Série Aprendizagem de Programação

1ª edição

Aprender a Aprender Programação:

Estratégias Metacognitivas para Melhorar o Aprendizado e o Ensino de Programação

Tiago Roberto Kautzmann

Este livro também está disponível para leitura no Kindle.

Encontre-o na Amazon (*amazon.com.br*) pesquisando pelo título.

Dados Internacionais de Catalogação na Publicação (CIP)
(Câmara Brasileira do Livro, SP, Brasil)

Kautzmann, Tiago Roberto
Aprender a aprender programação [livro eletrônico] : estratégias metacognitivas para melhorar o aprendizado e o ensino de programação / Tiago Roberto Kautzmann. -- 1. ed. -- União da Vitória, PR : Ed. do Autor, 2025. -- (Aprendizagem de programação)
PDF

Bibliografia.
ISBN 978-65-01-37735-3

1. Aprendizagem 2. Metacognição 3. Programação (Computadores) - Estudo e ensino I. Título. II. Série.

25-258752 CDD-005.107

Índices para catálogo sistemático:

1. Programação : Informática : Estudo e ensino 005.107

Eliete Marques da Silva - Bibliotecária - CRB-8/9380

Com o objetivo de aprimorar a clareza e a qualidade dos textos apresentados neste livro, utilizamos o ChatGPT como ferramenta auxiliar na revisão e no refinamento dos textos.

Sobre o autor

Tiago Roberto Kautzmann

Atualmente atua como Professor do Instituto Federal do Paraná (IFPR). Doutor em Computação Aplicada pela Unisinos - Universidade do Vale do Rio dos Sinos (2018 - 2022), com período sanduíche realizado em Marselha (França), junto à Aix-Marseille Université (2019). Mestre em Computação Aplicada pela Unisinos (2013 - 2015). Tecnólogo em Sistemas para Internet pela Universidade Feevale (2009 - 2011). Formação pedagógica pelo Programa Especial de Formação Pedagógica da Universidade Feevale (2012 - 2013). Realiza pesquisa científica em Inteligência Artificial aplicada à Educação. Professor há mais de 20 anos, ministra disciplinas sobre algoritmos, ciência de dados, inteligência artificial, estruturas de dados, programação orientada a objetos, desenvolvimento de software, banco de dados e lógica.

Contato: tkautzmann@gmail.com

Prefácio

Aprender a programar é um desafio que vai além da simples memorização de comandos e estruturas sintáticas. A programação exige raciocínio lógico, pensamento estruturado e, acima de tudo, autonomia no aprendizado. Neste contexto, a metacognição desempenha um papel fundamental. **Saber aprender, refletir sobre os próprios processos cognitivos e regular estratégias de estudo são habilidades que diferenciam um estudante que apenas codifica de um programador que compreende e domina seu próprio processo de aprendizado**.

Este livro tem como propósito apresentar estratégias metacognitivas aplicadas ao ensino e à aprendizagem de programação. A partir de conceitos fundamentados na literatura científica e da experiência do autor com turmas introdutórias de programação, o livro explora como a metacognição pode ser utilizada tanto por alunos quanto por professores para otimizar o ensino e o aprendizado.

Nos capítulos iniciais, são abordados os fundamentos da metacognição e seus principais componentes, como o conhecimento metacognitivo, o monitoramento dos processos de aprendizagem e a autorregulação da cognição. Em seguida, o livro apresenta a importância das habilidades metacognitivas no aprendizado da programação, destacando como essas competências podem influenciar na resolução de

problemas, na motivação dos alunos e na capacidade de aprender de forma independente.

Para os alunos, são apresentadas estratégias práticas, como o planejamento antes de codificar, a reflexão sobre erros, a comparação entre diferentes abordagens e o uso da verbalização do pensamento para consolidar o aprendizado. **Para os professore**s, são apresentadas estratégias que incentivam a autorregulação dos alunos, promovendo um ambiente de ensino que estimula a reflexão e o desenvolvimento de estratégias eficazes para aprender programação.

Mais do que um guia técnico, este livro busca promover uma mudança na forma como a programação é aprendida e ensinada. Ao incorporar princípios metacognitivos ao ensino de programação, espera-se que os alunos se tornem aprendizes mais conscientes, críticos e eficientes, e que os professores consigam potencializar a aprendizagem dos estudantes, ajudando-os a superar desafios com autonomia e confiança.

Que este livro sirva como um recurso interessante para todos aqueles que desejam aprimorar seu aprendizado ou sua prática docente na programação. Que cada leitor encontre aqui não apenas técnicas, mas um novo olhar sobre a aprendizagem, tornando o ato de programar um processo mais reflexivo, estratégico e significativo. Boa leitura!

O Autor.

Conteúdo

1. Introdução

Saber identificar o que se sabe e o que não se sabe sobre determinado domínio de conhecimento é ingrediente fundamental para o sucesso escolar. Alunos que possuem esta habilidade metacognitiva são mais propensos a remediar suas fraquezas de aprendizagem, mobilizando esforços para a aquisição dos conhecimentos deficitários (FOGARTY, 1994). Também são mais predispostos a buscarem ajuda, quando necessário (STAVRIANOPOULOS, 2007), e a estudarem estrategicamente, definindo objetivos, planejando, selecionando estratégias, monitorando e avaliando o progresso dos estudos (TOBIAS; EVERSON, 2002).

A habilidade do aluno de identificar o que sabe e o que não sabe é chamada por pesquisadores da área de Psicologia Educacional como habilidade de monitoramento do conhecimento. É considerada uma habilidade metacognitiva (conhecimento de uma pessoa sobre os próprios processos cognitivos), pois exige do aluno o monitoramento de um componente da sua cognição, a memória. Estudos indicam que o monitoramento preciso do conhecimento tem relação com o sucesso na aprendizagem em todos os cenários acadêmicos (TOBIAS; EVERSON, 2002).

No entanto, é conhecido que nem todos os alunos se engajam espontaneamente em um pensamento metacognitivo, ao menos que sejam explicitamente encorajados, através de atividades propostas por professores (SCHOENFELD, 1987; BERARDI-COLETTA; BUYER; DOMINOWSKI; RELLINGER, 1995; BRANSFORD et al., 1999; CHI; BASSOK; LEWIS; REIMANN; GLASER, 1989). Além disso, já foi demonstrado que processos metacognitivos podem ser melhorados através de treinamento (DESOETE; ROEYERS; DE CLERCQ, 2003; KRAMARSKI; MEVARECH; ARAMI, 2002). Dessa forma, pesquisadores recomendam o foco do ensino visando à melhora das habilidades metacognitivas dos alunos (BRANSFORD; BROWN; COCKING, 1999).

No contexto de aprendizagem de programação, as habilidades metacognitivas são essenciais para que os alunos possam lidar com a complexidade inerente ao desenvolvimento de software. **Programação não é apenas sobre escrever código; é um processo que exige análise de problemas, tomada de decisões e autorregulação do aprendizado**. Estudos indicam que alunos que utilizam estratégias metacognitivas conseguem monitorar melhor seus erros, ajustar suas abordagens e persistir diante de desafios complexos. Além disso, a metacognição está diretamente relacionada ao desenvolvimento de estratégias eficazes de depuração de código (PRATHER et al, 2020).

Aprender a programar também exige que o aluno saiba quando buscar ajuda e quando insistir na tentativa de resolver um problema sozinho. Segundo Tobias e Everson (2002), alunos que desenvolvem um monitoramento eficiente do conhecimento são mais propensos a empregar estratégias adequadas de aprendizado, evitando a dependência excessiva de professores ou tutores. Isso implica que a aprendizagem de programação não deve ser vista apenas como um exercício técnico, mas também como uma habilidade metacognitiva que pode ser aprimorada com a prática e o ensino adequado.

Muitos alunos iniciantes em programação enfrentam dificuldades porque não sabem como aprender programação de forma eficiente. Isso pode levar a frustrações e abandono prematuro das disciplinas introdutórias de programação nas faculdades e cursos técnicos. A pesquisa de Azevedo et al. (2013) destaca que estudantes que desenvolvem consciência sobre seus próprios processos de aprendizagem conseguem estabelecer metas claras, monitorar seu progresso e adaptar suas estratégias conforme necessário. Dessa forma, ensinar os alunos a aprimorar suas habilidades metacognitivas é tão importante quanto ensinar sintaxe e lógica de programação.

A aprendizagem autorregulada também desempenha um papel essencial. Zimmerman (2008) argumenta que aprendizes autorregulados são mais proativos no controle de seu aprendizado, aplicando estratégias metacognitivas como planejamento, monitoramento e avaliação para

melhorar seu desempenho. No ensino de programação, isso significa que um aluno que adota práticas metacognitivas tende a lidar melhor com desafios, compreender códigos mais complexos e ser mais eficiente na depuração de erros.

Se por um lado os alunos precisam desenvolver a capacidade de monitorar e regular seu próprio aprendizado, por outro, os professores desempenham um papel fundamental no incentivo e na modelagem dessas habilidades. Segundo Bransford, Brown e Cocking (1999), alunos não adotam naturalmente estratégias metacognitivas, a menos que sejam orientados e incentivados a fazê-lo. No ensino de programação, isso significa que as práticas pedagógicas devem incluir atividades que promovam a reflexão, a autorregulação e a experimentação de diferentes abordagens para resolver problemas.

Uma abordagem interessante para estimular a metacognição na programação é a Aprendizagem Baseada em Problemas (PBL, do inglês *Problem-Based Learning*), onde os alunos precisam analisar um problema específico, propor soluções, testar hipóteses e refletir sobre seus erros e acertos. Outra abordagem interessante, similar ao PBL, é a Aprendizagem Baseada em Projetos (PjBL, do inglês *Project-Based Learning*), onde os alunos aprendem a partir do desenvolvimento de projetos práticos, enfrentando desafios reais de programação, que exigem planejamento, tomada de decisões e autoavaliação constante. No contexto da programação, isso pode envolver a criação de softwares

funcionais, nos quais os estudantes precisam pesquisar tecnologias, dividir tarefas, testar implementações e ajustar estratégias conforme encontram dificuldades. Além de reforçar os conceitos técnicos, as duas abordagens (PBL e PjBL) estimulam a reflexão sobre o próprio processo de aprendizagem, ajudando os alunos a identificar lacunas no conhecimento, ajustar suas estratégias e desenvolver autonomia no aprendizado.

