Avaliando o Efeito da Criação de Testes na Aprendizagem e no Desempenho de Estudantes de Programação

André Almeida[1], **Dalton D. S. Guerrero**[1], **Wilkerson L. Andrade**[1]

[1]Universidade Federal de Campina Grande, Campina Grande - Paraíba, Brasil

andrealmeida@copin.ufcg.edu.br, dalton@dsc.ufcg.edu.br,

wilkerson@computacao.ufcg.edu.br

Abstract. *Programming requires transforming problems into systematic instructions. However, many students face difficulties early on, particularly in interpreting problem statements and constructing correct solutions. This study evaluates a strategy based on interaction with a reference solution accessible only through inputs and outputs, encouraging students to formulate their own tests. The results indicate that the strategy contributed to improved student performance, notably reducing failures, increasing success rates, and enhancing the reliability of solutions. These findings highlight the potential of the approach as a pedagogical support tool in programming education.*

Resumo. *Programar exige transformar problemas em instruções sistemáticas. No entanto, muitos estudantes enfrentam dificuldades logo na etapa inicial, relacionada à interpretação dos enunciados e à construção de soluções corretas. Este estudo avalia uma estratégia baseada na interação com uma solução de referência, acessível apenas por entradas e saídas, incentivando a formulação de testes pelos estudantes. Os resultados indicam que a estratégia contribuiu para a melhora no desempenho dos estudantes, com destaque para a redução de falhas, o aumento da taxa de sucesso e a maior confiabilidade das soluções. Tais evidências reforçam o potencial da proposta como apoio pedagógico ao ensino de programação.*

1. Introdução

Programar é a atividade de transformar um problema inicial em uma sequência clara e bem definida de etapas que, quando executadas, produzem um resultado. Essa habilidade é fundamental para estudantes em cursos introdutórios de programação [Özmen and Altun 2014], mas é também uma etapa desafiadora para iniciantes, que muitas vezes encontram dificuldades especialmente na compreensão inicial dos enunciados e na identificação dos elementos essenciais para construir soluções corretas.

Tradicionalmente, esses cursos apresentam conceitos de ciência da computação seguidos pela prática da programação em linguagens como Python, C ou Java [Hickey 2004], porém, o foco costuma recair sobre a sintaxe, enquanto a construção de algoritmos como ferramenta para resolver problemas recebe menos atenção. Como consequência, muitos estudantes adotam uma abordagem de tentativa e erro, o que limita o desenvolvimento de habilidades mais profundas de análise e compreensão [Edwards 2004].

DOI: 10.5753/sbie.2025.12058

Nesse cenário, o teste de software tem se destacado como uma estratégia eficaz para o ensino de programação. Edwards [Edwards 2004] defende que o uso de testes em cursos introdutórios favorece uma programação mais cuidadosa e consciente, além de estimular a compreensão do processo de criação de algoritmos. Contudo, como demonstram Scatalon et al. [Scatalon et al. 2020], a inserção do teste de software no ensino da programação pode ocorrer de maneiras variadas. É possível incluí-lo tanto em disciplinas introdutórias quanto em cursos mais avançados; os testes podem ser elaborados logo no início da resolução do problema ou apenas na etapa final, quando o programa já está quase finalizado. Portanto, ao optar por incluir testes no ensino, é necessário considerar essas diferentes possibilidades.

Considerando essas possibilidades, esta pesquisa busca apoiar o estudante iniciante no processo de interpretação das especificações dos problemas, por meio de uma abordagem que promove a interação com uma solução de referência acessível apenas pela interface de entradas e saídas, sem exposição do código-fonte. Essa interação permite que o estudante explore a especificação do problema testando diferentes entradas e observando as saídas. Essa dinâmica incentiva a formulação de hipóteses e a experimentação por meio de testes elaborados pelo próprio estudante, exigindo também que a solução proposta seja validada e aprovada pelos testes ocultos, conjuntos de casos de teste não disponibilizados aos estudantes, utilizados para avaliar a correção e robustez das soluções. Dessa forma, o processo de aprendizagem se aproxima das práticas formais de testes de software no contexto do ensino de programação.

Com o intuito de orientar a condução desta pesquisa, foram definidas questões de investigação específicas, detalhadas a seguir.

- QP1: A utilização da estratégia de apoio contribui para um aumento na taxa de sucesso final dos estudantes na resolução dos exercícios?
- QP2: O uso da estratégia proposta está associado a um melhor desempenho dos estudantes, considerando o percentual de testes aprovados?
- QP3: estudantes que utilizam a estratégia desenvolvem um maior número de testes corretos e cometem menos erros (falhas ou exceções) em comparação com aqueles que não utilizam a estratégia?
- QP4: Em que proporção os testes criados pelos estudantes abrangem os casos de teste definidos previamente pelo professor (tanto públicos quanto ocultos)?

Neste estudo, nosso objetivo é aprofundar a análise da eficácia da abordagem proposta como recurso para melhorar a compreensão dos exercícios propostos e, consequentemente, aumentar as chances de resolução correta. Ao permitir que o estudante experimente diferentes cenários de entrada e observe as respostas geradas pela solução de referência, busca-se fomentar um entendimento mais claro do problema, incentivando uma postura ativa, reflexiva e exploratória, características fundamentais para o desenvolvimento das habilidades exigidas em programação.

