IPCA

Relatório Fase 1 e 2 – Estruturas de Dados Avançadas

Estruturas de Dados Avançadas

Telmo Silva | A20456

Índice

Agradecimentos

A concretização deste projeto foi possível graças ao conhecimento adquirido ao longo da unidade curricular de **Estruturas de Dados Avançadas**, bem como ao apoio e orientação prestados pelos docentes responsáveis pela sua lecionação.

A todos os que, direta ou indiretamente, contribuíram para a realização deste trabalho — colegas, professores, ferramentas tecnológicas e recursos de apoio — expressa-se o mais sincero e reconhecido agradecimento.

Resumo

O presente trabalho de licenciatura visa a aplicação prática dos conhecimentos adquiridos na unidade curricular de **Estruturas de Dados Avançadas**, focando-se na implementação de uma solução modular para a gestão de antenas e cálculo de efeitos nefastos.

Após uma revisão teórica e análise de soluções existentes, desenvolveu-se uma aplicação em linguagem **C**, que permite a manipulação dinâmica de listas ligadas, a leitura de ficheiros de configuração, a identificação de localizações críticas e a visualização de informações numa grelha textual interativa. Na segunda fase, a estrutura foi expandida com **grafos**, representando relações entre antenas e permitindo a aplicação de algoritmos de procura (DFS, BFS), análise de caminhos e interseções de frequência.

Conclui-se que a solução proposta cumpre os requisitos estipulados, demonstra uma abordagem modular e eficiente, e proporciona uma base sólida para evoluções futuras, tanto académicas como práticas.

1. Introdução

O presente capítulo introduz o contexto, motivação, objetivos e metodologia do projeto desenvolvido no âmbito da unidade curricular de **Estruturas de Dados Avançadas** (EDA), inserida no 2.º semestre do 1.º ano da Licenciatura em Engenharia de Sistemas Informáticos.

Este projeto visa aplicar de forma prática os conhecimentos adquiridos ao longo da unidade curricular, nomeadamente no que diz respeito à utilização de estruturas de dados dinâmicas, leitura e escrita de ficheiros, modularização de código e geração de documentação técnica. Através do desenvolvimento de uma aplicação em linguagem C, o aluno teve oportunidade de consolidar competências fundamentais da programação estruturada e da engenharia de software.

A Fase 1 do projeto centrou-se na gestão de antenas através de **listas ligadas**, permitindo ao utilizador realizar operações como inserção, remoção, listagem e deteção de efeitos nefastos baseados na proximidade e frequência das antenas. A Fase 2 complementou este trabalho com a introdução de **estruturas de grafo**, modelando a interligação entre antenas e possibilitando a execução de algoritmos como **DFS (Depth-First Search)**, **BFS (Breadth-First Search)**, pesquisa de caminhos e deteção de interseções de frequências.

Este projeto permitiu desenvolver uma solução completa e modular, utilizando boas práticas de programação, documentação com Doxygen e testes funcionais com dados reais. O trabalho resultante está descrito neste relatório técnico e documentado num repositório Git de acesso público.

2. Estado da Arte

Este capítulo apresenta os conceitos fundamentais e tecnologias exploradas durante o desenvolvimento do projeto, bem como uma comparação com soluções técnicas existentes. A compreensão aprofundada destas abordagens foi essencial para fundamentar a escolha das estruturas de dados e algoritmos utilizados ao longo das Fases 1 e 2.

2.1 Conceitos e Fundamentos Teóricos

Durante a conceção do projeto, foi necessário estudar e aplicar vários conceitos fundamentais da área de estruturas de dados, com destaque para as **listas ligadas simples** e os **grafos não direcionados**.

As **listas ligadas** permitem a criação de sequências dinâmicas de elementos, onde cada nó aponta para o próximo. Esta estrutura revelou-se ideal para a Fase 1, dado que permite inserções e remoções eficientes de antenas, sem necessidade de realocação de memória.

Na Fase 2, recorreu-se à estrutura de **grafos**, representando antenas como vértices e as conexões entre elas como arestas. Os grafos foram implementados através de listas de adjacência, uma abordagem eficiente em termos de memória e desempenho. Para análise de conexões e acessibilidade entre antenas, aplicaram-se algoritmos clássicos como **DFS (Depth-First Search)** e **BFS (Breadth-First Search)**, fundamentais para percorrer e analisar redes.