Além disso, estratégias como a verbalização do pensamento (FLAVELL, MILLER & MILLER, 1999) e a comparação de diferentes abordagens podem ajudar os alunos a perceberem como suas escolhas afetam a eficiência e a clareza do código.

Outro ponto essencial é o *feedback* metacognitivo. Em vez de apenas corrigir erros no código dos alunos, os professores podem incentivá-los a refletir sobre o porquê de um erro ter ocorrido e quais estratégias podem ser usadas para evitá-lo no futuro. Essa prática ajuda os alunos a desenvolverem um pensamento crítico sobre seu próprio aprendizado, tornando-se programadores mais independentes e eficientes.

O que vou aprender neste livro?

Este livro tem como objetivo ajudar alunos e professores a entenderem e aplicarem estratégias metacognitivas na aprendizagem de

programação. Ao longo dos capítulos, serão explorados os seguintes assuntos:

- Os fundamentos da metacognição e da autorregulação da aprendizagem;
- A importância da habilidade de monitoramento do conhecimento na programação;
- Estratégias metacognitivas que os alunos poderiam adotar para melhorar seu aprendizado, incluindo planejamento, monitoramento e ajuste de estratégias;
- Práticas pedagógicas que os professores podem utilizar para incentivar a metacognição dos alunos, como a Aprendizagem Baseada em Problemas, o *feedback* metacognitivo e ensino explícito de estratégias de autorregulação;
- Reflexões sobre como a metacognição pode tornar o aprendizado de programação mais efetivo, engajador e autônomo.

Ao final da leitura, espera-se que o leitor esteja mais preparado para aprender e ensinar programação de maneira estratégica. Este livro não apenas apresenta conceitos teóricos, mas também oferece aplicações práticas e recomendações concretas, tornando-se um guia tanto para alunos iniciantes em disciplinas introdutórias de programação quanto para professores que desejam aprimorar suas práticas pedagógicas.

A quem se destina este livro?

Este livro é destinado a estudantes, professores e pesquisadores da área de programação e educação, especialmente aqueles interessados em compreender e aplicar estratégias metacognitivas para aprimorar o aprendizado e o ensino de programação. **Para os alunos** iniciantes, a obra oferece orientações práticas sobre como desenvolver habilidades metacognitivas, tornando o aprendizado mais eficiente e autônomo. **Para os professores**, apresenta estratégias pedagógicas que podem ser utilizadas para estimular a reflexão, a autorregulação da aprendizagem e a resolução estratégica de problemas no ensino de programação. Além disso, pesquisadores da área de informática na educação e do ensino de computação também poderão encontrar aqui subsídios teóricos e metodológicos sobre a importância da metacognição no desenvolvimento de habilidades computacionais.

Como estudar com este livro?

Para obter o máximo proveito deste livro, recomenda-se uma leitura ativa e reflexiva, associando os conceitos apresentados à sua própria experiência de aprendizagem ou ensino de programação. Os leitores podem anotar suas reflexões, destacar estratégias que

considerem mais úteis e testar sua aplicação em situações reais, seja ao resolver problemas de programação ou ao planejar atividades de ensino.

Cada capítulo apresenta princípios teóricos e estratégias práticas, permitindo que o leitor escolha o ritmo de estudo mais adequado ao seu contexto. Para estudantes, a sugestão é experimentar e adaptar as estratégias metacognitivas apresentadas, avaliando quais funcionam melhor para seu estilo de aprendizado. Para professores, o ideal é testar os métodos propostos em sala de aula, observando como os alunos respondem e ajustando as abordagens conforme necessário. O mais importante é não apenas ler, mas colocar em prática os conceitos.

2. Metacognição

A capacidade de refletir sobre os próprios processos de pensamento é um dos aspectos fundamentais da aprendizagem eficaz. Essa habilidade, conhecida como **metacognição**, permite aos indivíduos compreender e regular sua cognição — o conjunto de processos mentais responsáveis pela aquisição, processamento, armazenamento e uso do conhecimento —, otimizando o aprendizado e a resolução de problemas.

John Flavell foi o primeiro pesquisador a utilizar o termo metacognição (HARTMANN, 1998; TARRICONE, 2011), em um artigo de 1976, no qual descreve o construto: “Metacognição se refere ao próprio conhecimento sobre os próprios processos e produtos cognitivos ou qualquer coisa relacionada a eles” (FLAVELL, 1976 apud BROWN, 1977, p. 8, tradução nossa). Uma versão resumida define a metacognição como os pensamentos sobre os próprios pensamentos (FLAVELL, 1979).

O termo metacognição é composto pelas palavras “meta” e ”cognição”. O prefixo meta tem origem grega e significa “sobre”. Em relação à cognição, o trabalho de Flavell, Miller, P., Miller, S. (1999) refere-se a este construto como um conceito amplo demais para ser limitado a uma definição precisa e fechada.

Uma forma bastante simples de compreender o conceito de metacognição é responder a seguinte questão: quando foi a última vez que tentou lembrar o nome de uma pessoa, mas não conseguiu lembrar? Nesse caso, poderia não lembrar o nome do indivíduo, mas tinha absoluta certeza de que sabia o nome da pessoa e que em outro momento conseguiria lembrar. Este tipo de evento é metacognitivo porque o indivíduo está tendo um pensamento sobre um aspecto da cognição, no caso, o pensamento de que o nome da pessoa que se quer recordar está registrado na memória. Outro exemplo é quando uma pessoa decide fazer uma lista de compras para ir ao supermercado, porque sabe que poderá ter dificuldade de recordar todos os produtos de que necessita. Entender os limites da memória também é uma forma de metacognição, pois tem relação com crenças e conhecimento sobre a própria memória (DUNLOSKY; METCALFE, 2009).

No artigo seminal de 1976, Flavell ainda descreve alguns exemplos para explicar a metacognição: “[...] eu estou engajado em metacognição (metamemória, meta-aprendizado, meta-atenção, metalinguagem ou o que quer que seja) se eu notar que estou tendo mais problema aprendendo A do que B; se me dou conta que eu deveria checar duas vezes C antes de aceitá-lo como fato. [...] Se eu me tornar consciente de que é melhor eu tomar nota de D porque eu posso esquecê-lo. [...] (FLAVELL, 1976 apud TARRICONE, 2011, p. 2, tradução nossa).

Metacognição não deve ser confundida com pensamento crítico, porém, pensadores críticos provavelmente utilizarão de estratégias metacognitivas, mesmo que eles não saibam que estão utilizando (KOLENCIK; HILLWIG, 2011). As habilidades metacognitivas incluem o controle consciente da aprendizagem, o planejamento e a seleção de estratégias, o monitoramento dos progressos de aprendizagem, a correção de erros, análise da efetividade das estratégias de aprendizagem, e a modificação dos comportamentos e estratégias de aprendizagem quando necessário (RIDLEY et al., 1992). O próximo capítulo descreve os componentes da metacognição.

3. Componentes da metacognição

Segundo Pintrich, Wolters e Baxter (2000) e Dunlosky e Metcalfe (2009), os processos metacognitivos podem ser divididos em três componentes principais: a) **conhecimento metacognitivo**; b) **julgamento e monitoramento dos próprios processos de aprendizagem**; c) **autorregulação e controle da cognição**. A visão consensual é que todos estes três componentes da metacognição são necessários para regular a aprendizagem eficientemente e efetivamente (TOBIAS; EVERSON, 2009).

Conhecimento metacognitivo

Flavell (1979) descreve que o conhecimento metacognitivo consiste principalmente de conhecimentos ou crenças sobre os fatores ou variáveis que agem e interagem de forma que afetem o curso e os resultados dos empreendimentos cognitivos. O autor classifica os fatores e variáveis em três categorias: **pessoa**, **tarefa** e **estratégia**.

A **categoria pessoa** engloba tudo o que uma pessoa poderia vir a acreditar da natureza de si mesma e de outras pessoas como processadores metacognitivos. A crença de uma pessoa que ela aprende melhor escutando do que lendo e a crença de uma pessoa de que um dos

seus amigos tem melhor desempenho social do que outro amigo são exemplos da categoria pessoa.

A **categoria tarefa** inclui conhecimentos sobre como as variações das tarefas podem influenciar a cognição. Por exemplo, um enunciado de problema que fornece várias informações sobre o problema será geralmente mais fácil de ser resolvido do que um enunciado de problema que fornece poucas informações sobre o problema. Quando um aluno entende esta ideia, ela compõe o seu conhecimento metacognitivo sobre tarefas. Outro exemplo citado por Flavell é que os alunos podem saber que algumas tarefas envolvem empreendimentos cognitivos mais exigentes e difíceis do que outras tarefas, como lembrar uma história na íntegra, ao invés de apenas lembrar a essência da história (FLAVELL, 1979).

A **categoria estratégia** diz respeito ao conhecimento de estratégias para alcançar determinados objetivos. Um aluno poderia acreditar que uma boa forma de aprender e reter uma quantidade grande de informações é prestar atenção nos pontos principais e tentar repeti-los para si mesmo com suas próprias palavras (FLAVELL, 1979). Uma pessoa pode ter conhecimento de estratégias cognitivas mais sofisticadas, como o de passar mais tempo estudando materiais didáticos mais importantes ou menos familiares do que materiais menos importantes e já aprendidos (FLAVELL; MILLER, P.; MILLER, S., 1999). Outro exemplo de conhecimento metacognitivo da categoria

estratégia foi retirado do artigo de Barry Zimmerman (1998), que apresenta comportamentos autorregulados de escritores, músicos e esportistas. Consta uma estratégia utilizada pelo escritor norte-americano Ernest Miller Hemingway, relacionada à escrita. Segundo o artigo, Hemingway, todos os dias, parava de escrever no meio de uma sentença, propositalmente, porque ele descobriu que esta estratégia o fazia começar a escrever sem atrasos no dia seguinte.

