Integrantes:

Enzo Yokokura Araujo RM: 564177
Marco Antonio Ferreira Fonseca RM: 566434

Introdução:

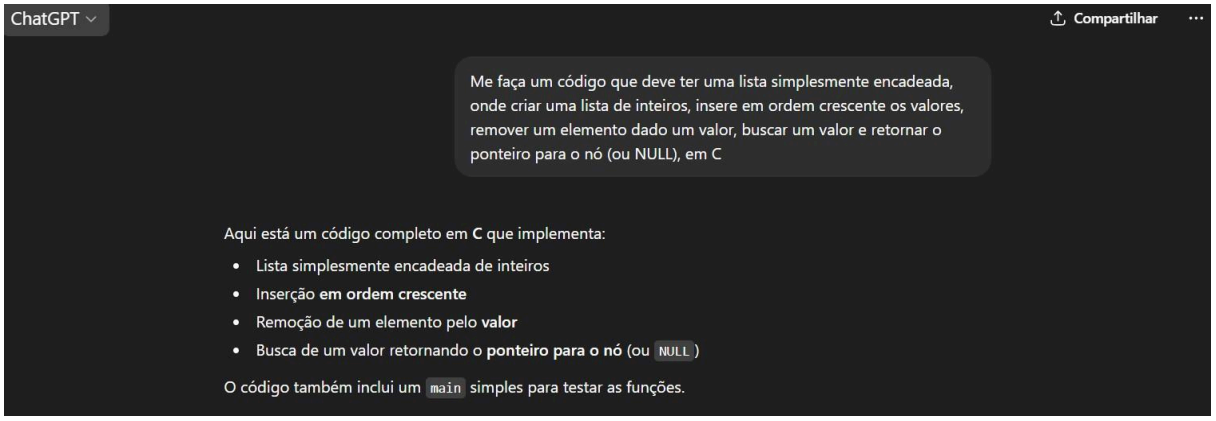
Usar ferramentas de "AI Companion" (como ChatGPT, Copilot, etc.) para resolver desafios de programação em C sobre estruturas de dados, e verifica em um mundo em que assistentes de IA escrevem, explicam e até otimizam códigos, assim utilizando elas como testes de eficiência e também verificando sua otimização

Metodologia de Uso da IA:

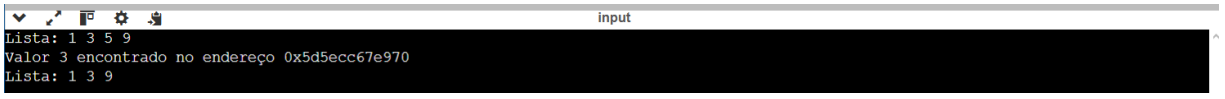
Foi utilizado a inteligência artificial da Open IA o ChatGPT, foram dadas as seguintes perguntas:

lista simplesmente encadeada:

"Me faça um código que deve ter uma lista simplesmente encadeada, onde criar uma lista de inteiros, insere em ordem crescente os valores, remover um elemento dado um valor, buscar um valor e retornar o ponteiro para o nó (ou NULL), em C "



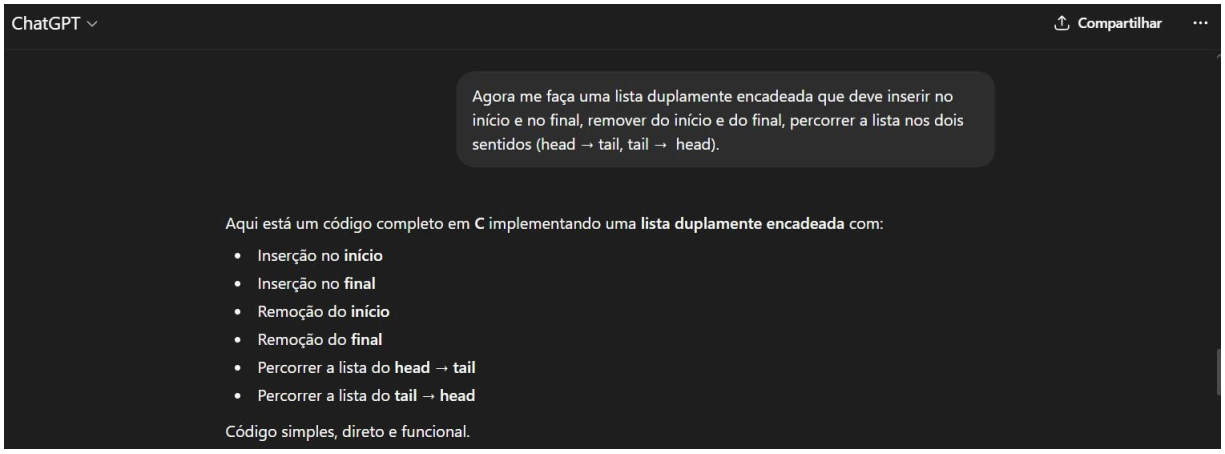
O código dado pela inteligência artificial deu a seguinte resposta:



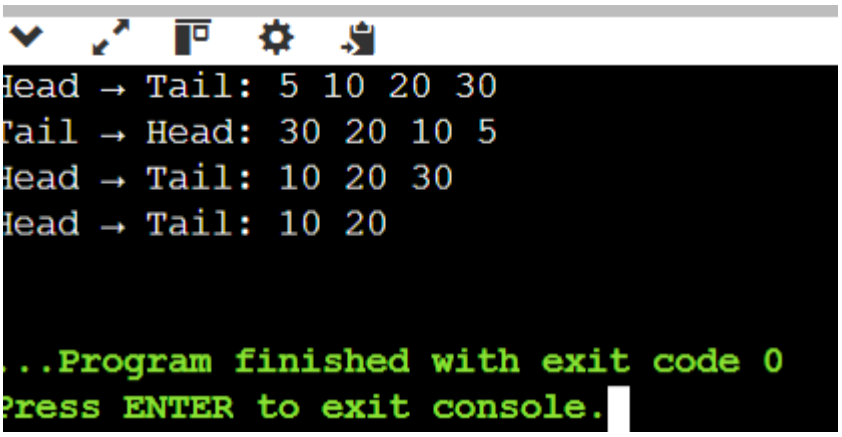
Em análise ele não deixou necessariamente o usuário sem acesso ao código das os valores, os valores serão sempre fixos no código e não maleáveis, para o usuário final utilizar sem mexer no código

Lista duplamente encadeada

"Agora me faça uma lista duplamente encadeada que deve inserir no início e no final,
remover do início e do final, percorrer a lista nos dois sentidos (head → tail, tail → head)"



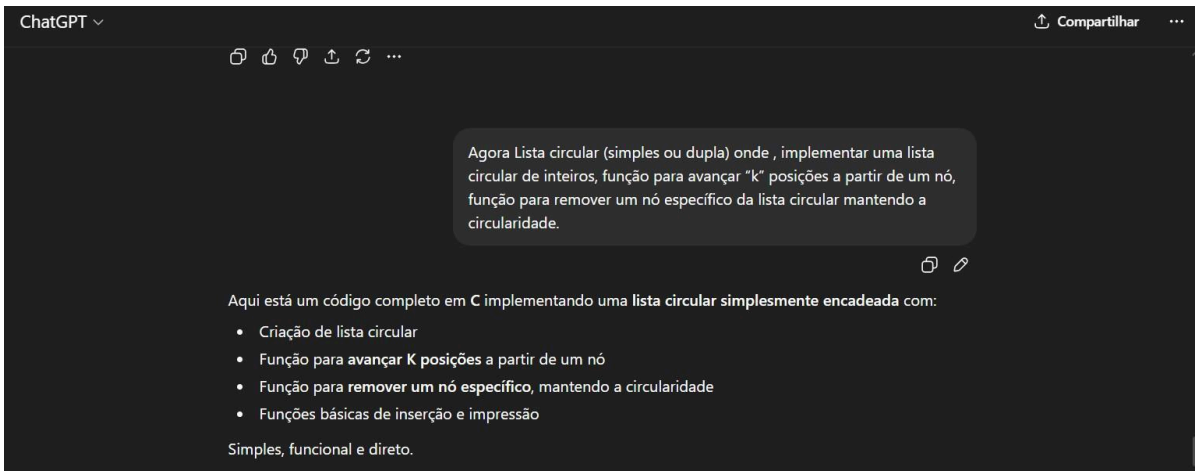
Resultado do código da inteligência artificial:



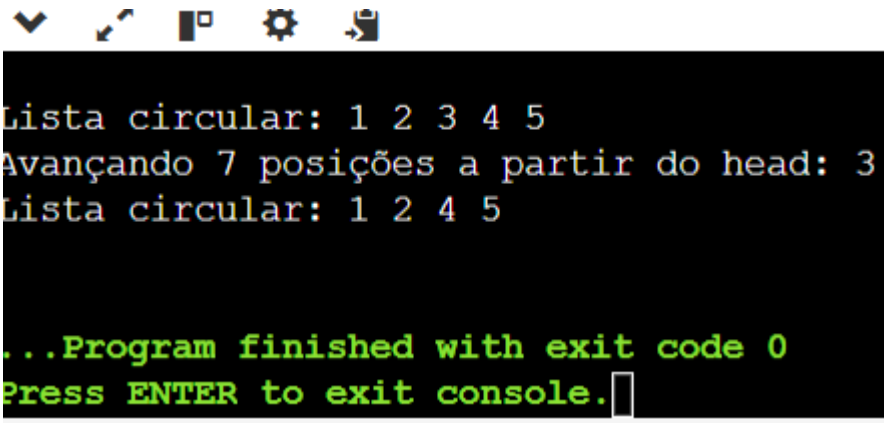
Em análise novamente ele não deixou necessariamente o usuário sem acesso ao código das os valores, os valores serão sempre fixos no código e não maleáveis, para o usuário final utilizar sem mexer no código.

Lista circular (simples ou dupla):

"Agora Lista circular (simples ou dupla) onde , implementar uma lista circular de inteiros, função para avançar "k" posições a partir de um nó, função para remover um nó específico da lista circular mantendo a circularidade."



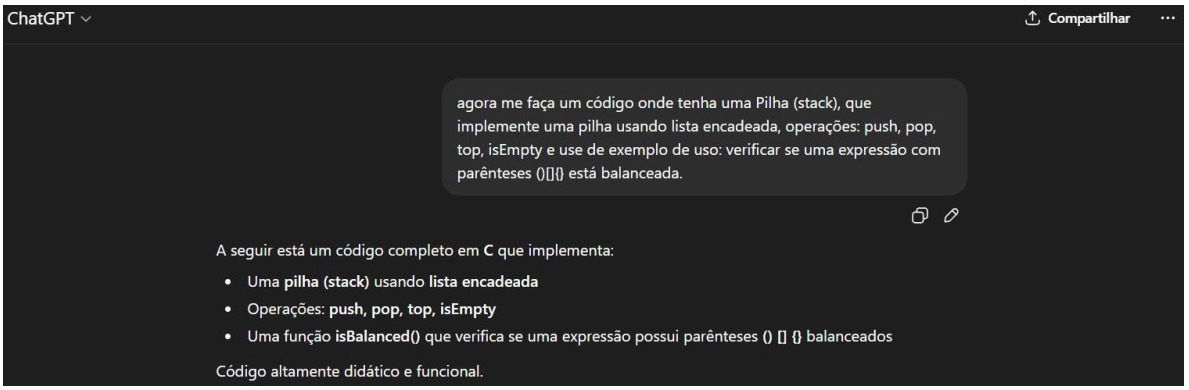
Resultado do código dado:



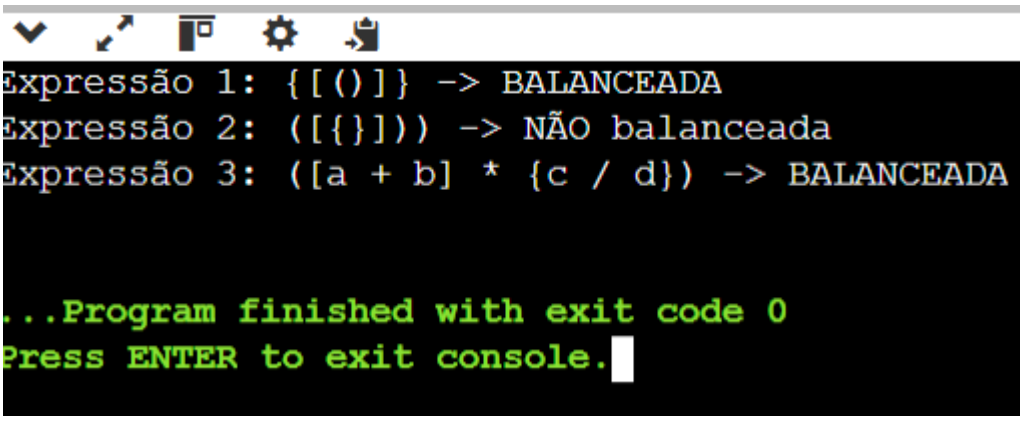
Em análise achamos o código novamente falho no requisito de deixar interativo com o usuário final, e um código pouco otimizado, contudo funcional.

Pilha (stack):

"agora me faça um código onde tenha uma Pilha (stack), que implemente uma pilha usando lista encadeada, operações: push, pop, top, isEmpty e use de exemplo de uso: verificar se uma expressão com parênteses ()[]{} está balanceada"



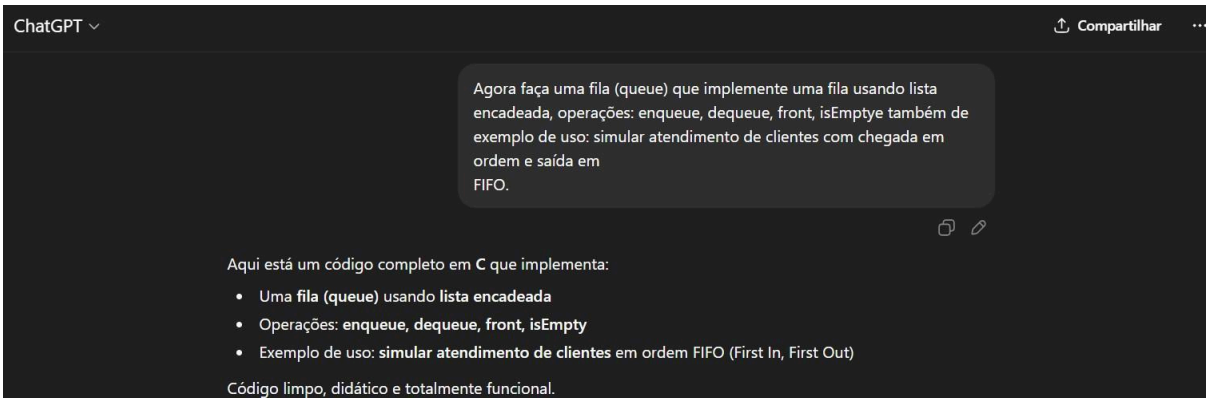
Resultado do código dado:



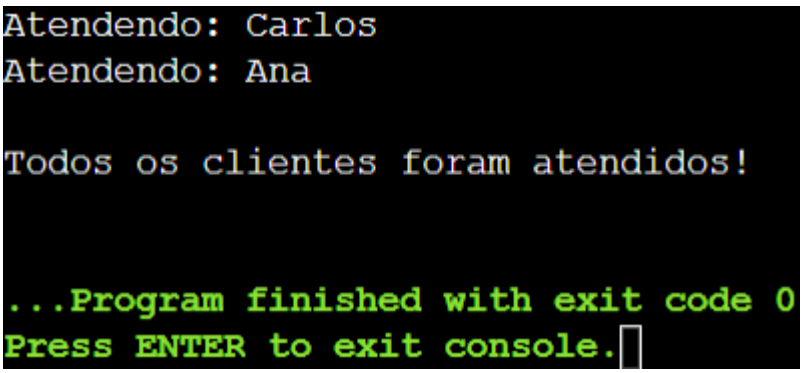
Em análise é possível verificar que o código é funcional, contudo, novamente sem integração dos dados do usuário final e mal otimizado, e simples

Fila (queue):

"Agora faça uma fila (queue) que implemente uma fila usando lista encadeada, operações: enqueue, dequeue, front, isEmptye também de exemplo de uso: simular atendimento de clientes com chegada em ordem e saída em FIFO. "



Resultado do código dado:



Em análise é possível verificar que o código é funcional, contudo, novamente sem integração dos dados do usuário final, e poucos dados utilizados.

