CBIE 2023
CONGRESSO BRASILEIRO DE INFORMÁTICA NA EDUCAÇÃO
Uma escola para o futuro: Tecnologia e conectividade a serviço da educação

Análise das Respostas do ChatGPT em Relação ao Conteúdo de Programação para Iniciantes

Luiz C. Pereira Filho[1], **Talita de P. C. de Souza**[1], **Luciano Bernardes de Paula**[1]

[1]Instituto Federal de São Paulo (IFSP)
Bragança Paulista – SP – Brasil

filho.luiz@aluno.ifsp.edu.br, {talita,lbernardes}@ifsp.edu.br

Abstract. *Teaching computer programming is considered important and, at the same time, challenging. Currently, AI tools have opened up a range of possibilities for use in education. The best known example is ChatGPT, which has natural language interaction and, in relation to basic programming content, demonstrates skills in creating, correcting and explaining codes. The aim of this paper was to analyze ChatGPT responses regarding initial programming content, in the context of beginning students. Qualitative tests were performed, in which the ChatGPT showed the potential to give correct answers and consistent explanations, and quantitative tests, with a success rate greater than 80%.*

Resumo. *O ensino de programação de computadores é considerado importante e, ao mesmo tempo, desafiador. Atualmente, ferramentas de IA abriram um leque de possibilidades para uso na educação. O exemplo mais conhecido é o ChatGPT, que possui interação em linguagem natural e, em relação ao conteúdo de programação básica, demonstra habilidades de criação, correção e explicação de códigos. O objetivo deste artigo foi analisar respostas do ChatGPT em relação ao conteúdo inicial de programação, no contexto de estudantes iniciantes. Foram realizados testes qualitativos, nos os quais o ChatGPT mostrou potencial para dar respostas corretas e explicações consistentes, e testes quantitativos, com aproveitamento superior a 80%.*

1. Introdução

Aprender programação, desde o entendimento do que é um algoritmo até a implementação em uma linguagem de programação, é um dos pontos fundamentais da Ciência da Computação. Entretanto, esse aprendizado é desafiador e possui diversos obstáculos.

O aprendizado de programação possui diversas dificuldades e desafios. Segundo [du Boulay 1986], dentre os motivos que dificultam o aprendizado da programação estão i) a falta de clareza ao estudante a respeito da importância de se aprender a programar, ii) a falta de entendimento de como o computador funciona, iii) a dificuldade de se aprender uma linguagem de programação, considerando sua sintaxe e semântica, iv) o entendimento de estruturas de uma linguagem de programação, tais como estruturas de decisão e repetição, e também iv) todo o pragmatismo de se programar, que envolve testes e rastreamento de erros (*debugging*).

Recentemente, foi colocado para uso público o ChatGPT[1] (*Chat Generative Pre-*

[1]Durante a escrita deste artigo, a versão disponibilizada gratuitamente para o público em geral era a versão 3.5, sendo essa a versão considerada ao longo deste artigo.

trained Transformer)[2], um modelo de linguagem baseado em inteligência artificial fornecido pela Open AI. O ChatGPT tem como diferencial o entendimento de linguagem natural e a geração de textos a partir de perguntas inseridas pelo usuário. Além disso, o ChatGPT possui a capacidade de gerar códigos fonte e explicá-los, a partir de enunciados, em diversas linguagens de programação. Uma vez que o ChatGPT se encontra disponível ao público de forma gratuita, é fato que estudantes iniciantes desta disciplina vão, cada vez mais, utilizá-lo para aprender.

Entretanto, o uso do ChatGPT na educação é um assunto ainda em discussão, com diversos pontos a serem considerados, como apresentado em [Lo 2023]. Um dos pontos a ser discutido é a qualidade atual das respostas geradas. Como informado pela própria OpenAI, o modelo, no estado atual, pode gerar informações erradas e conflituosas e ainda assim aparentar estar correto[3].