Julgamento e monitoramento dos próprios processos de aprendizagem

O componente metacognitivo de julgamento e monitoramento dos próprios processos de aprendizagem se refere à avaliação do andamento ou do atual estado de uma determinada atividade cognitiva (DUNLOSKY; METCALFE, 2009). Podem estar incluídos quatros processos metacognitivos: a) julgamentos de *ease of learning* (EOL); b) monitoramento da compreensão e da aprendizagem ou *judgements of learning* (JOL); c) *feeling of knowing* (FOK); e d) julgamentos de confiança. O processo EOL diz respeito a fazer uma avaliação sobre quão fácil ou difícil será realizar uma determinada tarefa de aprendizagem. O processo JOL trata de monitorar a compreensão da aprendizagem. O processo FOK é sobre ter a consciência de saber algo, mas ser incapaz de lembrá-lo. Os julgamentos de confiança são os

julgamentos sobre a correção ou adequabilidade de uma resposta (PINTRICH; WOLTERS; BAXTER, 2000).

Segundo Flavell, Miller, P., Miller, S. (1999), o monitoramento metacognitivo envolve, por vezes, as chamadas experiências metacognitivas, que são experiências afetivas ou cognitivas pertinentes a uma iniciativa cognitiva. O trabalho dos autores cita alguns exemplos, como a súbita percepção de uma pessoa de que não está entendendo o que está lendo. Esta percepção pode instigar ações adaptativas, como por exemplo, reler o texto, repensar o que já entende sobre o texto (ou achava que entendia), ler mais um pouco para ver se alguma coisa adiante esclarece o que veio antes ou pedir a ajuda de alguém.

Autorregulação e controle da cognição

O componente metacognitivo de autorregulação e controle da cognição se refere às atividades que os indivíduos se engajam a fim de adaptar e mudar sua cognição ou comportamento. São conhecidas como atividades de autorregulação e controle. Por exemplo, um aluno pode decidir parar uma atividade, decidir continuá-la ou mudá-la durante seu andamento. O conhecimento metacognitivo é importante neste componente, de forma que um aluno pode usar seu conhecimento sobre estratégias para superar determinadas dificuldades (DUNLOSKY; METCALFE, 2009).

Flavell, Miller, P., Miller, S. (1999) utilizam um exemplo bastante claro para o entendimento sobre como os componentes da metacognição se relacionam. Uma estudante de ensino médio acomoda-se em sua escrivaninha para fazer sua lição de casa. Ela pode planejar em que ordem fazer cada trabalho. Pode também fazer testes consigo mesma sobre alguns conhecimentos que serão aplicados na próxima prova, para ver o quanto ainda tem que estudar. A estudante também pode verificar se as suas fichas de vocabulário estão realmente lhe ajudando. Estas são atividades de monitoramento e autorregulação. Foram exemplos de atividades de autorregulação a utilização de estratégias, como quando ela faz um teste consigo mesma e utiliza fichas de vocabulário, e o planejamento, como quando ela planeja a ordem das tarefas.

Um exemplo de atividade de monitoramento é quando ela monitora se suas fichas de vocabulário estão lhe ajudando. As atividades de monitoramento e autorregulação se desenvolvem simultaneamente com o conhecimento metacognitivo. A estudante possui o conhecimento metacognitivo de que frequentemente comete erros de atenção nas operações de soma, e tal conhecimento a leva a sempre verificar duas vezes suas soluções para cada problema. Na direção oposta, sua autorregulação e seu monitoramento metacognitivos podem levar a novos conhecimentos metacognitivos, como quando ela aprende que sua memória para palavras beneficia-se mais de uma estratégia baseada no sentido da palavra do que na memorização.

Em relação às estratégias, Flavell, Miller, P., Miller, S. (1999) distinguem as estratégias cognitivas das estratégias metacognitivas. Enquanto que as estratégias cognitivas buscam ajudar a alcançar o objetivo de qualquer iniciativa cognitiva, as estratégias metacognitivas buscam oferecer informações sobre a iniciativa cognitiva ou seu progresso nela, ou seja, as estratégias metacognitivas servem para monitorar o progresso cognitivo. Um exemplo de estratégia metacognitiva é verificar duas vezes os próprios procedimentos e produtos cognitivos. Esta estratégia poderia ser aplicada em vários domínios, como em jogos de xadrez e resolução de problemas de Física ou de Matemática.

4. Aprendizagem autorregulada

Outra definição muito comum encontrada nas pesquisas que envolvem o construto da metacognição é o de **aprendizagem autorregulada**. Metacognição é um termo mais antigo, que foi definido nos anos 1970 e que, como explicado anteriormente, se refere ao conhecimento relacionado aos próprios processos cognitivos. A partir dos anos 1980, o conceito de aprendizagem autorregulada foi proposto por psicólogos para se referir às várias formas das pessoas monitorarem, controlarem e regularem sua aprendizagem (PINTRICH; WOLTERS; BAXTER, 2000). Segundo Zimmerman (1989), **a autorregulação se refere ao direcionamento de esforços pessoais para a aquisição de conhecimentos e habilidades, sem depender de professores, pais ou outros agentes de instrução**. Aprendizes autorregulados são proativos na definição de metas, na seleção e implantação de estratégias, no monitoramento da efetividade de suas ações na aprendizagem e no controle da aprendizagem (ZIMMERMAN, 2008).

Os alunos são autorregulados à medida que são participantes metacognitivos ativos em sua aprendizagem, além de outros aspectos motivacionais e comportamentais (ZIMMERMAN, 1989). O construto de aprendizagem autorregulada é subordinado ao construto da metacognição, pois incorpora tanto o monitoramento metacognitivo

quanto o controle metacognitivo. Com isso, presume-se que aprendizes autorregulados monitoram e controlam sua aprendizagem (AZEVEDO et al., 2013).

5. O relacionamento entre a metacognição e a cognição

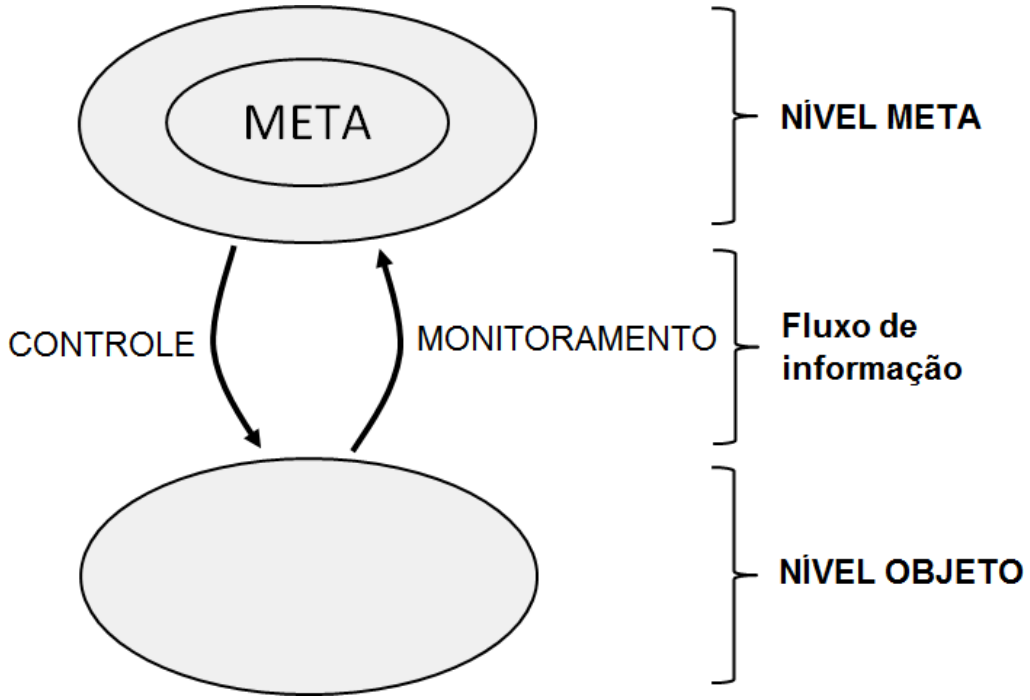
Nelson e Narens (1990) desenvolveram um *framework* sobre o relacionamento existente entre a metacognição e a cognição. O *framework* divide os processos cognitivos em dois ou mais níveis inter relacionados, mas Nelson e Narens (1990) trabalham com apenas dois níveis, que se chamam de **nível meta** e **nível objeto**.

O **nível objeto** representa os processos cognitivos em andamento, como a aprendizagem, o processamento da linguagem, a solução de problemas, entre outros. O **nível meta** contém um modelo que mantém a compreensão da pessoa sobre os processos cognitivos em andamento e que estão sendo engajados para completar uma determinada tarefa.

O **nível meta é a metacognição** e o **nível objeto é a cognição**. Há duas relações entre o nível objeto e o nível meta, chamados "controle" e "monitoramento", e que são definidos em função da direção do fluxo da informação entre os dois níveis (NELSON; NARENS, 1990). A Figura 1 apresenta o relacionamento entre os níveis. A noção básica relacionada ao controle é que a partir do modelo do nível meta é produzida uma atualização do nível objeto, ou seja, é produzido algum tipo de ação no nível objeto, como iniciar uma ação, continuar uma ação ou terminar

uma ação. O conhecimento contido no modelo do nível meta é formado através do monitoramento da pessoa sobre o nível objeto (DUNLOSKY; METCALFE, 2009).

Figura 1: Mecanismo teórico do relacionamento entre metacognição e cognição.



Fonte: Adaptado de Nelson e Narens (1990).

Em outras palavras, é através do monitoramento que o nível meta é modificado e através do controle que o nível objeto é modificado. No exemplo de Flavell, Miller, P., Miller, S. (1999) sobre uma estudante em escrivaninha, no capítulo 3, a aluna emprega esforços cognitivos (nível

objeto) relacionados à utilização de fichas de vocabulário. No nível meta, a estudante tem o conhecimento de que utilizar tais fichas de vocabulário é uma boa estratégia. Através do monitoramento sobre o nível objeto, a aluna pode inferir que não está tendo avanços na aprendizagem e, dessa forma, o monitoramento sobre sua cognição aciona a modificação do nível meta, que passará a registrar o conhecimento de que a utilização das fichas de vocabulário não é uma boa estratégia. Através desse novo conhecimento metacognitivo no nível meta, o controle vai modificar o nível objeto, pois a aluna vai empregar outro processo cognitivo.