2. Trabalhos Relacionados

A fim de contextualizar esta pesquisa e fundamentar sua proposta metodológica, esta seção apresenta uma revisão dos principais trabalhos relacionados ao uso de estratégias de apoio à aprendizagem de programação, com ênfase em abordagens baseadas em testes, sistemas de *feedback* automático e metodologias que buscam promover a compreensão

conceitual de estruturas fundamentais da programação. Diversas estratégias vêm sendo propostas na literatura com o objetivo de favorecer a compreensão de problemas e aprimorar os mecanismos de avaliação automática em cursos de introdução à programação.

Cabo [Cabo 2019] propõe uma abordagem fundamentada no modelo IPO (*Input-Process-Output*), na qual os estudantes devem prever os resultados de entradas sugeridas antes da codificação, permitindo identificar dificuldades cognitivas anteriores à implementação do código. Complementarmente, Wrenn e Krishnamurthi [Wrenn and Krishnamurthi 2019, Wrenn and Krishnamurthi 2021] propõem um mecanismo de *feedback* imediato baseado na definição de entradas e saídas esperadas pelos próprios estudantes. Seus estudos indicam que, embora essa estratégia contribua para o desenvolvimento da compreensão, ela também pode levar a uma aplicação mecânica dos testes, sem reflexão adequada sobre o comportamento dos programas.

Allen-Perez et al. [Allen-Perez et al. 2025] analisam os processos de testagem adotados por estudantes em cursos introdutórios de programação, identificando comportamentos como a testagem tardia e a ausência de casos de borda. Embora reconheçam a importância dos testes, os estudantes demonstram dificuldades em aplicá-los sistematicamente. O estudo sugere que o ensino explícito de práticas de testagem pode aprimorar a qualidade do código e fortalecer habilidades de depuração e análise de requisitos, evidenciando a relevância da testagem no contexto da educação em programação.

Pesquisas recentes ampliaram esse debate ao explorar diferentes dimensões do uso de testes no ensino de programação. Messer et al. [Messer et al. 2024] investigam como estudantes elaboram casos de teste em ambientes de programação assistida por ferramentas de IA, evidenciando tanto ganhos na cobertura dos testes quanto novos desafios relacionados à dependência excessiva de sugestões automáticas. Silva et al. [Silva et al. 2024] analisam o impacto de atividades baseadas em testes em cenários de ensino híbrido, destacando que a formulação de casos de teste pelos próprios estudantes favorece maior engajamento e compreensão conceitual, especialmente em turmas de grande escala. Já Gonçalves et al. [Gonçalves et al. 2025] discutem como a prática de testes pode ser utilizada como métrica de avaliação formativa, apontando que o desempenho dos estudantes em testes ocultos fornece indicadores relevantes sobre lacunas de compreensão que não emergem nos testes visíveis.

Em nossas pesquisas anteriores [Almeida et al. 2023, Almeida et al. 2024], investigamos uma estratégia pedagógica que integra a elaboração de testes ao processo de resolução de problemas. Diferentemente de abordagens que usam enunciados incompletos [Denny et al. 2025], adotamos problemas com enunciados completos, estimulando reflexão crítica, antecipação de cenários complexos e atenção a casos de borda. O presente trabalho incorpora métricas de desempenho, incluindo falhas, exceções e testes ocultos, permitindo uma avaliação mais abrangente do impacto dessa prática na qualidade das soluções produzidas.

3. Metodologia

A metodologia adotada neste estudo foi planejada e conduzida com rigor, visando assegurar a validade dos resultados obtidos. A seguir, são apresentados em detalhe o delineamento experimental, os materiais empregados e os instrumentos utilizados.

3.1. Plano Experimental

O estudo adotou um delineamento experimental controlado, comparando dois grupos: controle e experimental, cujos participantes foram selecionados aleatoriamente. Ambos receberam seis exercícios de programação em formato de miniteste, com duração máxima de duas horas, aplicados apenas após a apresentação formal dos conteúdos sobre estruturas de decisão, estruturas de repetição e listas na linguagem Python.

A principal diferença entre os grupos foi a aplicação de uma estratégia específica no grupo experimental, que permitia aos estudantes interagir com uma solução de referência sem acesso ao código-fonte, realizando quantas interações julgassem necessárias. Nessas interações, os estudantes forneciam entradas para o problema e observavam as saídas correspondentes (ver Figura 1). Por outro lado, o grupo controle seguiu a abordagem tradicional, na qual os estudantes liam o enunciado, implementavam a solução e submetiam o código ao sistema de avaliação. Em ambos os grupos, o número de submissões era limitado por meio de "fichas", um mecanismo que controlava a quantidade de envios e incentivava os estudantes a elaborarem suas soluções de forma mais cuidadosa e reflexiva.