Nesse contexto, este trabalho tem como objetivo apresentar a qualidade das respostas geradas pelo ChatGPT em relação a conteúdos iniciais de programação de computadores. A ideia é, a partir do ponto de vista de estudantes iniciantes, ou seja, que ainda não possuem experiência necessária para julgar as respostas do ChatGPT, entender a qualidade atual das respostas geradas, tanto em relação ao código gerado quanto às suas explicações. Para isso, foram feitos dois conjuntos de testes junto ao ChatGPT: i) testes qualitativos, nos quais, a partir de um pedido de código ao *prompt* do ChatGPT e interações com este, foram analisados a qualidade das respostas e das explicações e como novas interações com este alteraram suas respostas; e ii) testes quantitativos, nos quais foram inseridos diversos exercícios que contemplam conteúdos iniciais de programação e as respostas e explicações geradas foram analisadas, sem novas interações.

2. Trabalhos relacionados

Apesar do ChatGPT versão 3.5 ter sido lançado ao público muito recentemente (novembro de 2022), é possível encontrar trabalhos relacionados com o aqui apresentado.

Em [Finnie-Ansley et al. 2022], os autores testaram o Codex[4], um modelo anterior ao ChatGPT e específico para criação de códigos de programação, dos mesmos criadores do ChatGPT. Os autores realizaram basicamente dois testes: o primeiro com exercícios iniciais de programação (chamados pelos autores de CS1 - *Computer Science 1*) e o segundo com variações de enunciados do problema da chuva (*rainfall problem*), o qual os autores consideram um problema muito utilizado em programação para iniciantes. Um diferencial do artigo apresentado aqui é que, atualmente, o Codex foi incorporado ao ChatGPT, não sendo mais específico para criação de códigos de programação. Outro diferencial é que, no artigo citado, não é feita uma análise das explicações a respeito do código feito pelo Codex, mas somente do código gerado.

Em [Finnie-Ansley et al. 2023] é feita uma continuação do trabalho anterior, agora considerando exercícios mais avançados de programação (chamados pelos autores de CS2 - *Computer Science 2*), tais como metodologias, recursão, buscas, ordenação, etc, e segue a mesma linha do artigo anterior, focando no código gerado pelo Codex.

[2]Introduzindo o ChatGPT - https://openai.com - acessado em 09/07/2023.

[3]Limitações - https://openai.com/blog/chatgpt - acessado em 08/07/2023.

[4]OpenAI Codex - https://openai.com/blog/openai-codex - acessado em 10/07/2023.

Em [Sarsa et al. 2022] também é utilizado o Codex para gerar enunciados de exercícios e explicação para códigos já prontos. Diferentemente, neste artigo foram analisadas as respostas não só da explicação, mas também o código fonte gerado.

Uma diferença adicional que pode ser citada é que os artigos citados consideram as respostas e explicações do Codex para códigos gerados na linguagem Python, enquanto que no presente artigo é considerada a linguagem C, como explicado na Seção 5.

3. ChatGPT

O ChatGPT, versão 3.5, é um modelo de linguagem baseado em aprendizado de máquina (ML - *Machine Learning*), pertencente à categoria dos grandes modelos de linguagem (LLM - *Large Language Models*) [OpenAI 2023] e com a capacidade de compreensão de linguagem natural (NLU - *Natural Language Understanding*), facilitando a interação e a comunicação entre os usuários e o sistema [Aljanabi et al. 2023] . Na atual versão 3.5, o modelo incorpora técnicas avançadas, como o uso de aprendizado por reforço com *feedback* humano (RLHF - *Reinforcement Learning from Human Feedback*), para aprimorar ainda mais sua capacidade de gerar respostas significativas e humanas. O RLHF permite que o modelo aprenda com a orientação e *feedback* fornecidos por humanos, o que ajuda a melhorar a qualidade e a precisão das respostas geradas. Os modelos GPTs são caracterizados por tecnologias de propósito geral, possuindo um uso mais amplo [OpenAI 2023].

Os LLMs, categoria ao qual o ChatGPT pertence, são modelos probabilísticos [Cámara et al. 2023] de processamento de linguagem natural que podem gerar textos semelhantes aos gerados por humanos, além de também poder gerar códigos de programação com sucesso [Li et al. 2022]. Os textos são gerados a partir de *tokens*, que são previstos probabilisticamente baseados no contexto, sendo que os modelos do ChatGPT são treinados para fazer essa previsão de maneira otimizada [Lee 2023]. Um *token* representa uma sequência de caracteres de ocorrência mais frequente, sendo a unidade básica de texto que o modelo processa[5]. Os LLMs são gerados por treinamento extenso utilizando grandes volumes de dados textuais, e se utilizam algoritmos para identificar padrões e relacionamentos entre palavras e frases [Tsai et al. 2023].