Além das estruturas, foi necessário garantir a **modularização do código** em múltiplos ficheiros .c e .h, e a documentação clara com a ferramenta **Doxygen**, permitindo uma compreensão rápida das funcionalidades implementadas.

2.2 Soluções Técnicas Existentes

A resolução do problema proposto poderia recorrer a diversas alternativas técnicas, tais como **arrays dinâmicos**, **tabelas hash** ou até mesmo o uso de **bases de dados relacionais simples**. No entanto, estas abordagens apresentavam desvantagens para os requisitos do projeto.

Os **arrays dinâmicos**, embora rápidos para acesso aleatório, implicam realocação de memória em inserções/remoções frequentes, o que os torna menos eficientes neste contexto. As **tabelas hash**, apesar de rápidas na busca, são menos indicadas para representação de relações entre entidades.

Por outro lado, as **listas ligadas** ofereceram maior flexibilidade e simplicidade de implementação em C, tornando-se ideais para a gestão dinâmica das antenas. Quanto à representação das conexões, os **grafos com listas de adjacência** revelaram-se a solução mais adequada, dado o seu baixo custo de armazenamento e capacidade de representar múltiplas relações.

A escolha destas estruturas teve por base a eficiência, a clareza na implementação modular e a coerência com os objetivos da unidade curricular.

3. Trabalho Desenvolvido

Este capítulo descreve o trabalho realizado, incluindo a análise, especificação, estruturação do código e implementação das funcionalidades exigidas nas Fases 1 e 2.

3.1 Análise e Especificação

O projeto exigiu a criação de funcionalidades para gestão de antenas, leitura de ficheiros de configuração, deteção de interferências e, posteriormente, operações sobre grafos.

Requisitos Funcionais:

- Gerir dinamicamente antenas com base em listas ligadas;
- Carregar e guardar dados a partir de ficheiros de texto;
- Identificar localizações com efeito nefasto com base na proximidade de frequências iguais;
- Permitir visualização de dados na consola;
- Na Fase 2, aplicar conceitos de grafos para representar relações entre antenas;
- Implementar algoritmos de procura (DFS, BFS) e listagem de caminhos;

Arquitetura Lógica:

- O projeto foi modularizado em vários ficheiros (.c e .h);
- As estruturas de dados são isoladas em headers (antenas.h, grafo.h);
- As funcionalidades estão organizadas por fase e por tipo de operação.

Modelação:

- Lista ligada simples para armazenar antenas com campos de coordenadas e frequência;
- Segunda lista ligada para guardar localizações com efeito nefasto;
- Estrutura Grafo com vetor de vértices e listas de adjacências;

3.2 Implementação

A implementação foi feita em C, utilizando estruturas dinâmicas. Cada funcionalidade foi testada com diferentes dados e organizadas por módulos:

Funcionalidades da Fase 1:

- Inserção de antenas na lista ligada;
- Remoção de antenas com base em coordenadas;
- Leitura de ficheiro de configuração;
- Cálculo automático de efeitos nefastos;
- Visualização tabular e gráfica da grelha;

Funcionalidades da Fase 2:

- Implementação da estrutura Grafo e funções de criação e ligação de vértices;
- Algoritmos de procura: DFS e BFS;
- Listagem de todos os caminhos entre duas antenas;
- Deteção de interseções entre antenas com frequências distintas;
- Função para leitura automática de grelha de antenas e conversão para grafo;

4. Análise e Discussão de Resultados

Nesta secção apresentam-se os resultados obtidos com a execução das funcionalidades desenvolvidas na Fase 2. Foram utilizados dados do ficheiro 'antenas_base.txt' e realizadas operações de construção do grafo, procura em profundidade (DFS), procura em largura (BFS), listagem de todos os caminhos entre duas antenas e deteção de interseções de frequências distintas. As imagens seguintes mostram a saída real do programa com os testes efetuados.

```
Antena 0 (1,8) [O] -> 3 2 1
Antena 1 (2,5) [O] -> 3 2 0
Antena 2 (3,7) [O] -> 3 1 0
Antena 3 (4,4) [O] -> 2 1 0
Antena 4 (5,6) [A] -> 6 5
Antena 5 (8,8) [A] -> 6 4
Antena 6 (9,9) [A] -> 5 4
```