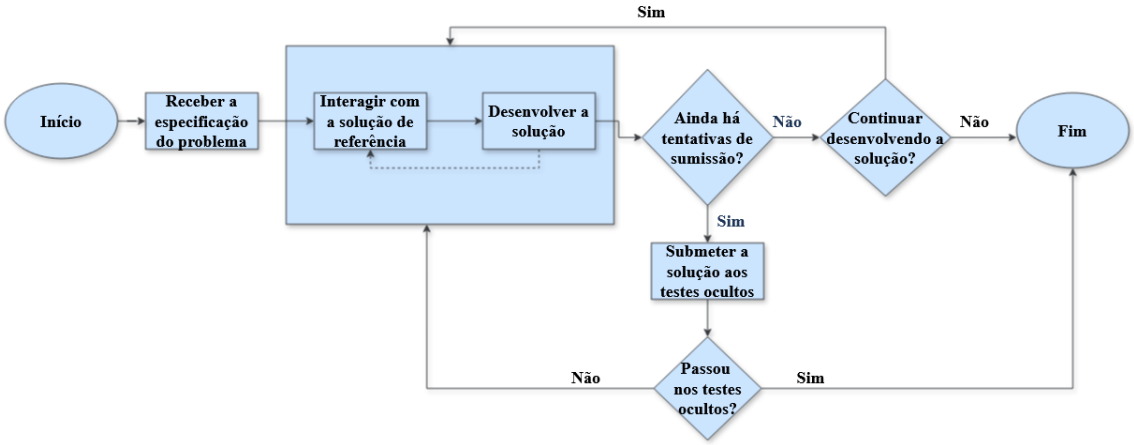

Figura 1. Processo de resolução com a inclusão da estratégia proposta.

Para avaliar as questões de pesquisa propostas neste estudo, foram observadas as seguintes variáveis apresentadas na Tabela 1, sendo as duas últimas inspiradas no trabalho de Horgan et al. [Horgan et al. 1994], que discute métricas relacionadas à cobertura de testes.

3.2. Interação com a Solução de Referência e Resultados sobre a Submissão

A Figura 2 reúne duas etapas do processo de avaliação automática. No lado esquerdo, ilustra a interação do estudante com a solução de referência via terminal: ao digitar o comando *p1-oracle*, correspondente ao exercício em questão, e inserir uma entrada, o sistema exibe a saída esperada, que é registrada como um teste criado pelo estudante para análise posterior. No lado direito, a figura apresenta uma visão geral dos resultados dos testes aplicados às submissões, onde símbolos indicam se cada teste foi bem-sucedido ("."), falhou ("F") ou gerou exceção ("e"). Essa visualização facilita a identificação rápida do desempenho dos estudantes, destacando soluções confiáveis, falhas frequentes e erros críticos.

Métrica	Descrição
Taxa de sucesso final	Proporção de submissões bem-sucedidas em relação ao total de últimas submissões realizadas por cada participante, considerando o desempenho final após o uso (ou não) da estratégia proposta.
Percentual de testes corretos	Proporção de testes ocultos que foram corretamente atendidos por uma submissão, em relação ao total de testes ocultos disponíveis.
Percentual de testes ocultos com falha ou exceção	Proporção de testes ocultos que resultam em falha (saída incorreta) ou que geram exceções durante a execução da solução, em relação ao total de testes ocultos aplicados à submissão.
Cobertura Planejada (CP)	Proporção de categorias de testes planejadas pelo professor que foram efetivamente cobertas pelos testes propostos pelo estudante. Definida como $CP = \frac{\|C \cap CA\|}{\|C\|}$, onde C é o conjunto de categorias planejadas e CA é o conjunto de categorias cobertas pelos testes do estudante.
Cobertura Estendida (CE)	Abrangência total das categorias contempladas pelos testes elaborados pelo estudante, considerando tanto as categorias planejadas pelo professor quanto possíveis categorias adicionais introduzidas pelo estudante. Definida como $CE = \frac{\|CA\|}{\|C \cup CA\|}$.

Tabela 1. Descrição das métricas utilizadas na análise das submissões.

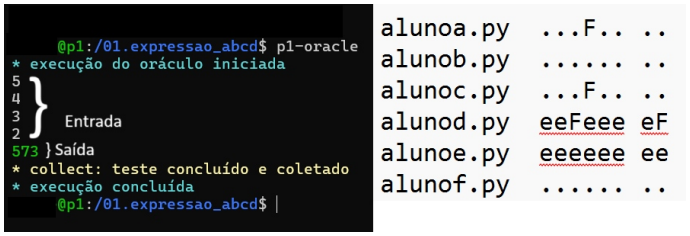

Figura 2. Exemplo de interação com a solução de referência (à esquerda) e dos resultados dos testes aplicados às submissões dos estudantes (à direita).

3.3. Exercícios Utilizados

Seis exercícios de programação foram aplicados, com o objetivo de avaliar a compreensão e a aplicação de conceitos fundamentais da linguagem Python, incluindo estruturas condicionais, laços de repetição e manipulação de listas e strings. Um resumo dos exercícios e dos conceitos abordados está disponível na Tabela 2, enquanto a classificação detalhada dos testes ocultos pode ser consultada na pasta do *Google Drive*[1].

