

Analizando Fluxogramas Gerados por LLMs para Problemas Introdutórios de Programação

Maria H. L. S. Acioli¹, Patricia C. de A. R. Tedesco¹, Evandro de B. Costa²

¹Centro de Informática – Universidade Federal de Pernambuco (UFPE)
Recife, PE – Brasil

²Instituto de Computação – Universidade Federal de Alagoas (UFAL)
Maceió, AL – Brasil

{mhlsa,pcart}@cin.ufpe.br, ebc.academico@gmail.com

Abstract. Flowcharts continue to be used in introductory programming education, supporting the visual construction of algorithms and the understanding of control flow. With the popularization of Large Language Models (LLMs) among students, it becomes relevant to investigate their ability to generate flowcharts. This study evaluates the ChatGPT 5.5 and Gemini 3.5 models on ten problems covering the main introductory programming topics. The results show that, even in basic tasks, the models still present limitations in producing correct flowcharts, reinforcing the potential of this activity as a tool for learning programming with less interference from generative AI.

Resumo. Fluxogramas continuam sendo utilizados na aprendizagem de programação introdutória, auxiliando na construção visual de algoritmos e na compreensão do controle de fluxo. Com a popularização dos Grandes Modelos de Linguagem (LLMs) entre estudantes, torna-se relevante investigar suas capacidades de gerar fluxogramas. Este estudo avalia os modelos ChatGPT 5.5 e Gemini 3.5 em dez problemas cobrindo os principais conteúdos de programação introdutória. Os resultados mostram que, mesmo em tarefas básicas, os modelos ainda apresentam limitações na produção de fluxogramas corretos, reforçando o potencial dessa atividade como ferramenta de aprendizagem de programação com menor interferência da IA generativa.

1. Introdução

A popularização dos Grandes Modelos de Linguagem (do inglês, *Large Language Models*, LLMs) tem provocado transformações significativas no ensino e na aprendizagem de programação. Pesquisas recentes demonstram que modelos como o ChatGPT e o Gemini são capazes de gerar código funcional para uma ampla variedade de problemas algorítmicos, respondendo com rapidez e precisão a partir de enunciados em linguagem natural [Dingle and Krulis 2024, Muepu et al. 2025, Ouh et al. 2023, Shuvo et al. 2025]. Se por um lado esse recurso pode apoiar a aprendizagem quando utilizado de forma reflexiva, por outro representa um desafio crescente para o *design* de atividades educacionais que promovam o desenvolvimento genuíno do raciocínio lógico e do pensamento computacional [Hong and Hong 2024].

O acesso imediato a soluções geradas por LLMs pode levar estudantes a contornar etapas essenciais do processo de resolução de problemas. Em vez de analisar o problema, decompô-lo e construir uma solução progressivamente, muitos passam a depender das respostas geradas, reduzindo a consulta a materiais de estudo [Xue et al. 2024] e adotando soluções prontas sem compreender os mecanismos subjacentes [Rahe

and Maalej 2025]. Tal comportamento compromete a formação do raciocínio algorítmico, que é um dos objetivos centrais do ensino introdutório de programação.

Considerando esse desafio, torna-se necessário adaptar as atividades de prática e avaliação para torná-las menos suscetíveis à resolução automática por ferramentas de inteligência artificial generativa [Hong and Hong 2024]. Uma maneira de fazer isso é não restringir o ensino apenas à escrita de código em linguagem de programação, abrindo espaço para a adoção de outras estratégias, como o emprego de fluxogramas.

O termo fluxograma refere-se aos *program flowcharts* definidos pela ISO 5807:1985, que estabelece símbolos padronizados para representar a sequência de operações envolvidas na resolução de um problema de programação [ISO 1985]. Trata-se de uma ferramenta que pode auxiliar o estudante na descrição de algoritmos, permitindo visualizar soluções em perspectivas diferentes.

Diante desse cenário, este trabalho investiga como os LLMs lidam com a tarefa de gerar fluxogramas para problemas introdutórios de programação. A capacidade dos modelos de gerar código tem sido amplamente estudada, mas sua habilidade de produzir representações algorítmicas visuais padronizadas permanece pouco explorada na literatura. Assim, surgem as seguintes questões de pesquisa:

- QP1: Em que medida os fluxogramas gerados por LLMs atendem aos padrões de representação e correspondem corretamente à solução do problema?
- QP2: Quais conteúdos introdutórios de programação apresentam melhores e piores resultados na geração de fluxogramas por LLMs?
- QP3: Quando os fluxogramas gerados por LLMs apresentam soluções incorretas, como eles comumente falham?
- QP4: Como diferentes LLMs e estruturas de *prompting* influenciam a qualidade dos fluxogramas gerados para problemas introdutórios de programação?

A fim de respondê-las, este estudo utiliza os LLMs OpenAI ChatGPT 5.5 e Google Gemini 3.5 para gerar fluxogramas de problemas de introdução à programação. Os modelos são utilizados sob diferentes estratégias de *prompting*, a saber, *zero-shot* e *few-shot prompting*. Na estratégia *zero-shot prompting*, o modelo recebe apenas a descrição da tarefa, enquanto no *few-shot prompting* são fornecidos exemplos antes da solicitação. Os fluxogramas gerados são então avaliados quanto à conformidade com os padrões de representação e à correção da solução produzida.

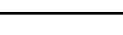
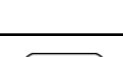
As contribuições esperadas deste trabalho são: (i) uma avaliação sistemática das capacidades e limitações de LLMs na geração de fluxogramas padronizados para programação introdutória; (ii) a identificação de falhas recorrentes, organizadas por tipo de conteúdo; e (iii) subsídios empíricos para o planejamento pedagógico de disciplinas de programação, em particular para o *design* de atividades que promovam o raciocínio algorítmico com menor risco de substituição por IA generativa.