Análise Final:

Para analisarmos de maneira mais coerente cada tópico dado pela iam iremos avaliar tópico por tópico.

A lista Simplesmente Encadeada Implementa uma lista de inteiros onde:

- Cada nó contém um valor e um ponteiro para o próximo.
- A função inserirOrdenado() insere valores já em ordem crescente.
- A função buscar() percorre a lista e retorna o ponteiro para o nó com o valor procurado.
- A função remover() remove um elemento específico pelo valor.
- Mantém comportamento padrão de listas simples: percorrimento linear e desalocação correta.

Lista Duplamente Encadeada (início/final e percursos bidirecionais)

Implementa uma lista onde:

- Cada nó aponta para o próximo e para o anterior.
- inserirInicio() adiciona um nó antes do head.
- inserirFim() adiciona um nó depois do tail.
- removerInicio() e removerFim() permitem remoção eficiente nas extremidades.
- É possível percorrer:
- head → tail
- tail → head

Lista Circular (avanço de K posições e remoção mantendo circularidade):

Implementa uma lista circular simples onde:

- O último nó aponta novamente para o head, formando um ciclo.
- avancarK() avança K posições partindo de qualquer nó (usando a circularidade).
- removerNo() exclui um nó específico mantendo o comportamento circular.
- A circularidade é preservada mesmo após remoções difíceis, como o

head. Pilha (Stack) com Lista Encadeada + Verificação de

Parênteses: Implementa uma pilha ligada (LIFO), com:

- push() → insere no topo

- pop() → remove do topo
- top() → consulta o topo sem remover
- isEmpty() → verifica se está vazia

Aplicação prática:

- isBalanced() verifica se uma expressão contém parênteses, colchetes e chaves balanceados.
- Utiliza a pilha para validar aberturas e fechamentos.

Fila (Queue) com Lista Encadeada + Simulação de Atendimento FIFO:

Implementa uma fila ligada com:

- enqueue() → insere no final
- dequeue() → remove do início
- front() → consulta o primeiro
- isEmpty() → verifica se está vazia

Exemplo real:

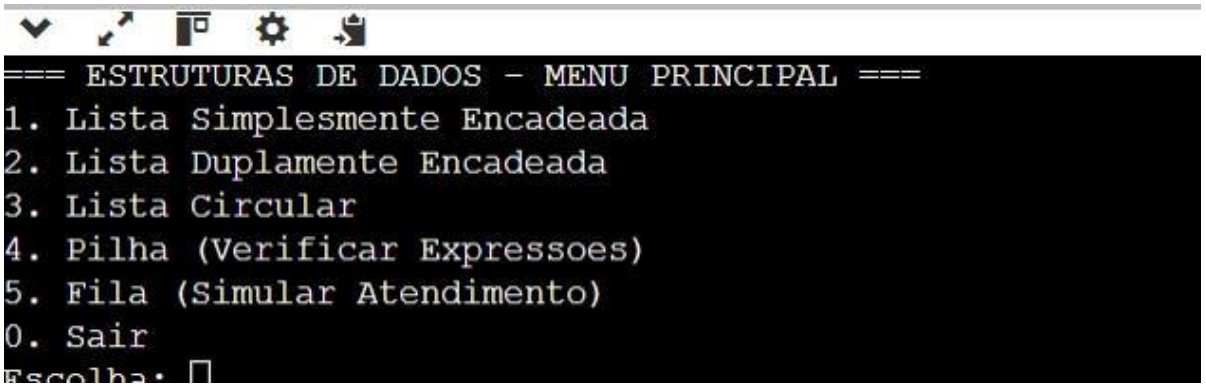
- Simulação de atendimento a clientes em ordem FIFO (First In, First Out).
- Cada cliente chega e entra na fila; o primeiro a chegar é o primeiro a ser atendido.