4. Resultados e Discussões

Participaram efetivamente 22 estudantes, 50% destes compondo o grupo de controle e 50% participando do grupo experimental, distribuídos aleatoriamente. Ambos os grupos tiveram o mesmo tempo para resolução dos seis exercícios propostos, com o acesso ao ferramental (interação com a solução de referência) incluído no processo de resolução apenas no grupo experimental. A pesquisa foi conduzida em conformidade com as normas éticas aplicáveis à investigação com seres humanos e registrada no Comitê de Ética em Pesquisa da instituição, sob o CAAE 89199725.6.0000.5182. Todos os participantes foram devidamente informados sobre os objetivos do estudo, a natureza voluntária de sua

[1]https://drive.google.com/drive/folders/1fRTXlToAHHmpZmigUoJPhBU9I8jJ6juK?usp=sharing

Exercício	Habilidades e Conceitos Envolvidos
1	Identificar letras coincidentes na mesma posição de duas palavras e produzir saída combinando caracteres e posições.
2	Implementar divisão inteira via subtrações sucessivas, controlando quociente, resto e sequência de operações.
3	Verificar se uma sequência de palavras está em ordem alfabética e indicar a primeira palavra fora de ordem.
4	Encontrar a maior palavra em uma sequência de frases e indicar sua posição.
5	Identificar elementos que aparecem exatamente três vezes em uma sequência de números.
6	Avaliar elementos simétricos de uma lista e aplicar regras específicas para gerar uma nova lista.

Tabela 2. Breve descrição dos exercícios.

participação e as medidas adotadas para garantir a confidencialidade e anonimato de seus dados.

Para a análise dos dados coletados neste estudo, utilizamos a linguagem de programação Python, que possibilitou o tratamento e a organização das informações de forma eficiente. Para investigar a significância estatística dos resultados, aplicamos testes de hipótese apropriados ao tipo de dado e à amostra disponível, como o teste exato de Fisher para proporções e o teste U de Mann-Whitney para comparações entre grupos, considerando o tamanho reduzido da amostra. Além disso, empregamos técnicas de análise descritiva para sumarizar os dados e facilitar a interpretação dos padrões observados. Essas ferramentas permitiram uma avaliação rigorosa e confiável do impacto da estratégia pedagógica proposta.

4.1. QP1 - Taxa de Sucesso Final

Para responder a esta questão de pesquisa, analisamos comparativamente o desempenho dos grupos controle e experimental com base na taxa de sucesso geral e também em cada exercício.

A análise da Tabela 3 evidencia uma diferença substancial no desempenho entre os grupos controle e experimental. Enquanto o grupo controle apresentou 46,67% de submissões corretas, o grupo experimental alcançou 82,76%, representando uma melhoria superior a 30 pontos percentuais. Esse resultado sugere que a intervenção aplicada ao grupo experimental teve um efeito positivo sobre a qualidade das submissões, reduzindo de forma significativa a proporção de submissões com falha, de 53,33% para 17,24%. A efetividade da estratégia adotada é reforçada pelos resultados do teste de hipótese, cujo p-valor de 0,0087 indica que a diferença observada entre os grupos é estatisticamente significativa.

Grupo	# Corretas (Passaram)	# Incorretas (Falharam)
Controle	46,67%	53,33%
Experimental	82,76%	17,24%

Tabela 3. Distribuição percentual de submissões corretas e incorretas por grupo.

A Figura 3 evidencia que o grupo experimental apresentou desempenho superior ao controle em quatro dos seis exercícios. Nos exercícios 1 e 2, o grupo experimental obteve mais acertos e menos falhas, sugerindo que a estratégia de apoio auxiliou os estudantes na organização da lógica, no controle de repetições e no gerenciamento de múltiplos valores simultaneamente. Nos exercícios 3 e 4, o grupo experimental também se destacou, indicando que a intervenção favoreceu a antecipação de possíveis cenários de teste e a validação da lógica antes da submissão, especialmente em problemas que exigem maior atenção à estrutura do código e às condições de borda. Nos exercícios 5 e 6, as diferenças foram menores, sugerindo que a complexidade conceitual reduzida desses problemas diminuiu o impacto da estratégia. Esses resultados indicam que a eficácia da intervenção é mais pronunciada em exercícios que demandam raciocínio estruturado e manipulação cuidadosa de variáveis, enquanto em tarefas mais diretas a diferença de desempenho entre os grupos é menos expressiva.

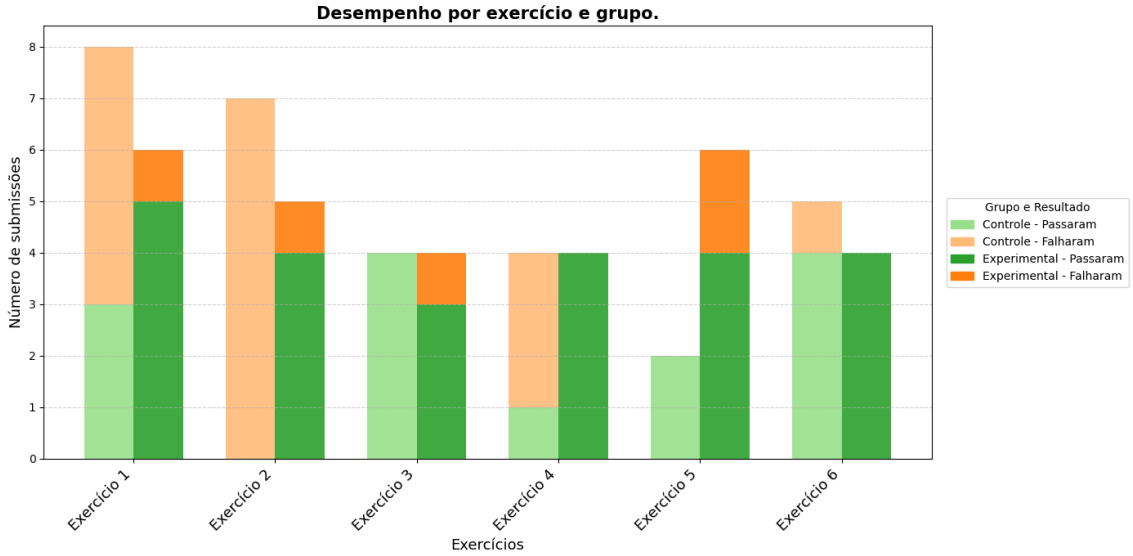

Figura 3. Distribuição do desempenho por exercício e grupo.

