Instituto Politécnico do Cavado e Ave

Ano Letivo: 2024/2025

Trabalho Prático

Estruturas de Dados Avançados

Professor:

Luís Ferreira

Nº/Aluno:

31501 - Tomás Afonso Cerqueira Gomes

Data:

Março de 2025

Índice

[Introdução 4](#_Toc194153691)

[Estrutura do Código 5](#_Toc194153692)

[Antenas 6](#_Toc194153693)

[Ficheiro antenas.h 6](#_Toc194153694)

[Ficheiro antenas.c 6](#_Toc194153695)

[carregarAntenasDeFicheiro 6](#_Toc194153696)

[inserirAntena 7](#_Toc194153697)

[removerAntena 8](#_Toc194153698)

[limparLista 8](#_Toc194153699)

[listarAntenas 9](#_Toc194153700)

[Gestão dos efeitos nefastos 10](#_Toc194153701)

[Ficheiro efeitos.h 10](#_Toc194153702)

[Ficheiro efeitos.c 10](#_Toc194153703)

[calcularEfeito 10](#_Toc194153704)

[efeitoExiste 11](#_Toc194153705)

[adicionarEfeito 12](#_Toc194153706)

[deduzirEfeitosNefastos 12](#_Toc194153707)

[listarEfeitos 13](#_Toc194153708)

[limparEfeitos 13](#_Toc194153709)

[Makefile 14](#_Toc194153710)

[Ferramentas Utilizadas 15](#_Toc194153711)

[Doxygen 15](#_Toc194153712)

[Visual Studio Code 15](#_Toc194153713)

[Conclusão 16](#_Toc194153714)

[Bibliogafia 17](#_Toc194153715)

Índice de Figuras

[Figura 1 – Estrutura do Código 5](#_Toc194154974)

[Figura 2 – Estrutura Antena 6](#_Toc194154975)

[Figura 3 – carregarAntenasDeFIcheiro 7](#_Toc194154976)

[Figura 4 – inserirAntena 7](#_Toc194154977)

[Figura 5 – removerAntena 8](#_Toc194154978)

[Figura 6 – limparLista 9](#_Toc194154979)

[Figura 7 – listarAntenas 9](#_Toc194154980)

[Figura 8 – Estrutura Efeito 10](#_Toc194154981)

[Figura 9 – calcularEfeito 11](#_Toc194154982)

[Figura 10 – efeitoExiste 11](#_Toc194154983)

[Figura 11 – adicionarEfeito 12](#_Toc194154984)

[Figura 12 – deduzirEfeitosNefastos 12](#_Toc194154985)

[Figura 13 – listarEfeitos 13](#_Toc194154986)

[Figura 14 – limparEfeitos 13](#_Toc194154987)

[Figura 15 – Makefile 14](#_Toc194154988)

# Introdução

Este trabalho tem como objetivo desenvolver uma aplicação em linguagem C que utiliza estruturas de dados avançadas, especificamente listas ligadas, para gerir antenas e calcular os efeitos nefastos causados pela sua interação. O

A aplicação permite carregar as antenas a partir de um ficheiro de texto, armazená-las em uma lista ligada e depois calcular a localização de efeitos nefastos entre as antenas. Esses efeitos são causados quando duas antenas da mesma frequência estão alinhadas, e a distância entre elas é tal que resulta em um ponto de efeito. O programa também permite listar as antenas carregadas e os efeitos nefastos encontrados, exibindo-os de forma organizada na consola.

Este trabalho explora os conceitos de listas ligadas, alocação dinâmica de memória e manipulação de ficheiros, proporcionando uma solução eficiente para o problema proposto.

Link do repositório do github: <https://github.com/tomascerqueiraa/TP---EDA.git>

# Estrutura do Código

O código foi organizado da forma correta e clara, utilizando boas práticas de programação para garantir facilidade de manutenção e leitura. Para isso, dividi o projeto em várias partes e utilizei funções específicas para cada tarefa. A estrutura do código segue a organização por funcionalidades o que facilita a compreensão.

Uma imagem com texto, captura de ecrã, Tipo de letra

Os conteúdos gerados por IA poderão estar incorretos.

