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

Os conteúdos gerados por IA poderão estar incorretos.**

Relatório do Projeto de Avaliação de estruturas de dados avançadas (Eda)

Listas ligadas

Nome: Diogo António Apolinário Costa

Nº: 6731

Curso: Licenciatura Engenharia de Sistemas Informáticos – PL

Professor: Luís Ferreira

**13-04-2025**

# Índice

1. [Introdução 3](#_Toc195471222)

[2. Estado da Arte 4](#_Toc195471223)

[3. Análise e Especificação 5](#_Toc195471224)

[4. Implementação 8](#_Toc195471225)

[5. Conclusão 12](#_Toc195471226)

[6. Bibliografia 13](#_Toc195471227)

# Introdução

Este projeto de avaliação foi proposto no âmbito da unidade curricular Estruturas de Dados Avançados para melhorar os conhecimentos relativos à definição e manipulação de estruturas de dados dinâmicas na linguagem de programação C. E, para isso, pretende-se desenvolver um programa em C que considere uma cidade com várias antenas, em que cada antena é sintonizada numa frequência específica que, por sua vez, é indicada por um caracter em que, quando há pares de antenas com a mesma frequência, é aplicado um efeito nefasto em duas localizações: uma de cada lado das antenas. Ao invés, quando existem antenas com frequências diferentes (caracteres diferentes) não criam localizações com efeito nefasto.

O problema proposto envolve a utilização de listas ligadas para representar as antenas e os seus efeitos nefastos, implementando funcionalidades como inserção, remoção, cálculo de zonas nefastas e manipulação de dados em arquivos binários e de texto. Tudo Isto, usando estruturas de dados dinâmicos, o que tornou mais complicada a criação das funções, mormente das funções de mostrar a matriz no ecrã e das funções de guardar em ficheiro.

# 

# 2. Estado da Arte

Para este trabalho será necessário utilizar listas ligadas, uma vez que, à partida, não se sabe o número de antenas e nefastes que serão criadas.

Uma lista ligada é uma estrutura de dados linear e dinâmica. É composta por várias listas que estão interligadas através de apontadores, ou seja, cada lista possui um apontador que direciona para o endereço de memória da próxima lista. Deste modo, as listas da estrutura não precisam de estar em posições contíguas da memória. Isto faz com que a estrutura se torne dinâmica, pois, a qualquer momento, é possível alocar uma nova lista e mudar os apontadores das listas já existentes, de modo que a nova lista seja inserida na estrutura com êxito, na posição desejada pelo programador (Lista Ligada – Wikipédia, a Enciclopédia Livre).

# 3. Análise e especificação

O projeto consistiu na criação, em linguagem C, de uma estrutura de dados eficiente para armazenar informações sobre antenas, como sejam a sua localização e frequência. As antenas são representadas numa lista ligada, onde cada elemento contém dados sobre a posição e o tipo de antena. A análise envolveu, também, a definição de algoritmos para identificar as zonas nefastas e a interação entre elas.

Como mostrado na Figura 1, as localizações dos efeitos nefastos estão marcadas com o caracter #.

Uma imagem com captura de ecrã, padrão, Retângulo, quadrado

Os conteúdos gerados por IA poderão estar incorretos.

Figura 1 – Antenas e Nefastes

Para isso, foi necessário usar operações como inserção e remoção de antenas e a implementação de métodos para calcular as zonas afetadas.

A estrutura de dados principal utilizada foi a 'ANTENA', que contém informações sobre a localização (linha e coluna), a frequência de cada antena e um apontador (next) para ligar à próxima lista. Posteriormente, foram criadas as funções para cada objetivo (criar, inserir, mostrar, calcular zonas nefastas, ler e guardar ficheiros). Para trabalhar com cada antena (criar e inserir, remover, mostrar, guardar em ficheiro), foi necessário considerar sempre a ordem, tendo em conta a prioridade de uma linha em relação a uma coluna e, no caso da mesma linha ou coluna, por ordem crescente das mesmas. Depois de inserir todas as antenas, o programa percorre todas as listas para as comparar com o mesmo caracter (frequência) e, caso encontre, cria o efeito nefasto pelo algoritmo escolhido, criando e inserindo novamente no lugar correto das listas ligadas (novamente pela ordem das linhas e das colunas).

A Figura 2 representa um esquema sobre como as listas ficavam ordenadas, mesmo adicionando uma antena ou efeito nefasto, o que se tornou uma função importante, de forma a poder manipulá-las corretamente.

