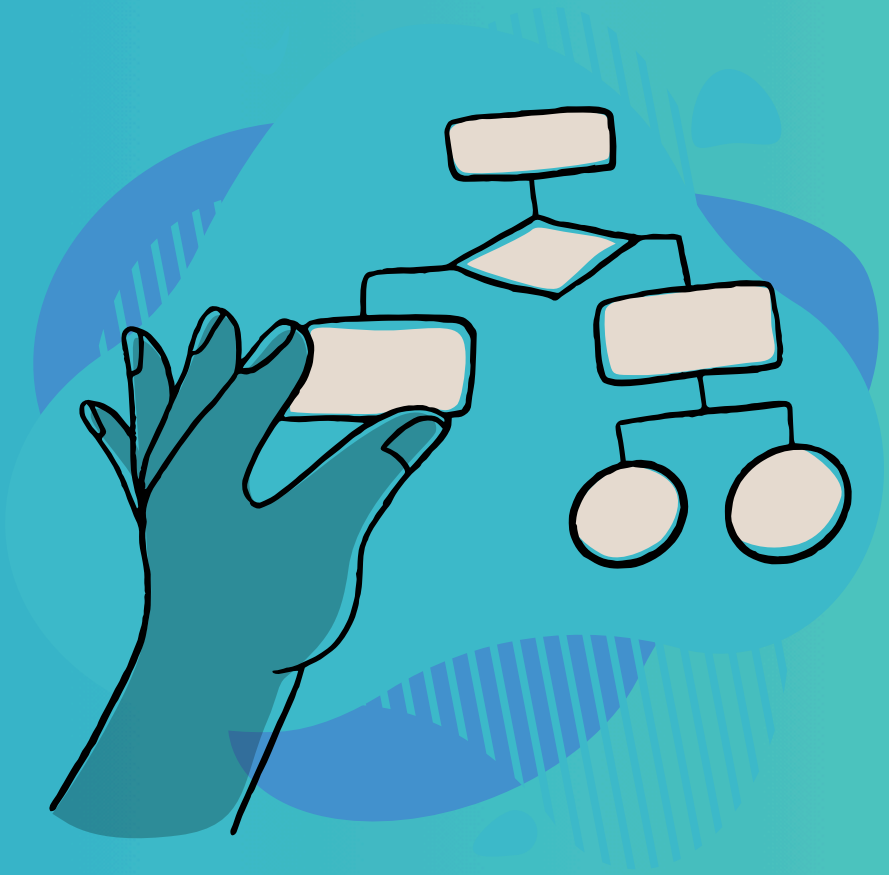
- **Passo a passo**
  - Organizar a turma em grupos.
  - Propor à turma um debate hipotético sobre a implementação de um projeto de cantina voltado ao desperdício zero de alimentos.
  - Conversar sobre as principais metas para o sucesso do cenário. Entre elas devem estar: a conscientização dos alunos, a compra de tonéis de compostagem, a formação dos funcionários da cantina e a instalação de lixeiras orgânicas.
  - Apresentação do problema:

"A direção da escola cortou o orçamento. Só temos tempo e dinheiro para resolver DUAS das quatro áreas. O resto do problema terá de esperar."

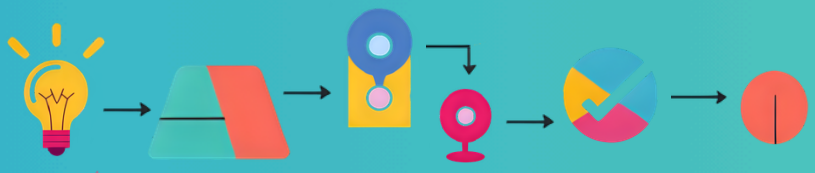
  - Cada grupo terá 5 minutos para escolher quais partes do problema devem ser atacadas primeiro para garantir a continuidade do projeto e preparar argumentos para defender a sua opinião para os demais.

- **Debate em sala**

    - Adiantaria a compra de caixas de compostagem se os funcionários não soubessem usar?
    - Os alunos saberiam da importância do uso das lixeiras orgânicas se não fossem conscientizados?
    - As caixas de compostagem seriam úteis se não tivesse lixo orgânico para abastecer?
    - Se os alunos não fossem conscientizados, os funcionários deveriam se esforçar para manter o projeto?



Capítulo 3:
Pilar do Pensamento Computacional: Reconhecimento de Padrões



TEMA 3

Por dentro da BNCC:

Habilidades:
EF01CO01
EF01CO02
EM13CO01

Com o capítulo 2, você descobriu como dividir um problemão em partes pode te ajudar a resolvê-lo de forma organizada e metódica. Aquele foi o Pilar do Pensamento Computacional responsável pela decomposição.

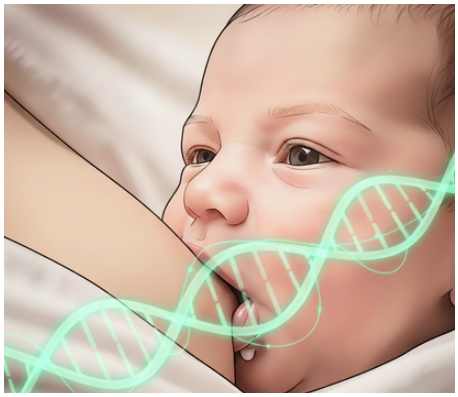
Agora que você aprendeu a decompor direitinho, o próximo passo é olhar para esses pequenos pedacinhos do seu problema e ver um **padrão**.

Neste capítulo, a ferramenta que iremos selecionar de dentro da nossa caixa será o **Reconhecimento de Padrões**. Vamos conhecê-la?

3.1. "Eu já vi isso antes"

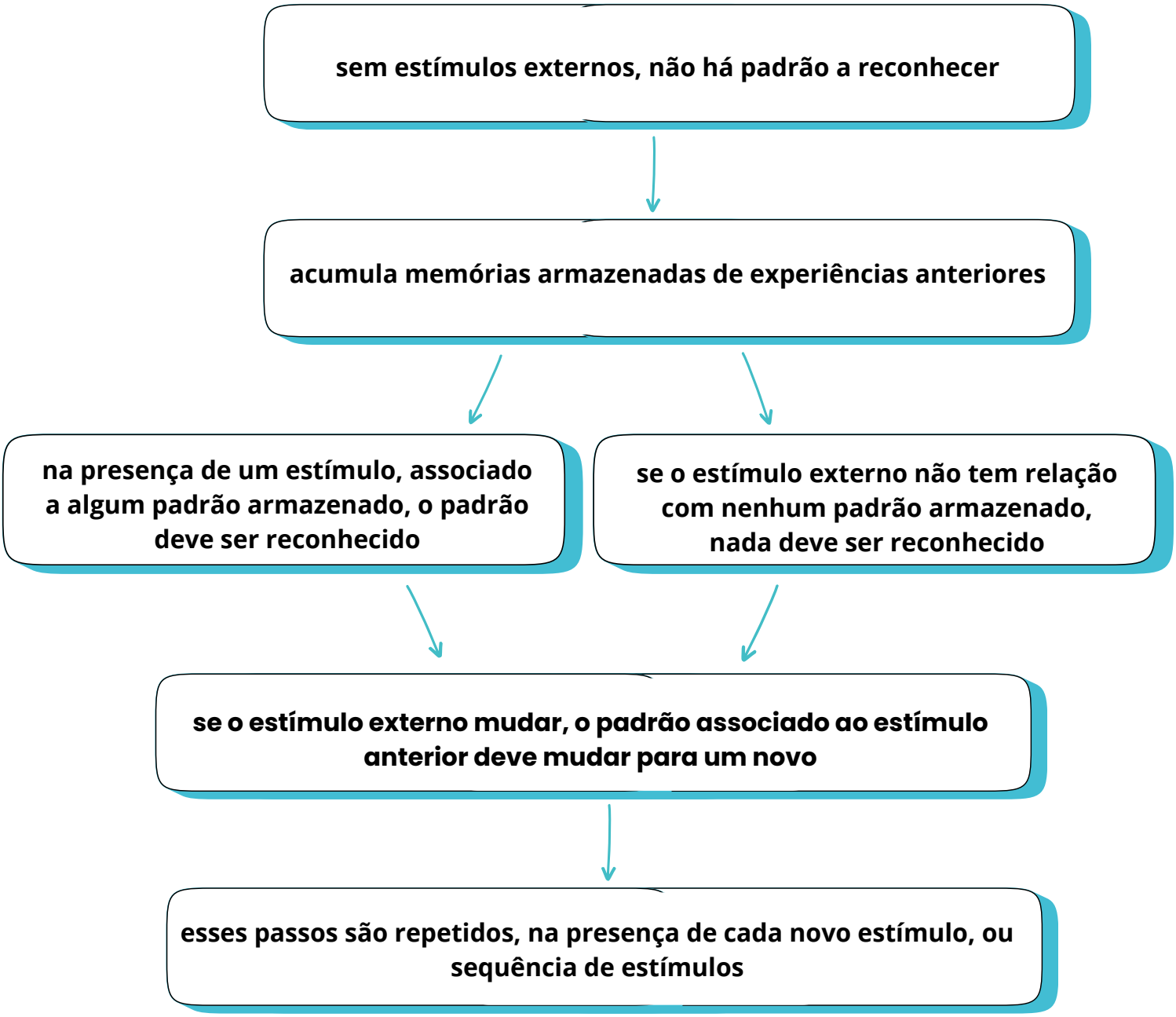
Quando falamos de computação, um dos atributos humanos mais cobiçados sempre foi a habilidade de percepção. Até mesmo os incrivelmente velozes processadores computacionais – chegando a 3,2 bilhões de ciclos de processamento por segundo – ainda não alcançaram a agilidade do processamento neural para decisões perceptivas.

Como, por exemplo, o reconhecimento de uma pessoa apenas por sua silhueta em um ambiente lotado, conclusão tomada dentro de algumas centenas de milissegundos. Essa habilidade de discernimento tem sido exercida consciente e inconscientemente pelos seres vivos durante todas as diversas etapas da jornada evolutiva.



Temos alguns padrões tão inerentes à sobrevivência que estão escritos no nosso DNA. Como o recém-nascido, que sabe sugar o leite materno assim que entra em sua boca, ou reflexos rápidos para escapar de situações de perigo. Nosso cérebro foi condicionado pela evolução a analisar rapidamente estímulos externos e desencadear uma reação.

Essa habilidade é a base para o **Reconhecimento de Padrões**. Nesse processo, em suma, o cérebro:



No dia a dia, nosso cérebro é uma máquina de encontrar padrões para nos ajudar a criar soluções eficientes. Fazemos isso o tempo todo para encontrar atalhos e não precisar reiniciar do mesmo ponto toda vez.

Reconhecer padrões é ter um olhar de detetive para encontrar semelhanças em coisas que parecem novas, mas que, no fundo, são muito parecidas com algo que já vimos antes.

Você provavelmente já usou essa técnica, sem nem saber, para guardar um montão de roupas logo após de lavá-las.



Olhar para aquela montanha de peças, todas misturadas – meias, camisetas, calças, toalhas – é sempre tão cansativo.

Mas será que é mesmo um desafio? O processo é simples: você encontra o par de uma meia e a dobra. Pega uma camiseta e aplica uma técnica específica.

De repente, você nota que está apenas **repetindo padrões**. No fim das contas, você não enfrentou 50 problemas diferentes, mas sim 3 ou 4 tipos de tarefas que se multiplicaram. O 'desafio' era apenas repetição.

Isso é Reconhecimento de Padrões! É a arte de olhar para muitos desafios diferentes e encontrar a "fórmula" ou o "jeito de fazer" que se repete em todos eles.

Na festa da Manu: você usou esse superpoder quando sua mente deu um estalo e você lembrou de coisas importantes.

Você recordou que seu irmão já tinha uma *playlist* pronta e que seu avô sempre se encarregava dos doces. Você reconheceu um padrão — "soluções que já existiam" — e pôde reaproveitá-las.

Para pensar!

Que rotinas ou padrões você repete todos os dias e que tornam sua vida mais fácil (como o caminho para a escola ou a ordem de se arrumar de manhã)?



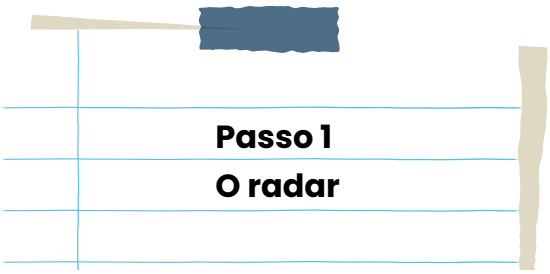
3.2. Plano de ação

Você já percebeu que, nos videogames, os chefões sempre têm um "padrão" de ataque? Ele ataca três vezes, gira e depois descansa por dois segundos. Quando você percebe isso, o jogo deixa de ser sorte e vira estratégia.

Na vida real e na computação, funciona do mesmo jeito. Usar o reconhecimento de padrões nos permite prever o futuro baseando-se no passado.



Vamos usar mais uma missão como exemplo:
"Preparar uma mochila de viagem."



Assim como você fez no passo a passo da decomposição, aqui no Reconhecimento de Padrões também é necessário saber de forma muito clara qual é o problema. Portanto, antes de sair apertando todos os botões do controle, disparando golpes contra o chefão, pare e observe. O que está acontecendo?

"Preciso separar tudo que for necessário levar para uma viagem de escola na qual irei passar um dia ensolarado em uma fazenda, no final do ano."

Você está coletando dados antes de agir. Dessa forma, logo ao analisar o problema geral, já é possível reconhecer suas especificações.

- Será um dia de calor, pois estará ensolarado e será no final do ano;
- Você passará um dia inteiro lá, então é essencial considerar isso para medir sem exageros e sem faltas suas necessidades;
- O destino é uma linda fazenda, com espaço para jogar bola, interagir com vários animaizinhos e desbravar a natureza. Existem roupas adequadas para aproveitar essas vantagens ao máximo.
- É uma viagem de escola, você precisa do seu uniforme de identificação.

Passo 2
Encontrar as repetições

Agora sim, depois de já fazer as identificações mais abrangentes — aquelas que podem ser feitas apenas considerando o problema inicial e conhecimentos de senso comum (conforto, influência natural, solicitações da escola) —, podemos partir para o próximo passo.

O que sempre acontece da mesma forma? Pare para pensar em todas as viagens que você já fez em sua vida:

- Viagem para a casa da Vó.
- Viagem para a praia.
- Viagem para conhecer seu priminho no interior.
- Viagem em família para conhecer outro estado.



Tendo todas as possibilidades dispostas à sua frente, fica mais fácil apontar qual que é mais semelhante à sua missão atual. Você percebe que quando foi visitar seus tios no interior, você ficou hospedado na fazenda deles.

Assim, finalmente, você encontra uma repetição de padrões que se aplicará no seu problema atual. **A "Viagem para o interior" é a que mais se parece com a "Viagem para a fazenda"!**

Passo 3
Variações: se adapte

Lembra do chefão do videogame? Às vezes, na segunda fase, ele aparece com uma cor diferente. O padrão de ataque é o mesmo (três golpes e gira), mas agora ele usa fogo em vez de gelo. O jogador esperto não reaprende tudo do zero; ele apenas se adapta.

Na sua missão da viagem, você faz a mesma coisa. Ao olhar para as viagens passadas, você percebeu que elas têm semelhanças com sua viagem atual. Mas cuidado! Elas não são idênticas. **Identifique o que mudou:**

O que é diferente:

Na casa do seu priminho, tem comida pronta. Na fazenda, talvez precise levar algo específico. Além disso, é uma viagem de escola, então você precisa usar o uniforme ou camiseta da turma. Você também não vai passar a noite como na outra viagem, então, pode descartar toda a bagagem de roupa de cama ou toalha para o banho.

O que é igual:

A necessidade de lanche para a viagem, levar documentos, andar no meio do mato e ver vários animais correndo soltos, botas para áreas enlameadas, boné para o sol, e roupa confortável.

Ajuste o padrão!

- Roupa de cama → Casaco extra (em caso de ficar muito frio)
- Lanche apenas para viagem → Lanche para comer durante o dia também
- + Repelente e boné (por causa do sol e insetos)

Agora, depois de identificar o padrão e ajustar as variações, só falta arrumar a mochila e pronto! Viu só? Ao usar o Reconhecimento de Padrões, você transformou uma tarefa que poderia ser estressante em um processo lógico e rápido.

E o melhor: ano que vem, se tiver uma excursão para um Parque Ecológico, seu cérebro vai gritar: *"Ei! Isso é quase igual à viagem da fazenda!"*. Você vai resgatar esse modelo mental e sua mochila estará pronta em cinco minutos!

3.3. Guia para o professor

Este capítulo abordou o segundo pilar do Pensamento Computacional: **Reconhecimento de Padrões**. Depois de aprender a dividir um problema (Decomposição), o estudante agora aprende a observar semelhanças entre situações diferentes, identificar repetições e reaproveitar soluções já usadas anteriormente.

O foco está em desenvolver a capacidade de:

- Detectar regularidades.
- Comparar situações.
- Generalizar estratégias.
- Adaptar soluções conhecidas a novos contextos.

Objetivos de aprendizagem

Ao final da aula, espera-se que os estudantes sejam capazes de:

- Compreender o conceito de Reconhecimento de Padrões.
- Identificar semelhanças entre situações aparentemente diferentes.
- Diferenciar o que é igual e o que é diferente em um problema.
- Utilizar experiências anteriores como base para resolver novos desafios.
- Adaptar estratégias conhecidas a pequenas variações.

Roteiro sugerido de aula (aproximadamente 50 min)

- **Abertura:** Voltando à história da "Festa", relembre o que foi abordado nos capítulos anteriores.

  - No Capítulo 1, a missão parecia impossível até que foi organizada uma lista organizando o pensamento.
  - No Capítulo 2 (Decomposição), o problema foi dividido em tarefas menores facilitando a sua resolução.
  - E agora, no Capítulo 3 (Padrões), é possível identificar nas partes divididas, o que se repete.

- **Pergunte:**

"Após elaborar a lista, quais itens Manu percebeu que já estavam prontos (playlist, doces do avô)?"

Isso é um padrão: soluções que se repetem e podem ser reaproveitadas.

- **Explique:** Apresente a situação do capítulo: uma montanha de roupas misturadas (meias, camisetas, calças, toalhas). Pergunte:

  - ***"Isso parece muitos problemas diferentes?"***
  - ***"O que você faz primeiro quando pega uma meia?"***
  - ***"E quando pega outra meia, você inventa outro jeito ou repete o mesmo?"***

- **Exploração:** Conduza para a ideia central do capítulo: o aluno deverá perceber que os "tipos de problemas" se repetem (dobrar meias, camisetas e calças, por exemplo).

- **Desenvolvimento:** Dinâmica: "O Método que se Repete"

    - Divida a turma em grupos.
    - Cada grupo recebe (no quadro/cartaz) a situação *Montanha de roupas* e deve criar uma tabela simples com duas colunas:
        - *"O que se repete?"* (o padrão)
        - *"Como eu faço?"* (a fórmula)
    - Exemplo esperado:
        - Padrão: *"meias aparecem várias vezes"* → fórmula: *"encontrar par e dobrar do mesmo jeito"*.
        - Padrão: *"camisetas aparecem várias vezes"* → fórmula: *"dobrar sempre do mesmo jeito"*.
    - Depois, faça outras rodadas com temas diversos, como *"preparar mochila de viagem"*, ou *"procurar um livro de estudos"*.

- **Visualização:** Após a atividade, explique que o critério de parada é quando a tarefa for tão simples que não gere dúvida sobre como executá-la.