#### Figura 1 – Estrutura do Código

O código foi dividido em diferentes funções e ficheiros, permitindo que cada parte do programa tenha uma função bem definida. Isso facilita a manutenção e a expansão do código no futuro.

# Antenas

Os ficheiros antenas.c e antenas.h são responsáveis pela gestão das antenas no sistema, incluindo a criação, armazenamento e exibição das antenas. A estrutura do código foi organizada e desenvolvido para utilizar funções específicas para cada tarefa, promovendo a clareza e a reutilização do código.

## Ficheiro antenas.h

O ficheiro antenas.h contém a estrutura Antena e a declaração das funções que manipulam essas antenas.

A estrutura Antena é o núcleo principal, pois representa uma antena no sistema. Contém um campo para armazenar a frequência e outros dois para armazenar as coordenadas de localização dessa antena (x e y), por outro lado ainda tem um outro campo que é um apontador para o próximo da lista, permitindo a ligação entre as antenas.

Este ficheiro têm também declaradas todas as funções para gerir da melhor forma todas as antenas.

Uma imagem com texto, captura de ecrã, Tipo de letra

Os conteúdos gerados por IA poderão estar incorretos.

#### Figura 2 – Estrutura Antena

## Ficheiro antenas.c

O ficheiro antenas.c contém a implementação das funções declaradas em antenas.h.

### carregarAntenasDeFicheiro

Esta função tem como objetivo carregar as antenas a partir de um ficheiro de texto e armazená-las numa lista ligada. A função lê cada linha do ficheiro e verifica cada caractere. Os outros caracteres são considerados frequências de antenas e são inseridos na lista. As coordenadas (x, y) correspondem à posição na linha e na coluna, respetivamente.

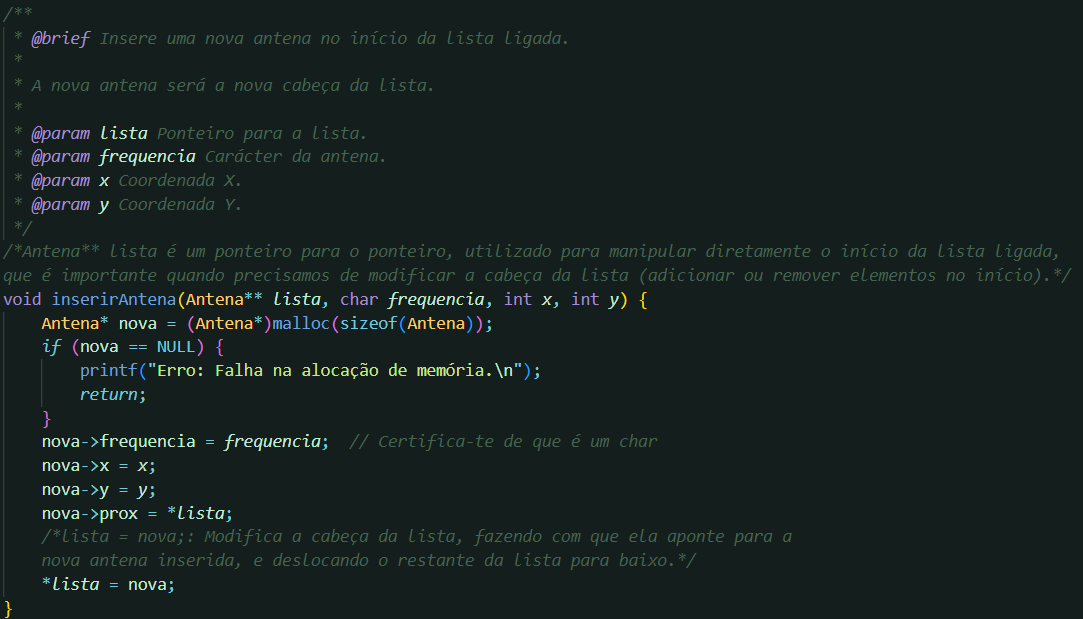
Uma imagem com texto, captura de ecrã, software

Os conteúdos gerados por IA poderão estar incorretos.