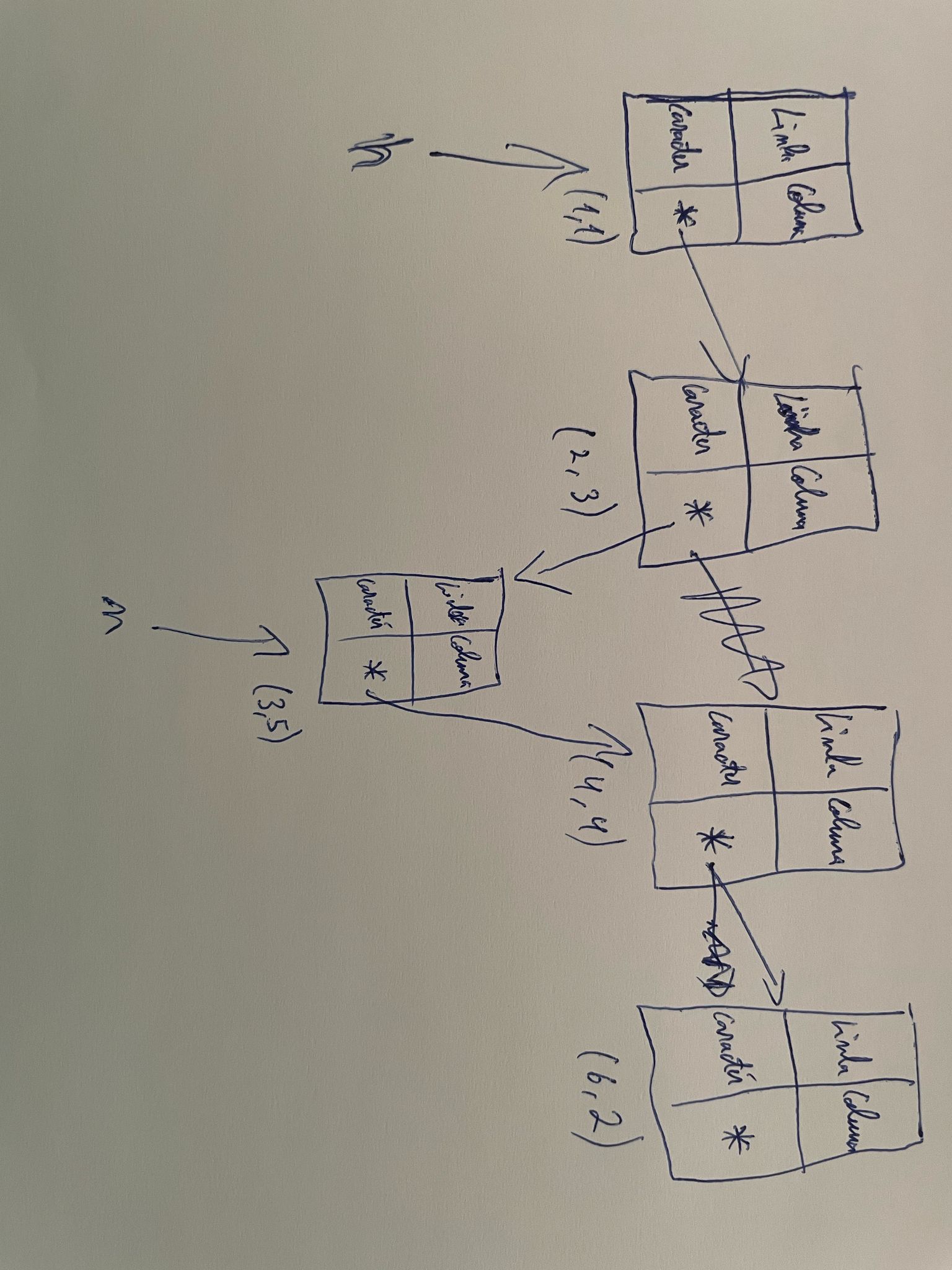


Figura 2 - Inserção de listas ligadas

Nesta figura 2, cada lista corresponde a cada retângulo, o qual contém o número de linha, a coluna, o caracter e o apontador para a seguinte. As letras “h” e “n” são os apontadores para as listas, o que permite um tratamento dinâmico das mesmas.  
  
 Para ver as matrizes, de acordo com o problema pré-estabelecido, criaram-se seguintes funções de mostrar: mostrar as antenas e mostrar as antenas com as respetivas nefastes - depois de aplicada a função de calcular as nefastes.