6. Monitoramento do conhecimento: uma importante habilidade metacognitiva

Nas últimas décadas, o grupo de pesquisa de Tobias e Everson desenvolveu estudos que buscaram entender um aspecto da metacognição, o **monitoramento do conhecimento**. A hipótese inicial do trabalho dos pesquisadores era de que alunos que não pudessem diferenciar entre o que sabem e o que não sabem, dificilmente se engajariam em atividades metacognitivas, como avaliar sua aprendizagem ou fazer planos para a aprendizagem (TOBIAS; EVERSON, 2002). Também não se espera que esses alunos empreguem estratégias de estudo e aprendizagem mais eficientes (TOBIAS; EVERSON, 2009). Estudos realizados durante vários anos apontaram positivamente para a hipótese inicial dos pesquisadores (TOBIAS; EVERSON, 2002).

Segundo Tobias e Everson (2002), a tarefa do aluno de monitorar o seu conhecimento é complexa. O aluno precisa recuperar o conhecimento declarativo ou procedural da memória de longo prazo, compará-lo com o material do problema que lhe é apresentado, e fazer um julgamento se possui conhecimento para resolver o problema.

Segundo o trabalho de Ullman (2004), o **conhecimento declarativo** está relacionado a conhecimentos sobre fatos (conhecimento semântico) e eventos (memória episódica), e parte deste conhecimento pode ser lembrado conscientemente. O **conhecimento procedural** geralmente não está disponível para acesso consciente, e tem relação, entre outras coisas, às habilidades e outros procedimentos, como a habilidade em jogos de videogame e o conhecimento para andar de bicicleta (ULLMAN, 2004).

A Figura 2 apresenta um modelo hierárquico desenvolvido por Tobias e Everson, em que a capacidade metacognitiva de monitoramento do conhecimento é um pré-requisito para ativar outras habilidades metacognitivas. Eles argumentam que habilidades metacognitivas não podem ser efetivamente aplicadas na falta de um monitoramento de conhecimento preciso e que alunos precisos em monitorar o seu conhecimento utilizam muito tempo e energia em conteúdos novos e não familiares. Em contraste, os alunos com processos de monitoramento de conhecimento menos efetivos provavelmente irão alocar seu tempo e recursos menos efetivamente e perderão um bom tempo estudando o que eles já sabem e terão mais dificuldades em dominar novos assuntos (TOBIAS et al., 1999 apud GAMA, 2004, p. 16).

Figura 2: Modelo de metacognição de Tobias e Everson.

Fonte: Traduzido de Tobias e Everson (2009).

O grupo de pesquisa de Tobias e Everson desenvolveu inúmeros estudos investigando a capacidade dos alunos de monitorar o seu conhecimento anterior. Os estudos apoiaram a hipótese de que alunos que diferenciam precisamente entre o que eles aprenderam anteriormente e o que eles ainda precisam aprender são mais capazes de focar sua atenção e outros recursos cognitivos sobre o material a ser aprendido (TOBIAS; EVERSON, 2002). A consciência dos alunos sobre o que eles sabem e o que não sabem permite que eles possam dar o primeiro passo para remediar suas áreas de conhecimento deficitárias

(FOGARTY, 1994). A importância de alunos conscientes sobre seu conhecimento também vai ao encontro de estudos de Boud e Falchikov (1989), que descrevem que uma das características dos aprendizes efetivos é que eles são realistas sobre os seus pontos fortes e fracos.

Estudos do grupo de Tobias e Everson indicam que a capacidade de monitorar o conhecimento tem relação com o sucesso dos alunos do Ensino Fundamental, Ensino Médio, Ensino Médio profissionalizante e Ensino Superior (TOBIAS; EVERSON, 2000).

Outros estudos ainda sugerem o relacionamento existente entre a habilidade de monitoramento do conhecimento e o **comportamento de busca de ajuda** dos alunos, chamado de comportamento de *help-seeking* (STAVRIANOPOULOS, 2007; TOBIAS; EVERSON, 2002). O processo de *help-seeking* é definido por Zimmerman (1998) como as escolhas de um aluno por materiais específicos, professores ou livros para auxiliar a si mesmo no aprendizado. Um aluno que utiliza estratégias de monitoramento de conhecimento tem mais probabilidade de buscar ajuda acadêmica quando necessário (STAVRIANOPOULOS, 2007). O estudo de Stavrianopoulos (2007) indicou que alunos que são bons monitores do seu conhecimento tendem a se concentrar no estudo de materiais que não são familiares a eles, do que se focar em materiais que eles já possuem conhecimento. Estes achados estão de acordo com os resultados encontrados por Tobias e Everson (2002), em que alunos precisos no monitoramento do conhecimento pediram mais ajuda

relacionada a palavras que eles indicaram serem desconhecidas, enquanto que alunos menos precisos no monitoramento do conhecimento buscaram mais ajuda sobre palavras que eles haviam estimado como sendo conhecidas (TOBIAS; EVERSON, 2002). Tobias e Everson argumentam que o comportamento de *help-seeking* sinaliza um nível de consciência metacognitiva, já que o aluno que busca ajuda pode ter percebido uma lacuna no seu conhecimento.

O comportamento de *help-seeking* sinaliza também uma intenção do aluno para resolver o problema de aprendizagem. Esta consciência sugere que o aluno possa diferenciar entre o que ele sabe e o que não sabe. Desta forma, Tobias e Everson (2002) apresentam evidências de que medidas de monitoramento de conhecimento se relacionam com atividades de *help-seeking*.

Modelos instrucionais que buscam treinar a habilidade do aluno de monitorar o próprio conhecimento podem ser de grande valor em ambientes em que os alunos possuem excesso de confiança no seu conhecimento. Um estudo apresentado por Blackwood (2013) investigou a capacidade de 307 estudantes de graduação de julgar o atual estado de seus conhecimentos. O estudo indicou uma tendência geral de excesso de confiança no conhecimento dos alunos participantes. Considerando a visão de Tobias e Everson (2002) de que a capacidade dos alunos de distinguir entre o que sabem e o que não sabem é importante para o sucesso acadêmico, a tendência de excesso de confiança pode ser

prejudicial para o sucesso de aprendizagem. Com isso, o trabalho de Blackwood (2013) sugere que intervenções que procurem melhorar a capacidade de monitorar o conhecimento podem ter um valor importante nos cenários de alunos com excesso de confiança.

7. A importâncias das habilidades metacognitivas na aprendizagem de programação

A literatura científica destaca que as habilidades metacognitivas são essenciais para a aprendizagem eficaz da programação, pois possibilitam que os alunos monitorem seu próprio progresso, ajustem suas estratégias de estudo e desenvolvam maior autonomia no aprendizado. A metacognição, entendida como a capacidade de refletir sobre os próprios processos cognitivos, desempenha um papel fundamental na resolução de problemas, um dos principais desafios enfrentados por iniciantes em programação (BORUCHOVITCH, 2007). Além disso, a programação de computadores pode ser utilizada como uma estratégia metacognitiva para estimular a busca por conhecimento de forma autônoma e consciente, especialmente em disciplinas como a Matemática e a Lógica Computacional (BATISTELA; TEIXEIRA, 2018).

Pesquisas indicam que a metacognição também influencia a motivação dos alunos, pois a capacidade de controlar e gerenciar os próprios processos de aprendizagem aumenta a percepção de responsabilidade sobre o próprio desempenho acadêmico, além de

fortalecer a confiança na própria capacidade de aprender (MARTINS; MACIEL, 2024). No ensino de programação, isso significa que alunos que desenvolvem maior consciência metacognitiva tendem a persistir mais na resolução de problemas complexos, explorando diferentes abordagens antes de buscar ajuda.

Além disso, o papel do professor na mediação do desenvolvimento metacognitivo é essencial. Estratégias pedagógicas que incentivam a reflexão sobre erros, a verbalização do pensamento durante a codificação e a comparação de diferentes soluções podem contribuir significativamente para a aprendizagem da programação (BUSNELLO; JOU; SPERB, 2007). O ensino explícito de habilidades metacognitivas, como planejamento, monitoramento e avaliação da aprendizagem, ainda é pouco explorado nas práticas educacionais, mas apresenta um grande potencial para melhorar o desempenho e a autonomia dos estudantes em programação (BATISTELA; TEIXEIRA, 2018).

Integrar a metacognição no ensino de programação não apenas pode melhorar o desempenho acadêmico, mas também desenvolver habilidades que são transferíveis para outras áreas do conhecimento.

8. Estratégias metacognitivas que o ALUNO poderia usar na aprendizagem de programação

A aprendizagem de programação exige não apenas o domínio de conceitos técnicos, mas também a capacidade de refletir sobre o próprio processo de aprendizagem, ajustando estratégias conforme a necessidade. Neste capítulo, são apresentadas **10 estratégias metacognitivas** que o aluno poderia adotar para melhorar sua compreensão e desempenho durante a aprendizagem de programação. As estratégias apresentadas foram elaboradas a partir de reflexões fundamentadas na literatura científica sobre metacognição e autorregulação, bem como na experiência do autor com alunos de turmas introdutórias de programação. O objetivo é oferecer orientações práticas que auxiliem os estudantes a desenvolverem um aprendizado mais eficiente, autônomo e consciente, tornando-os capazes de monitorar seu progresso, identificar dificuldades e utilizar técnicas que potencializam a sua aprendizagem.

1. Planejar antes de codificar

Antes de começar a programar um exercício de programação, o aluno poderia planejar sua abordagem de codificação. Isso inclui compreender os requisitos do problema que o exercício traz, identificar as entradas e as saídas esperadas, dividir a implementação em funções ou partes menores e selecionar a estrutura de dados e algoritmos mais adequados para a solução. Essa estratégia está relacionada ao conhecimento metacognitivo da categoria tarefa e estratégia (Flavell, 1979), pois envolve a capacidade de avaliar a complexidade do problema e tomar decisões conscientes sobre a melhor abordagem para resolvê-lo de maneira eficiente e organizada.