Entre os possíveis usos para o ChatGPT, a educação é um setor no qual a ferramenta pode ser aplicada, apoiando o ensino e aprendizado em diversos domínios, como direito, economia, matemática, medicina e programação [Lo 2023]. Tratando especificamente do domínio de programação, ela pode apoiar a escrita e a depuração de código, inclusive, dando explicações sobre o conteúdo gerado.

3.1. Criação de códigos com ChatGPT

Em relação à criação e explicação de código fonte pelo ChatGPT, o seu uso por meio de linguagem natural possibilita aos usuários inserir trechos de códigos ou comandos, sem a necessidade de se restringir a palavras-chave ou frases específicas, e obter respostas relevantes e contextualizadas com suas intenções [Aljanabi et al. 2023]. Entre os dados usados para o treinamento do modelo do ChatGPT, há um vasto conjunto de códigos escritos por programadores, que são usados para prever o próximo *token* da sequência na criação de novos códigos. Entre os modelos que compõem a versão 3.5, está o code-davinci-002,

[5]Índice do modelo para pesquisadores - https://platform.openai.com/ - acessado em 08/07/2023.

que é ideal para tarefas de conclusão de código puro. Esse modelo, além de traduzir linguagem natural para código, é capaz de inserir conclusões no código, suportando uma solicitação de até 8001 *tokens* por vez[6].

O ChatGTP também é capaz de rastrear a execução de código, fazer a correção de defeitos ou falhas, dando detalhes de como foi feita essa correção, prever a saída para o código e explicar conceitos fundamentais de programação, incluindo temas mais densos, como complexidade de algoritmos [MacNeil et al. 2022]. Essa combinação de recursos torna o ChatGPT uma ferramenta relevante e eficiente para a codificação, fornecendo respostas contextuais que contribuem para a aprendizagem de programação.

4. Metodologia

Em relação aos testes executados, foram realizados dois tipos: testes qualitativos e testes quantitativos, os quais são explicados a seguir.

Para os testes qualitativos, foram selecionados exercícios de programação que envolvessem conceitos considerados básicos: i) uso de variáveis e operadores: variáveis são áreas de memória nomeadas, nas quais é possível armazenar valores para uso posterior, inclusive utilizando operadores matemáticos; ii) estrutura de decisão: aquelas com as quais é possível inserir um teste lógico no código que define seu comportamento; iii) estruturas de repetição: aquelas com as quais é possível fazer com que uma ou mais instruções sejam executadas repetidamente em um código; iv) vetores e matrizes: estruturas de dados que podem armazenar diversos valores; e v) *strings*: sequência de caracteres armazenadas em vetores que representam palavras ou frases. Para cada um dos tópicos, foram propostos exercícios, utilizando o *prompt* do ChatGPT para que este gerasse a solução, a qual foi analisada em relação à corretude do código, sua sintaxe, clareza e qualidade da explicação. Nesse grupo de testes, foram feitas interações com o ChatGPT caso algum equívoco fosse identificado e suas novas respostas eram consideradas. É importante frisar que a análise, tanto do código quanto da explicação, foi feita considerando o contexto de estudantes iniciantes em programação. Mais detalhes são encontrados na Seção 5.

Para o conjunto de testes quantitativos, foram submetidos ao ChatGPT exercícios retirados de [Mizrahi 2008] e suas respostas foram analisadas, sem interações adicionais. Foram submetidos exercícios retirados dos seguintes capítulos: Cap. 2 - Operadores, Cap. 3 - Laços, Cap. 4 - Comando de decisão e Cap. 7 - Matrizes (que também considera vetores). Os exercícios sobre *strings* foram retirados do capítulo de matrizes, visto a falta de um capítulo específico para o tema. Todos os exercícios usados como entrada para o ChatGPT foram obtidos na seção de exercícios no final de cada capítulo, com exceção para os exercícios de matrizes, que foram retirados dos exemplos nas subseções do Cap. 7. Para cada exercício, uma nova sessão no ChatGPT foi criada, a fim de manter o isolamento entre as questões.

