# Trabalho Prático 3
# Algoritmos e Estrutura de Dados 2

Geraldo Arthur Detomi, Rhayan de Sousa Barcelos, Rodrigo José

## 1 Introdução

No mundo atual, a comunicação é fundamental e ocorre de diversas formas, seja por meio de mensagens de texto, chamadas telefônicas, *e-mails* ou até mesmo mensagens codificadas. Um desses códigos amplamente reconhecidos é o código Morse - desenvolvido pelo inventor Samuel Morse, que revolucionou as comunicações no século XIX.

Nesse contexto, este trabalho propõe a criação de um programa capaz de converter mensagens em código Morse para mensagens de texto e vice-versa. Para alcançar esse objetivo, utilizamos uma estrutura de dados conhecida como árvore Trie. A árvore Trie é uma estrutura eficiente para armazenar e pesquisar conjuntos de palavras ou cadeias de caracteres. Seu nome, "Trie", deriva da palavra *"retrieval"*, que significa recuperação em inglês, pois uma das principais aplicações dessa estrutura é a recuperação eficiente de palavras ou prefixos.

A utilização da árvore Trie nesse projeto permite uma conversão eficiente e precisa das mensagens em código Morse para o alfabeto latino. A estrutura da árvore é construída de forma que cada caractere do código Morse seja representado por um caminho específico percorrendo os nós da árvore. Isso garante um processo de conversão ágil e preciso. No caso deste projeto, trata-se de uma Trie Binária, pois as chaves utilizadas no código Morse têm apenas duas opções: ponto ou traço.

## 2 Implementação

A implementação foi feita separando os algoritmos e os testes em arquivos .h (*header*) e .c, além de utilizar enumerações para definir os caracteres "VAZIO", "TRACO"e "PONTO"que são amplamente utilizados no programa. Essa abordagem visa tornar o código mais legível e organizado. Inicialmente, o primeiro desafio foi a criação da árvore, definindo a estrutura de cada nó. Cada nó contém um ponteiro que aponta para um nó à esquerda e outro que aponta para um nó à direita, além de uma variável *char* para armazenar a respectiva letra que o nó irá representar, e um vetor de caracteres de tamanho máximo 6 para armazenar

o código Morse correspondente ao caractere. Para simplificar a identificação do nó raiz da árvore, foi utilizado um *typedef* de nó chamado ArvoreMorse. Em seguida, foram criadas algumas funções básicas para inicializar a árvore (*node* *inicializa_arvore()) e criar um novo nó (*node* criar_no(*char* caracter)), e posteriormente para imprimir a árvore (*void* imprime_arvore_pre_ordem(ArvoreMorse *arvore_morse)), que percorre a árvore utilizando o caminhamento pré-ordem, imprimindo cada nó da árvore e sua respectiva equivalência em código Morse.

Para inserir os nós na árvore, foi criada uma função que segue o mesmo método de inserção da árvore Trie. A função correspondente (*void* inserir_letra(Arvore Morse *arvore_morse, *char* *cod_morse, *char* letra)) recebe uma *string* que contém uma palavra em código Morse. Ela recebe o nó raiz (*arvore_morse) como ponto de partida para a navegação entre os ponteiros, e lê caractere por caractere dessa *string*. Se for um traço, navega para o ponteiro à direita; se o nó for nulo, cria-se um novo nó correspondente. Porém, se for um ponto, navega para o nó à esquerda. Se o nó à esquerda for nulo, cria-se um novo nó e atribui-se um *enum* "VAZIO", que corresponde ao caractere '\0'. Ao percorrer todas as letras da *string*, significa que já se navegou até o nó desejado, então cria-se um novo nó no local correspondente e atribui-se a letra passada como argumento (*char* letra).

Para realizar as conversões, foi criada uma função que converte um código Morse para seu respectivo caractere. Ela funciona seguindo a mesma lógica para inserir um nó na árvore, percorrendo caractere por caractere da *string* e navegando a partir do nó raiz. Se o caractere for um traço, navega-se para o ponteiro atual da direita; se o caractere for um ponto, navega-se para o ponteiro atual da esquerda. Ao percorrer todos os caracteres da *string*, significa que o nó atual atingiu o nó desejado, então retorna-se o respectivo caractere armazenado no nó. Para converter um caractere para código Morse, é utilizada uma busca em pré-ordem que percorre a árvore inteira procurando pelo nó que contém o caractere requisitado, e retorna a *string* (*char* *cod_morse[5]) que armazena sua respectiva equivalência em código Morse. Utilizando essas duas funções mencionadas acima, foram implementadas duas funções que recebem uma *string* inteira, convertem-na caractere por caractere e a retorna convertida, utilizando as funções mencionadas acima. São elas: *char* converte_str_alfa_para_morse(*char* *str_recebida, ArvoreMorse *arvore_morse) e *char* *converte_str_morse_para_alfa( *char* *str_recebida, ArvoreMorse *arvore_morse).