Exemplo prático:

Imagine que um aluno precisa escrever um programa que calcule a média de notas de um aluno e informe se ele foi aprovado ou reprovado. Antes de começar a programar, ele poderia pensar no problema e dividir a tarefa em pequenos passos: (1) solicitar as notas ao usuário, (2) calcular a média, (3) definir o critério de aprovação e (4) exibir o resultado. Em seguida, ele poderia decidir qual estrutura de dados usará (variáveis ou *arrays*, por exemplo) e quais comandos serão necessários (entrada, processamento e saída). Esse planejamento poderia evitar que

ele comece a programar de forma desorganizada e tenha que refazer partes do código (retrabalho).

2. Monitorar o próprio aprendizado

O aluno poderia constantemente avaliar se realmente compreendeu o problema, e se compreendeu, se consegue identificar e entender os conceitos de programação envolvidos na tarefa ou se apenas reconhece os termos (palavras ou grupo de palavras) usados no enunciado da tarefa. Essa estratégia se relaciona ao julgamento e monitoramento dos próprios processos de aprendizagem (DUNLOSKY; METCALFE, 2009), pois permite ao estudante avaliar seu nível de compreensão.

Exemplo prático:

Imagine que um aluno recebe a tarefa de implementar um algoritmo de ordenação em Python, como o *Bubble Sort*. Antes de começar a programar, ele lê o enunciado e percebe que já ouviu falar sobre algoritmos de ordenação, mas nunca implementou um. Em vez de apenas reconhecer o termo "Bubble Sort" e seguir diretamente para copiar um código da internet, ele se questiona:

- "Eu realmente entendo como esse algoritmo funciona?"
- "Se eu tivesse que explicá-lo para um colega, conseguiria?"

- "Se me pedirem para modificar o algoritmo para ordenar em ordem decrescente, saberia como fazer?"

Para avaliar sua compreensão, o aluno poderia tentar descrever verbalmente, ou desenhar num papel, como os números são comparados e trocados a cada iteração. Se perceber que não consegue explicar completamente o processo, ele identifica que precisa revisar a teoria antes de escrever o código. Esse exercício de autoavaliação e monitoramento o ajuda a evitar a ilusão de conhecimento e garante que ele compreende o problema antes de programá-lo.

3. Praticar a recuperação ativa

Ao invés de simplesmente reler materiais de estudo, o aluno poderia desafiar-se a lembrar e descrever conceitos sem consultar suas anotações. Isso poderia melhorar a retenção de informações e a autorregulação da aprendizagem, conectando-se ao processo de *feeling of knowing* (DUNLOSKY; METCALFE, 2009).

Exemplo prático:

Suponha que um aluno aprendeu sobre laços de repetição (*for* e *while*). Ao invés de apenas reler o material, ele poderia tentar escrever um programa simples que exibisse os números de 1 a 10 usando ambos

os laços. Caso tenha dificuldade, ele poderia refletir: "O que estou esquecendo? Como posso lembrar disso no futuro?" Esse exercício poderia ativar a memória e ajudar a fixar os conceitos.

4. Refletir sobre os erros e ajustar a estratégia

Sempre que cometer um erro, o aluno poderia analisá-lo e pensar em formas de evitá-lo no futuro. Isso poderia incluir revisar conceitos mal compreendidos. Esse processo reflete o controle metacognitivo, pois envolve ajustes estratégicos baseados na análise do desempenho (NELSON; NARENS, 1990).

Exemplo prático:

Se um estudante está tentando rodar um programa e recebe um erro de *IndexError*, em vez de apenas corrigir mecanicamente, ele poderia parar e refletir: "Por que isso aconteceu? O que eu posso fazer para evitar esse erro no futuro?" Ele poderia perceber que esqueceu de verificar os limites de um *array* antes de acessá-lo. Como solução, ele poderia adotar o hábito de sempre verificar o tamanho do *array* antes de acessar um índice.

5. Utilizar pseudocódigo e/ou diagramas antes de escrever código

Criar um pseudocódigo e/ou um fluxograma antes de escrever um programa poderia ajudar a estruturar o pensamento e prever possíveis dificuldades. Essa prática auxilia no monitoramento do pensamento antes da implementação, reforçando a ideia de monitoramento dos próprios processos de aprendizagem (PINTRICH, WOLTERS; BAXTER, 2000).

Exemplo prático:

Suponha que um aluno precise desenvolver um programa que determine se um número é primo. Em vez de começar a escrever o código imediatamente, ele decide criar um pseudocódigo para organizar seu raciocínio:

- Receber um número do usuário
- Verificar se o número é menor que 2 → Se for, exibir "Não é primo"
- Criar um loop que verifica se o número é divisível por algum valor entre 2 e a metade do mesmo número
    - Se encontrar um divisor, exibir "Não é primo" e encerrar

- Caso contrário, exibir "É primo"

Além disso, ele poderia desenhar um fluxograma para visualizar melhor a estrutura condicional e os loops envolvidos. Essas atividades poderiam ajudar o estudante a antecipar problemas e a organizar melhor o pensamento antes da implementação, reduzindo retrabalho.

6. Estabelecer metas de aprendizado e ajustar conforme necessário

O aluno poderia definir pequenas metas semanais, como aprender uma nova estrutura de controle ou resolver um conjunto de desafios. Se perceber que os objetivos são muito fáceis ou muito difíceis, ele poderia ajustá-los. Isso reforça a ideia de planejamento, monitoramento e ajuste da aprendizagem (ZIMMERMAN, 1989).

Exemplo prático:

Um estudante iniciante pode definir a meta de aprender a manipular arquivos em Python até o final da semana. Se perceber que atingiu o objetivo rapidamente, pode aumentar a dificuldade, como trabalhar com arquivos CSV. Se encontrar dificuldades, pode dividir a meta em partes menores, como primeiro aprender a abrir arquivos,

depois escrever dados em arquivos e, por fim, manipular grandes volumes de dados.

7. Comparar diferentes abordagens para resolver o mesmo problema

Em vez de se contentar com a primeira solução que encontrar, o aluno poderia tentar diferentes métodos de resolver o mesmo problema e avaliar qual é mais eficiente. Essa estratégia se relaciona com o conhecimento metacognitivo sobre estratégias (FLAVELL, 1979), permitindo ao aluno escolher a melhor técnica para diferentes contextos.

Exemplo prático:

Suponha que um aluno esteja desenvolvendo um programa para calcular a soma de todos os números pares dentro de uma lista em Python. A primeira abordagem que ele implementa é a seguinte:

```
def soma_pares(lista):
    soma = 0
    for num in lista:
        if num % 2 == 0:
            soma += num
    return soma
```

A princípio, o código funciona, mas ele decide explorar outras maneiras de resolver o problema e comparar as soluções. Ele testa uma abordagem utilizando *list comprehension* do Python, tornando o código mais conciso:

```
def soma_pares(lista):
    return sum([num for num in lista if num % 2 == 0])
```

Depois, ele experimenta a função *filter()*, uma abordagem funcional que pode ser mais eficiente em algumas situações:

```
def soma_pares(lista):
    return sum(filter(lambda num: num % 2 == 0, lista))
```

Ao comparar as três soluções, o aluno percebe que a primeira abordagem é mais didática e fácil de entender para iniciantes, enquanto a segunda e a terceira podem ser menos didáticas, mas mais compactas e mais eficientes em listas grandes. Essa experimentação ajuda o aluno a desenvolver flexibilidade cognitiva e a entender que um mesmo problema pode ter diversas soluções, sendo importante analisar qual estratégia é mais adequada para cada contexto. Essa prática o ajuda a melhorar sua capacidade de escolha e eficiência na programação.

8. Usar a técnica do “Pensar em Voz Alta”

Ao escrever código, o aluno poderia verbalizar seu pensamento, explicando suas decisões para si mesmo ou para colegas e o professor. Isso poderia ajudá-lo a detectar inconsistências e aprimorar sua compreensão. Essa abordagem está ligada ao monitoramento metacognitivo e à avaliação da compreensão (FLAVELL, MILLER & MILLER, 1999).

Exemplos práticos:

Ao escrever um programa para calcular o número fatorial de um número, o aluno poderia narrar seus passos: "Eu preciso de um loop que multiplique os números até N. O loop começa em 1 ou em 0? Preciso de uma variável acumuladora? Não sei. Ah, sim, preciso!" Esse processo poderia ajudar o aluno a identificar erros antes mesmo de rodar o código e a racionalizar melhor.

Em outro exemplo, imagine que um aluno esteja implementando um programa para verificar se uma palavra é um palíndromo - ou seja, se pode ser lida da mesma forma de trás para frente. Ele, então, decide **verbalizar seu pensamento** enquanto programa.

- "Primeiro, preciso receber a palavra do usuário."

- "Agora, preciso inverter a palavra para compará-la com a original. Mas como faço para inverter uma palavra?"
- "Se a palavra original for igual à invertida, então é um palíndromo."

Ao explicar cada decisão em voz alta, ele poderia perceber que esqueceu um detalhe: “Ahh, esqueci que a comparação deve ignorar maiusculas e minúsculas.”. Esse processo de verbalizar o raciocínio o ajudou a detectar um possível problema antes que ele se manifestasse no código. Além disso, se estivesse aprendendo em grupo com outros colegas ou com um professor, poderia compartilhar seu raciocínio e receber *feedback* sobre suas decisões. A estratégia do "pensar em voz alta" auxilia no monitoramento da compreensão e permite ao aluno avaliar se realmente entende cada etapa do programa ou se apenas está seguindo instruções mecanicamente.

9. Testar o código constantemente e refletir sobre os resultados

Em vez de esperar terminar todo o programa para testá-lo, o aluno poderia rodar pequenos trechos e verificar se funcionam como esperado. Isso reforça o monitoramento contínuo da cognição e da aprendizagem (DUNLOSKY; METCALFE, 2009), permitindo ajustes em tempo real.

Exemplo prático:

Imagine que um aluno está desenvolvendo um sistema de cadastro de usuários em Python, no qual o programa deve solicitar um nome e uma idade, armazená-los em um dicionário (uma estrutura de dados do Python) e exibir os dados cadastrados. Em vez de escrever todo o código de uma só vez e só então testar, ele poderia seguir uma abordagem iterativa, testando pequenos trechos do código à medida que progride.