Para padronizar os testes, para cada exercício, foi pedida a linguagem de programação na qual as respostas deveriam ser dadas, no caso linguagem C, e em qual sistema operacional as respostas seriam executadas, neste caso, Linux[7], acrescentando no início dos enunciados a sentença *"Escreva um programa em linguagem C para Linux..."*.

[6]Índice do modelo para pesquisadores - https://platform.openai.com/ - acessado em 08/07/2023.

[7]Isso é necessário pois existem bibliotecas da linguagem C que são exclusivas para sistemas Windows, o que poderia invalidar a resposta, uma vez que o ambiente de testes utilizou o sistema Linux.

5. Testes e resultados

Neste artigo, para a geração das respostas do ChatGPT, foi considerada a linguagem de programação C. Isso se deve a dois motivos: trata-se de uma linguagem que oferece maior liberdade ao programador, não tratando todas as exceções que um código possa gerar, ficando sob responsabilidade do programador, e é a linguagem utilizada na disciplina inicial de programação do curso da instituição dos autores deste artigo. Nas próximas subseções os resultados obtidos para cada tópico são apresentados.

5.1. Utilização de variáveis e operadores

Um dos conceitos iniciais da programação é a utilização de variáveis. Variáveis são nomes dados a áreas de memória, gerenciadas pelo sistema operacional, nas quais, em um programa de computador, é possível armazenar valores e reutilizá-los. Em relação aos operadores, é possível realizar cálculos com as quatro operações básicas da matemática (adição, subtração, multiplicação e divisão).

Um ponto importante é que, na linguagem C, utilizada neste artigo, as variáveis possuem tipo estático. Dessa forma, ao criar uma variável que receberá valores inteiros, esta deve ser do tipo *int*. Para valores de ponto flutuante (números reais), esta deve ser declarada como *float*, e assim por diante.

Em relação ao ChatGPT e variáveis, o programa proposto foi: *"faça um programa em C que recebe dois valores e apresente na tela a soma, a subtração e multiplicação e a divisão desses dois valores"*.

O objetivo desse exercício para os estudantes é verificar se o conceito de variável foi aprendido corretamente, assim como a sua utilização. O enunciado deixa em aberto em relação a quais tipos de variáveis devem ser usados, deixando a cargo do estudante.

Para esse exercício, o código gerado pelo ChatGPT estava claro e a explicação a respeito do código também estava correta. Entretanto, não foi mencionado na resposta o fato de que o segundo valor recebido não pode ser zero, pois resultaria em uma divisão por zero ao se apresentar a divisão entre os dois números. A resposta dada pelo ChatGPT não traz essa informação ao usuário, a respeito da necessidade da restrição no valor da segunda variável. Ao ser questionado a respeito dessa possibilidade, o ChatGPT corrigiu o código gerado, apresentando uma nova versão que tratava essa exceção, agora totalmente correto.

5.2. Estrutura de decisão

A estrutura de decisão, comumente conhecida como *if-else*, é uma estrutura muito comum em linguagens de programação. Utilizando esse tipo de estrutura é possível criar programas que, a partir de um teste lógico do tipo $x < y$ execute ou não uma instrução. Por exemplo, em um programa que calcula a média de um aluno, o teste pode ser *"a média final do aluno é maior ou igual a 6,0?"* e, em caso afirmativo, o programa apresenta a frase *"Aprovado"*, caso contrário apresenta *"Reprovado"*.

Para este teste, inicialmente foi pedido ao ChatGPT que *"faça um programa que receba 3 valores inteiros e os apresente em ordem crescente"*. Existem algumas formas de se fazer esse programa: receber os valores em três variáveis (A, B e C, por exemplo) e, sem alterar seus valores, apresentá-las na ordem crescente; ou trocar seus valores de

forma que A fique com o menor valor, B o segundo maior e C o maior. A estrutura de decisão é utilizada, nesse exercício, para se comparar os valores das variáveis A, B e C.