Figura 4.1 – Grafo carregado com antenas e ligações

```
(1,8) [O]
(4,4) [O]
(3,7) [O]
(2,5) [O]
```

Figura 4.2 – Resultado da DFS a partir da antena 0

```
(1,8) [O]
(4,4) [O]
(3,7) [O]
(2,5) [O]
```

Figura 4.3 – Resultado da BFS a partir da antena 0

```
(1,8) (4,4) (3,7)
(1,8) (4,4) (2,5) (3,7)
(1,8) (3,7)
(1,8) (2,5) (4,4) (3,7)
(1,8) (2,5) (3,7)
```

Figura 4.4 – Caminhos entre antenas 0 e 2

```
(1,8) [O] <-> (5,6) [A]
(1,8) [O] <-> (8,8) [A]
(1,8) [O] <-> (9,9) [A]
(2,5) [O] <-> (5,6) [A]
(2,5) [O] <-> (8,8) [A]
(2,5) [O] <-> (9,9) [A]
(3,7) [O] <-> (5,6) [A]
(3,7) [O] <-> (8,8) [A]
(3,7) [O] <-> (9,9) [A]
(4,4) [O] <-> (5,6) [A]
(4,4) [O] <-> (8,8) [A]
(4,4) [O] <-> (9,9) [A]
```

Figura 4.5 – Interseções entre frequências distintas (A e O)

5. Documentação Técnica com Doxygen

Foi utilizada a ferramenta Doxygen para gerar automaticamente a documentação técnica do projeto, com base nos comentários inseridos em cada módulo de código fonte. Foram documentadas todas as funções, estruturas de dados e cabeçalhos. A documentação permite navegar facilmente pelas funcionalidades e compreender a arquitetura do sistema.

As imagens seguintes ilustram partes da documentação gerada, incluindo a descrição do projeto, lista de funções, estruturas, e a página inicial.

Figura 5.1 – Página inicial do Doxygen

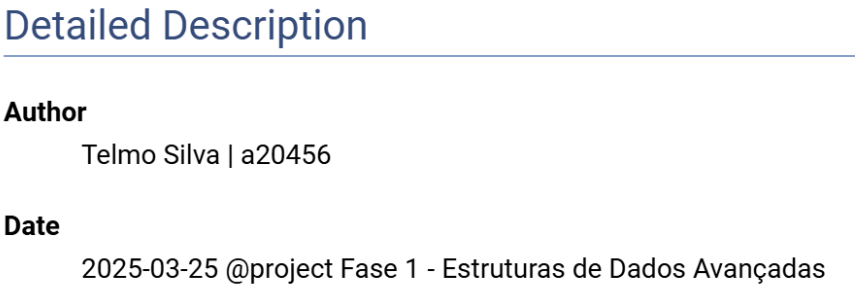


Figura 5.2 – Descrição geral do projeto

Here is a list of all functions with links to the files they belong to:

- a -

- atualizarMatrizComEfeitos() : **antenas.c**, **antenas.h**

- c -

- calcular_distancia() : **antenas.c**
- calcularEfeitoNefasto() : **antenas.c**, **antenas.h**
- carregarAntenasDeFicheiro() : **antenas.c**, **antenas.h**

- e -

- esta_alinhado() : **antenas.c**
- exibir_matriz_efeito_nefasto() : **antenas.c**, **antenas.h**
- exibir_matriz_posicionamento() : **antenas.c**, **antenas.h**
- exibir_menu() : **main.c**

- i -

- inicializar_lista() : **antenas.c**, **antenas.h**
- inicializar_matriz_efeito_nefasto() : **antenas.c**, **antenas.h**
- inicializarMatriz() : **antenas.c**, **antenas.h**
- inserir_antenas_iniciais() : **antenas.c**, **antenas.h**
- inserirAntena() : **antenas.c**, **antenas.h**

- j -

- jaExisteAntena() : **antenas.c**, **antenas.h**

Figura 5.3 – Lista de funções (parte 1)

- l -

- libertarListaAntenas() : **antenas.c**, **antenas.h**
- libertarListaEfeitos() : **antenas.c**, **antenas.h**
- libertarMatriz() : **antenas.c**, **antenas.h**
- limpar_tela() : **main.c**
- listarAntenas() : **antenas.c**, **antenas.h**
- listarEfeitos() : **antenas.c**, **antenas.h**