Este artigo está organizado da seguinte forma: a Seção 2 apresenta os conceitos básicos de fluxogramas e a Seção 3 discute os trabalhos relacionados. A metodologia adotada é descrita na Seção 4. A Seção 5 apresenta os resultados obtidos, enquanto a Seção 6 traz respostas às perguntas de pesquisa. Por fim, a Seção 7 conclui o artigo.

2. Fluxogramas

Fluxogramas representam a sequência de operações de um programa [ISO 1985], sendo ferramentas muito úteis no ensino introdutório, visto que auxiliam na compreensão das soluções. Constituem um recurso diferente do código em linguagem de programação, permitindo o desenvolvimento do pensamento computacional, da habilidade de resolver problemas e de depuração dos algoritmos, a partir de uma outra perspectiva. Para compor um fluxograma que representa um algoritmo de forma padronizada é necessário utilizar os símbolos definidos pela norma ISO 5807 [ISO 1985]. O Quadro 1 apresenta os símbolos considerados neste trabalho e seus respectivos significados.

Quadro 1. Principais símbolos dos fluxogramas (adaptado de ISO, 1985)

Símbolo	Nome	Utilidade
	Dados	Representar dados, tanto entrada como saída, qualquer que seja a mídia utilizada.
	Processo	Representar qualquer tipo de processamento, executando uma operação definida ou grupo de operações.
	Decisão	Representa uma decisão ou desvio no fluxo. Possui uma entrada e múltiplas saídas, das quais apenas uma é selecionada com base na condição descrita no símbolo. Cada saída deve ser identificada junto à linha correspondente.
	Linha	Representar o fluxo dos dados ou controles. Utiliza pontas de seta na extremidade para indicar a direção do fluxo onde necessário.
	Terminador	Representar início ou final de programa.

No contexto de programação, espera-se que os fluxogramas possuam início e fim definidos, que seus elementos estejam conectados por linhas de fluxo e que a sequência de execução do algoritmo seja claramente representada [Manzano 2004]. Embora a norma não especifique a linguagem a ser utilizada, recomenda-se o uso consistente dos símbolos e de identificações significativas ao longo do fluxograma [ISO 1985].

3. Trabalhos Relacionados

No contexto da relação entre fluxogramas e LLMs, destaca-se o estudo de He et al. (2025), que propôs o Flow2Code, uma base de avaliação composta por fluxogramas com o objetivo de medir a capacidade de LLMs em convertê-los para código-fonte. Os autores mostraram que os modelos enfrentam desafios na compreensão de representações gráficas de algoritmos.

Voltado para a educação, Darda and Jain (2024) desenvolveram uma ferramenta capaz de converter fluxogramas em código utilizando LLMs. O objetivo era auxiliar estudantes iniciantes que compreendem a lógica dos algoritmos, mas ainda apresentam dificuldades relacionadas à sintaxe das linguagens.

Apesar de possuírem objetivos distintos, os trabalhos de He et al. (2025) e Darda and Jain (2024) partem de fluxogramas para gerar código. O presente estudo compartilha o interesse pela avaliação de LLMs e pelo contexto educacional, mas investiga uma tarefa diferente, partindo de enunciados de programação em linguagem natural e tendo o fluxograma como artefato gerado.

Um trabalho que utiliza descrições em linguagem natural como entrada é o Prompt2Diagram, apresentado por Minu et al. (2025). Os autores desenvolveram um sistema baseado em LLMs capaz de gerar automaticamente diferentes tipos de diagramas, incluindo fluxogramas. Entretanto, o foco da proposta está em apoiar atividades de documentação.

Outro trabalho relacionado é o CodeFlow, apresentado por Zheng e Shi (2026). O sistema converte código produzido por estudantes em fluxogramas para fornecer *feedback* visual durante o processo de depuração. Em seus resultados preliminares, os autores destacam o potencial da abordagem e a necessidade de avaliações mais rigorosas para validar a qualidade dos diagramas gerados e seu impacto na aprendizagem.

Diante disso, não foram identificados trabalhos que investiguem a capacidade de diferentes LLMs em gerar fluxogramas de algoritmos a partir de enunciados de problemas de programação. Assim, o presente estudo busca contribuir para o preenchimento dessa lacuna na literatura.

4. Metodologia

4.1. Definição dos conteúdos e problemas

A metodologia inicia-se com a definição dos conteúdos de programação introdutória e a seleção dos problemas utilizados neste estudo. Para definir os conteúdos, dois aspectos principais foram considerados: as estruturas de algoritmos que podem ser representados por meio dos símbolos de fluxogramas adotados neste estudo e as competências relacionadas ao ensino de programação, previstas na Base Nacional Comum Curricular (BNCC) de Computação [SBC 2022].

Com base nisso, foram selecionados os seguintes conteúdos de programação introdutória: instruções sequenciais; estruturas condicionais; estruturas de repetição; manipulação de listas; funções e recursão. A partir daí, foram propostos dois problemas de programação para cada conteúdo. Os enunciados dos problemas foram elaborados pelos autores, com base em exemplos da BNCC de Computação [SBC 2022] e inspirados em exercícios comumente utilizados em sala de aula de nível introdutório. O Quadro 2 apresenta uma síntese dos problemas selecionados para o estudo.

Quadro 2. Problemas de programação

Conteúdo	ID	Descrição resumida
Instruções sequenciais	P01	Ler o ano de nascimento e o ano atual e informar a idade da pessoa.
	P02	Receber a altura e a base de um retângulo e informar sua área e perímetro.
Estruturas condicionais	P03	Solicitar a cor do semáforo e informar a ação correta: aguardar ou atravessar a rua.
	P04	Solicitar as quatro notas, calcular a média final e informar a situação do aluno.
Estruturas de repetição	P05	Exibir todos os números pares no intervalo de 0 a 1000.
	P06	Solicitar o número de andares e de janelas, exibir a sequência de lavagem das janelas, lavando todas as janelas de cada andar antes de passar para o próximo.
Manipulação de listas	P07	Receber uma lista de 10 números inteiros e um valor alvo e buscar a posição da primeira ocorrência do valor.
	P08	Receber uma lista de 10 números inteiros, aplicar o algoritmo <i>Bubble Sort</i> e exibir a lista ordenada.