#### Figura 3 – carregarAntenasDeFIcheiro

### inserirAntena

Esta função cria e insere uma nova antena no início da lista ligada. Ao ser chamada, ela aloca memória para a nova antena, preenche os dados (frequência e coordenadas) e insere esta no início da lista. O ponteiro da lista é redirecionado para apontar para a nova antena, fazendo dela a cabeça da lista.



#### Figura 4 – inserirAntena

### removerAntena

Esta função remove uma antena da lista ligada com base nas coordenadas fornecidas. Ao ser chamada, percorre a lista ligada à procura de uma antena com as coordenadas especificadas. Se a antena for encontrada, a função ajusta os ponteiros da lista para remover a antena e liberta a memória alocada. Se a antena não for encontrada, a função não faz nada. A função retorna 1 se a antena foi removida com sucesso e 0 caso contrário.

Uma imagem com texto, captura de ecrã, software, Sistema operativo

Os conteúdos gerados por IA poderão estar incorretos.

#### Figura 5 – removerAntena

### limparLista

Esta função percorre toda a lista de antenas e limpa a memória associada a cada antena. Para evitar lixo na memória, a função garante que todos os nós da lista sejam removidos e a memória alocada seja devidamente limpa.

Uma imagem com texto, captura de ecrã, Tipo de letra

Os conteúdos gerados por IA poderão estar incorretos.

#### Figura 6 – limparLista

### listarAntenas

A função listarAntenas imprime na consola uma tabela com as frequências e as coordenadas de cada antena armazenada na lista.

Uma imagem com texto, captura de ecrã, ecrã, software

Os conteúdos gerados por IA poderão estar incorretos.

#### Figura 7 – listarAntenas

# Gestão dos efeitos nefastos

A gestão dos efeitos nefastos foi desenvolvido de forma a detetar os efeitos causados pelas antenas que estão alinhadas no plano 2D. Esses efeitos nefastos são identificados quando duas antenas da mesma frequência estão alinhadas (na mesma linha, coluna ou diagonal) e uma delas está ao dobro da distância da outra, o que gera um ponto de efeito.

A gestão dos efeitos nefastos foi organizada nos ficheiros efeitos.c e efeitos.h, onde a lógica de cálculo e armazenamento dos efeitos foi implementada.

## Ficheiro efeitos.h

O ficheiro efeitos.h contém a declaração da estrutura Efeito e a declaração das funções relacionadas à deteção e gestão dos efeitos nefastos.

Uma imagem com texto, captura de ecrã, Tipo de letra

Os conteúdos gerados por IA poderão estar incorretos.

#### Figura 8 – Estrutura Efeito

## Ficheiro efeitos.c

O ficheiro efeitos.c contém a implementação das funções declaradas em efeitos.h. Trata da lógica principal para a obtenção dos efeitos nefastos entre as antenas e do armazenamento desses efeitos numa lista ligada.

### calcularEfeito

A função calcularEfeito tem como objetivo verificar se duas antenas da mesma frequência estão alinhadas e se a distância entre elas é tal que gera um efeito nefasto. O cálculo verifica se as antenas estão alinhadas (na mesma linha, coluna ou diagonal) e se a distância entre elas é o dobro. Caso isso seja verdade, calcula a localização do efeito.

Uma imagem com texto, captura de ecrã, software

Os conteúdos gerados por IA poderão estar incorretos.

#### Figura 9 – calcularEfeito

### efeitoExiste

A função efeitoExiste percorre a lista de efeitos para verificar se já existe um efeito com as coordenadas fornecidas. Se o efeito já existir, a função retorna 1, caso contrário, retorna 0. Isso é útil para garantir que não haja duplicação de efeitos nefastos na lista.

Uma imagem com texto, captura de ecrã, Tipo de letra

Os conteúdos gerados por IA poderão estar incorretos.