- m -

- main() : **main.c**, **main_sem_menu.c**
- marcarAntenasNaMatriz() : **antenas.c**, **antenas.h**

- r -

- removerAntena() : **antenas.c**, **antenas.h**

- s -

- salvar_antenas_arquivo() : **antenas.c**, **antenas.h**

- v -

- verificar_interferencia() : **antenas.c**, **antenas.h**

Figura 5.4 – Lista de funções (parte 2)

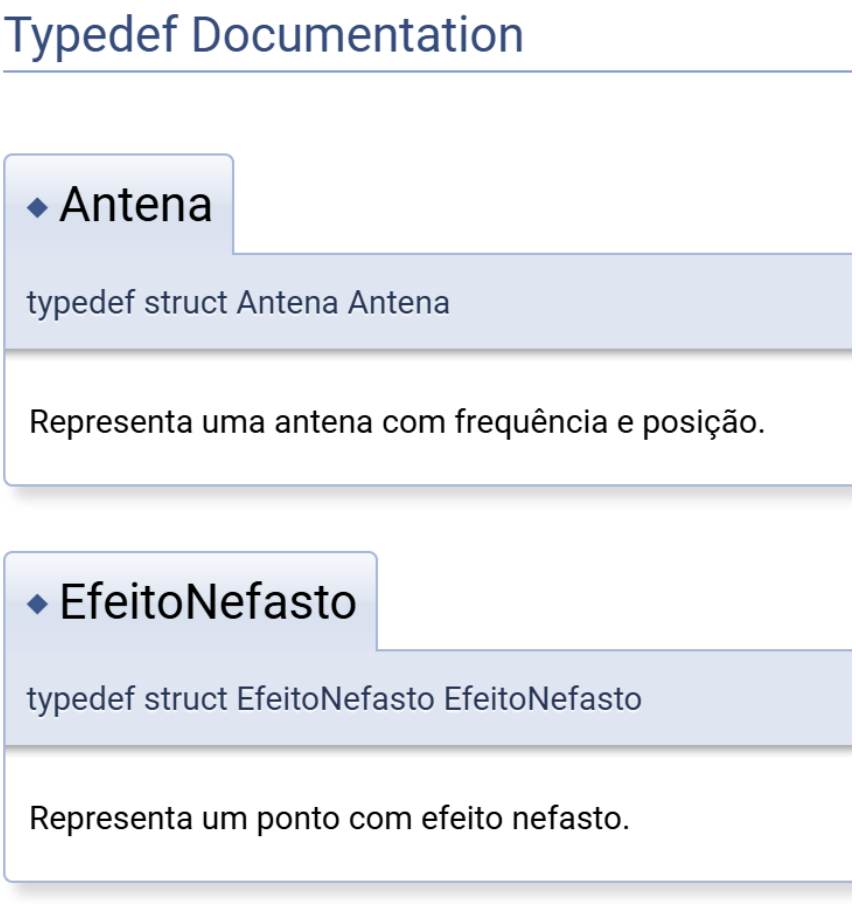


Figura 5.5 – Definições de estruturas de dados

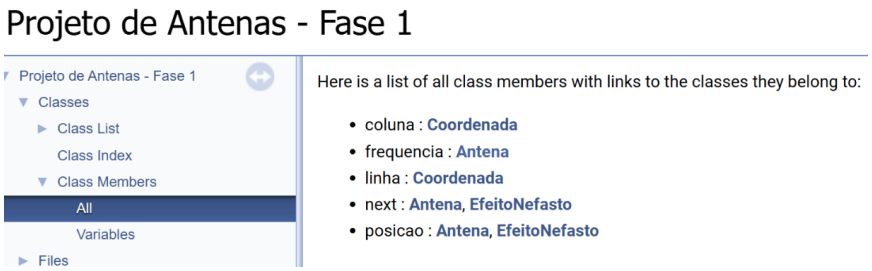
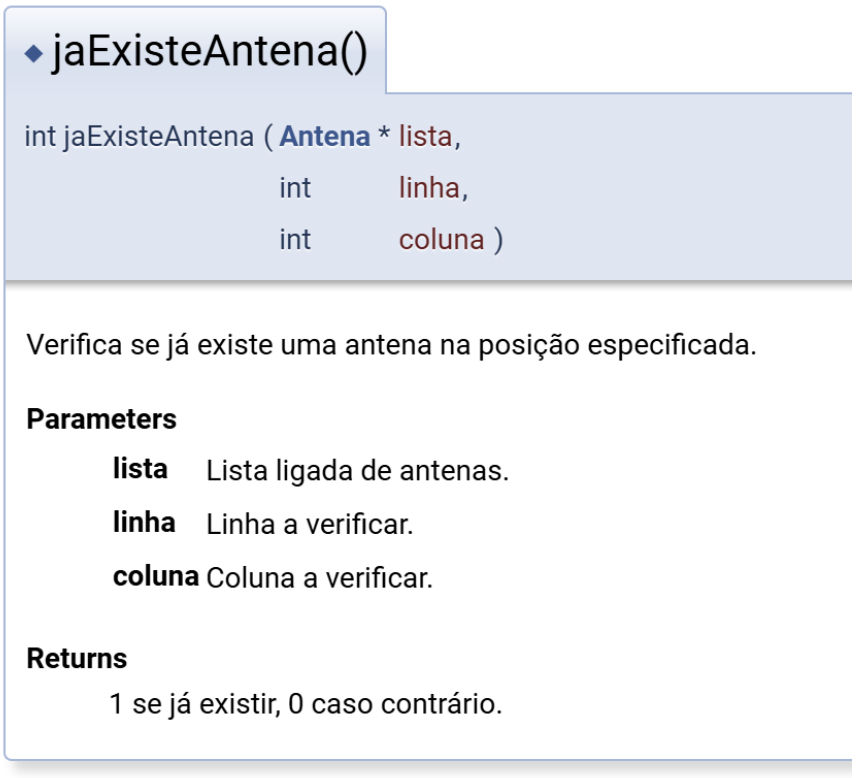


Figura 5.6 – Membros das estruturas



◆ jaExisteAntena()

```
int jaExisteAntena ( Antena * lista,
                     int     linha,
                     int     coluna )
```

Verifica se já existe uma antena na posição especificada.

Parameters

lista	Lista ligada de antenas.
linha	Linha a verificar.
coluna	Coluna a verificar.

Returns

1 se já existir, 0 caso contrário.

Figura 5.7 – Exemplo de função documentada

6. Conclusão

A realização do projeto de Estruturas de Dados Avançadas permitiu aplicar, de forma prática, os conceitos fundamentais abordados na unidade curricular. Durante a Fase 1, foram desenvolvidas funcionalidades com listas ligadas simples, leitura e escrita de ficheiros, e detecção de interferências entre antenas. Na Fase 2, introduziram-se estruturas mais complexas como grafos, com aplicação de algoritmos clássicos de procura e análise de conexões.