Na análise apresentada, considerou-se apenas a última submissão de cada participante em cada exercício. Consequentemente, os valores exibidos na Figura 3 refletem apenas os estudantes que efetivamente realizaram uma submissão. Caso algum participante não tenha enviado solução para determinado exercício, ele não foi contabilizado na frequência apresentada. Dessa forma, a contagem pode ser inferior ao total de 11 estudantes por grupo, não por inconsistência nos dados, mas em função da ausência de submissão de alguns estudantes. Esse procedimento permite avaliar de forma consistente o desempenho final de cada participante, evitando distorções causadas por múltiplas tentativas.

Em síntese, os resultados obtidos em relação à QP1 reforçam que a estratégia de apoio não apenas aumentou a taxa global de sucesso, mas também promoveu um desempenho mais consistente entre os exercícios, reduzindo a ocorrência de falhas e ampliando as oportunidades de acerto. Esse efeito sugere que a intervenção não se limitou a melhorar casos pontuais, mas exerceu influência abrangente sobre o processo de resolução dos estudantes, indicando ganhos efetivos em termos de compreensão conceitual e aplicação prática dos conteúdos de programação.

4.2. QP2 - Percentual de Testes Corretos

Para investigar se a estratégia proposta contribui para a melhora no desempenho dos estudantes, aqui examinamos a porcentagem de testes ocultos corretos como métrica de avaliação. Essa medida reflete a capacidade das soluções submetidas pelos estudantes de generalizarem corretamente, ou seja, de obterem sucesso em testes que não estavam visíveis durante o desenvolvimento. Ao considerar apenas as submissões finais de cada estudante por exercício, buscamos identificar diferenças no desempenho entre os grupos controle e experimental.

Conforme ilustrado na Figura 4, observa-se que a mediana da proporção de testes ocultos aceitos no grupo experimental (0,67) é ligeiramente superior à do grupo controle (0,55), sugerindo um desempenho central marginalmente mais elevado. No entanto, os resultados do grupo controle apresentam uma concentração maior entre aproximadamente 50% e 80%, o que indica uma distribuição mais consistente entre os participantes. Por outro lado, os dados do grupo experimental exibem maior dispersão, abrangendo uma faixa mais ampla de desempenho. Essa variabilidade sugere que, embora parte dos estudantes do grupo experimental não tenha obtido bons resultados, há evidências de que alguns indivíduos se beneficiaram da estratégia adotada, alcançando desempenhos superiores aos do grupo controle.

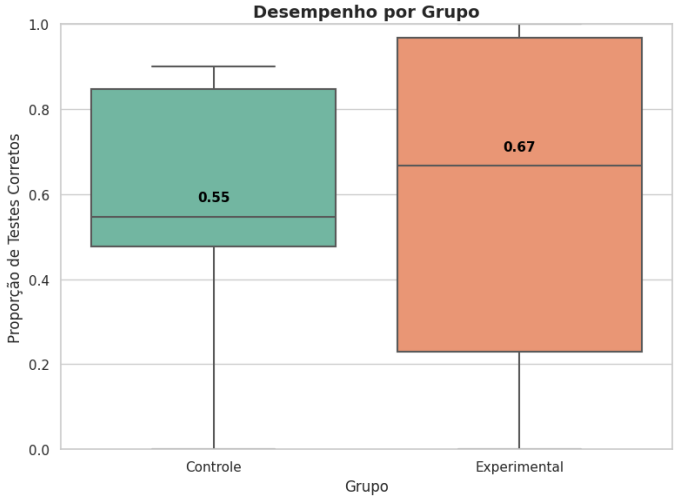

Figura 4. Distribuição da proporção de testes corretos por grupo.

Para investigar se essa diferença observada é estatisticamente significativa, foi aplicado o teste de Mann-Whitney U, resultando em um valor de p igual a 0,2550. Assim, em relação à QP2, conclui-se que não há evidências suficientes para afirmar que a estratégia contribuiu de forma consistente para a melhora no desempenho dos estudantes quando analisado o percentual de testes ocultos aprovados.

4.3. QP3 - Percentual de Testes Ocultos com Falha ou Exceção

Analisamos aqui o impacto da intervenção ao comparar o grupo experimental, que utilizou a estratégia de apoio, com o grupo controle. O objetivo é verificar se tal intervenção promoveu um aumento estatisticamente significativo na proporção de testes corretos, bem como uma redução na incidência de testes com falhas ou exceções.

A Figura 5 evidencia que o grupo experimental apresentou desempenho ligeiramente superior ao do grupo controle, com 71,7% dos testes ocultos executados nas soluções sendo aceitos, em comparação a 63,4% no grupo controle. Observa-se, ainda, uma redução superior a 50% na proporção de testes com falhas no grupo experimental, o que sugere uma melhora substancial na qualidade das soluções desenvolvidas. Por outro lado, no que diz respeito aos testes que resultaram em exceções, os dois grupos apresentaram resultados semelhantes, sem diferenças expressivas.