- **Socialização:** Cada grupo compartilha:

    - Um padrão identificado.
    - Uma fórmula criada.
    - Uma variação que precisou ser ajustada.

- **Reflexão final:** Conduza uma conversa sobre o exercício da festa. Pergunte:

    "Na festa, quais padrões foram reconhecidos?"

    - ***playlist pronta.***
    - ***avô cuidando dos doces.***
    - ***soluções já existentes.***
    - ***Qual foi o padrão mais útil que vocês encontraram?***

Conclua dizendo que reconhecer padrões é perceber que muitos desafios novos são parecidos com algo que já vivemos — e assim dá para resolver mais rápido.

Avaliação

Observe se os alunos desenvolveram o raciocínio durante a identificação de padrões e adaptações:

- Conseguem identificar corretamente o que se repete em diferentes situações.
- Diferenciam padrão de variação.
- Utilizam experiências anteriores como base para resolver novos desafios.
- Evitam começar do zero quando há possibilidade de reaproveitamento.
- Demonstram capacidade de comparação estruturada.

Durante a socialização, verifique se:

- O grupo explicou o padrão identificado.
- Houve clareza por parte do aluno entre o que é igual e o que é diferente.
- Conseguiram fazer adaptações lógicas com o novo contexto.

3.4. Sugestão de Atividades



Atividade 1 - Quem sou eu dos padrões (20 a 25 minutos)

- **Objetivo:**
  - O aluno deve ser capaz de visualizar que dados aparentemente caóticos possuem uma ordem oculta, um padrão e que encontrá-lo é a chave para a resolução.

- **Recursos necessários:**
  - Imagens de objetos que façam parte de um grupo que os caracteriza
  - Quadro
  - Marcador ou giz

- **Passo a passo:**
  - Dispor a turma em um círculo.

- Selecionar um tema super secreto para a turma. Por exemplo: "Coisas que flutuam", "Objetos que têm rodas", "Coisas que são vermelhas". E anotar em um papel que deve ficar escondido.

- Cada aluno deve receber uma imagem de algo que faz parte do tema secreto e não deve mostrar para os demais.

- Depois, um de cada vez, os alunos devem escolher um colega para fazer apenas uma pergunta sobre as características do objeto da imagem em comparação com o seu objeto, a fim de descobrir o padrão entre toda a turma.

- Desta forma, sem saber qual é o objeto uns dos outros e apenas sabendo qual é o seu e as dicas de características dadas pelos outros eles devem tentar encontrar a regra que conecta o grupo.

- Uma lista de ideias pode ser colocada no quadro para escolher a categoria de objetos mais provável por meio de votação.

- **Debate em sala:**

  - Após concluírem a atividade e descobrirem a regra. O(A) professor(a) deverá questionar a turma:
    - Foi fácil identificar o padrão com a primeira pergunta ou precisaram inspecionar vários desenhos para ter a certeza?
    - O que seriam os desenhos no mundo do computador?



Atividade 2 - O algoritmo Humano (20 a 25 minutos)

- **Objetivo:**

  - O aluno é convidado a refletir sobre como o seu próprio cérebro cria "atalhos" mentais baseados em repetições, simulando o funcionamento de uma Inteligência Artificial.

- **Recursos necessários:**

  - Folhas A4
  - Lápis ou caneta.

- **Passo a passo:**
  - Organizar a turma em grupos de 3 integrantes.
  - Cada grupo deve escolher um perfil estereotipado secreto. Ex.: *"o fanático por futebol"*, *"o viciado em videogames"*, *"o fã de música K-Pop"*.

"Se vocês fossem o algoritmo de um aplicativo de vídeos, que padrões de comportamento você imagina que este consumidor teria?"

  - Os alunos devem listar 3 comportamentos repetitivos desse perfil. Ex.: para *K-Pop*:
    1. Assiste a vídeos de coreografias;
    2. Pesquisa por letras de músicas em coreano;
    3. Segue canais de notícias de Seul.

  - Depois, devem criar uma "Regra de Previsão": *"se o utilizador fizer X e Y, então vamos sugerir Z"*.
  - As folhas com os comportamentos, sem especificar o perfil, são trocadas entre os grupos.
  - Os grupos rivais devem analisar os dados de comportamentos e tentar adivinhar o perfil estereotipado (reconhecer o padrão).

- **Debate em sala:**
  - Se um utilizador mudar de gostos repentinamente, quanto tempo demora até o antigo padrão desaparecer? Quantas novas escolhas ele precisa fazer?
  - Conseguem dar um exemplo de um falso padrão? Algo que parece uma regra, mas é apenas coincidência.



Atividade 3 - Debate sobre a máquina de fazer sucessos (15 a 25 minutos)

- **Objetivo:**
  - O aluno é convidado a refletir sobre como o reconhecimento de padrões é utilizado para criar produtos de grande sucesso, como músicas e debater o limite.

- **Recursos necessários:**
  - Quadro e marcador ou giz.
  - Um rádio para tocar breves refrões de músicas famosas.

- **Passo a passo:**

  - Organizar a turma em dois grandes grupos: os produtores, que defendem o padrão e os artistas, que defendem a quebra de padrão.
  - Apresentação do problema:

"Na indústria da música, os produtores, ao analisar canções, descobriram um padrão: quase todas as músicas Pop de sucesso usam os mesmos acordes musicais e têm a mesma estrutura. Hoje, programas de Inteligência Artificial conseguem reconhecer este padrão e criar o 'Hit Perfeito' em segundos."

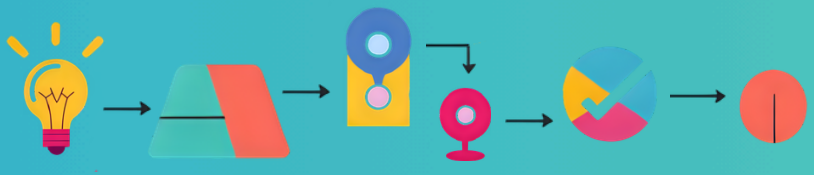
  - Cada um dos grupos terá 5 minutos para defender a sua ideia com argumentos.
  - Os produtores: se estamos acostumados com essa fórmula, por que mudar? É uma garantia de sucesso e lucro; as pessoas querem apenas dançar e se divertir.
  - Os Artistas: se todas as músicas seguissem a mesma fórmula, elas tornariam-se chatas e repetitivas, já que que um computador não é capaz de criar algo profundo, com uma particularidade emocional, pois possui limitação da criatividade.

- **Debate em sala:**

  - O professor(a) deverá atuar como mediador, lançando provocações para a turma.
    - Quando ouvem uma música nova na rádio e parece que já a conhecem, isso é um defeito da música ou é exatamente o motivo pelo qual gostam logo dela?
    - Será que um computador consegue reconhecer perfeitamente todos os padrões que nos fazem chorar em um filme ou dançar em uma música?
    - Imaginem que o mundo inteiro só consumia coisas feitas com o 'Padrão do Sucesso'. Como seria o nosso entretenimento daqui a 10 anos?
    - O Reconhecimento de Padrões é essencial para identificar mecanismos de sucesso e tendências de consumo. Mas será que deve ser utilizado em todas as circunstâncias, para criação de coisas novas, correndo o risco de se manter no velho?



Capítulo 4:
Pilares do Pensamento Computacional: Abstração



TEMA

4

Por dentro da BNCC:

Habilidades:
EF01CO01
EF01CO02
EM13CO01
EM13CO02

Agora que você já dominou a arte de quebrar grandes problemas em pedaços menores **(Decomposição)** e, no capítulo 3, aprendeu a usar o passado para resolver o futuro encontrando repetições **(Reconhecimento de Padrões)**, chegou a hora de conhecer mais uma ferramenta da nossa caixa.

Decompor problemas muitas vezes nos deixa com muitas pecinhas pequenas e difíceis de classificar, porém, nem sempre nós precisamos de todas essas peças para montar o quebra-cabeças.

Às vezes ficamos analisando detalhes que não são tão relevantes para o resultado final. Para enfrentar nosso inimigo sem nos sobrecarregarmos, precisamos da terceira ferramenta da nossa caixa: o Filtro do Essencial. No Pensamento Computacional, chamamos isso de **Abstração.**

4.1. Focando no que realmente importa

Imagine que você decompôs o problema da *"Mochila de Viagem"*, no capítulo anterior, e identificou o padrão de que *"toda viagem precisa de lanche"*. Mas, na hora de pensar no lanche, você começa a se preocupar com qual suco você prefere tomar no calor ou com a marca exata da bolacha que sua mãe comprou, a forma como você vai conservar a alface dentro do sanduíche ou há quanto tempo o pão está na prateleira da cozinha.

Parece loucura, não é? O fato é que, quanto mais paramos para pensar — ou damos abertura às divagações da mente —, mais percebemos detalhes específicos ao nosso redor. Se prestássemos atenção em absolutamente tudo, nosso cérebro travaria em cinco minutos.

Portanto, devemos aprender a filtrar todas essas informações camufladas e direcionar nosso foco para aquilo que é de maior importância no momento.

Para isso, usamos a ferramenta da **abstração**.

Esse pilar é responsável pelo superpoder de olhar para uma situação complexa e filtrar apenas as informações que são relevantes para o problema que você quer resolver agora, ignorando todo o resto.

Pense em um aplicativo de navegação veicular, que você já deve ter usado com sua família para mostrar a rota até uma nova localidade.



Você já reparou que ele não mostra a distância real entre um estabelecimento e outro no centro, nem mostra a inclinação exata de cada avenida, nem mesmo os nomes de cada condomínio residencial no caminho?

Talvez você pense que essas informações são muito importantes e deveriam aparecer nos mapas. Conclusão essa que não está menos correta.

Porém, deve concordar que se o seu objetivo é chegar a um lugar específico, as ruas e prédios paralelas ao seu caminho não são a coisa mais importante, não é mesmo? Se você tentasse usar um mapa de satélite super detalhado para chegar até seu destino, ficaria confuso.

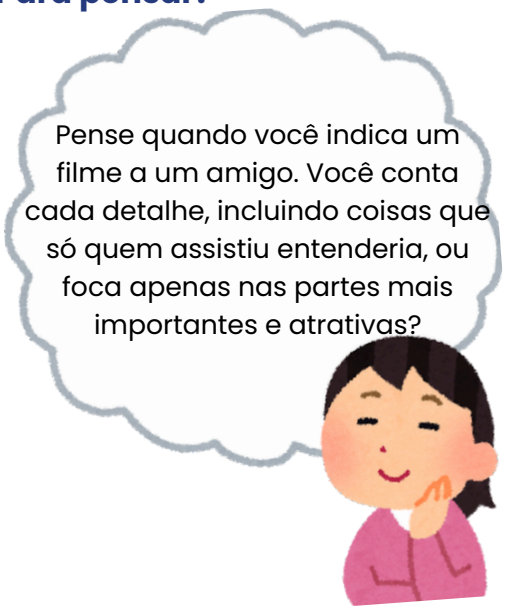
O mapa de navegação **abstrai** ou ignora a geografia real para focar apenas no que você precisa saber: *quanto tempo até o fim da viagem e qual é o melhor caminho.*

Isso é Abstração! É simplificar a realidade para criar um modelo que seja útil. É saber que, para jogar videogame, você precisa saber o que o botão "X" faz, mas não precisa saber como os circuitos eletrônicos dentro do controle funcionam.

Na festa da Manu: Ao perceber que a música e os doces já estavam quase resolvidos, pois já haviam sido utilizados em outra situação, você usou a abstração para focar no que era mais urgente, como escolher o tema e definir o orçamento, deixando de lado as preocupações menores.

Esses detalhes existiam, e você ainda teria de organizá-los depois, mas não eram os mais importantes para a sua missão da festa.

Para pensar!



A abstração é uma habilidade muito utilizada nas mais diversas áreas do conhecimento, apesar de ser um aspecto fundamental da engenharia de software.

Você sabia que...



Engenharia de *Software* é uma área da engenharia e computação que se preocupa com todos os aspectos da produção de software, desde os estágios iniciais de definições do sistema até a manutenção depois que ele entrou em operação.

Sommerville, I. **Software Engineering,** 10th Ed. Pearson, 2015.

Programadores criam "camadas" como ferramenta para lidar com a complexidade. Por exemplo, em um código que tem como objetivo processar registros, ao usar uma função deletar_registro(), o programador abstrai os detalhes de realização da ação, focando apenas no objetivo.

Paralelamente, o conceito de abstração não se limita à computação, sendo ferramenta fundamental em diversas áreas de atuação profissional:

- Um **arquiteto** utiliza a abstração como mecanismo de modelagem espacial. Ao elaborar um projeto, não deverá desenhar cada tijolo e parafuso em um esboço inicial, ele mantém o foco na representação do todo do ambiente físico. O objetivo é representar a função e o fluxo do ambiente.

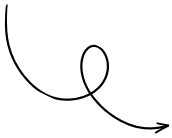
- Um **matemático** observa na soma de maças, 2 + 3 = 5, e na soma de minutos, 20 + 30 = 50, a possibilidade da abstração dos números específicos para criar uma fórmula algébrica a + b = c. Com isso transforma processos de contagem em objetos mentais, permitindo operar sobre eles sem precisar de referências físicas.

- O **legislador** redige as leis com o foco em conceitos abstratos. Ao criar uma lei para cobrir roubos de carros, cavalos ou dados, utiliza o conceito de "propriedade". Funciona como um sistema de etiquetagem universal: a lei define a categoria da infração, independentemente das características físicas do objeto envolvido.

- O **médico** usa abstração como ferramenta de filtragem diagnóstica ao ouvir a história do paciente. Ele ignora detalhes irrelevantes (a cor da roupa do paciente, opiniões pessoais) para identificar padrões de sintomas que correspondam a uma patologia conhecida. O raciocínio clínico abstrai o indivíduo momentaneamente para enxergar a doença.

Segundo Piaget, o famoso psicólogo que conhecemos no tópico 1.2 deste livro, existem três tipos de abstração: a **abstração empírica**, a **abstração pseudo-empírica** e a **abstração reflexiva**.

A **abstração empírica** depende da observação de cada indivíduo a respeito do objeto em questão. Características que já estão no objeto são percebidas externamente a depender do ponto de vista de cada indivíduo. O objeto não é transformado por essa ação. Há apenas um foco específico na parte do objeto que atende melhor ao que é procurado.



Imagine que seus olhos são um filtro do Instagram. O objeto está lá, você aplica o filtro "apenas cor" ou "apenas peso". O objeto não muda, mas sua percepção foca em um detalhe.

Esse tipo de abstração leva a reflexões sobre característica que levam a generalizações, isto é, a passagem do específico para o geral.

No caso da abstração pseudo-empírica, entram em jogo ações feitas nos objetos. Considere, por exemplo, uma abstração alcançada a partir de uma representação de algo.

Nesse caso, a abstração é empírica porque depende de características do objeto, mas também depende de modificações adicionadas a ele. Assim, a abstração diz menos sobre o objeto físico em si e mais sobre a ação de organizá-lo ou configurá-lo.

Imagine cinco pedrinhas espalhadas no chão. A propriedade "cinco" ou "uma linha reta de pedras" não está dentro da pedra. Para que elas formem uma linha, você precisa agir e enfileirá-las. Você aprende sobre "alinhamento" ou "contagem" observando o resultado da sua ação sobre os objetos.

Já a **abstração reflexiva** se origina no indivíduo e depende de suas ações, ou seja, o objeto físico sai completamente de cena. O conhecimento nasce totalmente dentro da sua cabeça.

Esse tipo de abstração está associado não apenas às ações propriamente ditas, mas também à construção de novas ideias baseadas em pensamentos anteriores.

Esse aspecto construtivo da abstração reflexiva exerce um papel fundamental no pensamento matemático avançado.

O conceito de **Função Matemática (f(x) = y)**. Você não precisa de pedras ou maçãs para entender isso. Você está relacionando a ação de pegar um número com a ação de transformá-lo em outro segundo uma regra. É uma construção puramente lógica e mental.

4.2. Plano de ação

A ideia de “ignorar detalhes” pode parecer contraprodutiva. Afinal, fomos ensinados que a perfeição mora nos detalhes e que deixá-los de lado é sinal de desleixo. Da mesma forma, para resolver um problema de localização é fácil pensar que precisamos descrever tudo o que existe no caminho.

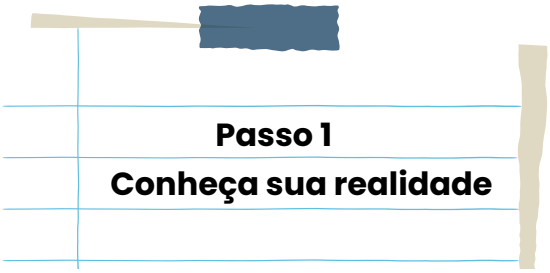
Porém, essa ideia não necessariamente representa um fato. Tentar prestar atenção até nos buracos da rua, ou na cor das casas, ou mesmo no cachorro latindo — é o caminho mais rápido para a confusão mental.

A chave para o sucesso desse passo dos Pilares do Pensamento Computacional é saber ignorar o ruído para focar na música. OK, mas como saber o que jogar fora e o que guardar? Como criar esse "filtro" na prática?

Vamos aprender a criar um roteiro passo a passo para você aplicar a abstração de forma eficiente.



Mas para isso precisaremos de uma nova missão:
"Desenhar um mapa para um amigo chegar à sua casa pela primeira vez"



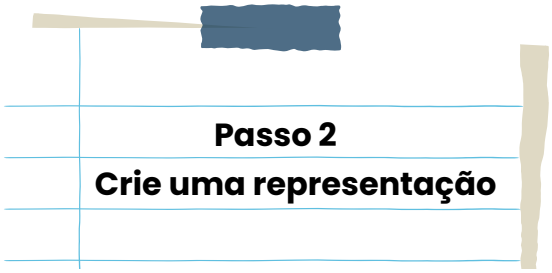
Antes de qualquer coisa, você precisa saber qual é a função exata daquilo que você vai criar.

"Guiar meu amigo da escola até minha casa da forma mais simples possível."

Se o seu objetivo fosse "fazer um desenho artístico da rua", você precisaria ter um cuidado especial com os detalhes das árvores e as cores das casas. Mas como o foco é "guiar", a beleza das casas não é tão relevante, apenas a localização delas. Percebe como o objetivo já define o que é útil?

Pense no mundo real, no trajeto completo. Tente visualizar o que existe no caminho, mas sem adicionar nenhum filtro ainda. É o momento de ver o grau de complexidade antes de simplificá-lo. No trajeto real existem:

- Centenas de carros passando.
- Cachorros de todas as raças latindo nos portões.
- A cor do asfalto.
- Uma padaria na esquina.
- Buracos na calçada.
- Um prédio azul bem alto.
- O nome de todas as 15 travessas.



Para conseguir lidar com o problema você precisa parar de olhar para a rua real e começar a encontrar símbolos que a retratem.

No universo da Abstração isso é chamado de Representação. Não tente desenhar aquilo que você anotou que viu no caminho. Precisamos de dados de navegação.

Agora é a hora da mágica. Pegue a lista feita no passo 1 e faça a pergunta de ouro da Abstração: *"Quais desses dados meu amigo precisa saber para não se perder?"* Se a resposta for *"não, ele não precisa disso"*, é ruído. Jogue fora.