O código foi modularizado, bem estruturado e documentado com Doxygen, permitindo uma leitura e manutenção facilitadas. Foram realizados testes práticos com grelhas de antenas simuladas, validados com prints reais apresentados neste relatório.

Todas as funcionalidades solicitadas no enunciado foram implementadas, testadas e integradas num projeto único. O projeto final encontra-se compactado num ficheiro ZIP que inclui o código-fonte, documentação técnica e este relatório.

Repositório Git: https://github.com/TelmoSilva20456/TP_Fase1_2_EDA_A20456

7. Referências

FERREIRA, João. Estruturas de Dados em C. Lisboa: FCA Editora, 2021.

DOXYGEN. Doxygen – Documentation Generator Tool. Disponível em: https://www.doxygen.nl. Acesso em: maio de 2025.

OPENAI. Assistente ChatGPT para suporte técnico ao desenvolvimento do projeto. Disponível em: https://chat.openai.com.

8. Anexos

No ficheiro ZIP entregue encontram-se incluídos os seguintes elementos complementares ao relatório:

- Código-fonte completo (*.c e *.h);
- Ficheiro de configuração utilizado (antenas_base.txt);
- Documentação técnica gerada automaticamente (pasta /docs/html);
- Prints da execução para validação das funcionalidades;
- Relatório técnico em formato .docx e .pdf;
- Projeto completo também disponível no repositório Git referido na Conclusão.