#### Figura 10 – efeitoExiste

### adicionarEfeito

A função adicionarEfeito tem como objetivo adicionar um novo efeito à lista de efeitos nefastos. Esta realiza a inserção de um novo nó na lista ligada de efeitos, sendo que antes de adicionar o efeito, é feita uma verificação para garantir que não há duplicação de efeitos. Se o efeito já existir na lista, a função não faz nenhuma alteração. Caso contrário, cria-se um novo nó com as coordenadas fornecidas e o insere-se no início da lista.

Uma imagem com texto, captura de ecrã, software

Os conteúdos gerados por IA poderão estar incorretos.

#### Figura 11 – adicionarEfeito

### deduzirEfeitosNefastos

A função deduzirEfeitosNefastos percorre todas as antenas da lista, comparando cada par de antenas. Para cada par, a função chama calcularEfeito para verificar se há um efeito nefasto entre elas. Quando um efeito é detetado, este é adicionado à lista de efeitos com a função adicionarEfeito.

Uma imagem com texto, captura de ecrã, software

Os conteúdos gerados por IA poderão estar incorretos.

#### Figura 12 – deduzirEfeitosNefastos

### listarEfeitos

A função listarEfeitos imprime no terminal a lista de efeitos nefastos. Exibe as coordenadas (x, y) de cada efeito encontrado, mostrando-os de forma tabular, facilitando a visualização dos resultados.

Uma imagem com texto, captura de ecrã, software

Os conteúdos gerados por IA poderão estar incorretos.

#### Figura 13 – listarEfeitos

### limparEfeitos

Esta função é responsável por limpar a memória alocada na lista de efeitos. Percorre a lista de efeitos e, para cada nó, utiliza o “free” para esvaziar a memória.

Uma imagem com texto, captura de ecrã, Tipo de letra, software

Os conteúdos gerados por IA poderão estar incorretos.

#### Figura 14 – limparEfeitos

# Makefile

O Makefile é uma ferramenta essencial para automatizar o processo de compilação e execução do projeto. Ao utilizar o make, conseguimos definir um conjunto de regras para garantir que o código seja compilado corretamente e de forma eficiente, sem a necessidade de realizar manualmente cada etapa da compilação.

Uma imagem com texto, captura de ecrã, software, Tipo de letra

Os conteúdos gerados por IA poderão estar incorretos.

#### Figura 15 – Makefile

Quando escrito make no terminal, o programa irá ser compilado, por sua vez se fizermos make run, o terminal irá executar o programa, ainda com mais algumas funcionalidades embutidas quando escrito make clean os ficheiros gerados quando ouve a compilação iram desaparecer. Para gerar a documentação automática do código basta utilizar o comando make doc que isso irá acontecer.

# Ferramentas Utilizadas

Durante o desenvolvimento deste projeto, foram utilizadas duas ferramentas principais para facilitar a codificação, geração de documentação e organização do trabalho: Doxygen e Visual Studio Code. Ambas as ferramentas desempenharam um papel fundamental na organização do código, rapidez e produtividade do mesmo.

## Doxygen

O Doxygen é uma ferramenta de geração automática de documentação que lê os comentários no código-fonte e cria documentação de forma automática. Facilita a criação de documentação sem a necessidade de escrever manualmente sobre o funcionamento das funções e estruturas de dados.

## Visual Studio Code

O VS Code foi utilizado como principal ambiente de desenvolvimento para escrever e editar o código C. O VS Code permite a execução de comandos diretamente no terminal integrado, como o make, o que facilita a compilação do código, execução do programa e geração de documentação sem sair do editor.

# Conclusão

Apesar dessas dificuldades, o projeto foi concluído com sucesso, e as soluções desenvolvidas respondem em conformidade aos problemas propostos.

Toda a lógica para calcular os efeitos nefastos entre antenas apresentou desafios, principalmente para garantir que a distância entre elas fosse corretamente calculada e que as antenas estivessem realmente alinhadas.

Este projeto proporcionou uma boa experiência de aprendizagem sobre estruturas de dados avançadas e gestão de memória em C, além de me permitir melhorar as minhas capacidades com a programação e na utilização de ferramentas de documentação e compilação.

# Bibliogafia

Chat gpt – <https://chatgpt.com/>

Gemini - <https://gemini.google.com/app>