Carros e cachorros? Eles mudam de lugar, não são elementos físicos, isso atrapalha. ✗ Eliminar

Cor do asfalto? É irrelevante para o trajeto. ✗ Eliminar

Buracos na calçada? Talvez você queira avisar seu amigo, não colocar no mapa. ✗ Eliminar

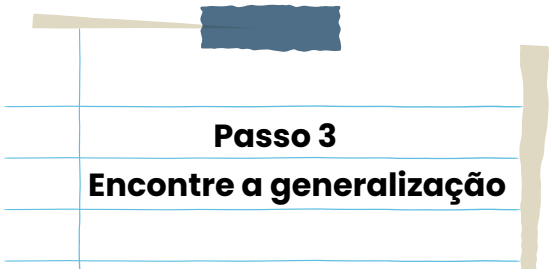
Nome de todas as 15 ruas? Inútil pra quem não conhece, só confunde. ✗ Eliminar

A padaria na esquina? Um ponto de referência para virar. ✓ Manter

O prédio azul? Confirmação que ele está no caminho certo. ✓ Manter

O nome da rua onde ele deve virar? Instrução importante para conclusão do trajeto. ✓ Manter

Desenhe ou escreva símbolos para representar o objeto real.
Dessa forma, você acabou de trocar o mundo real complexo por uma representação simbólica estritamente útil e simples.



Agora que você escolheu uma forma para representar uma rua, olhe para as outras esquinas. Perceba que, embora as características sejam um pouco diferentes (uma tem um semáforo, a outra tem uma placa de pare), a lógica de navegação é idêntica.

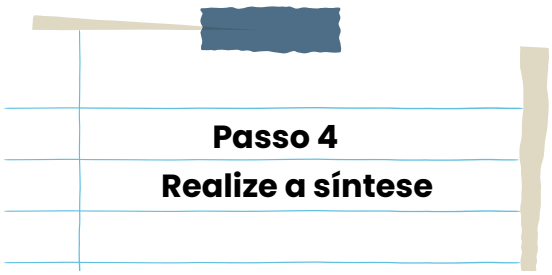
Isso é chamado de **Generalização**. É o ato de **sair do "específico" e ir para o "geral"**.

Para este processo a pergunta chave é: *"Essa instrução funciona para qualquer esquina?"* Para sair da primeira rua você diz *"Siga reto por 2 quadras e vire"*, essa estrutura se repete? Se sim, então não importa qual o nome da rua, o conceito de "Quadra" e o conceito de "Virar" são padrão nesse trajeto.

Crie uma regra visual única que cubra todos os casos. Não desenhe a "curva da farmácia" diferente da "curva da escola". Generalize:

- Toda curva será retratada por um ângulo de 90° no papel.
- Toda distância será medida em "Quadras", que você desenhará como blocos visuais, assim não importa se a quadra tem 100 metros e a outra tem 80 metros.

Viu só? Você acabou de abstrair as particularidades do terreno.



Aí vem o toque final. Pegue todos esses traços e regras e junte-as em um único objeto visual.

Essa é a **Síntese**. É quando o conjunto de instruções vira uma "coisa" única e compreensível.

Você deve se perguntar: *"Posso olhar para tudo isso de uma vez só e entender a imagem final?"*

O seu amigo não precisa ler: *"Ande com cuidado pela calçada esburacada, aprecie a vista do limoeiro no quintal da casa azul, agilize o passo ao passar pelo cão pastor alemão na rua..."*. **Ele olha para o desenho completo e vê o TRAJETO.**

Conecte os pontos e termine o desenho. As linhas e blocos, todos juntos, agora fazem parte de um novo objeto, o Mapa. Note como, com esses passos simples, foi possível sintetizar quilômetros de asfalto e casas em uma única folha de papel com alguns símbolos.

O seu ponto de chegada é que qualquer pessoa consiga olhar para o seu desenho e chegar a sua casa sem te ligar no meio do caminho perguntando *"em qual esquina eu viro?"*, é assim que você sabe que fez um bom trabalho e abstraiu corretamente.



- Se o seu mapa diz *"Vire onde tem um cachorro caramelo dormindo"*, isso é uma abstração ruim, detalhista e variável. Afinal, se o cachorro acordar e sair dali, o sistema quebra.
- Se o mapa diz *"Vire na segunda direita após a ponte"*, ele funciona sempre.

Momento da dica...



Use Ícones! Não desenhe uma padaria realista com pães na vitrine, substitua por um ícone de pão ou apenas um quadrado com a palavra "Pão".

Essa é a forma mais pura de abstração visual: uma imagem que carrega todo um significado instantaneamente, dessa forma economizando processamento visual para o que realmente interessa: o caminho!

4.3. Guia para o professor

Neste capítulo trabalhamos o terceiro pilar do Pensamento Computacional: **Abstração.** Já aprendemos a decompor problemas e reconhecer padrões, agora o estudante deverá desenvolver a capacidade de filtrar o que é essencial, ignorando os detalhes irrelevantes e focando no que realmente importa para resolver o problema.

Objetivos de aprendizagem

Ao final da aula, os estudantes deverão ser capazes de:

- Compreender o conceito de Abstração;
- Diferenciar informação relevante das não essenciais;
- Criar representações de situações reais;
- Criar instruções gerais para que funcionem em múltiplos contextos.

Roteiro sugerido de aula (aproximadamente 50 min)

- **Abertura:** reveja com os estudantes as sequências de pilares já estudados:

  - Capítulo 1 - organizando o pensamento.
  - Capítulo 2 (Decomposição) – dividir o problema.
  - Capítulo 3 (Padrões) – ver o que se repete.
  - Capítulo 4 (Abstração) – identificar o que importa.

- **Pergunte:**

"Se você quer chegar a um lugar, precisa saber a cor das casas?"
"Um mapa mostra tudo que existe na rua?".

- **Explique:** abstração é a habilidade de analisar uma situação e filtrar apenas as informações relevantes, ignorando o restante. Explique os quatro passos apresentados neste capítulo:

  - Conheça a realidade.
  - Crie uma representação.
  - Encontre a generalização.
  - Sintetize.

- **Exploração:** o objetivo define o que é relevante para a solução do problema.

- **Desenvolvimento:** dinâmica: "Filtrando o Essencial"

  - Separe a turma em grupos.
  - Os grupos devem desenhar um mapa para guiar alguém até um determinado local.
  - Peça que os alunos listem tudo o que existe no trajeto:

    - Carros
    - Cachorros
    - Cor do asfalto
    - Padaria
    - Buracos
    - Nome das ruas
    - etc.

- Após a criação dos mapas, os estudantes devem definir o que manter e o que eliminar, respondendo a seguinte pergunta: *"Isso é essencial para o objetivo?".*

- **Visualização:** transformar os elementos que os estudantes decidiram manter em símbolos.

  - Curvas → ângulo de 90°.
  - Quadras → blocos.
  - Padaria → ícone ou palavra simples.

- **Socialização:** cada grupo compartilha:

  - Um elemento eliminado e justifica.
  - Um elemento mantido e justifica.
  - Como representaram os componentes do trajeto.

O professor questiona:

"O que mudou quando vocês começaram a eliminar o que não era essencial?"
"O mapa ficou mais claro ou mais confuso?"

- **Reflexão final:** conduza uma conversa questionando:

"Se o mapa disser 'vire onde tem um cachorro caramelo dormindo', funciona sempre?"
"E se disser 'segunda direita após a ponte', funciona?"

Avaliação

A avaliação deve observar se o aluno:

- Diferencia detalhe irrelevante do que é essencial.
- Consegue justificar o porquê eliminou alguma coisa.
- Consegue criar representações simbólicas coerentes.
- Cria instruções gerais.
- Sintetiza informações de forma coerente.

4.4. Sugestão de Atividades

Atividade 1 - O cartunista computacional (20 a 25 minutos)

- **Objetivo:**
  - O aluno deve ser capaz de visualizar que a abstração é a arte de criar um modelo simplificado de uma realidade complexa, mantendo apenas a sua essência funcional, exatamente como os computadores processam gráficos e informações.

- **Recursos necessários:**
  - Folhas A4
  - Lápis
  - Borracha.
  - Imagem impressa caótica e detalhada de um objeto complexo.

- **Passo a passo:**
  - O professor deve distribuir impressa uma imagem de um objeto complexo (Ex: O motor de um carro, uma paisagem urbana cheia de pessoas e sinais, um animal muito detalhado).
  - Os alunos devem observar a imagem durante 1 minuto. Um computador "veria" ali milhões de *pixels* coloridos, o que exige muita memória para processar.
  - Os alunos têm 5 minutos para desenhar o mesmo objeto/cenário nas suas folhas, mas no formato "ícone de celular", apenas usando formas geométricas simples (quadrados, círculos, triângulos ou retas).
  - Após recolher as imagens originais, os desenhos geométricos são apresentados à turma que deve identificar o que a imagem representa"
  - Após concluírem a atividade, os alunos são convidados a partilhar como decidiram quais os detalhes a eliminar.

- **Debate em sala:**

  - Seria trabalhoso replicar os detalhes da fotografia? Você acha que seria também para o computador?
  - O desenho geométrico é mentiroso ou é apenas uma abstração da realidade?
  - Ao jogar um jogo 3D, você é capaz de perceber que estão interagindo com abstrações geométricas da vida real? Se os programadores tentassem recriar cada folha de grama com perfeição atómica, o que aconteceria com seu computador?



Atividade 2 - Múltiplas identidades / O filtro de dados (20 a 25 minutos)

- **Objetivo:**

  - O aluno é convidado a refletir sobre como diferentes sistemas computacionais abstraem o mesmo objeto do mundo real de formas completamente diferentes, dependendo do objetivo do programa.

- **Recursos necessários:**

  - Folhas A4
  - Lápis ou caneta
  - Quadro
  - Marcador ou giz

- **Passo a passo:**

  - Organizar a turma em 3 grandes grupos.
  - O professor apresenta o "Objeto do Mundo Real" que será analisado: um cãozinho de estimação chamado 'Bobi'. O Bobi é real, late, tem pelo castanho, gosta de brincar com bolinhas de tênis, tem 5 anos e pesa 10 kg.
  - Cada grupo vai representar uma instituição diferente. O servidor deles tem pouco espaço, logo, só podem guardar 3 coisas sobre o Bobi, características gerais.

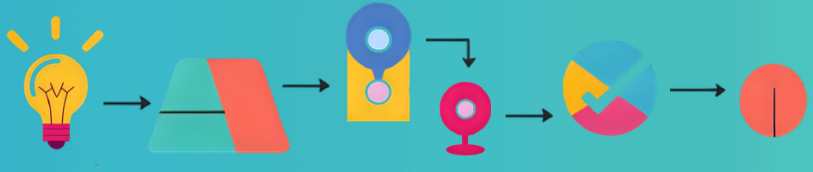
- Cada grupo vai assumir uma instituição: a clínica veterinária, o concurso de beleza canina e o registro civil.
- Após 5 minutos de discussão, cada grupo deve escrever no quadro as 3 características gerais de Bobi que são indispensáveis para a sua instituição

- **Debate em sala:**

  - Olhando para o quadro: o cão "Bobi" é exatamente o mesmo cãozinho físico. Mas por que razão parece que o computador do veterinário e o computador do registro civil criaram "Bobis" tão diferentes?
  - Quando vocês criam uma conta em um aplicativo, que parte da sua vida você "abstrai" do sistema? E quando preenchem um teste de matemática, que talentos a sua escola "abstrai"?



Capítulo 5:
Pilares do Pensamento Computacional: Pensamento Algorítmico



TEMA

5

Por dentro da BNCC:

Habilidades:
EF01CO01
EF01CO02
EM13CO01
EM13CO02
EM13CO04

No capítulo anterior, você usou a **Abstração** para fazer o mapa perfeito até sua casa, focando no que era essencial e ignorando os detalhes irrelevantes. Esse método te ajudou a perceber como as vezes as coisas não precisam ser tão complexas como parecem!

Mas será que no decorrer das páginas você notou como todos os pilares vistos até agora foram engrenagens de um projeto maior?

Você tem as peças separadas com a ferramenta da **decomposição**, os padrões identificados com **reconhecimento de padrões** e o plano otimizado pela **abstração** de irrelevâncias só falta uma ferramenta da caixa a ser explorada.



Talvez você não lembre qual é mas olhe bem para essas peças... elas estão paradas. Para que qualquer coisa realmente aconteça, alguém precisa transformar essas ideias em ação.

É aqui que entra o quarto e último pilar: o **Algoritmo**. O pensamento algorítmico é como a bateria, a energia que faz com que todo o plano traçado até aqui finalmente entre em ação.

Muitas pessoas tremem só de ouvir essa palavra, imaginando códigos complexos de *hackers* em filmes. Mas, segundo **Wing (2006)**, algoritmos são, em essência, uma série de passos ordenados e precisos para resolver um problema.

Se o Pensamento Computacional fosse uma cozinha:



- a **Decomposição** seriam os ingredientes separados
- o **Reconhecimento de Padrões** seria o uso de uma receita de família que você sabe que sempre funciona
- a **Abstração** seria o processo de corte e descarte dos ingredientes estragados
- o **Algoritmo** seria a preparação do prato.

Neste capítulo, vamos aprender o que essa receita precisa conter para se tornar um prato aclamado.

Prepare-se: é hora de fazer o seu modelo rodar.

5.1. Criando a receita do sucesso

Uma lista de ingredientes, por mais abstraída e organizada que esteja, nunca vai se transformar em um bolo sozinha. Um mapa, por mais milimetricamente preciso que seja, não te transporta subitamente até seu destino.

Você já tem os objetos, lapidados pelas ferramentas conhecidas até agora, mas sem uma ação metódica eles não são nada! Precisamos desesperadamente dos verbos dessa frase, o *como* fazer.

É aqui que entra o Pensamento Algorítmico.

Segundo Donald Knuth, considerado pai da análise de algoritmos, a definição de um algoritmo é similar ao de “receita”, “procedimento”, “rotina”, “método”, todas exprimindo a essência de conjunto de regras que dá uma sequência de operações para resolver um tipo específico de problema

Um algoritmo possui cinco características determinantes que não necessariamente estão presentes nos seus sinônimos:

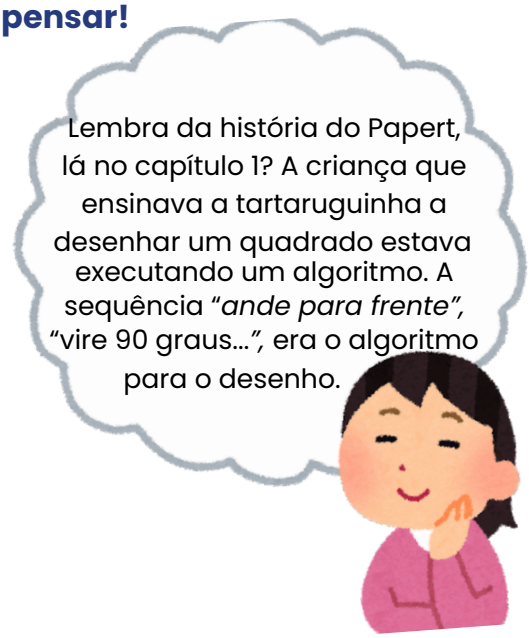
- **Finitude:** deve ter um limite de um número finito de passos.
- **Definição:** os passos devem ser precisos, não devem possuir instruções ambíguas ou espaço para considerações.
- **Entrada:** sempre deve possuir um estado inicial, uma situação, ocasião ou objeto, ou várias delas, que precisam ser modificadas.
- **Saída:** deve ter um resultado. Um ponto de chegada que seja diretamente relacionado e proporcional as entradas.
- **Efetividade:** um algoritmo é efetivo, todas as suas operações devem ser suficientemente básicas para que possam, em princípio, ser realizadas com exatidão e em um tempo finito.

O Algoritmo é o ato de traçar a rota. É a capacidade de pegar aquelas peças que você organizou e definir uma sequência de passos lógicos, precisos e finitos para alcançar um objetivo.

Na festa da Manu: com a sua lista de passos em mãos, você fez com que tudo acontecesse no tempo certo: o momento da música, DEPOIS, quando cortarem o bolo tocar o parabéns, se acabarem os salgadinhos na mesa, repor com doces, e assim por diante.

Você criou o algoritmo da festa perfeita!

Para pensar!



Muitas pessoas acham que os algoritmos são exclusivos de programadores e computadores da NASA. Mas a verdade é que, de uma forma ou outra, você os executa desde o momento em que acorda:

1. **Entrada:** o despertador toca.
2. **Decisão:** se estiver atrasado, vista a blusa que estava secando no varal. Senão, vista a camisa passada.
3. **Ação:** escove os dentes. (um movimento de repetição).
4. **Saída:** você termina sua higiene matinal e está pronto para fazer o café.

1. **Entrada:** preparar o café da manhã.
2. **Decisão:** se o pão for velho, faça uma torrada. Senão, passe manteiga no pão (movimento de repetição).
3. **Ação:** passe o café.
4. **Saída:** você se alimenta e está pronto para sair.

Essa é a estrutura básica de um algoritmo com estrutura de seleção e estrutura de repetição, nomes técnicos que você conhecerá no próximo capítulo, mas já pode perceber que não está tão distante do que você conhece.

Olhando para os algoritmos acima, podemos perceber que não foram dadas instruções muito precisas, embora, por reconhecimento de padrões você saiba o que fazer. O computador é uma máquina incrivelmente rápida, mas, ironicamente, ele não é "inteligente" como nós.

Ele não tem intuição. Ele não sabe passar café ou mesmo levantar da cama. Ele precisa que você diga exatamente o que fazer, quando fazer e quantas vezes repetir.

Imagine que você contratou um super robô assistente de última geração. Ele é rápido, atencioso e segue todos os seus comandos cegamente. Para sua infelicidade, porém, ele não tem senso comum ou inteligência

Se você disser *"Vire à direita"*, ele vira. Se você disser *"Pule da ponte"*, ele pula. Se você disser *"Passe manteiga no pão"*, mas não disser para **abrir o pote antes**, ele vai esfregar o pote de plástico fechado na fatia de pão.



Essa é a sacada! O robô é o seu computador. Ele precisa de um algoritmo tão claro que não deixa ambiguidades para a execução da tarefa, sempre chegando ao mesmo resultado.

Momento da dica...



Há um vídeo viral no YouTube que retrata bem a ideia. Um pai, no intuito de ensinar lógica de programação aos filhos, propõe um desafio aparentemente simples: eles devem escrever as instruções para fazer um sanduíche de manteiga de amendoim e geleia.

A graça começa quando o pai começa a agir como um robô e executar apenas exatamente o que estiver escrito no papel, sem usar nenhum tipo de bom senso ou intuição.

Procure no YouTube pelo vídeo: ***"Exact Instructions Challenge - THIS is why my kids hate me. | Josh Darnit"***

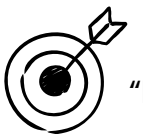
5.2. Plano de ação

Para nossa sorte, nós não somos robôs. Nosso cérebro é uma máquina incrível de improviso: se você vai pegar um copo d'água e a porta da cozinha está fechada, você a abre sem nem pensar. Se os copos estão sujos, você pega uma xícara.

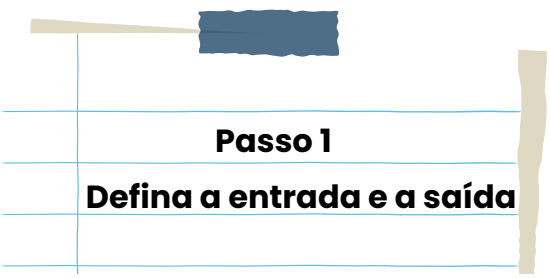
Portanto, criar um algoritmo pode parecer difícil porque estamos acostumados a pular etapas mentalmente para cumprir objetivos. Mas para criar uma solução replicável, precisamos ser metódicos e considerar todas as circunstâncias, você não pode tomar seus conhecimentos como base e sim o mínimo básico para realização.