Primeiro, ele poderia verificar se a entrada do usuário está sendo corretamente coletada:

```
nome = input("Digite seu nome: ")
idade = input("Digite sua idade: ")
print(f"Nome: {nome}, Idade: {idade}")
```

Como resultado esperado, o programa captura e exibe na tela os valores digitados. Então depois de confirmar que os inputs funcionam, ele adiciona a lógica para armazenar os dados:

```
usuario = {"nome": nome, "idade": idade}
print(usuario)
```

Como resultado esperado, impresso na tela, o dicionário deve conter os dados informados. O aluno, então, percebe que a idade deve ser armazenada como número inteiro. Ele modifica o código e o testa novamente:

```
usuario = {"nome": nome, "idade": int(idade)}
print(usuario)
```

Como resultado esperado, agora, a idade é um número, permitindo futuras operações matemáticas. Por fim, com tudo funcionando, ele implementa a exibição dos dados formatados corretamente:

```
print(f"Usuário cadastrado: {usuario['nome']} tem {usuario['idade']} anos.")
```

Como resultado esperado, o programa exibe os dados corretamente formatados. Essa estratégia reforça o monitoramento contínuo da cognição e da aprendizagem, permitindo que o aluno faça ajustes em tempo real (DUNLOSKY; METCALFE, 2009), evitando a frustração de encontrar vários erros ao final do desenvolvimento.

10. Buscar ajuda de forma estratégica

Uma habilidade essencial para o aprendizado da programação é **saber quando e como buscar ajuda**. Isso inclui a leitura da documentação oficial, a exploração de tutoriais confiáveis (vídeos, podcasts ou formato texto) e a consulta a colegas, professores, ou até mesmo IAs generativas como o ChatGPT. Desenvolver essa estratégia é fundamental para evitar a dependência excessiva de respostas prontas e fortalecer a autonomia. Essa prática está alinhada ao conceito de *help-seeking* como uma forma de monitoramento do conhecimento, permitindo que o aluno identifique suas dificuldades e busque recursos adequados para superá-las (TOBIAS; EVERSON, 2002).

Exemplo prático:

Um aluno iniciante está desenvolvendo um programa em Python que lê um arquivo CSV e processa seus dados. Ele sabe que precisa usar a biblioteca *pandas* do Python, mas encontra um erro ao tentar abrir o arquivo:

```
import pandas as pd

dados = pd.read_csv("dados.csv")
print(dados.head())
```

Ao rodar o código, ele recebe o erro:

```
FileNotFoundError: [Errno 2] No such file or directory: 'dados.csv'
```

Em vez de entrar em pânico e pedir ajuda imediatamente, ele poderia parar um tempo para refletir:

- "Será que o problema está no código ou no local onde salvei o arquivo?"
- "Eu já tentei resolver esse problema sozinho ou estou buscando ajuda cedo demais?"

Antes de perguntar para um colega, professor, ou até mesmo para alguma IA generativa como o ChatGPT, o aluno decide:

- Pesquisar na documentação do pandas para entender melhor a função *read_csv()*.
- Verificar se o arquivo está no diretório correto e testar passando o caminho completo do arquivo.
- Buscar o erro no Google e verificar possíveis soluções em fóruns como Stack Overflow.

Após esses passos, ele percebe que o erro acontece porque o arquivo não está no mesmo diretório do código. Ele então ajusta o caminho e o problema é resolvido. Caso não tivesse encontrado a solução, ele poderia então pedir ajuda a um professor ou colega, explicando o que já tentou e quais foram os resultados.

Esse comportamento demonstra um uso estratégico e eficiente da busca por ajuda, garantindo que ele aprenda com o processo e evite depender excessivamente de terceiros para resolver problemas.

9. Estratégias que o PROFESSOR poderia usar para ajudar os alunos a se beneficiarem da metacognição na aprendizagem de programação

O ensino de programação vai além da simples transmissão de conceitos e técnicas. Ele requer que o professor auxilie os alunos a desenvolverem habilidades metacognitivas, permitindo que monitorem seu próprio aprendizado, avaliem suas dificuldades e ajustem suas estratégias para aprender de forma mais eficiente. Neste capítulo, são **apresentadas 8 estratégias** que o professor poderia adotar para incentivar e potencializar o uso da metacognição na aprendizagem de programação. Uma **9ª estratégia** é apresentada, mas no próximo capítulo. As estratégias foram elaboradas a partir de reflexões fundamentadas na literatura científica sobre metacognição e autorregulação, bem como na experiência do autor com alunos de turmas introdutórias de programação. O objetivo é oferecer abordagens práticas para que os professores possam criar um ambiente de aprendizado que favoreça a reflexão, a autonomia e o desenvolvimento de estratégias eficazes por parte dos alunos.

1. Estimular o planejamento antes da programação

Antes de começar a codificar, o professor poderia incentivar os alunos a planejar suas soluções. Isso inclui compreender o problema, dividir a tarefa em partes menores, e selecionar as melhores estratégias. Essa abordagem ajuda os alunos a desenvolverem o conhecimento metacognitivo sobre tarefas e estratégias (FLAVELL, 1979).

Formas de aplicar:

- Antes de entregar um exercício de programação, o professor poderia pedir aos alunos que descrevessem verbalmente o problema.
- O professor poderia pedir aos alunos que expliquem por que escolheram determinada abordagem.
- Antes do aluno iniciar a implementação da solução para o exercício, o professor poderia fazer perguntas como: "Quais são as entradas e saídas do problema?", "Quais estruturas de controle você utilizaria?", "Quantas variáveis são necessárias?".

2. Promover a verbalização do pensamento

O professor poderia incentivar os alunos a verbalizar seu raciocínio enquanto escrevem o código. Isso poderia ajudá-los a detectar falhas de compreensão e a fortalecer a avaliação e monitoramento da aprendizagem (FLAVELL, MILLER & MILLER, 1999).

Formas de aplicar:

- O professor poderia propor atividades de "programação em pares", onde um aluno escreve o código e o outro o acompanha explicando cada passo em voz alta.
- O professor poderia pedir que os alunos apresentem suas soluções para a turma, explicando suas escolhas.
- Durante a correção de exercícios, o professor poderia escolher um código de um aluno e pedir para que ele explicasse como chegou naquela solução.

3. Criar oportunidades para a reflexão sobre erros

Ensinar os alunos a refletirem sobre os erros cometidos e ajustarem suas estratégias de estudo e programação. Esse processo está

diretamente ligado ao monitoramento da aprendizagem e controle metacognitivo (NELSON; NARENS, 1990).

Formas de aplicar:

- "Caça aos Bugs": O professor apresenta um código com erros intencionais e desafia os alunos a identificá-los, explicá-los e sugerir soluções. Essa atividade estimula o pensamento crítico e a depuração sistemática de código.
- Análise de erros em avaliações: Após a correção de provas ou trabalhos, incentivar os alunos a revisarem seus erros e escreverem uma explicação sobre o que ocorreu, qual foi a falha lógica ou sintática e como poderiam evitá-la no futuro. Essa prática ajuda a consolidar o aprendizado e a desenvolver o monitoramento metacognitivo.
- Ensino de depuração (*debugging*): O professor poderia ensinar técnicas de depuração eficazes, como o uso de *print statements*, depuradores integrados nas IDEs (*debuggers*) e o método de isolamento de erros. Poderia demonstrar na prática como analisar mensagens de erro, interpretar *logs* e testar pequenas partes do código para encontrar falhas de forma eficiente.

4. Incentivar a prática da recuperação ativa

Em vez de apenas reler materiais ou assistir a vídeos passivamente, os alunos poderiam se desafiar a lembrar conceitos sem consultar anotações.

Formas de aplicar:

- Antes de introduzir um novo conceito, o professor poderia perguntar aos alunos o que lembram sobre o conteúdo relacionado.
- O professor poderia criar atividades de "reconstrução de código", onde os alunos tentam reescrever um programa que viram anteriormente sem consultar o código original.
- O professor poderia utilizar *quizzes* rápidos no início e no final da aula para reforçar conceitos fundamentais.

5. Ensinar os alunos a testarem e compararem diferentes soluções

O professor poderia ensinar e incentivar os alunos a explorarem múltiplas abordagens para resolver um mesmo problema e a refletirem sobre qual é a mais eficiente.

Formas de aplicar:

- Pedir que os alunos resolvam um problema de duas maneiras diferentes e comparem a eficiência de cada abordagem. Exemplo:
    - Resolver um problema de soma acumulada usando um loop *for* e depois usando uma função *sum()*.
    - Implementar um algoritmo de busca sequencial e depois otimizar para busca binária.
- Propor debates sobre qual abordagem é melhor em diferentes cenários.

6. Desenvolver o hábito de questionamento e autoavaliação

Ensinar os alunos a sempre se questionarem se realmente compreendem um conceito ou apenas reconhecem as palavras em um enunciado de problema.

Formas de aplicar:

- Durante os exercícios, o professor poderia pedir aos alunos que respondam perguntas como:

- "Eu realmente entendi este conceito ou apenas reconheço aquelas palavras no enunciado?"
- "Se eu tivesse que explicar isso a alguém, conseguiria?"
- "Se este código apresentasse um erro, eu saberia identificar onde?"

7. Estabelecer metas de aprendizado e acompanhar a evolução dos alunos

O professor poderia incentivar os alunos a definirem metas semanais e ajustarem o nível de dificuldade conforme necessário, promovendo a autorregulação da aprendizagem (ZIMMERMAN, 1989).

Formas de aplicar:

- Pedir que cada aluno escreva uma meta de aprendizado para a semana, como "Entender bem a estrutura de loops" ou "Praticar a manipulação de listas em Python".
- Pedir aos alunos que façam um *check-in* no final da semana: "Você atingiu sua meta? Se não, qual foi a dificuldade? Como podemos ajustar sua estratégia?"
- Propor desafios opcionais para os alunos que avançam mais rápido, enquanto reforça os conceitos básicos para quem precisa.