Funções e recursão	P09	Receber um número inteiro não negativo, utilizar uma função iterativa para calcular seu fatorial e exibir o resultado.
	P10	Receber um número inteiro não negativo, utilizar uma função recursiva para calcular seu fatorial e exibir o resultado.

4.2. Abordagens de *prompting*

Após a definição dos problemas de programação, os LLMs OpenAI ChatGPT 5.5 e Google Gemini 3.5 foram utilizados para geração dos fluxogramas. Esses modelos foram adotados tendo em vista sua ampla disponibilidade ao público e a possibilidade de gerar fluxogramas diretamente em formato de imagem. A fim de investigar como a estrutura do *prompt* influencia a geração dos fluxogramas, foram utilizadas as abordagens *zero-shot prompting* e *few-shot prompting*.

Os experimentos foram realizados em maio de 2026 e compreenderam uma única execução de cada problema em cada modelo e abordagem de *prompting*, resultando em um total de 40 fluxogramas avaliados. Todas as gerações foram realizadas utilizando as configurações padrão das plataformas. Cada geração foi executada em uma nova conversa, com o objetivo de evitar que o contexto da conversa influenciasse os fluxogramas gerados.

Na abordagem *zero-shot prompting*, os modelos receberam uma instrução, sem apresentação de exemplos prévios. O *prompt* utilizado foi o seguinte: “Gere um fluxograma em formato de imagem para o seguinte algoritmo: <PROBLEMA>”. Em que <PROBLEMA> corresponde ao enunciado de um dos problemas de programação selecionados para o estudo.

Na abordagem *few-shot prompting*, os modelos receberam exemplos prévios antes da instrução para gerar um fluxograma e do enunciado do problema a ser resolvido. Foram utilizados três exemplos, acompanhados de seus respectivos fluxogramas, apresentados na Figura 1.

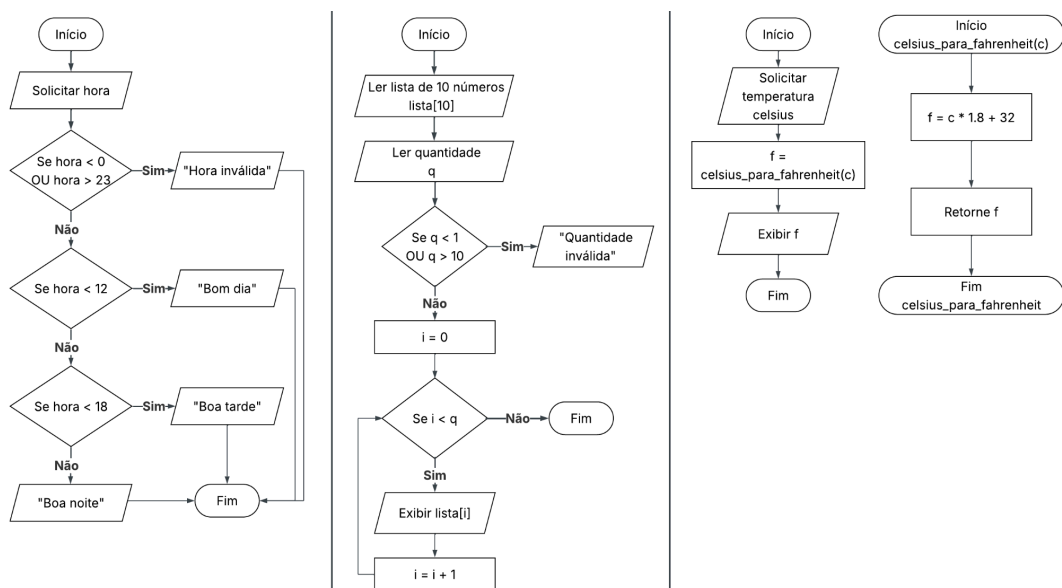


Figura 1. Fluxogramas utilizados como exemplos na estratégia *few-shot prompting*: (a) Exemplo 1, (b) Exemplo 2 e (c) Exemplo 3.

O primeiro exemplo envolvia estruturas condicionais por meio de um algoritmo que determinava uma saudação baseada na hora informada pelo usuário. O segundo exemplo explorava listas e estruturas de repetição, exibindo os primeiros elementos de uma lista de acordo com a quantidade informada. O terceiro exemplo utilizava uma função para converter temperaturas de graus Celsius para graus Fahrenheit. Dessa forma, os exemplos apresentavam nível de complexidade semelhante ao dos problemas selecionados e contemplavam os conteúdos definidos.

4.3. Avaliação dos Fluxogramas

Depois de gerados, os fluxogramas foram avaliados com base em um conjunto de critérios propostos a fim de verificar sua estrutura. Os critérios foram construídos com base nas especificações da norma ISO 5807 [ISO 1985] e nas recomendações de Manzano [2004] para sua aplicação no contexto de algoritmos. A avaliação foi realizada manualmente por um avaliador, e os critérios são apresentados no Quadro 3.

Quadro 3. Critérios para avaliação dos fluxogramas gerados

Critério	Descrição
Continuidade do fluxo	Verifica se o fluxo apresenta continuidade, de forma que todos os caminhos conduzam ao símbolo de fim.
Conectividade dos elementos	Verifica se não há símbolos soltos (desconectados) do fluxo ao qual pertencem.
Ausência de redundância	Verifica a inexistência de comandos duplicados ou ações redundantes.
Clareza textual	Avalia se os textos estão posicionados corretamente nos símbolos e se a linguagem utilizada permite a compreensão do algoritmo representado.
Uso correto dos terminadores	Verifica se início e fim são representados pelo símbolo Terminador.
Uso correto de entrada e saída	Verifica se os comandos de entrada e saída são representados pelo símbolo Dados (paralelograma).
Uso correto de processamento	Verifica se os comandos de processamento são representados pelo símbolo Processo (retângulo).
Uso correto das linhas de fluxo	Avalia se as linhas indicam direção única, respeitando a quantidade correta de conexões de cada símbolo e apontando para elementos logicamente válidos.
Uso correto de decisões	Avalia se as decisões são representadas pelo símbolo Decisão (losango) e se as saídas estão identificadas com “Sim” e “Não”, ou equivalente.
Representação de funções	Analisa se as funções são chamadas corretamente e se seus respectivos fluxogramas estão representados.