Não basta fazer uma lista de tarefas, precisamos de passos. Um algoritmo precisa de fluxo. Vamos ver como transformar sua lista em um algoritmo acessível?

Que tal aplicar a lógica algorítmica em um cenário clássico e que exige paciência de monge: ensinar tecnologia.



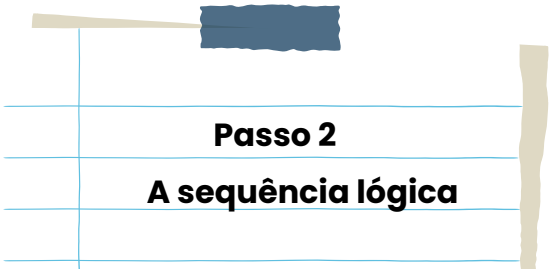
Aqui vai a nossa última missão:
"Ensinar sua mãe a criar uma conta no Google"



Todo algoritmo precisa saber as ferramentas a serem usadas e onde ele quer chegar.

Não há espaço para andar em círculos.

- **Entrada:** computador ou celular com internet, sua mãe, papel e caneta.
- **Saída:** um endereço de e-mail funcionando e a senha salva em local seguro.



Já ouviu aquele ditado popular misturado com axioma da matemática: ***"A ordem dos fatores não altera o produto"?***

Bom, **em algoritmos, altera sim**. O computador lê uma linha de cada vez.

Se você, por exemplo, guardar uma nova informação em uma pasta que antes tinha algo importante e depois tentar procurar, a informação antiga não vai estar mais lá.

O computador não vai achar. É preciso de ordem. Não adianta pedir pra sua mãe criar a senha antes de preencher o nome e dados.

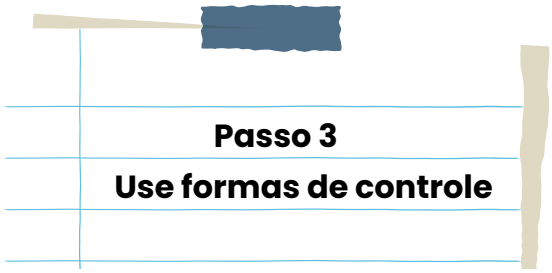
Veja a diferença entre uma lista bagunçada e um algoritmo:

1. Abrir o navegador de internet.
2. Digitar google.com.br na barra de endereços.
3. Clicar no botão azul "Fazer Login".
4. Clicar na opção "Criar conta".
5. Preencher Nome e Sobrenome.
6. Inserir a Data de Nascimento.



Note que cada passo depende do anterior.

Você não consegue fazer o passo 5 se não tiver feito o passo 1.



Aqui está o segredo que separa uma lista simples de um algoritmo inteligente.

A vida não é uma linha reta; contratempos acontecem. O endereço do e-mail pode já existir. A internet pode cair. Há formas de prever isso pelas **estruturas de controle.**

- **Capacidade de reação (SE -> ENTÃO -> SENÃO)**

  Confere a capacidade de reagir as adversidades.
  São as chamadas **bifurcações:** mais de uma opção de caminho a seguir.
  E se o nome da sua mãe for Maria dos Santos? Com um nome tão brasileiro e comum é quase impossível conseguir um usuário como @mariadossantos.

  *"**SE** o nome de usuário não está disponível, **ENTÃO** tente adicionar o número da sorte da sua mãe ao final: @mariadossantos7. **SENÃO** siga para a próxima página, pois o nome é válido"*

- **Poder da repetição (LOOP)**

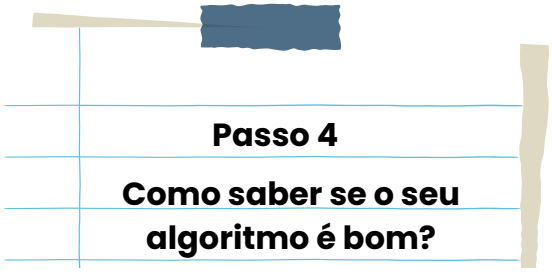
  Já pensou se você tivesse que explicar para sua mãe que ***SE*** o nome de usuário @mariadossantos7, não está disponível, ***ENTÃO*** ela deve usar @mariadossantos7. E se esse por vez não estiver também, ela deve escolher @mariadossantos8. Que se estiver indisponível novamente deve ser substituído por @mariadossantos9... Seria bem cansativo e ineficiente.

  Portanto, você pode apenas explicar para sua mãe que ***ENQUANTO*** não achar o nome de usuário disponível, ela deve continuar tentando.

  Na programação usamos laços de repetição (Loops) para criar uma regra que vale até o problema ser resolvido. Essa também é a lógica por trás dos avisos de senha fraca. O Google não aceita senhas como "123456" ou datas de aniversário. Portanto:

  ***ENQUANTO** a barra de força da senha não ficar verde, apague tudo. Misture letras maiúsculas, números e símbolos. Tente novamente.*

  O algoritmo prende o usuário nessa etapa até a condição ser satisfeita.



Passo 4

Como saber se o seu algoritmo é bom?

Antes de sentar com sua mãe (e gastar a paciência dela), faça o **"Teste do Robô"**.

Peça para um amigo seguir suas instruções **literalmente**, sem usar o bom senso.

- ·**Exemplo:** Você escreveu *"Digite a data de nascimento"*, mas não especificou o formato.
- **O Resultado:** Seu amigo digita *"hoje"* ou *"12 de maio de 2024"*.

O sistema do Google só aceita números (12/05/2024). O algoritmo falhou por falta de precisão.

Pare: Se o seu algoritmo prevê o erro do usuário (Decisão) e não deixa ele avançar com dados errados (Repetição), você criou um sistema robusto!

Quando você domina isso, você para de "torcer" para as coisas darem certo e começa a "programar" para que elas funcionem.

Momento da dica...



Use Fluxogramas!

Desenhe retângulos para as ações (Passo 2) e losangos para as perguntas (Passo 3).

Se você consegue desenhar o caminho com setas, você consegue resolver qualquer problema.

5.3. Guia para o professor

Neste capítulo trabalhamos o quarto pilar do Pensamento Computacional: **Pensamento Algorítmico.** Após dividir os problemas em partes menores, identificar os padrões e encontrar o que é essencial, o estudante está preparado para colocar o planejamento realizado em forma de algoritmo. O algoritmo é, portanto, uma sequência de passos bem definidos, com início e fim.

Objetivos de aprendizagem

Ao final da aula, os estudantes deverão ser capazes de:

- Compreender o conceito de Algoritmo.

- Construir uma sequência de passos bem definidos, identificando o início e o fim do algoritmo.
- Testar e revisar os algoritmos para que eles sejam bem precisos.

Roteiro sugerido de aula (aproximadamente 50 min)

- **Abertura:** reveja rapidamente com os estudantes as sequências de pilares já estudados:

  - Capítulo 1 - organizando o pensamento.
  - Capítulo 2 (Decomposição) – dividir o problema.
  - Capítulo 3 (Padrões) – ver o que se repete.
  - Capítulo 4 (Abstração) – identificar o que importa.
  - Capítulo 5 (Algoritmos) – criar passos bem ordenados.

- **Pergunte:**

  "Uma lista de ingredientes é capaz de fazer o bolo sozinha?"
  "Um mapa é capaz de levá-los a um destino?"

- **Explique:** O pensamento algorítmico é a capacidade de criar uma sequência de passos lógicos, com início e fim bem definidos, para atingir o objetivo desejado.

  Explique que o computador só executa o que é ordenado, ou seja se os passos não estiverem bem definidos, o computador não executará como se deseja.

- **Desenvolvimento:** **Dinâmica: "Teste do Robô"**

  - Separe a turma em grupos.
  - Um aluno será o programador e o outro será o robô (que executará os comandos exatamente como dado pelo aluno que representa o programador.
  - Lance um desafio para determinar ao robô como fazer um lanche ou arrumar sua mochila para a escola.
    - O robô deve executar literalmente cada comando.
  - Onde houver falhas, questione os alunos sobre como resolvê-las:
    - *Onde ocorreu a falha?*
    - *O comando foi correto?*
    - *A repetição foi adequada?*
    - *A decisão foi bem definida?*

- **Socialização:** Cada grupo compartilha:

    - Um erro identificado.
    - Uma forma de correção.
    - Quais características do algoritmo podem ser melhoradas.

O professor reafirma:

"Um algoritmo bem estruturado funciona sempre".

- **Reflexão final:** Conduza uma conversa questionando:

"Por que a ordem dos comandos importa?".
"O que acontece se o computador ou robô executar exatamente os comandos que você definiu?".

Avaliação

A avaliação deve observar se o aluno:

- Constrói um algoritmo coerente.
- Aplica comandos de decisão (SE/ENTÃO/SENÃO).
- Aplica comandos de repetição (ENQUANTO).
- Testa e revisa adequadamente o próprio algoritmo.

5.4. Sugestão de atividades



Atividade 1 – Rota do Entregador de Encomendas (20 a 25 min)

- **Objetivo:**

    - O aluno deve ser capaz de vivenciar um problema de otimização, entendendo que alguns problemas escalam de forma tão complexa que exigem otimizações especificas, forçando a busca por soluções alternativas.

- **Recursos necessários:**
    - Folhas A4 impressas com um mapa fictício contendo 5 pontos interligados por linhas com distâncias variadas anotadas.
    - Lápis e borracha.
    - Calculadoras (opcional).

- **Passo a passo:**
    - Organizar a turma em duplas e distribuir mapas do mesmo trajeto por dupla.
    - Missão: um entregador de aplicativo precisa sair do Ponto A, passar por outros 4 pontos (representando casas ou cidades) para deixar as encomendas, e voltar ao Ponto A percorrendo a menor distância total possível.
    - O entregador só deve passar por cada ponto de entrega no máximo uma vez.
    - Os alunos têm 10 minutos para traçar caminhos diferentes a lápis, somar as distâncias e tentar descobrir qual é a rota mais curta de todas.
    - O professor revela no quadro qual era, matematicamente, a rota mais curta.
    - Matemática e lógica: para 5 cidades, há 120 combinações de rotas possíveis. Mas se o entregador tivesse 10 entregas em diferentes pontos, o número de rotas possíveis saltaria para mais de 3 milhões!

- **Debate em sala:**
    - Seria trabalhoso replicar os detalhes da fotografia? Você acha que seria também para o computador?
    - O desenho geométrico é mentiroso ou é apenas uma abstração da realidade?
    - Nesses casos difíceis, o que é mais útil para o usuário: uma rota "perfeita" que o computador demora horas para descobrir, ou uma rota "boa o suficiente" calculada em 2 segundos?



Atividade 2 – O Tribunal dos Algoritmos (15 a 20 minutos)

- **Objetivo:**
    - O aluno deve ser capaz de analisar algoritmos defeituosos, identificar qual das 5 regras de Donald Knuth foi violada e reescrever a instrução para corrigir o problema.

- **Recursos necessários:**
  - Folhas A4 com os "Casos de Estudo" impressos.
  - Canetas ou lápis e papel.

- **Passo a passo:**
  - Organizar a turma em grupos de 3 a 4 alunos. Eles serão os "analistas de *software*".
  - O professor deve trazer o problema hipotético: *"Uma empresa de robótica faliu porque os seus robôs causaram desastres nas casas dos clientes."* A culpa foi dos programadores que esqueceram as regras de Knuth.
  - O professor distribui os 5 casos de falha para análise dos grupos:

    - **Robô Cozinheiro**: O robô recebeu a missão de fazer panquecas. A linha de código dizia: "Despeje massa na frigideira. Aqueça até a panqueca ficar com uma cor bonita e saborosa. Sirva no prato." O robô queimou a panqueca e ateou fogo à cozinha.
    - **Robô Arrumador**: O robô foi programado para organizar um quarto de criança. O código era: "1. Pegue um brinquedo do chão; 2. Coloque na caixa; 3. Volte ao passo 1." O quarto ficou limpo em 10 minutos, mas o robô continuou a tentar apanhar coisas invisíveis no chão até a bateria derreter 12 horas depois.
    - **Robô Matemático**: O robô foi programado para ajudar nas lições de casa da escola. O código dizia: "Some os dois números e guarde o resultado." O robô ficou parado a olhar para a parede sem fazer nada, porque a criança disse que eram maças a serem somadas e não números.
    - **Robô Entregador**: O robô precisava entregar uma encomenda urgente no Japão em 5 minutos. O código dizia: "Pegue na caixa, dê um salto gravitacional até Tóquio e entregue ao destinatário." O robô tentou saltar, mas ele não tem a capacidade de um salto gravitacional, caiu e a encomenda quebrou.
    - **Robô Calculadora:** O robô recebeu os números 5 e 10. O código dizia: "Multiplique os números e apague toda a sua memória para poupar energia." O robô fez o cálculo, apagou tudo, e quando o dono lhe perguntou o resultado, ele não soube responder.

  - Cada grupo deve analisar cada um dos casos, debater e sentenciar: qual regra (finitude, definição, entrada, saída ou efetividade) foi quebrada em cada caso?

- Após identificarem as quebras de regra, cada grupo deve escolher um dos casos e reescrever o algoritmo de forma correta, conforme visto no capitulo, cumprindo as 5 regras de Knuth.

- **Debate em sala:**
  - Após a correção, os grupos apresentam os seus vereditos. O professor(a) deverá questionar a turma:
    - No primeiro caso, como você reescreveria a instrução para que não houvesse "ambiguidade"?
    - No segundo caso, que forma de delimitação faltou para o robô parar?
    - No quarto caso, por que razão uma instrução não pode pedir o impossível? O que significa um passo ser "suficientemente básico"?
    - Olhando para estas 5 regras, concordam que programar não é sobre saber "falar com o computador", mas sim sobre saber pensar e traçar um plano de ação sem deixar nenhuma ponta solta?



Atividade 3 – Debate sobre a Rigidez da Máquina (15 a 25 minutos)

- **Objetivo:**
  - O aluno é convidado a refletir sobre como, na construção de algoritmos, regras muito rígidas podem criar soluções injustas no mundo real. Perceber o algoritmo como uma ferramenta que precisa de contemplar exceções.

- **Recursos necessários:**
  - Lousa
  - Marcador ou giz.

- **Passo a passo:**
  - Organize a turma em grupos.
  - Trazer um debate hipotético sobre a implementação de um novo projeto na escola: "um Algoritmo para a Formação de Grupos de Trabalho".

- A direção da escola decidiu aplicar um programa (algoritmo) que divide as turmas em grupos de forma "100% matemática e imparcial". A regra do algoritmo é:

    - *Passo 1. Organizar os alunos por ordem alfabética;*
    - *Passo 2. Agrupar de 4 em 4 alunos;*
    - *Passo 3. Não permitir trocas manuais arbitrárias.*

- Apresentação do problema:

"O algoritmo dividiu a sua turma em grupos. Separou melhores amigos, juntou alunos que se odeiam, e colocou 4 alunos com dificuldades em matemática juntos, sem ninguém para os ajudar. O algoritmo cumpriu as regras, mas o resultado não agradou a todos."

- Cada um dos grupos terá 5 minutos para debater seus argumentos.
- Um grupo deve defender a idealidade da decisão do algoritmo porque é "neutra".
- Os outros devem defender a "quebra" das regras do algoritmo para intervir humanamente e idealizar as preferências.

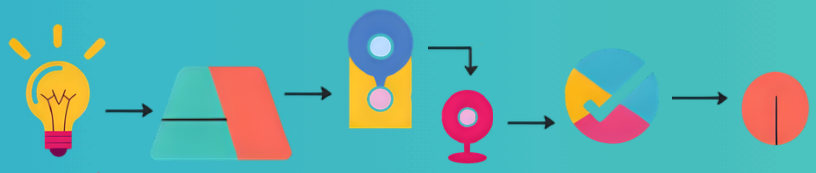
- **Debate em sala:**

    - Adiantaria ter um algoritmo super-rápido se ele não compreende a empatia ou as necessidades emocionais dos humanos?
    - Os algoritmos que usamos hoje, como as recomendações de vídeos curtos, às vezes erram de forma parecida, recomendando coisas que não se adequam?
    - Um algoritmo é neutro por natureza, ou ele pode carregar os preconceitos do programador que o escreveu? Se o programador não pensar nas exceções, o sistema falha com as pessoas?
    - É possível adicionar esse fator humano ao algoritmo de forma regrada?



Capítulo 6:

Princípios da lógica computacional



TEMA

6

Por dentro da BNCC:

Habilidades:

EF01CO01
EF01CO02
EM13CO01
EM13CO02
EM13CO04
EM13CO05
EM13CO19
EM13CO21

Após explorarmos os quatro pilares fundamentais do pensamento computacional, chegamos ao ponto que permite a articulação de todos esses conceitos: **a lógica computacional.**

Mais do que apenas um conjunto de regras matemáticas, a lógica funciona como a **"gramática"** do raciocínio estruturado fornecendo as ferramentas necessárias para validar os argumentos e garantir que as instruções dadas a uma máquina sejam coerentes, sem ambiguidades e exatas.

6.1. O enigma da porta misteriosa

Imagine que você está em um jogo de aventura. Diante de você há uma porta gigante com duas alavancas. Um aviso diz:

"A porta só abrirá se as duas alavancas forem puxadas ao mesmo tempo."

Você tenta puxar uma — nada acontece. Puxa a outra — nada também. Mas quando puxa as duas juntas... a porta se abre!

Sem perceber, você acabou de usar lógica computacional.



Segundo **Jeannette Wing (2006)**, o pensamento computacional envolve a capacidade de formular problemas e suas soluções de forma que um computador — humano ou máquina — possa executá-las de maneira eficaz.

Essa lógica define as condições, as sequências e as regras que determinam o que vai acontecer. É assim que o computador "pensa": **ele segue instruções com base em decisões lógicas.**

6.2 Conceitos fundamentais da computação

Um computador não tem imaginação — ele segue instruções.

Não basta dizermos *"arrume o quarto"*, o computador precisa que você diga:

1. Pegue as roupas do chão.
2. Dobre as roupas.
3. Guarde-as no armário.

Esse tipo de pensamento ordenado é o que chamamos de **lógica computacional** — uma forma de pensar de maneira sequencial e sem ambiguidades.

Lógica Computacional é a capacidade de pensar de forma estruturada e sequencial, criando instruções sem ambiguidades, que podem ser executadas de maneira eficiente.

De acordo com **Seymour Papert (1980)**, aprender a pensar como um computador significa aprender a pensar de forma estruturada e lógica — um processo que reforça a resolução de problemas.

6.3. Eixos da lógica computacional

Imagine que você está organizando uma festa com seus amigos. Há muito o que fazer — escolher a música, preparar os lanches, arrumar o espaço — e tudo precisa acontecer na ordem certa para que a festa seja um sucesso. Esse é o primeiro pilar da lógica computacional: **a sequência.**

6.3.1 Sequência - a ordem que importa

A sequência é o primeiro pilar da lógica computacional. Ela representa a ordem em que as ações devem acontecer para alcançar um objetivo. Sem uma boa sequência de passos, até tarefas simples podem dar errado. Quando você prepara uma festa, por exemplo, precisa seguir uma ordem lógica.

- Escolher o local da festa;
- Fazer a lista de convidados;
- Comprar os ingredientes e materiais;
- Decorar o ambiente;
- Só então, dar início à festa.

Se inverter a ordem — por exemplo, tentar decorar antes de escolher o local — as coisas podem sair bem confusas.

A ordem sequencial garante que cada passo aconteça no momento certo.



No computador, é a sequência que garante que um programa executará corretamente.

Vamos exercitar?



Crie um algoritmo narrativo mostrando a sequência lógica para fazer um sanduíche.

Dica: Liste os passos em ordem. Depois, leia e veja se faria sentido executar exatamente os passos que você definiu.