O último passo foi definir o modo de leitura do arquivo e como utilizar o programa. O arquivo é lido linha por linha, ignorando linhas em branco. Cada linha lida é enviada para as funções responsáveis por converter a *string* de código Morse para alfanumérico e vice-versa. Essas funções retornam a *string* no formato desejado, substituindo espaços por '/' conforme solicitado no trabalho, no caso de uma conversão de alfanumérico para Morse. Além disso, durante as conversões, todas as letras maiúsculas são convertidas para minúsculas, garantindo

consistência no resultado final. Também foi criada uma função chamada (*void* desalocar_arvore(ArvoreMorse *arvore_morse)) que libera toda a memória gasta pela árvore, liberando todos os seus nós, para evitar vazamento de memória.

Foi definida a utilização do programa por meio de linhas de comando no terminal, recebendo argumentos com o nome do arquivo ou sua localização por extenso, além da ação que o programa deve realizar. Por exemplo: "./*main* -i"exibe os caracteres disponíveis na árvore Morse e seus respectivos códigos Morse; "./*main* input/arquivo_alfa.txt -a"converte um arquivo com texto em alfanumérico para Morse; "./*main* input/arquivo_morse.txt -m"converte um arquivo com texto em Morse para alfanumérico. Os resultados da conversão são exibidos no terminal e também é gerado um arquivo de saída na pasta '/*out*' do programa. Além disso, existem exemplos de imagens demonstrando como executar o programa, localizadas na pasta /documentação. O arquivo *README.md*, localizado na pasta raiz, fornece instruções detalhadas sobre como utilizar o programa de forma correta.

# 3 Resultados

A primeira vista, a tarefa de transformar mensagens em código Morse para texto alfabético e vice-versa pode parecer complexa. No entanto, graças ao programa desenvolvido, essa tarefa se tornou simples e acessível. Os resultados obtidos ao executar o programa foram satisfatórios, uma vez que ele cumpriu seus objetivos de forma correta. Todos os testes de codificação para código Morse e decodificação para texto alfabético foram bem-sucedidos. Além disso, o programa foi projetado para liberar adequadamente a memória alocada dinamicamente, evitando problemas de vazamento de memória.

Com precisão, o programa foi capaz de converter as mensagens de texto alfabético para código Morse e vice-versa, seguindo as regras estabelecidas pelo código Morse. Além disso, a funcionalidade de impressão da árvore Morse foi verificada e os resultados foram consistentes.

Portanto, é seguro afirmar que a implementação foi adequada e que o programa se tornou uma fonte confiável para situações que envolvem o uso do código Morse.

# 4 Conclusão

Por meio da implementação e análise do tipo de árvore adequado, assim como da aplicação do código Morse, este trabalho representou uma excelente oportunidade para aprofundar nosso entendimento sobre o funcionamento das árvores e exemplificar uma de suas inúmeras utilidades. O desafio de desenvolver um programa capaz de decodificar o código Morse nos proporcionou uma base sólida

e abrangente para lidar com outros cenários que exigem o uso dessa estrutura. O conhecimento adquirido é inestimável para nossa formação como programadores, pois aprimora nossa capacidade de resolver problemas e fortalece nossas habilidades de programação.

Durante a seleção e implementação do tipo adequado de árvore, pudemos compreender diversos aspectos relacionados às árvores. Isso envolveu um estudo detalhado das características de cada tipo, levando em consideração fatores como eficiência, complexidade e capacidade de armazenamento de dados, a fim de escolher a opção mais adequada para a situação. Além disso, a aplicação do código Morse exigiu a compreensão dos padrões associados a ele, pois foi necessário estabelecer a correspondência entre esses padrões e os símbolos alfanuméricos correspondentes.

Dessa forma, o trabalho foi executado com êxito, contribuindo para o desenvolvimento de nossas habilidades lógicas como programadores, além de estimular nossa capacidade de resolver problemas. Conseguimos compreender a estruturação e o funcionamento das árvores, bem como a conversão entre código Morse e símbolos alfanuméricos.

# 5 Bibliografia

CPLUSPLUS, 2023. Disponível em: https://cplusplus.com/reference/ Acesso em: 24 junho. 2023.