A resposta apresentada pelo ChatGPT utilizou-se de funções e ponteiros para realizar a troca de valores entre as variáveis. Essa resposta estava correta e clara, porém, considerando estudantes iniciantes, é improvável que esses já estejam familiarizados com os conceitos de funções e ponteiros, o que pode gerar confusão. Foi então pedido *"repita o código anterior, mas sem usar troca de valores entre as variáveis"*. O ChatGPT apresentou uma nova versão, dessa vez sem usar troca de valores entre variáveis, que estava correta e clara, tanto em relação ao código quanto à explicação.

Para o código que usou ponteiros, foi pedido *"repita o código, mas sem usar ponteiros"*. A resposta foi interessante: dentro de uma mesma resposta, o ChatGPT criou um código utilizando ponteiros e funções e, por três vezes, pediu desculpas pelo equívoco e gerou um novo código e explicação, todos os três ainda com ponteiros e funções. Foi questionado *"a resposta anterior ainda utiliza ponteiros"*, o que gerou uma nova resposta com funções e ponteiros. Foi, então, pedido explicitamente *"faça sem usar ponteiros ou funções"*, o que, finalmente, gerou uma resposta que utilizava troca de valores entre variáveis sem uso de ponteiros e funções e estava correta, tanto no código e explicação.

5.3. Estrutura de repetição

As estruturas de repetição (também chamadas de laços de repetição) são utilizadas em linguagens de programação para se repetir uma ou mais instruções uma quantidade prevista ou imprevista de vezes. As principais estruturas de repetição são os laços *for*, *while* e *do-while* (uma variação do laço *while*), e todas são baseadas em um teste lógico que controla quantas vezes as instruções serão repetidas. O laço *for* é mais indicado quando sabe-se de antemão a quantidade de vezes que as instruções devem ser repetidas (por exemplo, apresentar uma contagem na tela de 0 a 10) e os laços *while* e *do-while* para quando não há previsão de quantas vezes as instruções serão repetidas (por exemplo, repetir as instruções até que o valor inserido pelo usuário seja válido).

O seguinte problema relacionado com laços de repetição foi proposto: *"faça um programa que simule dois semáforos de trânsito. O primeiro semáforo muda de estado (vermelho para verde ou vice-versa) a cada x segundos. O segundo semáforo muda de estado a cada y segundos, sendo $x < y$. Indique, para um período de 1000 segundos, quantas vezes os dois semáforos estarão no mesmo estado (vermelho ou verde) ao mesmo tempo. Considere os semáforos iniciando iguais."*.

Um ponto importante nesse tipo de problema é estimular o pensamento abstrato e a criatividade do estudante. Uma pergunta comum feita pelos estudantes iniciais, ao se depararem com esse problema, é como devem fazer para marcar o tempo no programa. Entretanto, apesar do problema proposto definir que o programa deve considerar um período em segundos, para obter a resposta não é necessário que a solução seja feita considerando um período de tempo real. O correto é criar uma contagem de 0 a 1000, com a ajuda dos laços de repetição, e monitorar os estados dos dois semáforos a cada iteração.

A resposta dada pelo ChatGPT estava correta e não considerava o tempo real de 1000 segundos, e sim um contador de 0 até esse valor, o que era esperado. Entretanto, na primeira resposta retornada, não havia controle para que o valor de x fosse menor que y e nem que os dois valores fossem maiores que 0. Foi perguntado *"o código acima possui*

alguma restrição a respeito dos valores de x e y?". As respostas foram basicamente duas: que *"os valores de x e y sejam valores positivos maiores que zero para evitar loops infinitos"*, o que é correto, porém não foi tratado no código e que *"pode haver um número excessivo de iterações dentro do período de 1000 segundos. Isso pode acontecer se x e y forem múltiplos um do outro ou se estiverem muito próximos um do outro."*, o que não faz sentido e não está correto. Foi, então, dito para o ChatGPT *"não foi considerado que x deve ser menor que y e que não podem ser negativos"* e dessa vez a resposta gerada considerava, corretamente, as restrições em relação aos valores de x e y.