6.3.2 Decisão (condicional) → Quando há uma escolha a ser feita.

A decisão é o segundo pilar da lógica computacional. Ela aparece quando precisamos escolher entre caminhos diferentes, dependendo de uma condição. Ela permite escolher caminhos diferentes, de acordo com as condições do momento.

Na lógica computacional, é o famoso ***"Se... então..."***

"Se uma condição for verdadeira, faça isso; senão, faça aquilo."
Por exemplo: *"se estiver frio, vista um casaco."*



Voltando ao nosso exemplo, durante os preparativos da festa, você pode se deparar com escolhas.

Algoritmo: escolher o Local da Festa

1. Verificar a previsão do tempo.
2. Se estiver chovendo, montar a festa dentro de casa.
3. Senão, montar a festa no quintal.
4. Continuar com a decoração no local escolhido.

Por exemplo: *"O tempo está bom para uma festa ao ar livre?"* ou *"Há gelo suficiente nas bebidas?"*

Essas perguntas levam à decisões lógicas: se algo for verdadeiro, faça uma ação; senão, faça outra. No algoritmo acima se percebe claramente a condição nos itens 2 e 3. O que vai influenciar a decisão: *"Está chovendo ou não?"*.

Vamos exercitar?



Crie um algoritmo narrativo que inclua pelo menos uma condição ao fazer o lanche.

Por exemplo: Se estiver com fome, fazer um lanche; senão, estudar.

Essa decisão muda o caminho. Em um programa de computador, essa escolha é feita com comandos condicionais — ***if... else.***

6.3.3 Repetição - quando uma ação deve se repetir até uma condição mudar

A repetição é o terceiro pilar da lógica computacional. Ela permite repetir determinadas ações várias vezes, até que uma condição mude.

No problema da festa, há mais um detalhe: os convidados vão chegar, e você quer garantir que tudo esteja perfeito até lá. Então, você verifica várias vezes:

- Enquanto houver balões para encher, continue enchendo.

Algoritmo: Encher os Balões da Festa

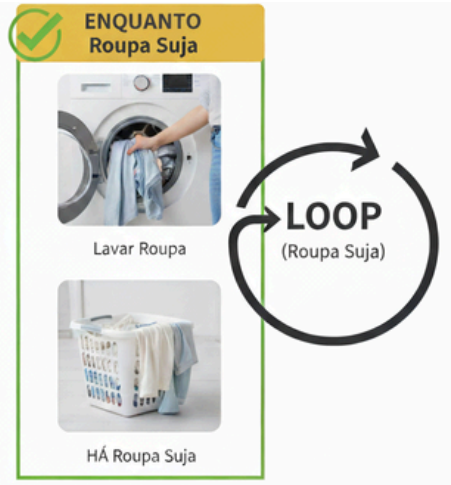
1. Pegar um balão.
2. Enquanto ainda houver balões no pacote, fazer o seguinte:
   a. Encher um balão.
   b. Amarrar o balão.
   c. Guardar o balão cheio.
3. Quando o pacote acabar, parar de encher.
4. Usar os balões na decoração.

Em programação, essas ações são realizadas por estruturas chamadas de **laços de repetição**, que fazem o computador **"repetir até que algo aconteça".**

A instrução do item 2: *"enquanto ainda houver balões"* mostra que a ação será repetida até que a condição mude (ou seja, até os balões acabarem).

Essa é a lógica da repetição — fazer algo várias vezes sem precisar escrever o mesmo comando de novo.

Exemplo: *"Enquanto houver roupa suja, continue lavando."*



Esses pilares formam a base de todo raciocínio lógico, seja em um programa de computador ou em uma simples rotina do dia a dia.

Vamos exercitar?



Crie um algoritmo narrativo que inclua uma ação repetitiva.

Por exemplo: Enquanto estiver tocando sua música favorita, continuar dançando.

6.4. Conectando os três princípios

A combinação dos três pilares: sequência, decisão e repetição permitem a elaboração de algoritmos consistentes.

Preparar uma Festa Completa:

1. Escolher o local e fazer a lista de convidados **(sequência).**
2. Se estiver chovendo, preparar o evento dentro de casa; senão, montar no quintal **(decisão)**.
3. Enquanto ainda houver balões para encher, continue enchendo **(repetição)**.
4. Quando tudo estiver pronto, começar a festa.

Esta estrutura nos leva a pensar em etapas, escolher o melhor caminho e repetir sequências automaticamente até atingir o objetivo. Agora que você já conhece cada pilar individualmente, veja como todos se unem na prática.

A lógica computacional está presente em tudo o que fazemos. Planejar uma festa, estudar para uma prova ou até jogar um game envolve seguir uma sequência, tomar decisões e repetir ações até alcançar o objetivo.

Pensar computacionalmente é ver o mundo como um conjunto de passos organizados — é um jeito de transformar ideias em ações e ações em resultados.

Você sabia que...



O conceito dos três pilares da lógica computacional — sequência, decisão e repetição — surgiu ainda nas primeiras linguagens de programação, na década de 1960.

Programadores como ***Edsger Dijkstra*** defenderam que qualquer programa, por mais complexo que seja, pode ser construído apenas com essas três estruturas.

Essa ideia ficou conhecida como **"Programação Estruturada"**, e é a base de como todos os softwares modernos são escritos até hoje.

Por isso, quando você cria um algoritmo com esses três pilares, está aplicando o mesmo tipo de raciocínio usado por programadores profissionais!

6.5. Guia para o professor

Neste capítulo 6, integramos os quatro pilares estudados anteriormente (Decomposição, Reconhecimento de Padrões, Abstração e Algoritmos) por meio das estruturas sequenciais, de decisão e repetição.

A lógica computacional funciona exatamente como uma outra linguagem qualquer, que tem suas regras para ser compreendida.

Objetivos de aprendizagem

Ao final da aula, os estudantes deverão ser capazes de:

- Compreender os conceitos de sequência, decisão e repetição.
- Criar algoritmos narrativos simples aplicando cada estrutura.
- Representar visualmente algoritmos utilizando cores ou fluxogramas.
- Desenvolver raciocínio lógico e clareza na formulação de instruções.

Roteiro sugerido de aula (aproximadamente 50 min)

- **Abertura:** apresente a história da "Festa Misteriosa", retomando o exemplo da organização de tarefas do capítulo 1.
- **Pergunte:**

"O que acontece se invertermos a ordem?"
"Quando precisamos repetir uma ação?"

- **Explique:** a lógica computacional é a capacidade de organizar ações, aplicando três princípios fundamentais:
    - Sequência – as ações devem acontecer em uma ordem específica.
    - Decisão - há momentos em que é preciso escolher um caminho com base em uma condição ("Se... então...").
    - Repetição - algumas ações precisam ser executadas várias vezes até que uma condição mude ("Enquanto...").

- **Desenvolvimento:** dinâmica: "Teste do Robô"
    - Separe a turma em grupos.

- Peça que os grupos transformem o algoritmo narrativo em uma representação visual (em cartaz ou blocos coloridos) mostrando as etapas do raciocínio lógico, com cores diferentes para cada pilar:
  - Azul: sequência.
  - Amarelo: decisão.
  - Verde: repetição.

- **Visualização:** mostre como as ações se conectam como blocos estruturados. Explique que todo programa de computador é construído a partir dessas três estruturas.

- **Socialização:** cada grupo apresenta seu processo, destacando as dificuldades e as decisões lógicas tomadas.

- **Reflexão final:** conduza uma conversa sobre como a lógica ajuda a pensar de forma mais clara e organizada.

Avaliação

- Observe a participação e o envolvimento dos alunos nas atividades práticas e nas discussões em grupo.
- Verifique se compreenderam os três pilares — sequência, decisão e repetição — e se conseguiram aplicá-los corretamente em seus algoritmos.
- Avalie a clareza das sequências e das decisões lógicas criadas pelos grupos.
- Promova uma autoavaliação reflexiva, perguntando:

"O que aprendi sobre como a sequência, a decisão e a repetição organizam o pensamento lógico?"

"Como transformar uma tarefa cotidiana em um algoritmo me ajudou a pensar de forma mais estruturada?"

6.6. Sugestão de Atividades



Atividade 1 - Sequência (20 a 25 minutos)

- **Objetivo:**
  - O aluno deverá compreender que toda tarefa pode ser organizada em uma sequência lógica de passos para alcançar determinado resultado

- **Recursos necessários:**
  - Papel e lápis.
  - Modelos de fluxograma (opcional).

- **Passo a passo**
  - Organize a turma em grupos.
  - Cada grupo deverá escolher uma atividade do dia a dia, por exemplo: escovar os dentes; chegar à escola; estudar para a prova; ou buscar o lanche.
  - Os alunos deverão fazer uma lista numerada (o que chamamos de algoritmo narrativo) definindo cada passo para executar a tarefa escolhida.
  - Lembre os alunos que devem ser o mais claros possíveis em cada item da lista (comando).

- **Debate em sala**

Após concluírem a atividade, um aluno do grupo deverá ler em voz alta o algoritmo criado.

O professor(a) deverá questionar a turma:
- É possível executar a tarefa com as instruções definidas pelo grupo?
- Há alguma sugestão para deixar a execução mais adequada?
- Alguém faria diferente?



Atividade 2 - Decisão (20 a 25 minutos)

- **Objetivo:**
  - O aluno deverá entender como as decisões lógicas funcionam.

- **Recursos necessários:**
  - Papel e lápis.
  - Modelos de fluxograma (opcional).

- **Passo a passo**
    - Professor(a) apresenta aos alunos uma situação-problema, por exemplo: *"Um estudante acorda atraso para ir à escola..."*.
    - Peça aos alunos que respondam por meio de um algoritmo narrativo:
        - O que acontece se o aluno acordar atrasado?
        - O que acontece se não acordar atrasado?
        - E se o aluno perder o transporte?
        - E se o aluno chegar atrasado na escola?
    - Os alunos deverão organizar suas respostas utilizando os termos (SE/SENÃO) e apresentar um fluxograma usando setas e símbolos para demonstração.

- **Debate em sala**

Após concluírem a atividade, o professor(a) deverá destacar:

- Que toda decisão depende uma uma condição lógica (verdadeira ou falsa).
- Que, como na vida real, o computador também faz escolhas baseando-se em condições.



Atividade 3 - Repetição (20 a 25 minutos)

- **Objetivo:**
    - O aluno deverá identificar as ações que se repetem e a importância da condição de parada.

- **Recursos necessários:**
    - Papel e lápis.
    - Modelos de fluxograma (opcional).

- **Passo a passo**
    - Proponha uma tarefa repetitiva, como contar os alunos da sala de aula ou organizar as cadeiras da sala.
    - Peça aos alunos para descreverem a tarefa de duas formas:

- Escrevendo cada repetição separadamente.
- Usando uma frase que indique a repetição, por exemplo:
  - *"Repita até acabar..."*;
  - *"Enquanto houver alunos..."*;
  - *"Para cada cadeira da sala..."*;

- **Debate em sala**

  - Compare as duas descrições.
  - Mostre como a estrutura de repetição facilita a instrução.
  - Lembre os alunos que toda estrutura de repetição deve ter um início e um fim definido.



Atividade 4 - Pensando como um computador (30 a 40 minutos)

- **Objetivo:**

  - O aluno deverá ser capaz de integrar sequência, decisão e repetição na resolução de problemas.

- **Recursos necessários:**

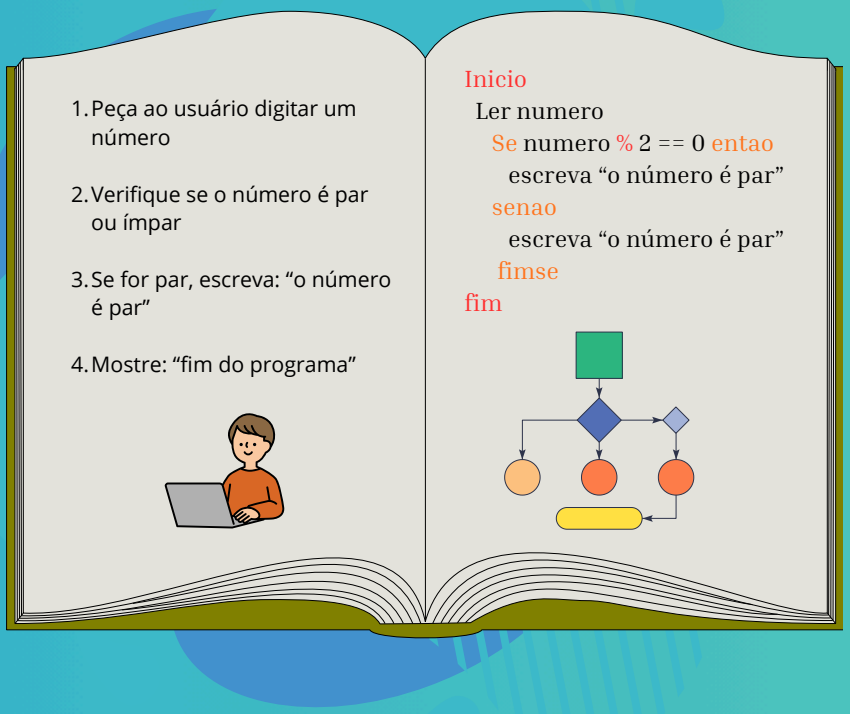
  - Papel e lápis.
  - Modelos de fluxograma (opcional).

- **Passo a passo**

  - Professor(a) solicita aos alunos que criem uma situação que envolva:
    - um sequência de ações.
    - pelo menos uma decisão.
    - pelo menos uma repetição.
  - Exemplo de situações: fazer um café, trocar um pneu, retirar um livro na biblioteca, etc.
  - Os alunos deverão, ao final da atividade, identificar onde ocorreu cada conceito na lógica apresentada.

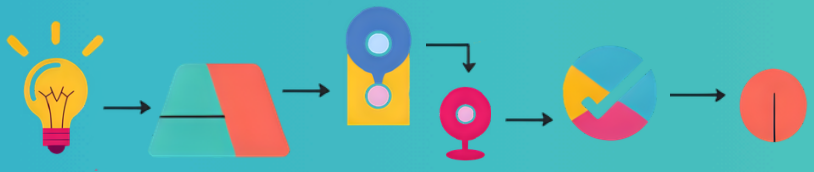
- **Debate em sala**

    - Se você fosse um computador executando esse algoritmo, em qual passo teria mais dificuldade de entender a instrução?
    - Existe algum passo que poderia gerar dúvida na hora da execução?
    - O que aprendemos com essa atividade sobre como os computadores resolvem os problemas?



Capítulo 7:

Representações da Lógica Computacional



TEMA 7

Por dentro da BNCC:

Habilidades:

EF01CO01
EF01CO02
EM13CO01
EM13CO02
EM13CO04
EM13CO05
EM13CO11
EM13CO19

No capítulo 6, apresentamos a lógica computacional, explorando os princípios de sequência, decisão e repetição.

Após entender a lógica básica, neste capítulo, iniciaremos a aplicação mais próxima da linguagem de programação, aprendendo as **representações da lógica computacional.**

Neste capítulo vamos transformar a lógica computacional em **instruções práticas**. Para tanto, vamos iniciar com os algoritmos narrativos, os pseudocódigos, os fluxogramas e até a lógica de blocos em que você poderá testar sua lógica de forma prática.

7.1. Algoritmos narrativos: contando passos para o pensamento

Como explica Brackmann et *al.* (2017), compreender as estruturas de sequência, decisão e repetição é essencial para que os alunos possam transferir a lógica do pensamento computacional para diferentes áreas do conhecimento.

Então, antes de escrever códigos, é importante entender o que é um **algoritmo**. A palavra parece complicada, mas significa algo simples:

Você já usa algoritmos o tempo todo!

Quando ensina alguém a fazer um bolo, está descrevendo um algoritmo:

Um algoritmo é um conjunto de instruções organizadas para realizar uma tarefa.

1. Quebre os ovos.
2. Misture com farinha e açúcar.
3. Coloque no forno.
4. Espere assar.
5. Retire e sirva.

Cada passo é uma instrução, e todos juntos formam uma sequência lógica. Quando descrevemos esses passos usando palavras e histórias, temos um algoritmo narrativo.

Exemplo de Algoritmo Narrativo: "tomando um suco"

O texto a seguir é um algoritmo narrativo — uma forma natural e simples de expressar uma sequência lógica.

Para fazer um suco de laranja:

- Primeiro, pegue as laranjas;
- Depois, corte-as ao meio;
- Esprema o suco em um copo;
- Se quiser, adicione açúcar;
- Mexa bem e beba!



Ele mostra claramente a ordem das ações e até uma condição (*"se quiser, adicione açúcar"*).

A partir dos algoritmos narrativos, podemos dar o próximo passo: representar o mesmo raciocínio de forma estruturada, transformando a narração em **pseudocódigo**.

```
Início
    Ler dados de entrada
    Processar (fazer algo com os dados)
    Escrever o resultado (saída)
Fim
```

7.2. Pseudocódigo: falando com o computador em português

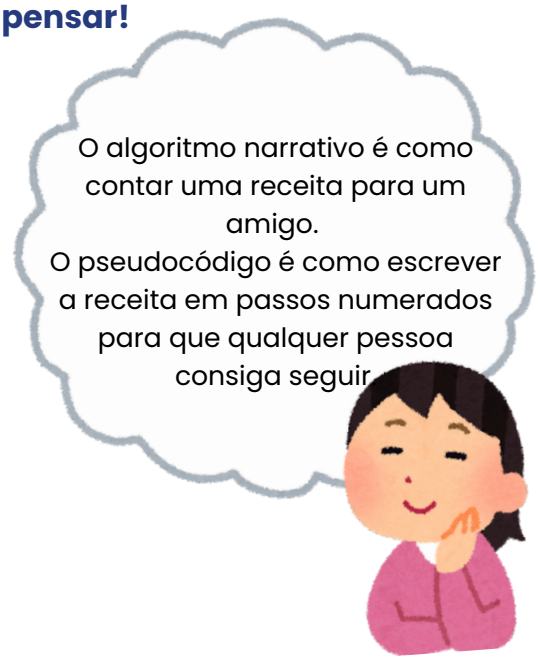
Depois de aprender a criar algoritmos narrativos, chegou a hora de transformá-los em uma forma mais próxima da linguagem dos computadores — o pseudocódigo.

Mas não se assuste com o nome: “pseudo” significa “falso”, e “código” vem de “programa”. Ou seja, pseudocódigo é um jeito de escrever um programa de forma parecida com o português, sem precisar usar nenhuma linguagem de programação de verdade.

O que é o pseudocódigo?

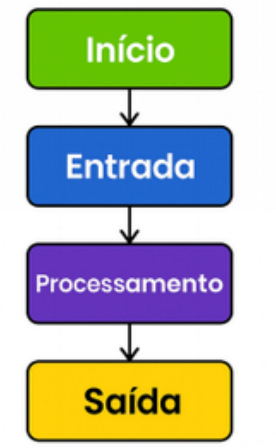
O pseudocódigo serve como ponte entre o raciocínio humano e a linguagem do computador. Ele ajuda a organizar as ideias, deixando claro o que deve acontecer e em que ordem. É como transformar uma história (o algoritmo narrativo) em um conjunto de instruções precisas.

Para pensar!



Pensando em etapas

Um pseudocódigo segue sempre uma sequência de passos bem definidos, como uma receita. Veja sua estrutura básica:



Todo algoritmo segue uma sequência lógica do começo ao fim.

Essas quatro partes — **Início, Entrada, Processamento e Saída** — estão presentes em praticamente qualquer programa.

