Instituto De Ciências Matemáticas e de Computação

Departamento de Ciências de Computação

SCC0220 - Laboratório de Introdução à Ciência da Computação II

Turma BCC – Prof. Jean Roberto Ponciano

*Entrega 01*

**Nome: Christyan Paniago Nantes – N° USP: 15635906**

**Data: 22/08/2025**

**Esse template foi montado por: Christyan Paniago Nantes**

## Descrição da atividade:

Resolver o problema de verificar palíndromos de 3 formas, iterativa, recursiva e iterativa feita por inteligência artificial. Na versão da IA. deve-se melhorar ou piorar o desempenho do código gerado pela IA.

No relatório deve estar presente:

- As 3 versões do programa.

- Diferenças das versões iterativas.

- Analise de eficiência (recursiva vs iterativa && iterativa vs iterativa IA)

- Analise de simplicidade

- Diferenças da análise de requisitos da IA vs pessoal.

## Iterativo e Recursivo (Feito Por Mim):



## Iterativo (Feito Pela IA):

**Modelo**: Chat Gpt 5-auto

**Link do chat**: <https://chatgpt.com/s/t_68a8bdcdf8248191b12cc09d554881f9>

**Prompt**: Entregável 01 – Palíndromos

Você consegue ler um texto que é igual tanto se você o ler da direita para a esquerda quanto da esquerda para a direita? Se sim, você está diante de um palíndromo! Exemplos incluem: “asa” e“socorram-me, subi no onibus em Marrocos”. Neste primeiro entregável, você deve providenciar três versões de um programa em C que receba uma string qualquer e imprima na saída “Sim”, se a string for palíndromo, ou “Não”, caso contrário. O programa só deve considerar os caracteres alfanuméricos da string dada, ou seja, deve ignorar espaços, pontos, vírgulas, entre outros caracteres especiais. Letras maiúsculas e minúsculas não devem ser consideradas diferentes. Para verificar se uma string é palíndromo: 1. Veja se o primeiro caractere é igual ao último; 2. Se forem iguais, veja se a substring gerada após remover o primeiro e o último caracteres é um palíndromo.

Escreva o código para resolver o problema dos palidromos, somente a versao iterativa. padrão c, e deve receber varias strings até o fim do programa

**Código**:



## Análises:

**Diferenças das versões iterativas:**

Na minha versão eu implementei de forma separada a limpeza da string e a verificação de palíndromo. A IA agrupou ambas etapas e realizou a mesma ideia de two pointers, mas comparava apenas os pointers válidos (alfabeto ou número).

Isso complica a leitura do código, mas elimina a necessidade de uma string alternativa (utilizada no meu código) e um processo extra, o que pode salvar uma mínima quantidade de tempo.

Na versão da IA foram utilizadas variáveis unsigned char e size\_t, o tecnicamente são mais corretas , visto que as funções da ctype.h só garantem retorno estável se receberam um unsigned char (se forem fora do range do ascii, podem dar erro) e o size\_t é o tamanho de uma string (pois pode estourar um int).

Mas no escopo do trabalho, era reconhecido que não teria esses caracteres inválidos e que a string não seria grande, então se torna algo redundante que só complica a leitura do código.

Também a IA utilizou int = {0,1} como booleano, quando eu utilizei a biblioteca stdbool, ambas funcionam normalmente, mas booleano facilita a leitura do código.

Além disso, a IA removeu o “/n”, que acredito não ser necessário, visto que o “/0” estaria presente antes e portanto a remoção se torna desnecessária.

**Analise de eficiência (recursiva vs iterativa && iterativa vs iterativa IA):**

A comparação da recursiva vs iterativa, se tem que a iterativa consumira menores recursos e será mais eficiente, visto que não existe o overhead das chamadas recursivas das funções (memória e chamadas). Além de que a versão iterativa se torna mais simples de visualizar mentalmente, visto que conseguimos visualizar o caminho inteiro, sem pensar nas diversas chamadas das funções.

A comparação da iterativa vs iterativa IA, se tem que a IA verificou mais rigorosamente os protótipos das funções da biblioteca, e extrapolou casos que poderiam gerar erros. Mas esses erros não aconteceriam no escopo do exercício.

No tempo de execução (pelo runcodes), o tempo se manteve o mesmo com mínimas variações (0.0001 seg), visto que os tamanhos das strings aparentam ser pequenas, assim não se demonstra a diferença dos algoritmos.