8. Ensinar os alunos a buscar ajuda de forma estratégica

O professor poderia mostrar aos alunos como buscar ajuda da maneira correta, garantindo que não se tornem dependentes de respostas prontas de terceiros, mas sim que desenvolvam a autonomia na aprendizagem.

Formas de aplicar:

- Ensinar os alunos a fazerem perguntas bem formuladas ao professor ou colegas, explicando o que já tentaram e onde encontraram dificuldades, antes de pedir ajuda.
- Mostrar como usar a documentação oficial de linguagens de programação e incentivá-los a consultá-la antes de pedir ajuda para colegas e professores.
- Criar um sistema de "perguntas e respostas" na turma, onde alunos podem ajudar uns aos outros antes de recorrer ao professor.

10. Aprendizagem Baseada em Problemas (PBL) e Aprendizagem Baseada em Projetos (PjBL)

Há uma última estratégia para compartilhar com os professores, mas achei melhor separá-la para um capítulo próprio. Neste capítulo, apresento discussões sobre a Aprendizagem Baseada em Problemas (PBL, do inglês *Problem-Based Learning*) e a Aprendizagem Baseada em Projetos (PjBL, do inglês *Project-Based Learning*).

Vamos começar pela Aprendizagem Baseada em Problemas. O PBL é uma abordagem pedagógica que professores e currículos acadêmicos poderiam explorar para melhorar o engajamento dos alunos, suas habilidades metacognitivas e de autorregulação e, principalmente, a aprendizagem. A Aprendizagem Baseada em Problemas é uma abordagem pedagógica que coloca o aluno no centro do processo de aprendizagem, desafiando-o a resolver problemas complexos e realistas enquanto desenvolve conhecimento e habilidades essenciais para sua formação.

No ensino de programação, o PBL pode ser uma ferramenta poderosa para estimular habilidades metacognitivas, pois exige que os alunos planejem, monitorem e avaliem continuamente suas próprias

estratégias para resolver desafios (SAVERY; DUFFY, 1995). O professor pode utilizar o PBL para promover a reflexão ativa sobre conceitos e técnicas, incentivando os alunos a identificar lacunas em seu conhecimento, formular hipóteses e testar diferentes soluções.

Um exemplo prático dessa abordagem seria apresentar um problema real, como o de ordenar uma lista de palavras, e pedir que os alunos identifiquem quais estruturas de dados e algoritmos seriam mais adequados para resolver o desafio. Durante o processo, os estudantes são incentivados a verbalizar suas estratégias, comparar abordagens e refletir sobre suas escolhas, desenvolvendo assim habilidades de autorregulação da aprendizagem (HMELO-SILVER, 2004).

Além de estimular o pensamento crítico, o PBL permite que os alunos adquiram uma consciência mais precisa sobre suas próprias dificuldades e progressos. De acordo com Barrows (1986), um dos criadores da metodologia PBL, os alunos que aprendem por meio dessa abordagem tendem a desenvolver maior autonomia e autoconfiança, pois aprendem a buscar informações de forma independente e a ajustar suas estratégias conforme necessário.

Para que o PBL seja eficaz na aprendizagem de programação, o professor deve assumir o papel de facilitador, guiando os alunos com perguntas estratégicas que os levem a refletir sobre suas decisões. Questões como "Por que essa estrutura de dados é a mais adequada para este problema?", "Quais foram os desafios encontrados e como você os

superou?" e "Se tivesse que refatorar seu código, o que faria diferente?" ajudam a desenvolver a autorregulação metacognitiva, promovendo um aprendizado mais profundo e significativo (SCHMIDT; ROTGANS; YEW, 2011).

Assim, ao integrar o PBL no ensino de programação, o professor não apenas ensina conceitos técnicos, mas também ajuda os alunos a se tornarem aprendizes autônomos e estratégicos, capacitando-os a enfrentar desafios cada vez mais complexos ao longo de sua formação acadêmica e profissional.

Aprendizagem Baseada em Projetos (PjBL)

Uma abordagem similar ao PBL que também poderia ser aplicada, e que é muito interessante para cursos de programação, é a Aprendizagem Baseada em Projetos (PjBL, do inglês *Project-Based Learning*).

A Aprendizagem Baseada em Problemas (PBL) e a Aprendizagem Baseada em Projetos (PjBL) são abordagens pedagógicas ativas que compartilham princípios fundamentais, como a ênfase na autonomia do aluno, na resolução de problemas e na construção do conhecimento de forma contextualizada. No entanto, enquanto o PBL se concentra na resolução de um problema específico (ordenação de uma lista de palavras), a Aprendizagem Baseada em Projetos envolve a criação de

um produto final, um projeto que pode ser um software, um módulo de software, um aplicativo ou uma funcionalidade, ao longo de um período mais extenso (THOMAS, 2000).

No ensino de programação, essas abordagens podem ser combinadas de forma estratégica: o PBL pode ser utilizado nas fases iniciais para estimular o raciocínio crítico e a exploração de diferentes soluções para desafios computacionais, enquanto o PjBL pode consolidar esse conhecimento por meio do desenvolvimento de projetos completos, nos quais os alunos aplicam as habilidades adquiridas em um contexto mais amplo e integrador. Ambas as metodologias fortalecem a metacognição e a autorregulação, permitindo que os alunos reflitam sobre suas escolhas, ajustem suas estratégias de aprendizagem e desenvolvam maior autonomia ao longo do processo de ensino.

Para que a Aprendizagem Baseada em Problemas (PBL) e a Aprendizagem Baseada em Projetos (PjBL) sejam aplicadas na prática, os professores podem adaptar seus planos de ensino para atender essas abordagens. Além disso, mudanças mais estruturais podem ser feitas nos Planos Pedagógicos de Curso (PPC), ajustando a metodologia de ensino para que essas abordagens sejam incorporadas de forma sistemática no currículo. Isso pode envolver a reformulação de disciplinas, a inclusão de avaliações baseadas em projetos e a flexibilização da carga horária para permitir um aprendizado mais ativo e centrado no aluno. Uma estratégia importante nesse processo é promover a interdisciplinaridade

entre as disciplinas, possibilitando que os estudantes apliquem conhecimentos de diferentes áreas na solução de problemas complexos. Dessa forma, a aprendizagem se torna mais contextualizada e alinhada com os desafios reais do mercado, incentivando o pensamento crítico, a colaboração e a autonomia dos alunos.

11. Considerações finais

Aprender a programar é um desafio intelectual que exige persistência, estratégia e reflexão contínua. No entanto, muitos estudantes encontram dificuldades não apenas na assimilação de conceitos técnicos, mas também na forma como organizam e regulam seu próprio aprendizado. A metacognição, ao permitir que o aluno monitore, avalie e ajuste suas estratégias de estudo, surge como um elemento-chave para transformar a experiência de aprendizado, tornando-a mais eficaz e autônoma.

Neste livro, discutimos estratégias metacognitivas que podem ser adotadas por alunos para aprimorar sua aprendizagem, bem como abordagens que professores podem implementar para incentivar a reflexão e a autorregulação nos seus estudantes. Exploramos a importância do planejamento antes da codificação, da verbalização do pensamento, da experimentação de diferentes abordagens, do monitoramento do progresso e da busca estratégica por ajuda, entre outras práticas fundamentais. Além disso, destacamos metodologias ativas como a Aprendizagem Baseada em Problemas (PBL) e a Aprendizagem Baseada em Projetos (PjBL), que incentivam a resolução de problemas de maneira autônoma e estruturada.

Ao longo das discussões, ficou claro que aprender a programar vai além do domínio técnico, sendo também um processo cognitivo e metacognitivo. Alunos que refletem sobre seu próprio aprendizado absorvem melhor os conteúdos, tornam-se mais resilientes diante de desafios e mais preparados para acompanhar a evolução tecnológica.

Para professores, a adoção de estratégias que promovam a metacognição em sala de aula é essencial para formar aprendizes mais independentes e críticos. A introdução de atividades que estimulem a autorregulação da aprendizagem, como análise de erros, depuração guiada, revisão reflexiva e planejamento estruturado do código, pode fazer uma grande diferença na maneira como os alunos percebem, controlam e aprimoram seu próprio processo de aprendizado.

Este livro buscou oferecer um guia prático e fundamentado para aqueles que desejam aprimorar a forma como aprendem e ensinam programação. A metacognição não é um conceito restrito ao campo da Psicologia da Educação — ela é uma ferramenta poderosa que pode capacitar estudantes e educadores a transformarem sua abordagem em relação ao conhecimento. Ao adotar uma mentalidade metacognitiva, tanto alunos quanto professores poderão tornar o ensino e a aprendizagem de programação mais estratégicos, eficientes e, acima de tudo, significativos.

Por fim, o autor deste livro ficaria muito feliz em receber *feedbacks* de alunos e professores sobre a aplicação das estratégias

discutidas neste livro, através do e-mail do autor: tkautzmann@gmail.com. Compreender como essas abordagens funcionaram com os leitores, quais desafios surgiram e quais adaptações foram feitas pode enriquecer ainda mais as futuras reflexões sobre o ensino e a aprendizagem da programação. Se você experimentou alguma estratégia apresentada aqui, compartilhe suas percepções, dificuldades e sugestões. Seu retorno é essencial para aprimorar o conhecimento sobre o uso da metacognição no ensino de programação e para ajudar a construir práticas educacionais cada vez mais eficazes e acessíveis.

Referências

AZEVEDO, R.; TAUB, M.; MUDRICK, N. Understanding and reasoning about real-time dynamic uncertainty in complex tasks using metacognitive strategies. *Educational Psychologist*, v. 48, n. 1, p. 25-41, 2013.

AZEVEDO, R.; HARLEY, J.; TREVORS, G.; DUFFY, M.; FEYZI-BEHNAGH, R.; BOUCHET, F.; LANDIS, R.. Using Trace Data to Examine the Complex Roles of Cognitive, Metacognitive, and Emotional Self-Regulatory Processes During Learning with Multi-agent Systems. In: AZEVEDO, R.; ALEVEN, V. (Eds.). International Handbook of Metacognition and Learning Technologies. New York: Springer, cap. 28, p. 427-449, 2013.