5.4. Vetores e matrizes

Vetores são como várias variáveis do mesmo tipo agrupadas, sendo cada elemento do vetor acessado por meio de um índice de posição, por exemplo *v[0]*, *v[1]* e assim por diante. Matrizes são vetores com mais de uma dimensão, sendo as mais comuns com duas dimensões. Cada posição de uma matriz bidimensional pode ser acessada via dois índices, um de linha e outro de coluna, por exemplo *m[0][2]*.

Foi passado o seguinte problema ao ChatGPT: *"escreva um programa em C que receba dez valores diferentes do tipo float e os apresente na tela, primeiro na ordem em que foram inseridos e depois na ordem contrária de inserção. Utilize vetor"*. A resposta oferecida pelo ChatGPT foi exata e a explicação clara.

Foi proposto para o ChatGPT um segundo exercício: *"faça um programa em C que utilize uma matriz e preencha-a valores aleatórios entre 0 e 500. Em seguida o programa deve apresentar o maior e o menor valor encontrado na matriz"*. O código gerado apresentou uma matriz 3x3 e era claro e correto, assim como sua explicação.

5.5. *Strings*

Strings são cadeias de caracteres (letras, algarismos e caracteres especiais) que formam uma ou mais palavras como, por exemplo, um nome completo, um endereço, etc. Em algumas linguagens de programação, existe o tipo de dado *string*, que tem esse propósito. Entretanto, na linguagem C, as *strings* são vetores do tipo *char*, sendo que em cada posição é armazenado um caractere que forma aquela *string*.

Foi proposto o seguinte exercício para o ChatGPT: *"escreva um programa em C que recebe duas palavras e testa se essas palavras são iguais"*. A resposta dada pelo ChatGPT estava correta, assim como sua explicação, porém, esta possuía um problema importante: foi usada a função *scanf* com o parâmetro %s para o recebimento das palavras. Essa função é considerada insegura para esse fim, pois permite que a sequência de caracteres inserida seja maior que o número de posições existentes no vetor de caracteres que irá recebê-la, o que pode gerar um estouro de memória (*buffer overflow*).

Ao ser questionado a respeito do que pode ser melhorado no código apresentado, o ChatGPT citou esse problema e sugeriu uma função segura, no caso a função *fgets*, inclusive gerando uma nova versão do código, sem utilizar *scanf* para o recebimento da *string*. É interessante citar que, no caso de um estudante iniciante em programação, este não teria o conhecimento necessário a respeito da vulnerabilidade da função em questão e, portanto, assumiria que a resposta oferecida estaria totalmente correta.

5.6. Testes quantitativos

Para cada exercício proposto ao ChatGPT, foi avaliado se os códigos gerados compilavam e executavam, se as saídas estavam corretas em relação aos enunciados dos exercícios assim como suas explicações. Os resultados podem ser vistos na Tabela 1, na qual podem ser conferidos os conceitos de programação testados nos exercícios (coluna Tópicos), a quantidade de exercícios para cada tópico (coluna Quantidades), quantos exercícios submetidos compilaram e executaram sem a ocorrência de erros (coluna Compilou/Executou), a quantidade de exercícios que apresentaram uma resposta correta ao exercício proposto (coluna Correção) e a quantidade de exercícios para os quais a explicação estava correta (coluna Explicação). Como ambiente de testes foi usada a plataforma Replit[8], na qual é possível fazer a compilação usando o compilador GCC[9] (*GNU Compiler Collection*) na versão 11.3.0, e a execução no sistema operacional Linux Ubuntu 22.04[10].

Tópicos	**Quantidades**	**Compilou/Executou**	**Correção**	**Explicação**
Variáveis e Operadores	9	9	9	9
Estruturas de Repetição	6	6	6	6
Estruturas de Decisão	10	10	10	10
Strings	15	15	8	10
Vetores e Matrizes	6	6	5	5
Totais	**46**	**46**	**38**	**40**

Tabela 1. Testes quantitativos.

Para os tópicos de variáveis e operadores, estruturas de repetição e estruturas de decisão, foram propostos 9, 6 e 10 exercícios, respectivamente, para o ChatGPT. As respostas foram todas corretas, tanto para compilações e execuções quanto para as saídas das execuções e explicações sobre os códigos gerados. Para as explicações com poucos detalhes, foi feita uma nova solicitação após a primeira resposta com a sentença *"explique o código."*, com o intuito de estimular o ChatGPT a dar mais detalhes sobre o código gerado. Com isso, os três tópicos obtiveram 100% de respostas corretas que satisfazem os requisitos dos enunciados, com códigos e explicações corretas.