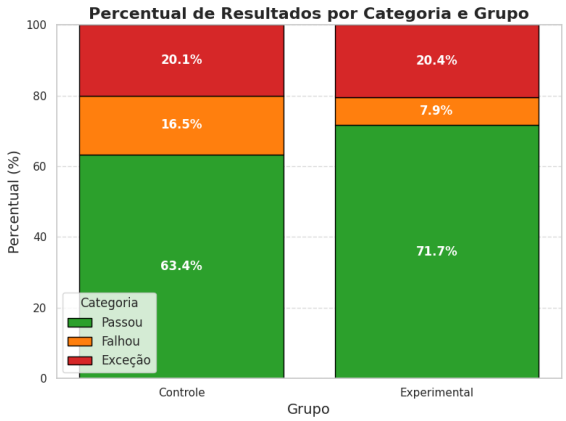

Figura 5. Distribuição percentual de testes ocultos com falha ou exceção por grupo experimental e controle.

Esses achados são corroborados pelos testes de hipótese aplicados: para a categoria *passou*, foi identificada uma diferença estatisticamente significativa entre os grupos ($p = 0{,}0222$); o mesmo se verificou para a categoria *falhou*, com uma diferença ainda mais pronunciada ($p = 0{,}0008$). Por outro lado, para a categoria *exceção*, não houve diferença estatística significativa entre os grupos ($p = 0{,}9274$).

Ainda foi possível realizar a análise individual de cada exercício para identificar em quais problemas a intervenção teve maior efeito. A Figura 6 apresenta esses resultados detalhados. A análise estatística por exercício mostrou que, na maioria dos casos, não houve evidências suficientes para afirmar que a estratégia adotada impactou significativamente a quantidade de testes corretos, falhos ou com exceção.

Contudo, dois exercícios se destacaram com resultados estatisticamente significativos: o Exercício 2 e o Exercício 6. No Exercício 2, que envolve a divisão por subtrações sucessivas, os estudantes precisam gerenciar simultaneamente múltiplos valores e garantir que a sequência de subtrações seja corretamente registrada e exibida. A estratégia de apoio parece ter auxiliado os estudantes a organizar a lógica do *loop* e atualizar corretamente as variáveis, refletindo no maior número de testes corretos e menor incidência de falhas.

No Exercício 6, que exige análise de elementos equidistantes das extremidades de uma sequência e aplicação de regras condicionais específicas, como divisibilidade por 3 ou 5, os estudantes precisaram manter atenção ao raciocínio lógico e à validação de múltiplas condições simultaneamente. Nesse caso, o uso da estratégia permitiu que antecipassem cenários complexos e estruturassem a lógica do programa de forma mais precisa. Nos demais exercícios, os p-valores elevados, todos acima de 0,35, indicam ausência de

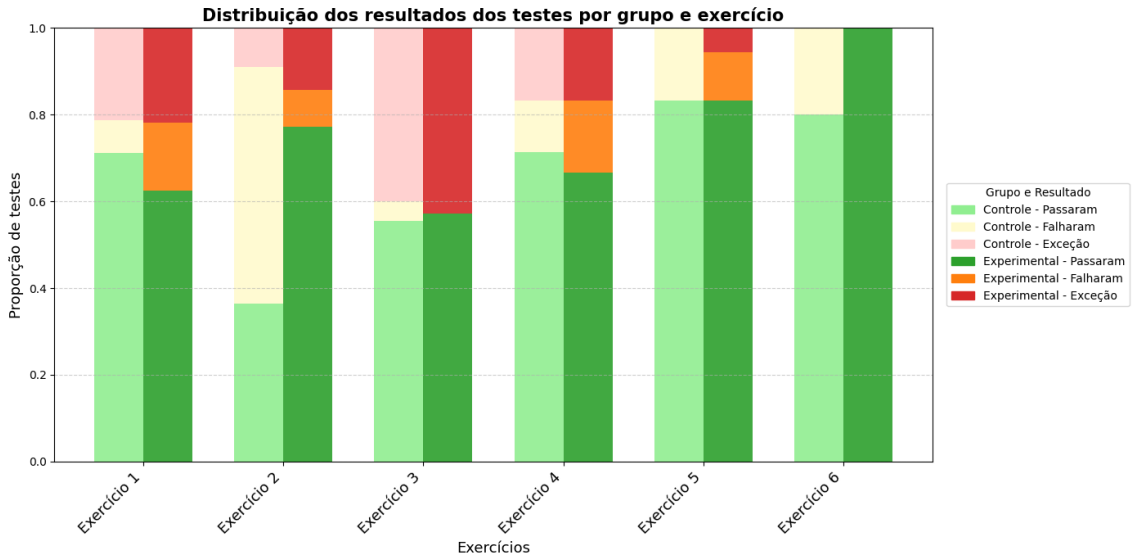

Figura 6. Distribuição do resultado dos testes ocultos por grupo e por exercício.

diferença estatística significativa entre os grupos. Esses achados reforçam que a estratégia de apoio apresenta maior potencial em exercícios que exigem manipulação cuidadosa de variáveis, controle de fluxo iterativo e planejamento antecipado da lógica de solução.