A função para calcular as nefastes - sempre duas para cada par de antenas - foi usada, aplicando o algoritmo seguinte:

- Para a primeira nefaste, que se localiza antes de ambas as Antenas, observou-se a seguinte fórmula: valor da linha da primeira Antena menos o valor da linha da segunda Antena mais o valor da linha da primeira novamente. Neste caso, para saber o valor da linha. Por sua vez, para saber o valor da coluna, foi usado o mesmo cálculo, mas, aqui, para os valores das colunas.

- Para a segunda nefaste, que se localiza depois de ambas as Antenas, usou-se o mesmo cálculo matemático, mas, neste caso, o que mudou foi a ordem das Antenas nesse cálculo, que se passa a explicar: valor da linha da segunda Antena menos o valor da linha da primeira Antena, mais o valor da linha da segunda. Para o valor da coluna, calcula-se exatamente da mesma forma, mas considerando-se os valores das colunas.

Este foi o método usado no cálculo das nefastes.

# 4. Implementação

Neste capítulo, será explicado o trabalho criado e a obtenção do resultado final.

Cada lista tem as mesmas referências, com a seguinte struct.

typedef struct ANTENA {

int linha;

int coluna;

char nome;

struct ANTENA\* next;

}ANTENA;

Tanto as Antenas, como as Nefastes, apenas necessitam dessas 4 características.

Cada lista é criada com a função criaAntena, em que devolve uma antena/nefaste criada em memória com os parâmetros da struct (número da linha, da coluna e o caracter(nome)).

O cabeçalho da função é:

ANTENA\* CriaAntena(int linha, int coluna, char nome);

A função de inserir Antena devolve a Antena/Nefaste inserida no “sítio certo” da lista ligada e tem como parâmetros o apontador para o início da lista ordenada, bem como o apontador para a nova Antena/Nefaste que se quer inserir.

O cabeçalho da função é:

ANTENA\* inserirAntena(ANTENA\* h, ANTENA\* n);

Esta função torna-se uma das mais importantes, visto que, estando as listas ordenadas, facilita a manipulação das mesmas com as funções restantes. Para inserir a lista ordenada há 3 situações possíveis: quando é inserida no início da lista, no meio ou no fim. E, para isso, foi necessário colocar em código todas essas condições, comparando, primeiramente, a linha da nova Antena/Nefaste com a linha da primeira lista da lista ordenada. Caso essa seja inferior, significa que é a primeira da lista ligada. Caso a linha da nova seja igual à linha da primeira da lista, comparamos pelas colunas: se a coluna da nova for inferior à coluna da primeira da lista, insere-se novamente no início e passa a primeira da lista a ser a nova. Depois, confere-se, com um ciclo, quando a linha da nova é superior a todas as linhas dos elementos da lista (até ao fim da lista), essa nova passa a ser a última da lista e insere-se no fim. Por sua vez, quando a linha da nova é igual a alguma linha dos elementos da lista, compara-se a coluna e, caso esta seja maior do que a coluna desse elemento (e onde a next desse mesmo elemento é nulo) , então essa nova é a última da lista.

Até aqui, vimos os casos da inserção no início ou no fim, estando em falta, ainda, a possibilidade da lista ser inserida no meio de outras. Caso isso aconteça, é necessário alterar o apontador da lista exatamente anterior para ficar a apontar para essa nova lista, bem como colocar o apontador dessa nova a apontar para a lista exatamente a seguir. O excerto de código infra mostra isso mesmo, sendo que o aux2 é o apontador que se encontra a apontar para a lista exatamente a seguir à nova e o aux é o apontador que se encontra a apontar para a lista anterior.

n->next = aux2;

aux->next = n;

No fim de usada esta função, são devolvidas as listas todas ordenadas, já com a nova inserida.

A função de remover com o cabeçalho:

ANTENA\* removerAntena(ANTENA\* h, int linha, int coluna);

Vai, novamente, comparar as linhas e as colunas da Antena que o utilizador insere com as linhas e colunas da lista ligada e, caso haja essa Antena, será retirada da lista ligada. Para retirar uma antena identificada pelo utilizador e a lista continuar toda ligada, efetuam-se as ligações necessárias e usa-se “free”. Caso essa Antena que o utilizador identificou não exista, nada acontecerá e aparecerá a mensagem “Antena não existe”.

A função de calcular zonas nefastas é, também, uma função muito importante, sendo, aliás, onde assenta o principal problema deste trabalho prático.