Os fluxogramas receberam uma pontuação de 0 a 10, correspondente à quantidade de critérios em conformidade. Quando um critério não se aplicava ao problema avaliado, ele era considerado atendido para fins de pontuação. Os fluxogramas que atingiam a pontuação máxima (10) eram então submetidos a um teste de execução passo a passo, a fim de verificar se a lógica proposta de fato resolvia o problema de programação. Essa etapa foi necessária porque a conformidade com os critérios não garantia, por si só, que o algoritmo estivesse correto. Assim, a pontuação representava apenas o resultado da avaliação estrutural, enquanto a classificação do fluxograma como correto dependia também da validação de sua lógica. Somente os

fluxogramas que atenderam a todos os critérios de conformidade e que também resolveram corretamente o problema foram classificados como corretos.

5. Resultados

O Quadro 4 resume a avaliação dos fluxogramas gerados pelo ChatGPT 5.5 utilizando a abordagem *zero-shot prompting*. Para fins de sumarização, foram apresentados apenas os critérios que apresentaram falhas em pelo menos um fluxograma. A marcação “✓” significa que o critério foi atendido, enquanto “✗” indica falha. O resultado C indica que o fluxograma estava correto e I, incorreto.

Quadro 4. Avaliação dos fluxogramas do ChatGPT 5.5 (*zero-shot prompting*)

Fluxograma do Problema:	P01	P02	P03	P04	P05	P06	P07	P08	P09	P10
Continuidade do fluxo	✓	✓	✓	✓	✓	✗	✗	✓	✓	✗
Uso correto de entrada e saída	✓	✓	✗	✗	✓	✓	✓	✓	✓	✗
Uso correto de processamento	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗
Uso correto das linhas de fluxo	✓	✓	✓	✓	✓	✗	✗	✗	✗	✓
Pontuação	10	10	9	9	10	8	8	9	8	7
Resultado	C	C	I	I	C	I	I	I	I	I

Para ilustrar a falha mais recorrente, relacionada às linhas de fluxo, o fluxograma P06 pode ser observado na Figura 2 (a). Nesse caso, a linha entre a decisão “andar ≤ N” e a saída “Exibir ‘Fim da lavagem...’” é utilizada incorretamente, pois apresenta dupla orientação. Já o fluxograma P10, que apresentou mais falhas, pode ser observado na Figura 2 (b). O comando “Retorne 1” deveria ser representado por um retângulo de processamento. Além disso, nesse ponto, o fluxo não é devidamente direcionado ao fim. Também, o comando “Exiba 'N deve ser inteiro não negativo!’” deveria utilizar um paralelogramo, para representar saída.

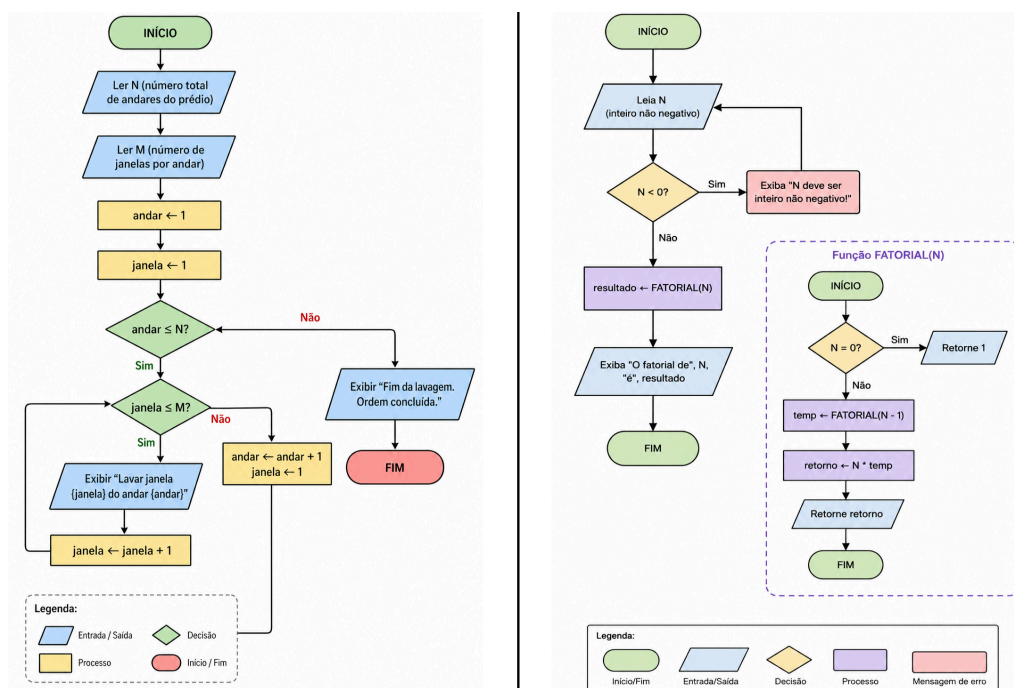


Figura 2. Fluxogramas gerados pelo ChatGPT 5.5 utilizando *zero-shot prompting* para os problemas: (a) P06 - Janelas e (b) P10 - Fatorial Recursivo.

O Quadro 5 resume a avaliação dos fluxogramas gerados pelo Gemini 3.5 utilizando a abordagem *zero-shot prompting*. As marcações seguem o mesmo padrão do quadro anterior. A notação “-” indica critério não aplicável.