BARROWS, H. S. A taxonomy of problem-based learning methods. *Medical Education*, v. 20, n. 6, p. 481-486, 1986.

BATISTELA, F.; TEIXEIRA, A. C. A programação de computadores como processo metacognitivo: uma experiência na escola de hackers. *Revista Intersaberes*, v. 13, n. 30, p. 1-15, 2018. Disponível em: https://periodicos.uninter.com/revistaintersaberes/article/view/11341. Acesso em: 10 mar. 2025.

BLACKWOOD, T.. Business undergraduates' knowledge monitoring accuracy: how much do they know about how much they know? Teaching in Higher Education, New York, v. 18, n. 1, p. 65-77, 2013.

BERARDI-COLLETA, B.; BUYER, L.S.; DOMINOWSKI, R.L.; RELLINGER, E.R.. Metacognition and problem solving: A process-oriented approach. Journal of Experimental Psychology, v. 21, n.1, p. 205-223, 1995.

BORUCHOVITCH, E. A metacognição e a construção do conhecimento: sua importância na formação de professores. In: CONGRESSO NACIONAL DE PSICOLOGIA ESCOLAR E EDUCACIONAL, 8; SIMPÓSIO CRIATIVIDADE, METACOGNIÇÃO E APRENDIZAGEM À DISTÂNCIA NA GESTÃO DO CONHECIMENTO, 2007, São João Del Rei. *Anais*... São João Del Rei, 2007. CD-ROM.

BOUD, D.; FALCHIKOV, N.. Quantitative Studies of Student Self-Assessment in Higher Education: a Critical Analysis of Findings. Higher Education, New York, v. 18, n. 5, p. 529-549, 1989. Disponível em: <http://link.springer.com/article/10.1007/BF00138746>. Acesso em: 11 out. 2014.

BRANSFORD, J. D.; BROWN, A. L.; COCKING, R. R. *How People Learn: Brain, Mind, Experience, and School.* Washington, D.C.: National Academy Press, 1999.

BROWN, A. L.. Knowing When, Where, and How to Remember: A Problem of Metacognition. Technical Report n. 47. Washington: National InstiFLAVELL, J.. Metacognition and cognitive monitoring: A new area of cognitive-developmental inquiry. American Psychologist, New York, v. 34, n. 10, p. 906–911, 1979. Disponível em: <https://saltworks.stanford.edu/assets/mg885pk5612.pdf>. Acesso em: 8 set. 2014.tute of Education, 1977.

BUSNELLO, F. B.; JOU, G. I.; SPERB, T. M. Desenvolvimento de habilidades metacognitivas: capacitação de professores de ensino fundamental. *Psicologia: Reflexão e Crítica*, v. 20, n. 2, p. 348-356, 2007. Disponível em: https://www.scielo.br/j/prc/a/h9wn3swpXPZ6QwzcV7FfS8D/. Acesso em: 10 mar. 2025.

CHI, M. T. H.; BASSOK, M.; LEWIS, M. W.; REIMANN, P.; GLASER, R.. Self-Explanations: How Students Study and Use Examples in Learning to Solve Problems. Cognitive Science, Austin, TX, v. 182, p. 145-182, 1989.

DESOETE, A.; ROEYERS, H.; DE CLERCQ, A.. Can Offline Metacognition Enhance Mathematical Problem Solving? Journal of Educational Psychology, Washington, v. 95, n. 1, p. 188-200, 2003.

DUNLOSKY, J.; METCALFE, J.. Metacognition. Los Angeles: SAGE, 2009.

FLAVELL, J. H.; MILLER, P. H.; MILLER, S. A.. Desenvolvimento Cognitivo. Porto Alegre: Artmed, 1999.

FOGARTY, R. J.. How to Teach for Metacognitive Reflection. Glenview: Pearson, 1994.

GAMA, C. A.. Integrating Metacognition Instruction in Interactive Learning Environments. 2004. 246 f. Thesis (PhD) -- University of Sussex, Brighton, UK, 2004. Disponível em: <http://homes.dcc.ufba.br/~claudiag/thesis/Thesis_Gama.pdf>. Acesso em: 7 out. 2014.

HARTMAN, H. J.. Metacognition in teaching and learning: An introduction. Instructional Science, Netherlands, v. 26, n. 1-2, p. 1-3, 1998.

HMELO-SILVER, C. E. Problem-based learning: What and how do students learn? *Educational Psychology Review*, v. 16, n. 3, p. 235-266, 2004.

KRAMARSKI, B.; MEVARECH, Z. R.; ARAMI, M.. The effects of Metacognitive Instruction on Solving Mathematical Authentic Tasks. Educational Studies in Mathematics, Netherlands, v. 49, p. 225-250, 2002.

KOLENCIK, P. L.; HILLWIG, S. A.. Encouraging Metacognition: Supporting learners through metacognitive teaching strategies. New York: Peter Lang Publishing, 2011.

MARTINS, C. O.; MACIEL, C. M. L. A. Estratégias metacognitivas no desenvolvimento da consciência fonológica: reflexões de professoras alfabetizadoras de Cuiabá. Revista Prática Docente, v. 9, p. e24012, 2024.

NELSON, T. O.; NARENS, L.. Metamemory: A Theoretical Framework and New Findings. In: BOWER, Gordon H. (Ed.). The Psychology of Learning and Motivation. New York, NY: Academic Press, v. 26, cap. 4, p. 125-173, 1990.

PINTRICH, P. R.; WOLTERS, C. A.; BAXTER, G. P.. Assessing Metacognition and Self-Regulated Learning. In: SCHRAW, Gregory; IMPARA, James C.. Issues in the measurement of metacognition. Lincoln, NE: Buros Institute of Mental Measurements, p. 43-97, 2000.

PRATHER, J.; BECKER, B. A.; CRAIG, M.; DENNY, P.; LOKSA, D.; MARGULIEUX, L. What do we think we think we are doing? Metacognition and self-regulation in programming. In: **INTERNATIONAL COMPUTING EDUCATION RESEARCH CONFERENCE (ICER), 2020**, Proceedings [...]. New York: ACM, 2020. p. 2-13.

RIDLEY, D. S.; SCHUTZ, P. A.; GLANZ, R. S.; WEINSTEIN, C. E.. Self-regulated learning: the interactive influence of metacognitive awareness and goal-setting. The Journal of Experimental Education, Mawah, NJ, v. 60, n. 4, p. 293-306, 1992.

SAVERY, J. R.; DUFFY, T. M. Problem-based learning: An instructional model and its constructivist framework. *Educational Technology*, v. 35, n. 5, p. 31-38, 1995.

SCHMIDT, H. G.; ROTGANS, J. I.; YEW, E. H. J. The process of problem-based learning: What works and why. *Medical Education*, v. 45, n. 8, p. 792-806, 2011.

SCHOENFELD, A. H.. What's all the Fuss About Metacognition? In: SCHOENFELD, A. H. (Ed.). Cognitive Science and Mathematics Education. Hillsdale, NJ: Lawrence Erlbaum Associates, cap. 8, p. 189-215, 1987.

STAVRIANOPOULOS, K.. Adolescent's Metacognitive Knowledge Monitoring and Academic Help Seeking: The Role of Motivation Orientation. College Student Journal, Alabama, v. 41, n. 2, p. 444-453, 2007.

TARRICONE, P.. The Taxonomy of Metacognition. New York: Psychology Press, 2011.

THOMAS, J. W. A review of research on project-based learning. *The Autodesk Foundation*, p. 1-44, 2000.

TOBIAS, S.; EVERSON, H.. Assessing Metacognitive Knowledge Monitoring. In: SCHRAW, G.; IMPARA, J. C. (Ed.). Issues in the measurement of metacognition. Lincoln, NE: Buros Institute of Mental Measurements and Erlbaum Associates, 2000. cap. 4, p. 147-222. Disponível em: <http://digitalcommons.unl.edu/burosmetacognition/5/>. Acesso em: 10 out. 2014.

TOBIAS, S.; EVERSON, H.. Knowing What You Know and What You Don't: Further Research on Metacognitive Knowledge Monitoring (Research Report). New York: The College Board. n. 2002-3, p. 1-25, 2002.

TOBIAS, S.; EVERSON, H. T. *Knowing what you know: The role of metacognition in education.* Washington, D.C.: American Psychological Association, 2002.

TOBIAS, S.; EVERSON, H.. The Importance of Knowing What You Know: A Knowledge Monitoring Framework for Studying Metacognition in Education. In: HACKER, D. J.; DUNLOSKY, J.; GRAESSER, A. C. (Ed.). Handbook of Metacognition in Education. New York, NY: Routledge, cap. 7, p. 107-127, 2009.

ULLMAN, M. T.. Contributions of memory circuits to language: the declarative/procedural model. Cognition, The Netherlands, v. 92, p. 231–270, 2004.

ZIMMERMAN, B. J.. Academic Studying and the Development of Personal Skill: A Self-Regulatory Perspective. Educational Psychologist, New York, NY, v. 33, p. 73-86, 1998.

ZIMMERMAN, B. J. Investigating self-regulation and motivation: Historical background, methodological developments, and future prospects. *American Educational Research Journal*, v. 45, n. 1, p. 166-183, 2008.

Aprender a Aprender Programação:

Estratégias Metacognitivas para Melhorar o Aprendizado e o Ensino de Programação

1ª edição

Aprender a programar vai muito além de memorizar comandos e escrever códigos – requer reflexão, planejamento e autorregulação. No entanto, muitos estudantes enfrentam dificuldades não por falta de esforço, mas por não saberem aprender de maneira eficaz.

Neste livro, você descobrirá estratégias metacognitivas que transformarão sua forma de aprender programação, se for um aluno, ou de ensiná-la, se for um professor. Baseado em pesquisas científicas e na experiência do autor com alunos de turmas introdutórias de programação, este livro apresenta técnicas para desenvolver autonomia na aprendizagem, aprimorar a resolução de problemas e melhorar a retenção do conhecimento.

Se você é um estudante que deseja aprender programação com mais eficiência ou um professor que busca estratégias para ajudar seus alunos a desenvolverem pensamento crítico e autonomia, esta obra é para você!

Tiago Roberto Kautzmann

Série Aprendizagem de Programação