Em resumo, em relação à QP3, conclui-se que a estratégia contribuiu para que os estudantes desenvolvessem um maior número de testes corretos e reduzissem falhas, ainda que seus efeitos não tenham se manifestado uniformemente em todos os exercícios analisados.

4.4. QP4 - Cobertura Planejada e Cobertura Estendida

Aqui, buscamos compreender em que medida os testes propostos pelos estudantes, tanto os inicialmente planejados quanto os estendidos ao longo da atividade, cobrem os casos de teste definidos pelo professor, incluindo tanto os públicos quanto os ocultos. Essa análise, com os dados do grupo experimental, permite avaliar a capacidade dos estudantes de antecipar cenários relevantes de execução e de criar testes que representem adequadamente a diversidade de situações esperadas. Ao investigar a sobreposição entre os testes dos estudantes e os testes de referência, buscamos evidenciar o grau de alinhamento entre as estratégias de validação dos estudantes e os objetivos pedagógicos traçados.

Para avaliar a cobertura dos testes propostos pelos estudantes em relação aos cenários definidos pelo professor, foi adotado um processo em múltiplas etapas. Inicialmente, identificaram-se os cenários representados nos testes ocultos. Em seguida, os testes dos estudantes foram comparados individualmente com esses cenários, permitindo calcular a frequência de cobertura em níveis individual e coletivo. Testes que não correspondiam a nenhum cenário pré-definido foram classificados como cenários extras, indicando exploração adicional por parte dos estudantes. A análise foi quantificada por meio das métricas Cobertura Planejada (CP) e Cobertura Estendida (CE), cujos resultados, referentes apenas ao grupo experimental, estão apresentados na Tabela 4.

Observa-se que, em dois dos cinco exercícios com dados disponíveis, houve um aumento modesto na cobertura quando considerados os testes adicionais propostos pelos

Exercício	CP (Cobertura Planejada)	CE (Cobertura Estendida)
Exercício 1	0,57	0,57
Exercício 2	0,67	0,75
Exercício 3	0,83	0,83
Exercício 4	0,75	0,8
Exercício 5	1,00	1,00
Exercício 6	-	-

Tabela 4. Comparação entre a cobertura planejada (CP) e estendida (CE) para cada exercício (grupo experimental).

estudantes, como nos casos do Exercício 2 (CP = 0,67; CE = 0,75) e do Exercício 4 (CP = 0,75; CE = 0,80). No Exercício 5, a cobertura foi total desde os testes planejados (CP = 1,00; CE = 1,00), não havendo incremento, embora a abrangência tenha sido mantida. Já nos Exercícios 1 e 3, os valores de CP e CE permaneceram iguais, indicando que os testes criados pelos estudantes não contribuíram para a inclusão de novos cenários além dos já previstos. O Exercício 6 foi excluído da análise por não ter recebido nenhum teste proposto. Esses resultados sugerem que, embora a extensão dos testes nem sempre leve à ampliação da cobertura, ela pode, em alguns casos, contribuir para a exploração de cenários adicionais relevantes, reforçando o potencial da estratégia para fomentar a diversidade na formulação de testes.

A análise dos testes propostos pelos estudantes indica uma forte concentração em casos triviais ou muito semelhantes entre si, muitas vezes reproduzindo diretamente os testes públicos fornecidos ou apresentando variações mínimas desses exemplos. Em alguns exercícios, observou-se que a maior parte dos testes se enquadrava em uma única categoria, sugerindo uma exploração bastante limitada do espaço de entrada. Casos extremos como entradas vazias, repetições excessivas ou valores nos limites do domínio esperado foram raramente representados, evidenciando dificuldades dos estudantes em formular cenários que desafiem de forma mais ampla e realista o comportamento das soluções.

Quanto à cobertura dos testes ocultos, os resultados revelam que ela foi geralmente baixa. Foram poucos os casos em que os estudantes conseguiram antecipar cenários mais complexos definidos pelo professor, e alguns testes ocultos jamais foram cobertos por qualquer estudante ao longo da atividade. Isso aponta lacunas importantes no raciocínio exploratório e sugere que, sem estímulos ou direcionamentos adicionais, os estudantes tendem a restringir seus testes a padrões visíveis e familiares, deixando de contemplar aspectos críticos e menos óbvios do problema proposto.

Em síntese, os resultados referentes à QP4 indicam que, embora a estratégia tenha favorecido certa ampliação na cobertura dos cenários de teste em casos específicos, os estudantes ainda demonstraram limitações na capacidade de antecipar situações mais complexas e desafiadoras.

5. Conclusões e Trabalhos Futuros

O ensino de programação representa um dos pilares fundamentais na formação de profissionais em áreas correlatas à ciência da computação e engenharia. Entretanto, a comple-

xidade intrínseca à construção do pensamento algorítmico torna essa disciplina particularmente desafiadora para estudantes iniciantes, que frequentemente apresentam dificuldades não apenas na sintaxe das linguagens, mas sobretudo na compreensão profunda dos problemas a serem resolvidos. Diante desse contexto, estratégias que incorporem práticas de teste de software como elemento formativo surgem como promissoras, uma vez que estimulam a reflexão crítica, a experimentação orientada e o aprimoramento contínuo das soluções propostas pelos estudantes.