Quadro 5. Avaliação dos fluxogramas do Gemini 3.5 (*zero-shot prompting*)

Fluxograma do Problema:	P01	P02	P03	P04	P05	P06	P07	P08	P09	P10
Continuidade do fluxo	✓	✓	✓	✓	✓	✗	✗	✗	✗	✓
Conectividade dos elementos	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓
Ausência de redundância	✓	✓	✓	✓	✓	✓	✗	✓	✗	✓
Clareza textual	✓	✓	✓	✗	✓	✗	✗	✗	✓	✓
Uso correto de entrada e saída	✓	✓	✗	✗	✓	✗	✓	✓	✗	✓
Uso correto de processamento	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓
Uso correto das linhas de fluxo	✓	✓	✗	✗	✓	✗	✗	✗	✗	✗
Uso correto de decisões	-	-	✗	✓	✗	✗	✓	✓	✓	✓
Representação de funções	-	-	-	-	-	-	-	-	✗	✓
Pontuação	10	10	7	7	9	5	6	6	4	9
Resultado	C	C	I	I	I	I	I	I	I	I

O fluxograma P09, que obteve a menor pontuação, pode ser visualizado na Figura 3 e ilustra diversas falhas identificadas nos critérios de avaliação, incluindo problemas na representação da função, no uso das linhas de fluxo e nos comandos de entrada e saída, além de inconsistências relacionadas à continuidade do fluxo.

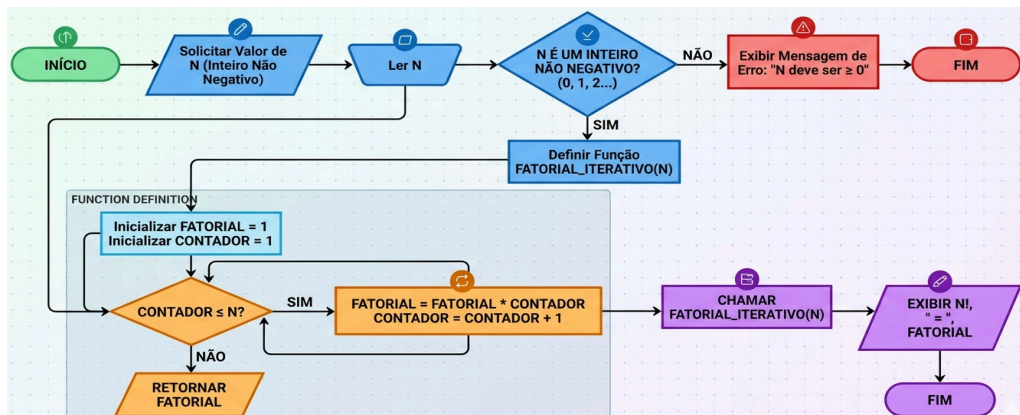


Figura 3. Fluxograma gerado pelo Gemini 3.5 utilizando *zero-shot prompting* para o problema P09 - Fatorial Iterativo.

Em relação aos resultados da abordagem *few-shot prompting*, o Quadro 6 apresenta a avaliação dos fluxogramas gerados pelo ChatGPT 5.5.

Quadro 6. Avaliação dos fluxogramas do ChatGPT 5.5 (*few-shot prompting*)

Fluxograma do Problema:	P01	P02	P03	P04	P05	P06	P07	P08	P09	P10
Ausência de redundância	✓	✓	✗	✓	✓	✓	✓	✓	✓	✓
Uso correto das linhas de fluxo	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓
Pontuação	10	10	9	10	10	10	10	9	10	10
Resultado	C	C	I	C	C	C	C	I	C	C

Apenas dois fluxogramas apresentaram falhas nos critérios de avaliação. No fluxograma P03, visto na Figura 4 (a), ocorre a duplicação de um comando de saída “Exibir ‘Aguardar na calçada’”, que é executado duas vezes quando a cor é vermelha. No fluxograma P08, mostrado na Figura 4 (b), há um erro na utilização da linha de fluxo que parte do comando “ $j = j + 1$ ”, pois a seta aponta para outra linha em vez de um símbolo. Além disso, uma linha identificada com “Sim” está sem orientação.

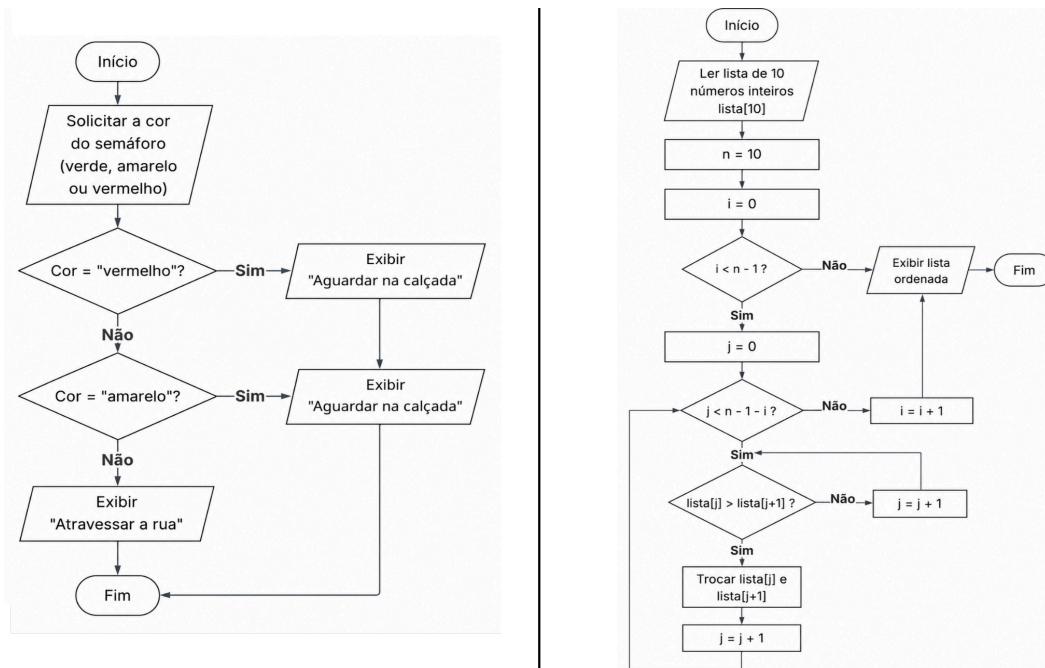


Figura 4. Fluxogramas gerados pelo ChatGPT 5.5 utilizando *few-shot prompting* para os problemas: (a) P03 - Semáforo e (b) P08 - Ordenação.