Fazendo suco de laranja, agora em pseudocódigo:

Imagine que o computador precisa preparar um suco de laranjas. O pseudocódigo seria algo assim:

```
Início
    Pegar laranjas
    Cortar ao meio
    Espremer suco no copo
    Perguntar: deseja adicionar açúcar?
    Se resposta = sim então
        Adicionar açúcar
    Mexer bem
    Beber suco
Fim
```

Perceba como o pseudocódigo deixa o raciocínio ainda mais organizado. Cada passo está em uma linha, e a decisão (adicionar ou não açúcar) aparece claramente com o comando *"Se... então..."*.

Observe:

- **Entrada:** a pergunta *"Deseja adicionar açúcar?"* é o momento em que o programa recebe uma informação do usuário.
- **Processamento:** "Se resposta = sim" é a decisão lógica — o algoritmo escolhe o caminho a seguir.
- **Saída:** "Beber suco" representa o resultado final da execução.

Momento da dica...



Antes de escrever pseudocódigos longos, descreva a tarefa em palavras simples.

Depois, vá transformando cada verbo em uma instrução direta e curta.

Essa é a transição natural entre pensamento humano e raciocínio lógico

Estruturas básicas de um pseudocódigo

Como vimos até aqui, o pseudocódigo é uma linguagem mais parecida com a de um programa de computador. Para criá-lo usamos algumas estruturas comuns:

Estrutura	Função	Exemplo
Início / Fim	Delimitar o algoritmo	Início ... Fim
Ler	Entrada de dados	Ler idade
Escrever	Saída de informações	Escrever "Bem-vindo!"
Se... então... senão	Tomar decisões	Se nota ≥ 6 então "Aprovado"
Enquanto / Para	Repetir ações	Enquanto contador ≤ 5 faça ...

Com as estruturas apresentadas na tabela acima é possível descrever pseudocódigos que facilitarão sua transcrição para qualquer linguagem de programação no futuro.

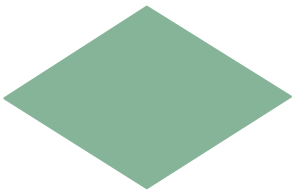
7.3. Fluxograma: desenhando o raciocínio

Agora, imagine transformar o algoritmo em um mapa visual. Cada ação vira uma forma geométrica e as setas mostram o caminho do raciocínio.
Esse mapa visual é o **fluxograma** — uma ferramenta poderosa para visualizar decisões e repetições e entender, de forma rápida, o que acontece em cada etapa.

Cada forma tem um significado:



Retângulo → uma ação



Losango → uma decisão (tipo "sim" ou "não")

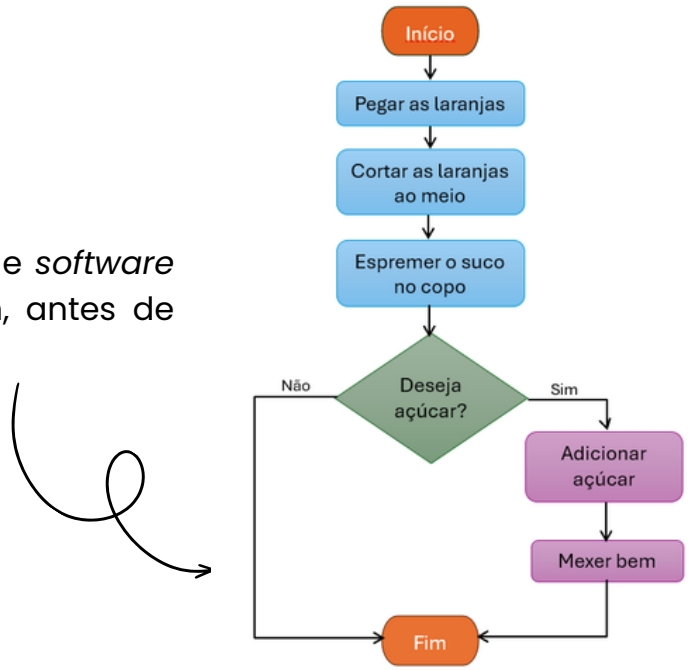


Setas → mostram o caminho

Exemplo: "Fazer suco de laranja"

Início
Pegar laranjas
Cortar ao meio
Espremer suco no copo
Perguntar: ***deseja adicionar açúcar?***
Se resposta = sim então
Adicionar açúcar
Mexer bem
Beber suco
Fim

Muitos programadores e *designers* de *software* começam desenhando ideias assim, antes de escrever uma linha de código.



7.4. Lógica de blocos

Antes de escrever códigos de verdade, podemos representar ações usando blocos visuais que se encaixam como peças de um quebra-cabeça colorido.

Cada bloco tem uma função específica: alguns iniciam ações, outros repetem comandos, verificam condições ou encerram tarefas.

Esses blocos formam uma linguagem visual simples e intuitiva, que permite "ver o pensamento lógico acontecer" — um passo essencial antes de aprender a programar com palavras e símbolos.

A lógica que se monta com as mãos

Imagine que você quer ensinar um robô virtual a andar, pular e cumprimentar alguém. Você pode construir o raciocínio assim:



Bloco 1: "Quando o jogo começar" → define o **início da ação.**

Bloco 2: "Ande 10 passos" → uma **ação direta**.

Bloco 3: "Diga 'Olá, tudo bem?' por 2 segundos" → uma **mensagem**.

Bloco 4: "Espere 1 segundo" → cria uma **pausa lógica**.

Bloco 5: "Repita 3 vezes" → representa a **repetição de um comando**.

Ao encaixar esses blocos, você cria um miniprograma — uma sequência lógica que o computador pode executar sem erros.

Aplicações práticas em sala de aula

Mesmo antes de usar o computador, a lógica de blocos pode ser **simulada em papel:**

- Os alunos podem recortar blocos coloridos e encaixá-los manualmente, representando ações;
- Cada cor representa um tipo de bloco:

Verde = Início
Azul = ação
Roxo = diálogo
Laranja = repetição
Amarelo = decisão
Vermelho = fim

Depois, os alunos podem comparar suas "pilhas de blocos" e discutir quais sequências funcionam e quais precisam ser corrigidas — como um "*debug* visual".

Você sabia que...



O primeiro computador moderno, **chamado ENIAC**, só funcionava com milhares de instruções digitadas manualmente!

Para que ele fizesse uma simples soma, era preciso planejar toda a sequência lógica de passos.

Hoje, o computador faz isso em segundos, mas a lógica por trás é a mesma que usamos em pseudocódigos e fluxogramas.

7.5. Guia para o professor

Neste capítulo saímos da lógica simples para a sua representação por meio de algoritmos narrativos, pseudocódigos e blocos visuais. O principal objetivo está em colocar numa linguagem mais próxima da programação toda lógica desenvolvida nos capítulos anteriores.

Objetivos de aprendizagem

Ao final da aula, os estudantes deverão ser capazes de:

- Compreender os fundamentos da lógica computacional.
- Transformar ações cotidianas em algoritmos narrativos, pseudocódigos e fluxogramas.
- Criar lógica de blocos, reforçando o raciocínio sequencial.
- Construir soluções lógicas e criativas.

Roteiro sugerido de aula (aproximadamente 50 min)

- **Abertura:** apresente a história da “Porta Misteriosa” para introduzir o conceito de lógica computacional.

- **Pergunte:**

"O que precisa acontecer antes que a porta se abra?"
“Existe apenas um caminho possível?”
“O que acontece se uma condição não for atendida?”

- **Explique:** a lógica computacional não é apenas sobre computadores, mas sobre organizar o pensamento de forma estruturada.

- **Exploração:**
  - Separe a turma em grupos.
  - Proponha que descrevam uma tarefa cotidiana (por exemplo: fazer um lanche, escovar os dentes ou organizar a mochila) em formato de algoritmo narrativo.
  - Oriente com perguntas:

"Qual é o primeiro passo?"
"A ordem pode ser alterada?"
"Existe algum momento de escolha?"
"Há alguma ação que precisa ser repetida?"

- **Desenvolvimento:** dinâmica: "Seu Pseudocódigo"
    - Peça que cada grupo transforme o algoritmo narrativo em pseudocódigo, utilizando comandos estruturados (INÍCIO, FIM, SE, SENÃO, ENQUANTO).
    - Em seguida, solicite que os estudantes representem o mesmo algoritmo em formato de fluxograma, utilizando papel, cartolina ou blocos coloridos.

- **Visualização:** apresente a ilustração dos blocos.
    - Explique como cada bloco representa uma estrutura lógica:
        - Bloco de ação → sequência.
        - Bloco de decisão → condição.
        - Bloco de repetição → laço.
    - Mostre que a lógica de blocos facilita a visualização do raciocínio, organizando o fluxo das ações de maneira clara e estruturada

- **Socialização:** cada grupo apresenta seu processo, destacando as dificuldades e as decisões lógicas tomadas.

- **Reflexão final:** conduza uma conversa sobre como o pensamento lógico ajuda a organizar ideias e resolver problemas com clareza. Pergunte:

"O que mudou quando transformamos a ideia em pseudocódigo?"
"Qual representação foi mais clara?"
"Por que a lógica precisa ser precisa?"

- **Avaliação:**
    - Observe a participação e o envolvimento dos alunos nas atividades práticas.
    - Verifique se compreenderam a diferença entre algoritmo narrativo, pseudocódigo e fluxograma.
    - Avalie a clareza das sequências e das decisões lógicas criadas pelos grupos.
    - Promova uma autoavaliação reflexiva, perguntando:

"O que aprendi sobre pensar de forma lógica e organizada?"
"Como a lógica de blocos me ajudou a visualizar o raciocínio?

7.6. Sugestão de Atividades



Atividade 1 - Decisão (20 a 25 minutos)

- **Objetivo:**
  - Compreender a estrutura lógica das ações e decisões, utilizando algoritmos narrativos, pseudocódigos e fluxogramas.

- **Recursos necessários**
  - Papel e lápis
  - Cartolina e canetas coloridas
  - Modelos de fluxograma (opcional)

- **Passo a passo**
  - Escolha um desafio simples: *"preparar o lanche"*, *"escovar os dentes"*, *"ir para a escola"*.
  - Escreva um algoritmo narrativo contando a sequência das ações.
  - Transforme o algoritmo narrativo em pseudocódigo, usando palavras curtas e objetivas.
  - Desenhe o fluxograma do mesmo processo.
  - Troque o trabalho com outro grupo e tente seguir as instruções exatamente como estão.
  - Depois, discutam: o que funcionou? O que ficou confuso? O que poderia ser mais claro?

- **Debate em sala**
  - O algoritmo criado por outro grupo foi fácil de entender e executar?
  - Qual das formas (algoritmo narrativo, pseudocódigo ou fluxograma) facilitou mais a compreensão da tarefa?
  - Se um computador tivesse que executar o algoritmo apresentado, ele conseguiria fazer a tarefa apenas com as instruções escritas?



Atividade 2 - Repetição (20 a 25 minutos)

- **Objetivo:**
    - Os alunos deverão compreender a estrutura lógica das ações e decisões, utilizando algoritmos narrativos, pseudocódigos e fluxogramas.

- **Recursos necessários:**
    - Papel e lápis.
    - Cartolina e canetas coloridas
    - Modelos de fluxograma (opcional).

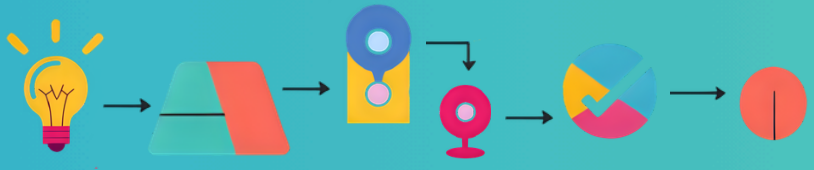
- **Passo a passo**
    - Escreva um algoritmo narrativo definindo a sequência das ações.
    - Transforme o algoritmo narrativo em pseudocódigo, usando palavras curtas e objetivas.
    - Desenhe o fluxograma do mesmo processo.
    - Troque o trabalho com outro grupo e tente seguir as instruções exatamente como estão.

- **Debate em sala**
    - O que funcionou?
    - O que ficou confuso?
    - O que poderia ser mais claro?



Capítulo 8:

Introdução ao Scratch



TEMA

8

Por dentro da BNCC:

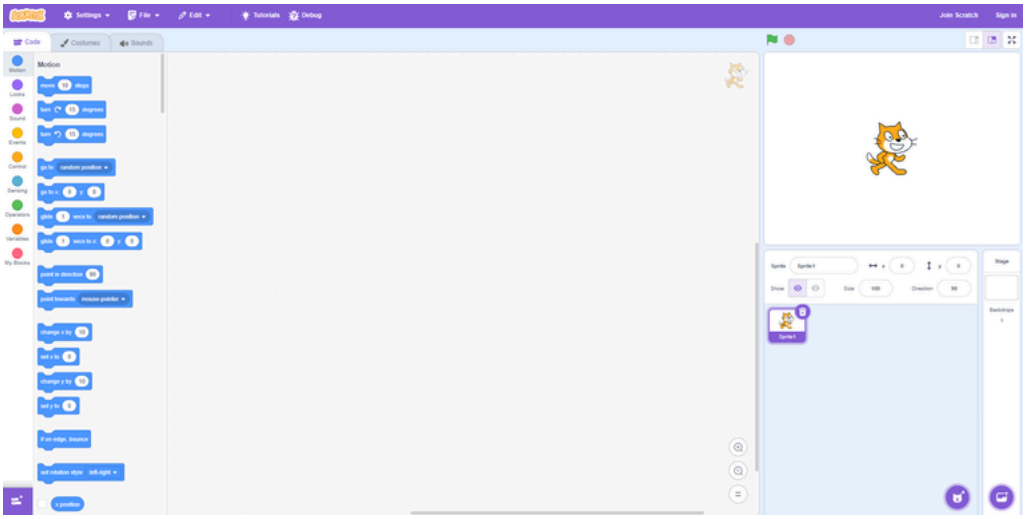
Habilidades:

EF01CO01
EF01CO02
EM13CO01
EM13CO024
EM13CO4
EM13CO05
EM13CO11
EM13CO11

8.1. O que é *Scratch*?

O *Scratch* é considerado uma **linguagem de programação**, criada no ano de 2007, pelo Instituto de Tecnologia de *Massachusetts* (*Massachusetts Institute of Technology* - MIT).

Possui uma interface dinâmica, interativa, com ícones e janelas de fácil manuseio, além de uma usabilidade prática e com atrativas para crianças.



Vamos lá, sabemos que somente ler nem sempre é a melhor forma de aprendizado, por isso enquanto acompanha a leitura a partir de agora, siga os passos a seguir:



1. Entre no site http://scratch.mit.edu/

2. Clique em Aderir ao *Scratch*

3. Crie um nome de usuário e escolha uma senha para seu acesso

4. Preencha seus dados reais conforme solicitado

5. Indique seu e-mail

6. Parabéns. Se tudo deu certo você já está na comunidade do *Scratch*.

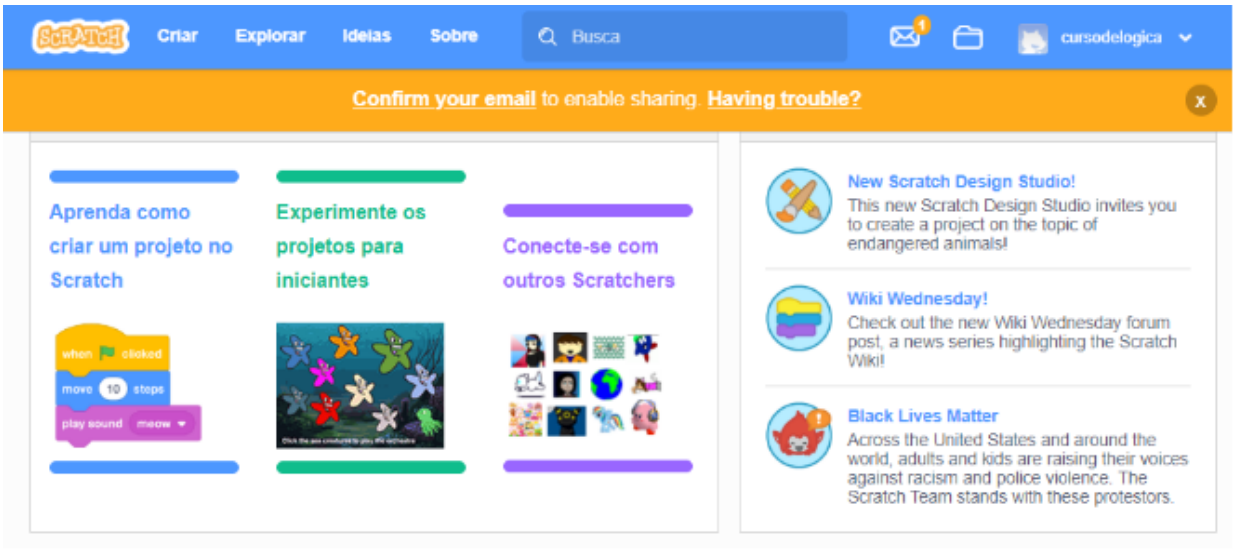
Você sabia que...



Você pode elaborar seus projetos sem criar uma conta, sem fazer todos esses passos:

- Basta acessar o site e clicar em **Criar** no canto superior esquerdo da tela.

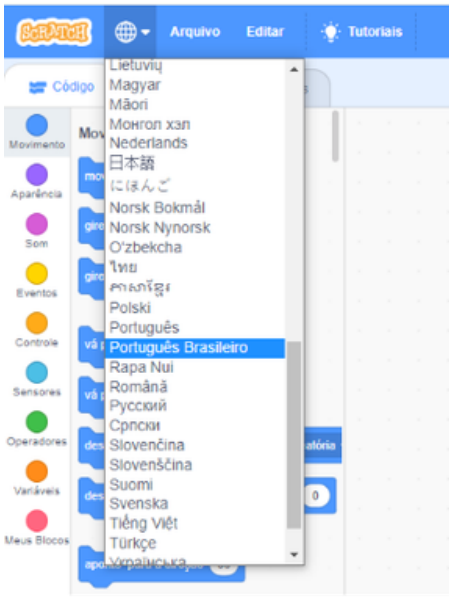
Mas lembre-se: sem o cadastro não será possível compartilhar seus projetos com outros professores.



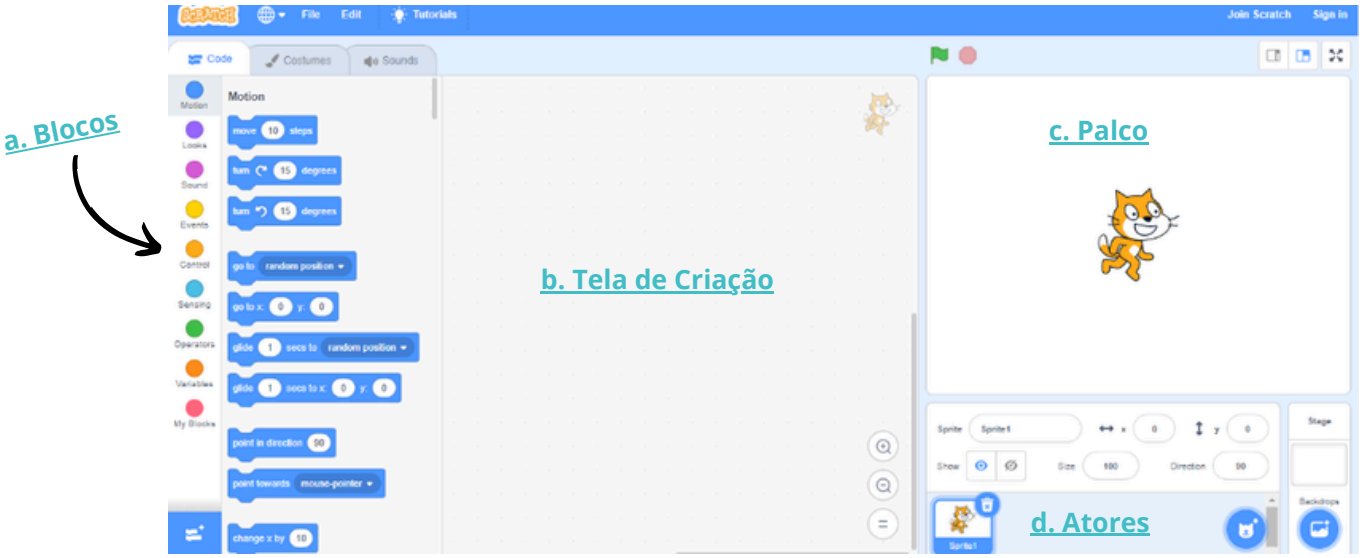
7. Agora você deve clicar em ***Confirm your email*** na tela principal. Somente após isso é que você poderá compartilhar seus projetos.

