|  |
| --- |
| **RELATÓRIO** PROJETO ESTRUTURA DE DADOS AVANÇADOS |

|  |
| --- |
| **Tiago Barroso Fontes 33222** |

|  |
| --- |
| Licenciatura em Engenharia de Sistemas Informáticos |

Índice

[Introdução 5](#_Toc194267748)

[1. Estruturas de dados 6](#_Toc194267749)

[1.1. Estrutura Antena 6](#_Toc194267750)

[1.2. Estrutura EfeitoNefasto 6](#_Toc194267751)

[2. Algoritmos e Funções Principais 7](#_Toc194267752)

[2.1. Criação de Antenas 7](#_Toc194267753)

[2.2. Carregamento dos dados 7](#_Toc194267754)

[2.3. Cálculo dos Efeitos Nefasto 8](#_Toc194267755)

[2.4. Gestão da Memória 9](#_Toc194267756)

[2.5. Função Principal 9](#_Toc194267757)

[3. Testes e Validações 10](#_Toc194267758)

[3.1. Resultados Obtidos 10](#_Toc194267759)

[GitHub 11](#_Toc194267760)

[Conclusão 12](#_Toc194267761)

Índice de Figuras

[Fig. 1 - Estrutura de dados Antena 6](#_Toc194268117)

[Fig. 2 - Estrutura de dados EfeitoNefasto 6](#_Toc194268118)

[Fig. 3 - Função de criação das Antenas 7](#_Toc194268119)

[Fig. 4 - Função de carregamento de dados 7](#_Toc194268120)

[Fig. 5 - Função de Calcular o Efeito Nefasto 8](#_Toc194268121)

[Fig. 6 - Funções da gestão da memória 9](#_Toc194268122)

[Fig. 7 - Função Main (Função Principal) 9](#_Toc194268123)

[Fig. 8 - Matriz utilizada no teste do programa 10](#_Toc194268124)

[Fig. 9 - Resultado obtido da matriz utilizada no teste do programa 10](#_Toc194268125)

Introdução

Este projeto desenvolve um sistema em C para análise de interferência entre antenas, utilizando a manipulação de estruturas de dados dinâmicas. O trabalho aborda um problema real de telecomunicações, onde se pretende identificar zonas de interferência em redes de antenas distribuídas numa matriz bidimensional.

A solução implementada aplica conceitos de estruturas de dados dinâmicas para resolver eficientemente o problema proposto. O desenvolvimento seguiu uma abordagem modular, garantindo código limpo e de fácil manutenção, com possibilidade de futuras expansões para estruturas mais complexas.

# Estruturas de dados

O sistema foi construído com base em duas estruturas de dados principais, a estrutura Antena e a estrutura EfeitoNefasto.

## Estrutura Antena

Cada elemento da lista de antenas contém as informações essenciais para o cálculo dos efeitos Nefasto:

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

Os conteúdos gerados por IA poderão estar incorretos.

Fig. 1 - Estrutura de dados Antena

As coordenadas (x,y) que representação a posição exata na matriz e o caracter que representa sua frequência de transmissão. O ponteiro para a próxima antena estabelece a ligação que forma a cadeia de elementos, permitindo o armazenamento dinâmico conforme novas antenas são adicionadas ao sistema.

## Estrutura EfeitoNefasto

Para cada ponto identificado com o efeito nefasto, o sistema armazena a coordenada. A ligação entre os diferentes pontos de interferência é mantida através de um apontador para o próximo elemento, criando assim uma lista independente.

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

Os conteúdos gerados por IA poderão estar incorretos.

Fig. 2 - Estrutura de dados EfeitoNefasto

# Algoritmos e Funções Principais

## Criação de Antenas

Esta função recebe como parâmetros as coordenadas (coluna, linha) e a frequência da antena, retornando um apontador para a estrutura criada. A função inclui verificação de erros na alocação de memória. É utilizada como função auxiliar durante o carregamento dos dados.

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

Os conteúdos gerados por IA poderão estar incorretos.

Fig. 3 - Função de criação das Antenas

## Carregamento dos dados

Função do sistema que realiza a leitura e interpretação do ficheiro de entrada. Processa o ficheiro linha a linha, identificando as posições das antenas e cria a lista ligada. Implementa o tratamento de erros para formatos.

Uma imagem com texto, captura de ecrã, software

Os conteúdos gerados por IA poderão estar incorretos.

Fig. 4 - Função de carregamento de dados

## Cálculo dos Efeitos Nefasto

Esta função analisa todas as combinações possíveis de pares de antenas para identificar pontos de interferência. Utiliza um algoritmo de comparação sistemática entre antenas da mesma frequência, aplicando as regras geométricas especificadas no enunciado para determinar as posições afetadas.

Uma imagem com texto, captura de ecrã, software

Os conteúdos gerados por IA poderão estar incorretos.

Fig. 5 - Função de Calcular o Efeito Nefasto

## Gestão da Memória

Funções complementares que ajudam na gestão de recursos, percorrendo as listas ligadas e libertando a memória alocada. São essenciais para prevenir memory leaks, especialmente em execuções contínuas ou com grandes volumes de dados.

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

Os conteúdos gerados por IA poderão estar incorretos.

Fig. 6 - Funções da gestão da memória

## Função Principal

A função executa o programa, seguindo uma sequência lógica de operações.

Uma imagem com texto, captura de ecrã

Os conteúdos gerados por IA poderão estar incorretos.

Fig. 7 - Função Main (Função Principal)

# Testes e Validações

Para validar o sistema, utilizamos como caso de teste uma matriz 12x12 contendo 14 antenas de diferentes frequências, posicionadas em padrões que permitissem testar diversos cenários de interferência.

Uma imagem com texto, captura de ecrã, preto e branco, Tipo de letra

Os conteúdos gerados por IA poderão estar incorretos.

Fig. 8 - Matriz utilizada no teste do programa

## Resultados Obtidos

O sistema processou corretamente o ficheiro de entrada, identificando todas as 14 antenas com as suas respetivas coordenadas e os efeitos nefastos, o sistema detetou 5 localizações críticas.

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

Os conteúdos gerados por IA poderão estar incorretos.

Fig. 9 - Resultado obtido da matriz utilizada no teste do programa

GitHub

https://github.com/Fontes9/Projeto-EDA

Conclusão

A implementação com listas ligadas é eficiente para gerir as antenas e calcular os efeitos nefastos. A estrutura dinâmica permitiu maior flexibilidade no armazenamento e processamento de dados, enquanto a modularização garantiu código organizado e manutenível.

Os conhecimentos adquiridos nessa etapa serão fundamentais para a transição para grafos, mostrando como estruturas de dados diferentes podem resolver o mesmo problema com abordagens diferentes.