Por fim, o Quadro 7 mostra os resultados do Gemini 3.5.

Quadro 7. Avaliação dos fluxogramas do Gemini 3.5 (*few-shot prompting*)

Fluxograma do Problema:	P01	P02	P03	P04	P05	P06	P07	P08	P09	P10
Continuidade do fluxo	✓	✓	✓	✗	✓	✗	✓	✓	✓	✓
Conectividade dos elementos	✓	✗	✓	✓	✓	✓	✓	✓	✓	✓
Ausência de redundância	✓	✗	✓	✗	✗	✗	✓	✓	✓	✓
Clareza textual	✓	✓	✓	✓	✓	✗	✓	✓	✓	✓
Uso correto dos terminadores	✓	✗	✓	✓	✓	✓	✓	✓	✓	✓
Uso correto de processamento	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗
Uso correto das linhas de fluxo	✓	✓	✓	✓	✓	✗	✗	✗	✗	✓
Uso correto de decisões	-	-	✗	✗	✗	✗	✓	✗	✓	✓
Pontuação	10	7	9	7	8	5	9	8	9	9
Resultado	C	I	I	I	I	I	I	I	I	I

Dessa vez, alguns erros foram cometidos na geração do fluxograma P02, apresentado na Figura 5 (a). Nota-se que há elementos desconectados e duplicação do comando “Fim”, que também foi representado incorretamente por um símbolo de processamento, em vez do terminador adequado.

Já o fluxograma P06, exibido na Figura 5 (b), obteve a menor pontuação. A decisão “ $A \leq \text{Total de Andares}$ ” aparece em duplicidade, além disso, após a primeira ocorrência desse símbolo, o fluxo é encaminhado para ações desenhadas à direita, contudo, não há mais direcionamento que leve ao fim do algoritmo. Outras falhas ocorrem por linhas de fluxo ambíguas ou indefinidas, omissão da saída “Não” em uma decisão e o termo “ExibirExibir”, que compromete a clareza da linguagem.

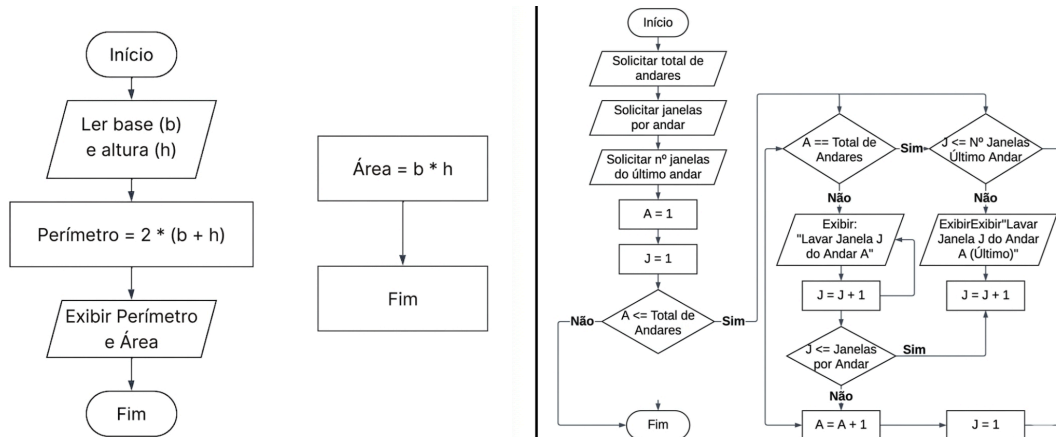


Figura 5. Fluxogramas gerados pelo Gemini 3.5 utilizando *few-shot prompting* para os problemas: (a) P02 - Retângulo e (b) P06 - Janelas.

6. Respostas às Perguntas de Pesquisa

Em relação à pergunta **QP1**, que investiga em que medida os fluxogramas produzidos atendem aos padrões de representação e solucionam os problemas propostos, apenas 14 dos 40 casos avaliados foram classificados como corretos, correspondendo a 35% do total. Essa proporção indica que os LLMs estudados são capazes de gerar fluxogramas adequados, embora ainda existam limitações relacionadas ao modelo utilizado, à abordagem de *prompting* e ao nível do problema.

Quanto aos conteúdos de programação que apresentaram melhores resultados (**QP2**), a Tabela 1 apresenta as médias das pontuações obtidas em cada conteúdo. Os problemas envolvendo apenas instruções sequenciais alcançaram a maior média, com 9,6 pontos. Em contrapartida, estruturas de repetição e manipulação de listas apresentaram o menor desempenho, ambas com média de 8,1 pontos.