No tópico de variáveis e operadores, em um dos exercícios, o ChatGPT fez uso de uma função. Nas soluções para estruturas de repetição, o ChatGPT fez uso da estrutura *for* em 3 das 6 respostas, utilizou o *while* em 2 respostas e, em uma questão fez o uso de recursão, conceito que não é considerado inicial. Em 2 exercícios foram utilizadas funções para encapsular os códigos com as lógicas que atenderam aos enunciados. Nas soluções dos exercícios de estruturas de decisão, foram usadas funções em duas respostas.

O tópico de *strings* foi testado com 15 questões. Todas as respostas compilaram e executaram corretamente; 8 foram consideradas corretas, pois apresentaram saídas corretas conforme os enunciados, e 7 foram consideradas incorretas: 2 por apresentarem o uso

[8]Replit - https://replit.com/ - acessado em 09/07/2023.

[9]GNU Gcc - https://gcc.gnu.org/ - acessado em 09/07/2023

[10]Ubuntu - https://ubuntu.com/blog/tag/22-04-lts - acessado em 09/07/2023

do *scanf* para leitura de *strings* e 5 por por apresentarem saídas incorretas em relação aos enunciados. Em relação às explicações, 5 respostas foram consideradas com a explicação incorreta por não seguiram as saídas esperadas pelos enunciados dos exercícios.

O tópico de matrizes foi testado com 6 exemplos, sendo que 5 apresentaram compilações e execuções, saídas de execuções e explicações corretas, tendo 80% de aproveitamento. Em 4 exemplos, o ChatGPT fez uso de funções para encapsular as lógicas das soluções, chegando a escrever 3 funções em um mesmo programa. Em 4 programas foram definidas constantes numéricas, que são definidas para valores que são inalteráveis durante a execução do programa [Mizrahi 2008].

De um total de 46 questões apresentadas para o ChatGPT, 100% foram resolvidas com sintaxe correta, compilando e executando; 38 questões, ou 82,6%, atenderam corretamente os requisitos dos enunciados, apresentando saídas corretas para as entradas; e, por fim, 40 questões, ou 86,9%, foram explicadas corretamente.

6. Discussão dos resultados

O aproveitamento geral do ChatGPT foi superior a 80%, sendo que para as questões dos temas mais iniciais, como variáveis e operadores e estruturas de decisão e repetição, o acerto foi de 100%. Nos exercícios que envolve manipulação de *strings*, que é um tema mais complexo, principalmente na linguagem C, o aproveitamento foi próximos dos 50%. Como visto na Seção 5, ao ser questionado a respeito de algum problema no código apresentado, geralmente o ChatGPT apresenta uma solução melhorada para o problema, porém, para um aluno iniciante em programação, é de se questionar se este teria essa percepção em relação à resposta obtida ou se simplesmente aceitaria como correta.

No tópico de vetores e matrizes, que é um tema de complexidade assim como a manipulação de *strings*, o aproveitamento foi de 80%. Em 4 respostas, a função *scanf*, considerada insegura para a leitura de cadeia de caracteres, foi usada para capturar a entrada digitado. Nessas situações, o aluno precisa ter conhecimento sobre esse problema para que possa substituir as funções por outras mais adequadas, inclusive podendo usar o próprio ChatGPT para obter essa correção no código. Porém, novamente, é questionável que um estudante iniciante teria esse conhecimento.

Em 16 respostas, foram usadas funções para agrupar e encapsular partes do código. Todas as funções foram nomeadas em inglês e seus nomes foram coesos em relação ao propósito de cada uma. Como funções não é um dos assuntos iniciais, é necessário que o aluno busque esse conhecimento para compreender as respostas fornecidas, alterando os códigos ou solicitando as alterações para adequar as soluções aos enunciados dos exercícios. É importante destacar que o ChatGPT foi treinado com bases de códigos disponíveis em repositórios da Internet, com todos os níveis de conhecimento, portanto, esse comportamento de criar funções pode ser considerado comum, dependendo do contexto e da complexidade do exercício proposto.