8. Após criada a conta, você pode clicar em ***Criar*** para elaborar o seu primeiro projeto na linguagem de programação Scratch.

9. Mas antes, configure o idioma para Português Brasileiro, clicando no ícone de globo e depois escolhendo o idioma.



8.2. Conhecendo a ferramenta



a. Blocos: são os comandos que vamos utilizar para criar nossos jogos; eles são divididos em categorias:

- Movimento
- Aparência
- Sons
- Eventos
- Controles
- Sensores
- Operadores
- Variáveis
- Meus blocos

Vamos detalhar rapidamente cada uma destas categorias:

Movimento

Serve para controlar a posição e a orientação do seu personagem (ator).

O que faz: Faz o ator andar, girar, ir para uma posição específica na tela ou apontar para uma direção.
Exemplo: *"Mova 10 passos"*.

Aparência

Controla o que o usuário vê e como o personagem se comunica visualmente.

O que faz: altera fantasias (roupas/posições), muda o tamanho, aplica efeitos visuais (como ficar fantasmagórico) e faz o ator exibir balões de fala ou pensamento.
Exemplo: *"Diga 'Olá!' por 2 segundos"*.

Som Responsável por toda a parte auditiva do projeto

O que faz: toca sons gravados ou da biblioteca, altera o volume e muda o tom (frequência) da música ou efeito sonoro.
Exemplo: *"Toque o som 'Miau' até o fim"*.

Eventos São os gatilhos que dão início às ações. Sem eles, o código não sabe quando começar.

O que faz: define o que acontece quando a bandeira verde é clicada, quando uma tecla é pressionada ou quando o ator é clicado.
Exemplo: *"Quando a bandeira verde for clicada"*.

Controle É o 'cérebro' da lógica, onde você decide o fluxo do programa.

O que faz: cria esperas, repetições (laços como "sempre" ou "repita 10 vezes") e condições (o famoso e já comentado "se...então").
Exemplo: *"Sempre: Mova 10 passos"*.

Sensores Permite que o programa interaja com o mundo ao redor (*mouse*, teclado e outros atores).

O que faz: detecta se o ator está tocando em uma cor, se uma tecla está pressionada ou qual é a posição do *mouse*.
Exemplo: *"Tocando em [Ponteiro do mouse]?"*.

Operadores Utilizado para cálculos matemáticos e comparações lógicas.

O que faz: faz somas, subtrações, escolhe números aleatórios e compara se um valor é maior, menor ou igual a outro.
Exemplo: *"Número aleatório entre 1 e 10"*.

Variáveis Serve para guardar informações que podem mudar durante o jogo.

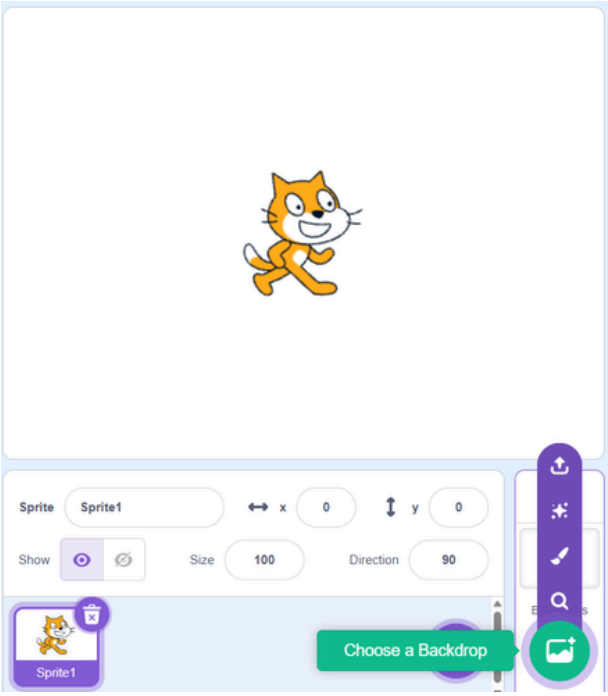
O que faz: armazena pontuações, tempo restante, nomes de jogadores ou vidas.
Exemplo: *"Adicione 1 a {Pontuação}"*.

Meus blocos Permite que você crie seus próprios comandos personalizados.

O que faz: Você agrupa vários blocos existentes e dá um nome a eles (como criar uma "função"). Isso ajuda a organizar códigos muito grandes e evitar repetições.
Exemplo: Criar um bloco chamado "Pular" que engloba subir e descer.".

---

b. Tela de criação: onde serão colocados/agrupados os blocos e editados de acordo com o objetivo do jogo.



c. Palco: é uma pré visualização do jogo. Nesta tela o usuário consegue ver como está ficando sua produção.

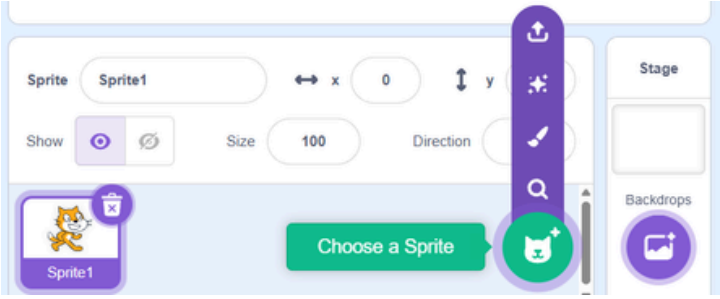
Logo acima do palco temos os seguintes botões:

Representa o *play* do jogo.

Representa o *pause* do jogo.

É possível alterar a imagem de fundo do palco, acessando o menu ***choose a backdrop*** logo abaixo.

d. Atores: neste espaço irão aparecer todos os personagens que você escolher adicionar no seu jogo.



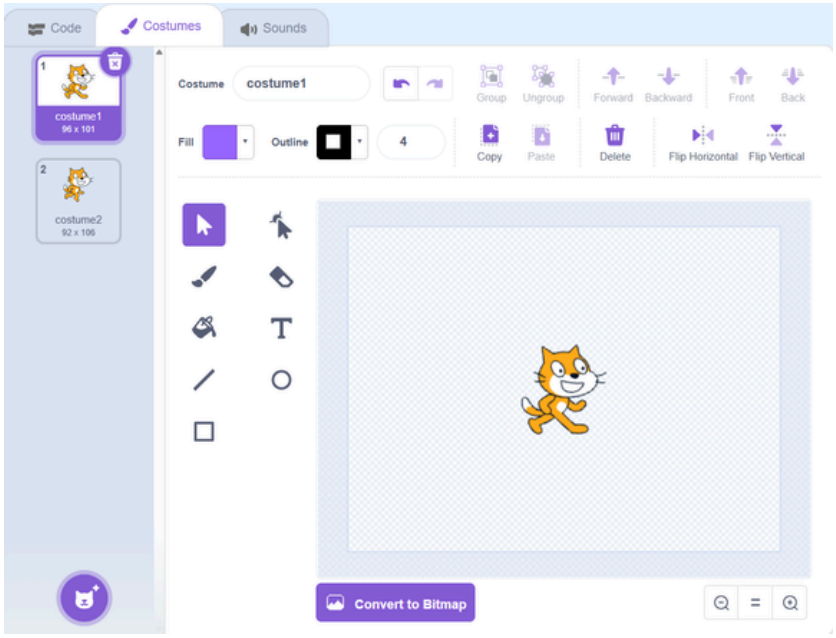
Para trocar o personagem, você deve clicar na opção ***choose a sprite*** que se encontra logo abaixo do palco.

O Scratch disponibiliza mais de 300 opções de personagens diferentes, entre eles: pessoas, objetos, animais, números e letras.

Também é possível editar as fantasias (que podemos entender como as diversas posições que o mesmo personagem pode apresentar) e editar seus respectivos sons.

Para isso, basta acessar:

- ***Costumes:*** para alterar a fantasia.
- ***Sounds:*** para alterar o sons emitidos pelos personagens.



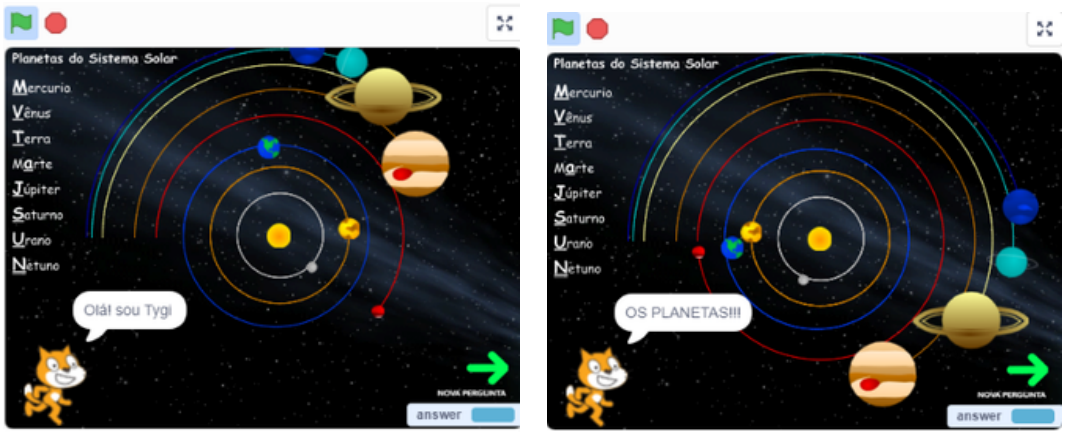
8.3. Casos reais na educação

Agora veremos alguns jogos desenvolvidos por membros da comunidade *Scratch* que podem ser encontrados no próprio site da ferramenta, acessando a **aba *Project* (ou 'Explorar')**.

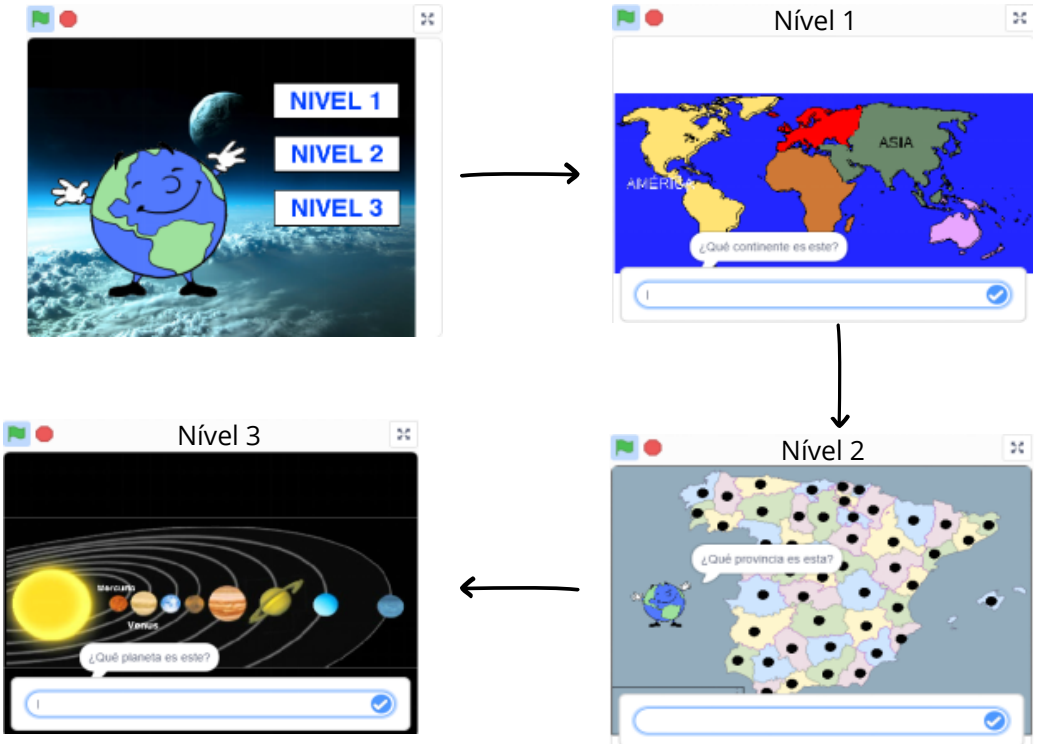
- **Matemática:** com o objetivo de ensinar a operação de adição, o jogo possui uma dinâmica bem simples e intuitiva. O projeto está disponível em: https://scratch.mit.edu/projects/65710744/



- **Ciências:** Este jogo tem o objetivo de ensinar sobre os planetas do Sistema Solar; o jogador digita o nome de um deles, e o jogo se encarrega de explicar seu significado e suas características. O projeto está disponível em: https://scratch.mit.edu/projects/257023022/



- **Geografia:** com três níveis, este jogo busca ensinar os nomes dos continentes (nível 1), as províncias (nível 2) e os planetas (nível 3). O projeto está disponível em: https://scratch.mit.edu/projects/11461682/



8.4. Vamos praticar!

Agora que você conhece a ferramenta e sabe de todas as suas possibilidades, vamos criar seu primeiro jogo *Scratch*. Siga os passos a seguir para criar os dois projetos:

1. Fazendo o gato miar (ou o ator falar)

a. Clique em ***Sprite*** que neste caso é o gato.

b. Clique na aba **Código**.

c. Clique na categoria **Eventos**.

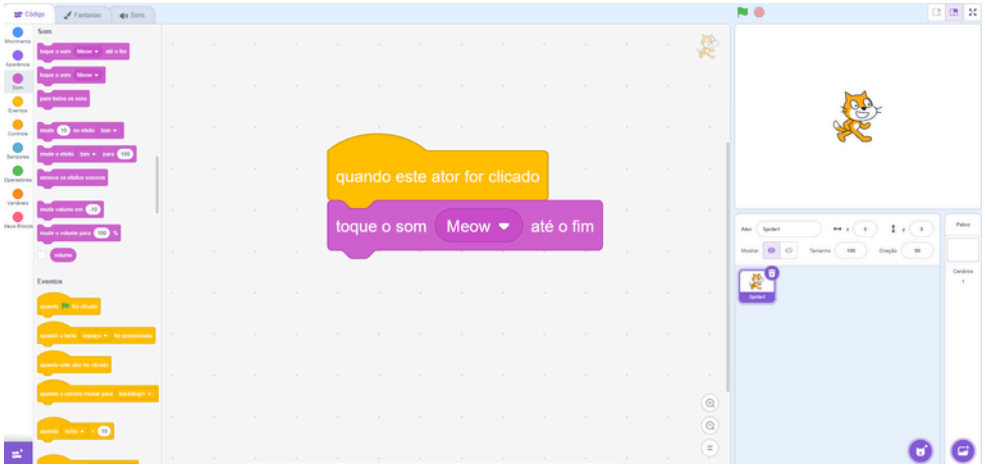
d. Clique e arraste o bloco '**quando este ator for clicado'** até a parte central da tela (edição).

e. Depois clique na categoria **Som**.

f. Selecione o bloco '**toque o som'** *meow* '**até o fim'** e o encaixe exatamente abaixo do bloco '**quando este ator for clicado'**.

g. Depois clique no gato que está no palco para ouvi-lo miar.

Parabéns. Este foi seu primeiro jogo criado no Scratch.



Vamos tentar mais um? Siga os passos novamente, mas antes leia as instruções do quadro ao lado:

Vamos precisar escolher mais um cenário:

- No canto inferior direito, clique no ícone de uma figura com o sinal de **'+'**;
- Escolha entre as opções o cenário que quiser;

O cenário ***backdrop1*** é o que representa o fundo branco, por isso, é com ele que vamos começar!

Agora, vamos escolher uma nova fantasia:

- Clique sobre o ator;
- Depois em **Fantasias**, mais acima da tela;
- No canto inferior esquerdo, clique no ícone do gato com o sinal de **'+'**
- Você pode escolher outro personagem ou apenas modificar a posição do gato.

Por padrão, o Scratch sempre inicia com o personagem **Cat-a**.

2. Mudando o cenário

a. Clique na aba **Código**

b. Clique na categoria **Eventos**

c. Clique e arraste o bloco '**quando bandeira verde for clicado'** até a parte central da tela (edição)

d. Depois clique em **Aparência**

e. Clique e arraste o bloco '**mude cenário para backdrop1'** até a parte central da tela (edição) e o encaixe exatamente abaixo do bloco '**quando bandeira verde for clicado'**.

f. Clique novamente em **Eventos**

g. Clique e arraste o bloco '**quando a tecla espaço for pressionada'** até a parte central da tela (edição). Note que seu formato não permite encaixar na parte de cima, somente embaixo.



A categoria **EVENTOS** é a única que será utilizada sempre no início dos blocos.

h. Clique em Aparência

i. Clique e arraste os seguintes blocos:

- **Diga Olá por 2 segundos**

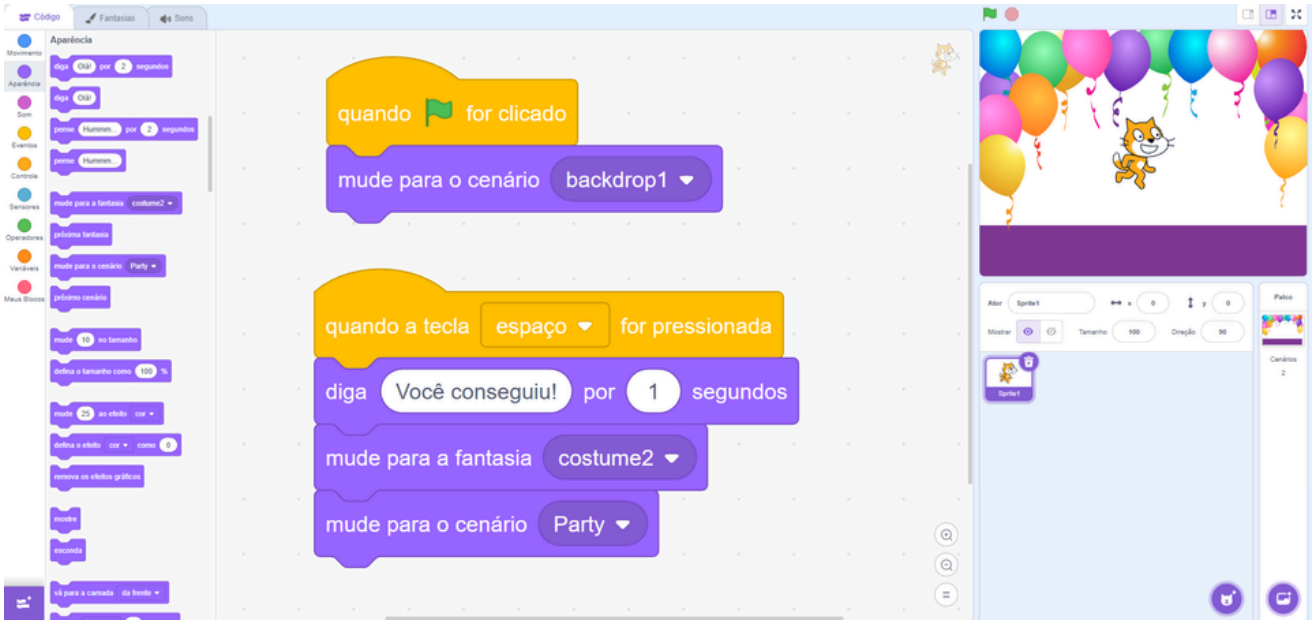
Altere a frase e o tempo; você pode fazer isso, clicando sobre eles, apagando e digitando a nova frase; o bloco deverá ficar da seguinte maneira: '**Diga Você Conseguiu por 1 segundo'**

- **Mude para a fantasia costume1**

Note que ao lado do nome **Costume1** aparece uma setinha apontando para baixo, clique nela e você verá o nome da nova fantasia que escolheu para seu personagem.