Tabela 1. Pontuações médias por conteúdo, modelo e abordagem de *prompting*

	<i>Zero-shot prompting</i>		<i>Few-shot prompting</i>		Média por conteúdo
	ChatGPT 5.5	Gemini 3.5	ChatGPT 5.5	Gemini 3.5	
Instruções sequenciais	10	10	10	8,5	9,6
Estruturas condicionais	9	7	9,5	8	8,4
Estruturas de repetição	9	7	10	6,5	8,1
Manipulação de listas	8,5	6	9,5	8,5	8,1
Funções e recursão	7,5	6,5	10	9	8,3
Média por modelo e <i>prompting</i>	8,8	7,3	9,8	8,1	

Naturalmente, os problemas que envolvem apenas instruções sequenciais utilizam uma estrutura de representação mais simples, pois não exigem a simbologia associada à decisão e suas linhas de fluxos alternativos. Por outro lado, para os problemas de repetição, esses elementos são de uso obrigatório. Problemas que envolvem manipulação de listas também exigem estruturas de repetição para percorrer seus elementos.

A contagem dos critérios de avaliação não atendidos permite identificar as falhas mais comuns nos fluxogramas incorretos, respondendo à **QP3**. A falha mais recorrente está relacionada ao uso das linhas de fluxo, com 16 ocorrências. Em seguida, destacam-se problemas de continuidade do fluxo (9 ocorrências), representação de decisões e caminhos alternativos (8 ocorrências) e comandos duplicados ou redundantes (7 ocorrências).

Quanto à **QP4**, a qualidade dos fluxogramas foi influenciada tanto pelo LLM quanto pela estrutura de *prompting* adotada. Os resultados apontam que a utilização dos exemplos teve impacto positivo. Como visto na Tabela 1, a pontuação média do ChatGPT 5.5 passou de 8,8 pontos utilizando *zero-shot prompting* para 9,8 pontos com *few-shot prompting*. Já a pontuação do Gemini 3.5 aumentou de 7,3 para 8,1 pontos.

Além dos efeitos quantitativos, foram observadas mudanças visuais e textuais nos fluxogramas gerados. Na segunda abordagem, os modelos passaram a adotar um estilo mais próximo dos exemplos fornecidos, adaptando o formato dos símbolos e reproduzindo características como a ausência de cores e estilo de linguagem mais consistente entre os diagramas.

Entretanto, a abordagem *few-shot prompting* provocou um efeito colateral observado no fluxograma P02 gerado pelo Gemini 3.5, visto na Figura 5 (a). Após receber os exemplos, o modelo gerou um diagrama com elementos desconectados. Uma possível explicação é a influência do terceiro exemplo, que continha uma função representada em um fluxo complementar, levando o modelo a reproduzir parte dessa estrutura em um problema que não exigia função.

Por fim, em relação às diferenças de desempenho entre os modelos, os resultados obtidos neste estudo indicam que o ChatGPT 5.5 apresentou desempenho superior ao Gemini 3.5, uma vez que obteve as maiores pontuações médias (Tabela 1) e gerou 11 fluxogramas corretos, contra apenas 3. A Figura 6 ilustra esse comportamento ao longo dos dez problemas avaliados, evidenciando que os ganhos obtidos com *few-shot prompting* foram mais consistentes, especialmente para o ChatGPT 5.5.

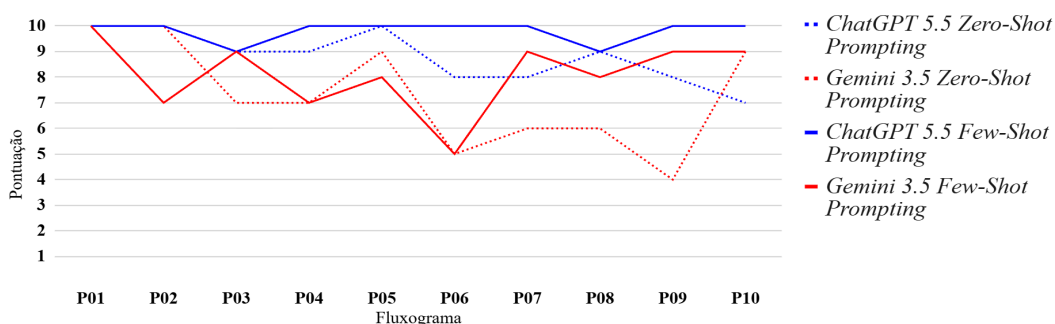


Figura 6. Pontuações dos fluxogramas gerados pelos modelos e abordagens

7. Conclusões

Este estudo avaliou os LLMs ChatGPT 5.5 e Gemini 3.5 em diferentes abordagens de *prompting* para a geração de fluxogramas a partir de enunciados de problemas de programação de nível introdutório. Os resultados permitiram avaliar se os modelos são capazes de gerar fluxogramas dentro dos padrões estabelecidos pela ISO 5807 e se suas soluções representam algoritmos válidos para os respectivos problemas. As perguntas de pesquisa foram respondidas, permitindo o alcance dos objetivos propostos.

Foi possível observar que os modelos são capazes de gerar fluxogramas compatíveis com os problemas propostos, porém ainda apresentam limitações importantes. Nas condições adotadas neste estudo, apenas 35% dos fluxogramas gerados atenderam integralmente aos critérios de avaliação, indicando que essas tecnologias ainda não oferecem geração automática totalmente confiável, sem necessidade de revisão ou supervisão humana. Entretanto, os resultados também mostram que técnicas de *prompting* podem contribuir para a qualidade das respostas, uma vez que o ChatGPT 5.5 alcançou média de 9,8 pontos na abordagem *few-shot prompting*, cometendo apenas duas falhas entre os dez fluxogramas gerados.

Esse resultado pode ser interpretado como uma oportunidade educacional, visto que a necessidade de analisar, identificar falhas e corrigir os fluxogramas gerados impede o estudante de aceitar diretamente a solução produzida, levando-o a um exercício mais crítico e a reflexões sobre a lógica de programação. Ademais, os resultados sugerem que gerações de maior qualidade dependem da aplicação de *prompts* mais elaborados, reduzindo a obtenção de respostas adequadas a partir de instruções simples.