Os resultados deste estudo indicam que a estratégia baseada na formulação e uso de testes teve impacto positivo no desempenho dos estudantes, com o grupo experimental apresentando menos falhas, mais testes ocultos aceitos e maior taxa de sucesso final. No entanto, a análise da cobertura dos testes revelou limitações no raciocínio exploratório, com poucos casos-limite e baixa diversidade de entradas. Isso aponta para a necessidade de estratégias pedagógicas complementares que estimulem a criação de testes mais variados e a ampliação da compreensão dos estudantes sobre o espaço de entradas.

Para trabalhos futuros, propomos o desenvolvimento e a investigação de estratégias complementares que estimulem a ampliação do raciocínio exploratório dos estudantes durante a formulação de testes, buscando incentivar a criação de casos mais diversificados e desafiadores, incluindo cenários-limite e situações inesperadas. Ademais, é relevante avaliar intervenções que promovam o desenvolvimento do pensamento crítico e da capacidade analítica no contexto da construção de testes, integrando abordagens que favoreçam a exploração sistemática do espaço de entradas.

6. Ameaças à Validade

Além do tamanho reduzido da amostra, outras ameaças à validade devem ser consideradas. Os participantes foram alocados aleatoriamente, o que contribui para reduzir possíveis diferenças pré-existentes entre os grupos experimental e controle, como variações em conhecimento prévio, habilidades de programação ou níveis de motivação. Apesar de não termos coletado informações demográficas detalhadas, a aleatoriedade na alocação tende a balancear os grupos, sugerindo que diferenças de desempenho observadas possam ser atribuídas principalmente à intervenção aplicada.

Adicionalmente, a intervenção foi conduzida em um contexto específico, envolvendo exercícios, linguagem de programação e metodologia particulares. Esse aspecto pode limitar a generalização dos resultados para outros cursos, instituições ou níveis de ensino, especialmente quando as características do currículo, do ambiente de aprendizagem ou do perfil dos estudantes diferirem significativamente das condições do estudo.

Referências

Allen-Perez, G., Millan, L., Nghiem, B., Wu, K., Shah, A., and Soosai Raj, A. G. (2025). An analysis of students' testing processes in cs1. In *Proceedings of the 56th ACM Technical Symposium on Computer Science Education V. 1*, pages 46–52.

Almeida, A., Andrade, W., and Guerrero, D. (2024). Ampliando a compreensão de problemas em programação: Avaliação de uma estratégia educacional baseada em teste de software. In *Anais do XXXV Simpósio Brasileiro de Informática na Educação*, pages 276–288, Porto Alegre, RS, Brasil. SBC.

Almeida, A., Araújo, E., and Figueiredo, J. (2023). Investigando o uso de testes para apoiar a resolução de problemas de programação. In *Anais do XXXIV Simpósio Brasileiro de Informática na Educação*, pages 357–367, Porto Alegre, RS, Brasil. SBC.

Cabo, C. (2019). Fostering problem understanding as a precursor to problem-solving in computer programming. In *2019 IEEE Frontiers in Education Conference (FIE)*, pages 1–9. IEEE.

Denny, P., Kumar, V., MacNeil, S., Prather, J., and Leinonen, J. (2025). Probing the unknown: Exploring student interactions with probeable problems at scale in introductory programming. *arXiv preprint arXiv:2504.11723*.

Edwards, S. H. (2004). Using software testing to move students from trial-and-error to reflection-in-action. In *Proceedings of the 35th SIGCSE technical symposium on Computer science education*, pages 26–30.

Gonçalves, S. C. L., Moreira, R., Backes, A. R., Rodrigues Moreira, L. F., and Martinhago, A. Z. (2025). Programming in brazilian higher education and high school: A systematic literature review. In *Latin American Conference on Learning Technologies*, pages 287–303. Springer.

Hickey, T. J. (2004). Scheme-based web programming as a basis for a cs0 curriculum. In *Proceedings of the 35th SIGCSE technical symposium on Computer science education*, pages 353–357.

Horgan, J., London, S., and Lyu, M. (1994). Achieving software quality with testing coverage measures. *Computer*, 27(9):60–69.

Messer, M., Brown, N. C., Kölling, M., and Shi, M. (2024). Automated grading and feedback tools for programming education: A systematic review. *ACM Transactions on Computing Education*, 24(1):1–43.

Özmen, B. and Altun, A. (2014). Undergraduate students' experiences in programming: difficulties and obstacles. *Turkish Online Journal of Qualitative Inquiry*, 5(3):1–27.

Scatalon, L. P., Garcia, R. E., and Barbosa, E. F. (2020). Teaching practices of software testing in programming education. In *2020 IEEE Frontiers in Education Conference (FIE)*, pages 1–9. Ieee.

Silva, L., Fortes, R., and Ribeiro, R. G. (2024). Geração automática de casos de teste para auto-graders baseada em execução simbólica. In *Anais do XXXV Simpósio Brasileiro de Informática na Educação*, pages 2325–2338, Porto Alegre, RS, Brasil. SBC.

Wrenn, J. and Krishnamurthi, S. (2019). Executable examples for programming problem comprehension. In *Proceedings of the 2019 ACM Conference on International Computing Education Research*, ICER '19, page 131–139, New York, NY, USA. Association for Computing Machinery.

Wrenn, J. and Krishnamurthi, S. (2021). Reading between the lines: Student help-seeking for (un) specified behaviors. In *Proceedings of the 21st Koli Calling International Conference on Computing Education Research*, pages 1–6.