- **Mude para o cenário *Party***

Da mesma forma que a fantasia, neste bloco irá aparecer a setinha para baixo; ao clicar nela você verá os cenários que você já havia escolhido/selecionado anteriormente.



Não esqueça de encaixar um bloco no outro e que todos devem ser encaixados no bloco **'QUANDO TECLA ESPAÇO FOR PRESSIONADA'**

Com o Scratch podemos aplicar os conceitos introdutórios de algoritmos, como: sequência (já representado nos exercícios anteriores), condição e repetição. Vamos agora praticar um exemplo para estes dois últimos:

3. O desenho do quadrado: algoritmo de repetição

Momento da dica...



Primeiro, adicione a extensão '**Caneta**' clicando no botão azul no canto inferior esquerdo da tela do Scratch

a. Clique na aba **Código**

b. Clique na categoria **Eventos**

c. Clique e arraste o bloco **‘quando bandeira verde for clicado’** até a parte central da tela (edição)

d. Depois clique em **Caneta**

e. Clique e arraste o bloco ‘**apague tudo**' (para limpar desenhos anteriores) e depois clique e arraste o bloco **‘use a caneta’**

f. Clique novamente em **Controle**

g. Clique e arraste o bloco ‘**repita 10 vezes’** até a parte central da tela (edição). Altere o valor para 4

h. Clique em **Movimento**

i. Clique e arraste o bloco **‘mova 10 passos’** até a parte central da tela (edição) e encaixe dentro do bloco **‘repita’.** Altera o valor para 100

j. Clique e arraste o bloco **‘gire para a direita (90) graus’** até a parte central da tela (edição) e encaixe na sequência dentro do bloco **‘repita’**

k. Clique em **Controle**

l. Clique e arraste o bloco **‘espere 1 segundo’** até a parte central da tela (edição) e encaixe na sequência dentro do bloco **‘repita’**



O código sem a repetição teria 8 blocos longos de **"mova"** e **"gire"**.

O bloco **repita** reduz isso para apenas 3 blocos

4. O porteiro da festa: algoritmo de condição

a. Clique em **Eventos**

b. Clique e arraste o bloco **‘quando bandeira verde for clicado’** até a parte central da tela (edição)

c. Clique em **Sensores**

d. Clique e arraste o bloco **‘pergunte (Qual seu nome?) e espere’** até a parte central da tela (edição) e encaixe no bloco anterior. Altere a pergunta para **‘Quanto é 5 + 5?’**

e. Clique em **Controle**

f. Clique e arraste o bloco **‘se... então... senão...’** até a parte central da tela (edição) e encaixe no bloco anterior

g. Clique em **Operadores**

h. Clique e arraste o bloco de ‘**igualdade < [ ] = [50] >’** até a parte central da tela (edição) e encaixe dentro da forma hexagonal do bloco **‘se’**

i. Clique em **Sensores**

j. Clique e arraste o bloco **‘resposta’** até o primeiro espaço da igualdade

k. No segundo espaço da igualdade, digite **‘10’**

l. Clique em **Aparência**

m. Clique e arraste o bloco **‘Diga Olá! por 2 segundos’** até a parte central da tela (edição) e encaixe no primeiro espaço do bloco de condição, logo abaixo do **‘se’**

n. Altere o texto para **‘Acesso liberado! Você conseguiu!’**



o. Clique e arraste o bloco **‘Diga Olá! por 2 segundos’** até a parte central da tela (edição) e encaixe no segundo espaço do bloco de condição, logo abaixo do **‘senão’**

p. Altere o texto para **‘Acesso negado! Tente novamente!’**

8.5. Guia para o professor

Por fim, chegamos ao capítulo da prática no ambiente de programação Scratch. O foco aqui não é apenas aprender um nova ferramenta, mas exercitar todos os conceitos aprendido no capítulos anteriores.

Objetivos de aprendizagem

Ao final da aula, os estudantes deverão ser capazes de:

- Compreender a estrutura básica do Scratch.
- Identificar as categorias de blocos e suas funções.
- Aplicar estruturas sequenciais básicas.
- Compreender a relação entre lógica estudada anteriormente e a ferramenta Scratch.

Roteiro sugerido de aula (aproximadamente 50 min)

- **Abertura:** apresente o Scratch como uma ferramenta que organiza o pensamento computacional em blocos encaixáveis.

- **Explique:** explique que cada bloco no Scratch representa uma instrução lógica. Destaque as categorias principais apresentadas no capítulo:

    - Eventos → determinam quando algo começa.
    - Movimento → controla deslocamento.
    - Aparência → altera visual e mensagens.
    - Som → executa áudio.
    - Controle → contém repetição e decisão.
    - Operadores → cálculos e comparações.
    - Variáveis → armazenam informações.

- **Desenvolvimento:** oriente os alunos a:

    - Criar uma conta (opcional).
    - Configurar idioma.
    - Criar o primeiro projeto:
        - Evento: “quando este ator for clicado”
        - Ação: “toque o som até o fim”.

- Depois, proponha a atividade de mudança de cenário e fala utilizando:
  - Evento “Quando bandeira verde for clicada”.
  - Evento “Quando tecla espaço for pressionada”.
  - Blocos de aparência e fantasia.
- Proponha que modifiquem:
  - A frase exibida
  - O cenário
  - O personagem
  - O som
  - O tempo de execução
- Pergunte:

“O que acontece se alterar a ordem?”
“E se remover o evento?”
“O que acontece se não encaixar corretamente?”

- **Socialização:** cada grupo apresenta o projeto criado, a estrutura que utilizaram, apontando as dificuldades e os ajustes que foram necessários.

- **Reflexão final:** conduza uma conversa sobre o Scratch, perguntando:

O Scratch é diferente do que imaginaram?”
“A lógica ficou mais clara quando visualizaram os blocos?”
“Onde houve decisão ou repetição?”

Avaliação

- Observe se os estudantes compreenderam as funções dos blocos.
- Se desenvolvera a capacidade de montar sequências coerentes.
- Se apresentaram autonomia na manipulação da ferramenta.
- Se conseguiram encontrar e corrigir erros.
- Promova uma autoavaliação reflexiva, perguntando:

“Meu projeto inicia corretamente?”
“Consegui aplicar os conceitos aprendidos nos capítulos anteriores?

8.6. Sugestão de Atividades



Atividade 1 - Explorando os Blocos do Scratch (20 a 25 minutos)

- **Objetivo:**
  - Compreender que cada bloco do Scratch representa uma instrução lógica para produzir uma ação no programa.
- **Recursos necessários**
  - Computadores com acesso ao Scratch.
  - Internet (opcional).
  - Projetor multimídia (opcional).
- **Passo a passo**
  - Organize os alunos em grupos.
  - Peça que cada grupo abra o ambiente Scratch e explore as categorias de blocos apresentadas no capítulo:
    - Movimento
    - Aparência
    - Som
    - Eventos
    - Controle
    - Sensores
    - Operadores
    - Variáveis
  - Solicite que cada grupo escolha uma categoria de blocos.
  - Os alunos deverão montar um pequeno exemplo utilizando pelo menos três blocos diferentes da categoria escolhida.
  - Ao final da atividade, cada grupo deverá demonstrar o que seu programa faz.
- **Debate em sala**
  - Qual bloco inicia a execução do programa?
  - O que acontece se o bloco de evento for removido?
  - Todos os blocos podem ser usados em qualquer ordem?



Atividade 2 - Interação com Personagem (20 a 25 minutos)

- **Objetivo:**
  - Compreender como eventos e ações podem ser combinados para criar interações no Scratch.

- **Recursos necessários**
  - Computadores com acesso ao Scratch.
  - Internet (opcional).
  - Projetor multimídia (opcional).

- **Passo a passo**
  - Os alunos devem escolher um personagem (ator) do Scratch.
  - Peça que criem um projeto com as seguintes características:
    - Um evento.
    - Uma ação visual.
    - Um som.
  - Exemplo de estrutura:
    - Evento: Quando o ator for clicado.
    - Ação: O personagem fala uma frase.
    - Complemento: Um som é executado.
  - Os alunos devem modificar:
    - o texto da fala
    - o tempo da mensagem
    - o som utilizado
  - Ao final, cada aluno (ou grupo) deverá apresentar sua criação para a turma.

- **Debate em sala**
  - O que acontece quando clicamos várias vezes no personagem?
  - O que muda quando alteramos a ordem dos blocos?
  - Como o Scratch sabe quando executar o código?

Atividade 3 - Movimento Repetitivo (20 a 25 minutos)

- **Objetivo:**
  - Compreender como as estruturas de repetição funcionam na programação visual.

- **Recursos necessários**
  - Computadores com acesso ao Scratch.
  - Internet (opcional).
  - Projetor multimídia (opcional).

- **Passo a passo**
  - Os alunos devem escolher um personagem (ator) do Scratch.
  - Peça que criem um movimento simples utilizando blocos de Movimento.
  - Em seguida, peça que utilizem um bloco de Controle para repetir o movimento.
  - Em seguida, peça que utilizem um bloco de Controle para repetir o movimento.
    - Exemplo:
      - Repita 4 vezes
      - Mova 50 passos
      - Gire 90 graus
  - Peça que observem o resultado no Scratch.
  - Em seguida, peça que alterem o número de repetições e analisem o que acontece.

- **Debate em sala**
  - O que muda quando alteramos o número de repetições?
  - Seria possível fazer o mesmo movimento sem usar o bloco “repita”?
  - Qual forma deixaria o programa mais organizado?



Atividade 4 - Meu Primeiro Projeto (20 a 25 minutos)

- **Objetivo:**
  - Integrar os conceitos estudados no capítulo para desenvolver um pequeno projeto no Scratch.

- **Recursos necessários**
  - Computadores com acesso ao Scratch.
  - Internet (opcional).
  - Projetor multimídia (opcional).

- **Passo a passo**
  - Organize os alunos em grupos.
  - Cada grupo deverá criar um projeto que contenha:
    - um evento.
    - uma sequência de ações.
    - pelo menos uma decisão ou repetição.
  - Sugestões de projetos:
    - Um personagem que responde a uma pergunta.
    - Um jogo simples de perguntas e respostas.
    - Um personagem que se movimenta ao pressionar uma tecla.
    - Uma animação curta.
  - Após finalizar o projeto, os alunos deverão explicar:
    - qual evento inicia o programa.
    - quais blocos foram utilizados.
    - qual era o objetivo do projeto.

- **Debate em sala**
  - O que funcionou bem no projeto?
  - O que foi mais difícil de montar?
  - Em qual parte vocês usaram lógica de decisão ou repetição?
  - O que vocês melhorariam no projeto se tivessem mais tempo?

- **Reflexão final**



Ao longo deste livro aprendemos a organizar ideias em sequências, tomar decisões e repetir ações. Vamos pensar juntos:

De que forma essas habilidades podem ajudar na resolução de problemas do dia a dia?

Referências

BRACKMANN, Christian P.; ROMÁN-GONZÁLEZ, Marcos; ROBLES, Gregorio; MORENO-LEÓN, Jesús; CASALI, Ana; BARONE, Dante. **Development of Computational Thinking Skills through Unplugged Activities in Primary School**. In: Proceedings of the 12th Workshop in Primary and Secondary Computing Education (WiPSCE '17), Nijmegen, Netherlands, November 8–10, 2017. ACM, 2017. DOI:10.1145/3137065.3137069.

DIJKSTRA, Edsger W. **Go To Statement Considered Harmful.** Communications of the ACM, New York, v. 11, n. 3, p. 147–148, mar. 1968. Disponível em: https://homepages.cwi.nl/~storm/teaching/reader/Dijkstra68.pdf. Acesso em: 06 fevereiro. 2026.

MALONEY, J.; RESNICK, M.; RUSK, N.; SILVERMAN, B. & EASTMOND, E. The Scratch Programming Language and Environment. **ACM Transactions on Computing Education**, v 10, n. 4, 2010.

PAPERT, Seymour. **Mindstorms**: Children, Computers, and Powerful Ideas. New York: Basic Books, 1980. Disponível em: https://worrydream.com/refs/Papert_1980_-_Mindstorms,_1st_ed.pdf. Acesso em: 06 fevereiro. 2026.

SOMMERVILLE, I. **Software Engineering**, 10th Ed. Pearson, 2015.

WING, Jeannette M. **Computational thinking**. Communications of the ACM, New York, v. 49, n. 3, p. 33–35, mar. 2006. Disponível em: https://www.cs.cmu.edu/~15110-s13/Wing06-ct.pdf. Acesso em: 06 fevereiro. 2026.

Links úteis

Scratch - Starter Projects.
Disponível em: <http://www.scratchbrasil.net.br/index.php/materiais/tutoriais.html>.

Exact Instructions Challenge - THIS is why my kids hate me. | Josh Darnit.
Disponível em: <https://www.youtube.com/watch?v=cDA3_5982h8>.

Sobre as autoras:

Gabrieli Jorge Barbosa



Graduanda em Engenharia de Computação na Universidade Federal de Santa Catarina (UFSC). Bolsista de Iniciação Cientifica do CNPq no projeto Meninas Digitais: Integrando Universidades, Institutos e Escolas. É membro do Laboratório de Tecnologias Computacionais (LabTeC-UFSC). Atualmente, se dedica com foco especial na implementação do Pensamento Computacional nas escolas, seguindo as diretrizes da BNCC.

Marta Adriana Machado da Silva



Doutora em Engenharia e Gestão do Conhecimento (UFSC) e Mestre e Bacharel em Ciências da Computação (UFSC). Com mais de 20 anos de experiência no Ensino Superior, atua como Docente, sendo autora de 3 livros didáticos (publicados pela UNESC em 2022): “Raciocínio Lógico e Computacional”, "Programação de Computadores" e "Estrutura de Dados". Possui vasta experiência em produção científica, com 7 capítulos de livros e 13 artigos completos publicados em periódicos especializados (como ETD e RENTE). Sua pesquisa inclui temas de Inteligência Artificial, inovação educacional, Realidade Aumentada (RA) e Laboratórios Remotos.

Sobre as autoras:

Tatiana Nilson dos Santos



Bolsista de pós-doutorado Júnior do CNPq no Programa de Pós-Graduação em Tecnologias da Informação e Comunicação (PPGTIC) na Universidade Federal de Santa Catarina (UFSC). Membro do Laboratório de Tecnologias Computacionais (LabTeC-UFSC) e auxilia na coordenação do projeto Meninas Digitais: Integrando Universidades, Institutos e Escolas. Possui doutorado em Engenharia e Gestão do Conhecimento com ênfase em mídias digitais para surdos, mestrado em Tecnologias da Informação e Comunicação, com ênfase em Tecnologias Educacionais, ambos pela UFSC, especialização emTecnologias para Educação Profissional e Tecnológica, pelo Instituto Federal de Santa Catarina (IFSC). Suas pesquisas envolvem games na educação, ludicidade e tecnologias aplicadas ao processo de ensino e aprendizagem, gamificação, e-learning, inteligência artificial.

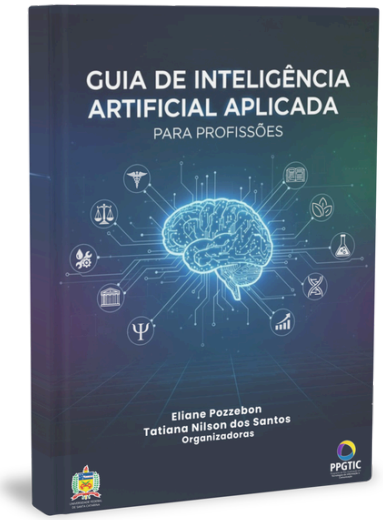
Eliane Pozzebon

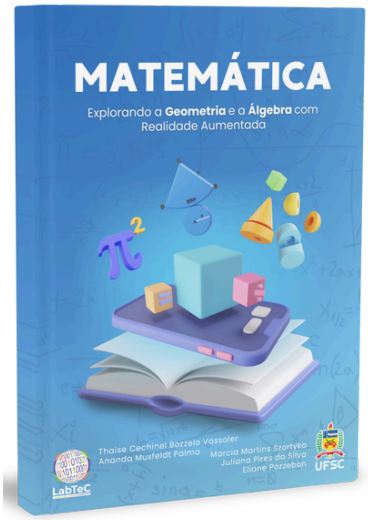


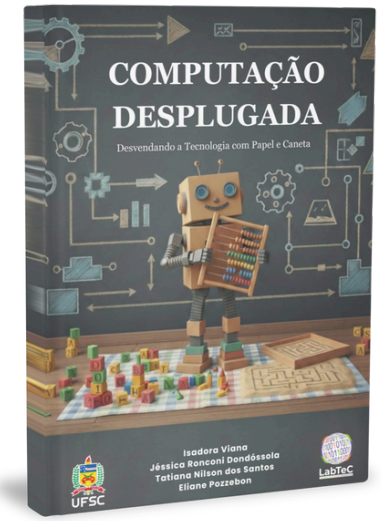
Professora Titular do Departamento de Computação da Universidade Federal de Santa Catarina (UFSC) e atua na área de Inteligência Artificial. Coordena o Laboratório de Tecnologias Computacionais (LabTeC-UFSC) e lidera o Grupo de Pesquisa em Tecnologias Computacionais certificado pelo CNPq. Possui pós-doutorado em Inteligência Artificial pela Universidade Federal do Rio Grande do Sul (UFRGS), doutorado em Engenharia Elétrica e mestrado em Ciência da Computação pela UFSC. Suas pesquisas envolvem inteligência artificial aplicada à educação e à saúde, tecnologias imersivas, reconhecimento facial e iniciativas voltadas à inclusão de mulheres na computação.

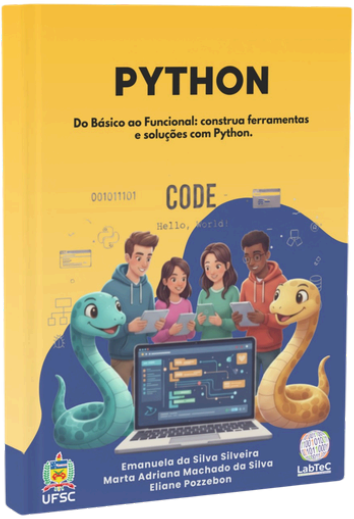
Gostou desta leitura? Confira também:

Visite https://labtec.ufsc.br/ebooks para conhecer a coleção completa e conferir outros lançamentos.









Gostou desta leitura? Confira também:

Visite https://labtec.ufsc.br/ebooks para conhecer a coleção completa e conferir outros lançamentos.