Um ponto importante é que, alguns erros cometidos pelo ChatGPT podem ser facilmente corrigidos, inclusive pelo próprio ChatGPT, porém demanda conhecimento prévio de quem recebe a resposta. Entretanto, para os erros apontados que, por exemplo, vão contra o enunciado do problema proposto, é preciso analisar o código gerado, o que demandaria ainda mais conhecimento prévio do estudante.

7. Conclusão

Este trabalho apresentou uma análise das respostas dadas pelo ChatGPT em relação a conteúdos iniciais de programação, considerando o ponto de vista de um estudante inexperiente em relação a esse assunto. Foram feitos testes qualitativos e quantitativos e foi possível compreender que, no momento atual de escrita deste artigo, o ChatGPT possui potencial como auxiliar no processo de aprendizagem de programação, porém suas respostas não podem ser consideradas totalmente corretas e demandam certo conhecimento prévio de quem o usa para analisá-las e aproveitá-las.

Referências

Aljanabi, M., Ghazi, M., Ali, A. H., Abed, S. A., and ChatGpt (2023). Chatgpt: Open possibilities. *Iraqi Journal For Computer Science and Mathematics*, 4(1):62–64.

Cámara, J., Troya, J., Burgueño, L., and Vallecillo, A. (2023). On the assessment of generative ai in modeling tasks: an experience report with chatgpt and uml. *Software and Systems Modeling*, 22(3):781–793.

du Boulay, J. B. H. (1986). Some difficulties of learning to program. *Journal of Educational Computing Research*, 2(1):57–73.

Finnie-Ansley, J., Denny, P., Becker, B. A., Luxton-Reilly, A., and Prather, J. (2022). The robots are coming: Exploring the implications of openai codex on introductory programming. ACE '22, page 10–19, New York, NY, USA. ACM.

Finnie-Ansley, J., Denny, P., Luxton-Reilly, A., Santos, E. A., Prather, J., and Becker, B. A. (2023). My ai wants to know if this will be on the exam: Testing openai's codex on cs2 programming exercises. In *Proceedings of the 25th Australasian Computing Education Conference*, ACE '23, page 97–104, New York, NY, USA. ACM.

Lee, M. (2023). A mathematical investigation of hallucination and creativity in gpt models. *Mathematics*, 11(10).

Li, Y., Choi, D., Chung, J., Kushman, N., Schrittwieser, J., Leblond, R., Eccles, T., Keeling, J., Gimeno, F., Lago, A. D., Hubert, T., Choy, P., de Masson d'Autume, C., Babuschkin, I., Chen, X., Huang, P.-S., Welbl, J., Gowal, S., Cherepanov, A., Molloy, J., Mankowitz, D. J., Robson, E. S., Kohli, P., de Freitas, N., Kavukcuoglu, K., and Vinyals, O. (2022). Competition-level code generation with alphacode. *Science*, 378(6624):1092–1097.

Lo, C. K. (2023). What is the impact of chatgpt on education? a rapid review of the literature. *Education Sciences*, 13(4).

MacNeil, S., Tran, A., Mogil, D., Bernstein, S., Ross, E., and Huang, Z. (2022). Generating diverse code explanations using the gpt-3 large language model. In *Proceedings of the 2022 ACM Conference on International Computing Education Research - Volume 2*, ICER '22, page 37–39, New York, NY, USA. Association for Computing Machinery.

Mizrahi, V. V. (2008). *Treinamento em Linguagem C*, volume 1. Person Prentice Hall.

OpenAI (2023). Chatgpt. `https://openai.com/chatgpt`. Acessado em 10/07/2023.

Sarsa, S., Denny, P., Hellas, A., and Leinonen, J. (2022). Automatic generation of programming exercises and code explanations using large language models. In *Proc. of the 2022 ACM Conf. on International Computing Education Research V.1*. ACM.

Tsai, M.-L., Ong, C. W., and Chen, C.-L. (2023). Exploring the use of large language models (llms) in chemical engineering education: Building core course problem models with chat-gpt. *Education for Chemical Engineers*, 44:71–95.