Entre as limitações deste estudo, destaca-se que os resultados obtidos estão restritos ao conjunto de problemas e aos modelos avaliados, de modo que não podem ser generalizados para todos os contextos de ensino de programação ou para outros LLMs. Além disso, a ausência de múltiplos avaliadores pode ter introduzido viés na avaliação. Outra limitação decorre da natureza não determinística dos modelos generativos, de forma que os resultados apresentados refletem o comportamento observado em uma única execução de cada experimento e podem variar em novas execuções.

Diante disso, como trabalhos futuros, sugere-se a ampliação da investigação para comparar múltiplas gerações de um mesmo problema, possibilitando análises estatísticas mais robustas, além da avaliação de outros modelos e da comparação entre fluxogramas gerados por LLMs e fluxogramas produzidos por especialistas humanos. Recomenda-se ainda a ampliação do banco de problemas e a realização de estudos em sala de aula nos quais os estudantes sejam desafiados a elaborar fluxogramas com e sem o auxílio de LLMs, buscando compreender suas percepções sobre a confiabilidade das respostas, a necessidade de analisar criticamente os resultados e o impacto disso na aprendizagem.

Por fim, este estudo finaliza destacando a utilização de fluxogramas como uma ferramenta recomendada para o ensino de programação, reforçando seu potencial como atividade de menor interferência da IA generativa, uma vez que exige a análise crítica das soluções produzidas, inclusive em problemas introdutórios.

Disponibilidade de Artefatos

Os enunciados completos dos problemas, os *prompts* utilizados e todos os fluxogramas gerados estão disponíveis para acesso através do repositório público: <https://github.com/mhlsa-ynn/llm-flowcharts>.

Declaração sobre uso de Inteligência Artificial

Foi utilizada a ferramenta ChatGPT para apoio na revisão estrutural do texto, incluindo verificação gramatical, sugestão de sinônimos, ajustes de fluidez e discussão de alternativas de redação. Após o uso da ferramenta, todo o conteúdo gerado foi cuidadosamente avaliado pelos autores, podendo ser modificado, adaptado ou descartado, a fim de decidir de forma criteriosa pela adoção ou não das sugestões apresentadas. Ressalta-se que os autores assumem total responsabilidade pelo conteúdo desta publicação, sendo as ideias, a redação original e o desenvolvimento da pesquisa de sua inteira autoria.

Referências

- Darda, A. and Jain, R. (2024) “Code Generation from Flowchart using Optical Character Recognition & Large Language Model”, Authorea Preprints.
- Dingle, A. and Krulis, M. (2024) “Tackling Students' Coding Assignments with LLMs”, Proceedings of the 1st International Workshop on Large Language Models for Code, Lisbon, Portugal, p. 94-101.
- He, M., Zeng, J., Jiang, Y., Zhang, W., Liu, Z., Shi, X. and Zhou, A. (2025) “Flow2Code: Evaluating Large Language Models for Flowchart-based Code Generation Capability”, Findings of the Association for Computational Linguistics: ACL 2025, p. 8124-8146.
- Hong, A. and Hong, G. (2024) “The Effectiveness of Coding LLMs and the Challenges in Teaching CS1/2: A Case Study”, Journal of Computing Sciences in Colleges, Vol. 40, No. 1, p. 122-131.
- ISO (1985) "Information Processing - Documentation Symbols and Conventions for Data, Program and System Flowcharts, Program Network Charts and System Resources Charts", ISO 5807:1985, Geneva, Switzerland.
- Manzano, J. A. N. G. (2004) “Revisão e Discussão da Norma ISO 5807-1985 (E) Proposta para Padronização Formal da Representação Gráfica da Linha de Raciocínio Lógico Utilizada no Desenvolvimento da Programação de Computadores a Ser Definida no Brasil”, Revista Eletrônica Thesis, Vol. 1, p. 1-31.
- Minu, M. S., Babu, A., Senthil, R. and Ahmed, T. A. S. S. (2025) “Prompt2Diagram: Transforming Natural Language into Visual Diagrams Using Advanced NLP and LLM”, Proceedings of the 1st International Conference on Research and Development in Information, Communication, and Computing Technologies (ICRDICCT 2025), v. 4, p. 161-169.
- Muepu, D. M., Watanobe, Y., Ibn, A. M. F. and Shahajada, M. M. (2025) “Comparative Evaluation of ChatGPT, Gemini, and DeepSeek in Educational Problem Solving”,

IEEE 18th International Symposium on Embedded Multicore/Many-core Systems-on-Chip (MCSoc), p. 530-537.

Ouh, E. L., Gan, B. K. S., Jin Shim, K. and Wlodkowski, S. (2023) “ChatGPT, Can You Generate Solutions for my Coding Exercises? An Evaluation on its Effectiveness in an Undergraduate Java Programming Course”, Proceedings of the 2023 Conference on Innovation and Technology in Computer Science Education, Turku, Finland, p. 54-60.

Rahe, C. and Maalej, W. (2025) “How Do Programming Students Use Generative AI?”, Proceedings of the ACM on Software Engineering, Vol. 2, p. 978-1000.

SBC (2022) “Computação: Complemento à BNCC”, Sociedade Brasileira de Computação, Brasil.

Shuvo, U. A., Dip, S. A., Vaskar, N. R. and Al Islam, A. B. M. A. (2025) “Assessing ChatGPT’s Code Generation Capabilities with Short vs Long Context Programming Problems”, Proceedings of the 11th International Conference on Networking, Systems, and Security, NSysS '24, Association for Computing Machinery, New York, NY, USA, p. 32-40.

Xue, Y., Chen, H., Bai, G. R., Tairas, R. and Huang, Y. (2024) “Does ChatGPT Help With Introductory Programming? An Experiment of Students Using ChatGPT in CS1”, Proceedings of the 46th International Conference on Software Engineering: Software Engineering Education and Training, Lisbon, Portugal, p. 331-341.

Zheng, K. and Shi, Y. (2026) “CodeFlow: LLM-Generated Flowchart Feedback for Programming Students”, Proceedings of the 57th ACM Technical Symposium on Computer Science Education V.2, p. 1609-1610.