Agora deixando mais simples de entendimento:

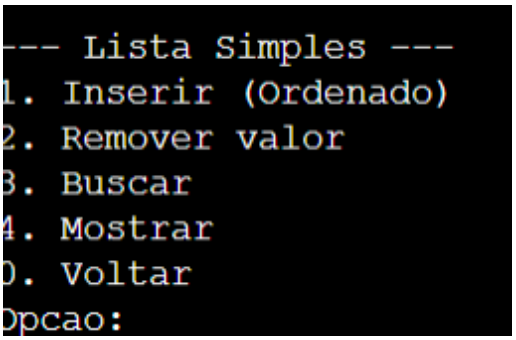
- Lista simples: inserção ordenada, busca e remoção.
- Lista dupla: inserção/remoção nas extremidades e percursos bidirecionais.
- Lista circular: navegação contínua (K passos) e remoção mantendo ciclo.
- Pilha: operações LIFO com aplicação para validar parênteses.
- Fila: operações FIFO com simulação real de atendimento.

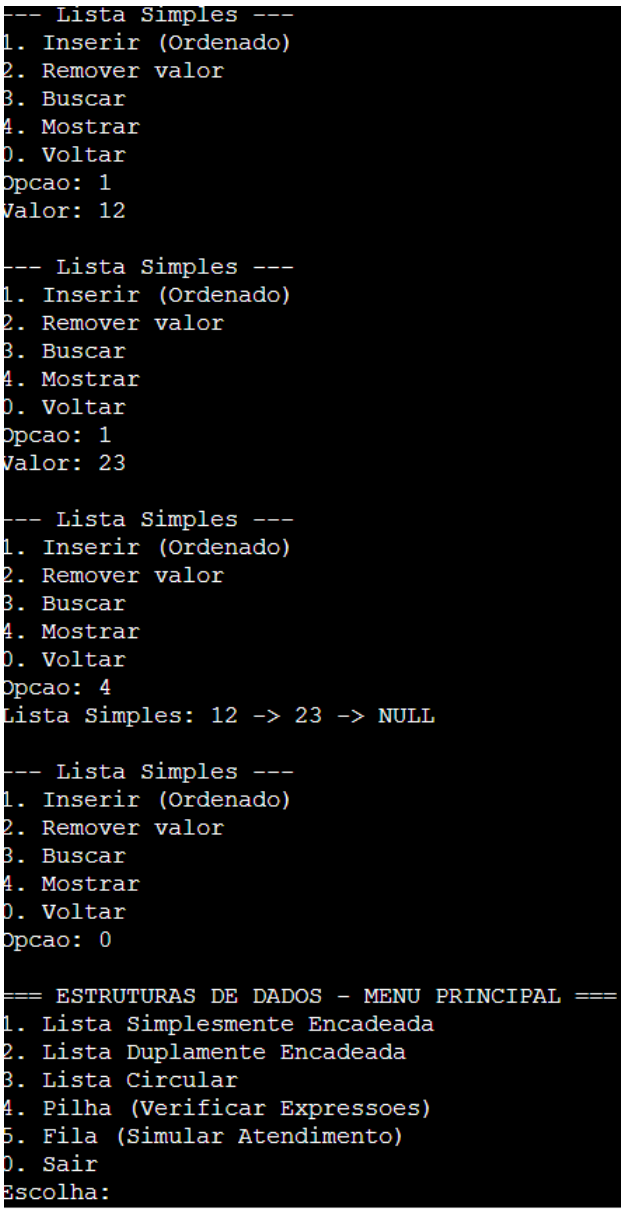
Resolução do Grupo e Código utilizado

O grupo decidiu reutilizar algumas partes do código da IA, contudo mais otimizado e principalmente decidiu englobar todos os pedidos em um só código e deixando que o usuário final controle, como um painel de controle, segue imagem abaixo:



Cada lista, pilha ou fila, tem acesso e distribuição dos valores que o usuário dar, deixando para utilidade do próprio usuário final alguns exemplos:





Para não ocupar muito espaço deixarei o arquivo em anexo de todos os códigos utilizados, como o código final do grupo.

Conclusões Finais

É possível concluir que a utilização das inteligências artificiais é bem válida, contudo, muita das vezes possui erros "bobos" ou má otimizado, é muito bom para poupar tempo, mas longe de substituir a capacidade humana atual de otimização, ou seja perfeito como ferramenta, contudo inadmissível substituição humana atualmente, talvez no futuro com um bom treinamento de um vasto banco de dados mais otimizado talvez substitua