O cabeçalho da mesma é:

ANTENA\* calcularZonasNefastas(ANTENA\* h);

Esta função, como falado anteriormente, usou um algoritmo específico para calcular as zonas nefastas. Essas zonas nefastas só são calculadas quando existem, no mínimo, duas Antenas do mesmo tipo, com o mesmo caracter. Para tal, foi necessário utilizar um ciclo para percorrer todas as listas ligadas e verificar se existe mais do que uma Antena com o mesmo caracter (sinal/frequência). Caso encontre uma igual, aplicará o algoritmo, criará a lista da primeira Nefaste (com a função de criar) e inserirá ordenada nas listas ligadas (com a função de inserir). Como o ciclo percorre todas as listas ligadas, vai passar novamente nessas antenas, mas de forma inversa, vai aplicar, novamente, o algoritmo e calcular a outra zona nefasta. Tal como na anterior, vai criar e inserir nas listas ligadas essa nova zona nefasta. As zonas nefastas são identificadas com o caracter ‘#’. Poderiam haver situações em que, com os cálculos usados, uma nova zona nefasta ficaria com a linha e a coluna exatamente igual a uma Antena/Nefaste já existente. Perante isso, dá-se prioridade às Antenas e não seria criada essa Nefaste por troca da Antena. Caso uma zona nefasta também fique com o mesmo número de linha e coluna de outra já existente, apenas seria criada uma, por forma a não repetir. No final de calculadas as zonas nefastas com esta função e inseridas corretamente, são devolvidas as listas ligadas com as zonas nefastas. A função de mostrar a matriz com as zonas nefastas, recebe como parâmetros as listas ligadas das antenas e as listas ligadas das zonas nefastas.

int mostrarMatrizComNefastas(ANTENA\* h, ANTENA\* hNefastas);

Para melhor visualização, apurou-se qual a linha e a coluna menor e maior das duas listas ligadas, por forma a mostrar no terminal todas as Antenas e todas as Zonas Nefastas, até aquelas com linhas e colunas negativas.

No código utilizaram-se métodos para percorrer as listas para inserir os caracteres das Antenas e das Zonas Nefastas no sítio certo, conforme a linha e a coluna. Caso não encontre nem uma Antena nem uma Nefaste nessa linha e coluna, aparece um ponto ( . ), apenas para ser visualmente mais percetível para o utilizador.

Mostrando, por exemplo, na Figura 3.

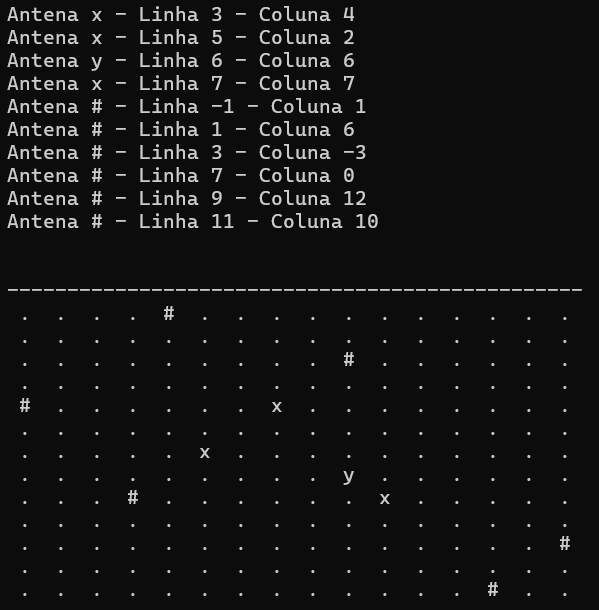


Figura 3 - Mostra as Antenas e as Zonas Nefastas

Para ler do ficheiro, foram criadas duas funções: uma para ler do ficheiro .txt (por exemplo, a Figura 4) e outra do ficheiro .bin. O que a função faz é: abrir o ficheiro e percorrer tudo até encontrar caracteres diferentes de pontos ( . ), ou quebras de linha (\n), ou zonas nefastas (#), ou, ainda, até não encontrar mais nada, criando e inserindo essa Antena na lista ligada. Continua a fazer o mesmo para criar e inserir todas as Antenas. Depois de “ver” todo o ficheiro, fecha-o com o fclose.

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

Os conteúdos gerados por IA poderão estar incorretos.

Figura 4 - Exemplo de um ficheiro .txt

Usando a função de mostrar a matriz a partir do ficheiro, que é praticamente igual à do mostrar do terminal, conseguimos ver a matriz no ecrã e, depois, com a função calcular zonas nefastas, conseguimos fazer tudo novamente, mas, neste caso, as listas ligadas com as Antenas “vêm” do ficheiro e não do terminal.

Por fim, foram criadas as funções para guardar nos ficheiros: uma para o ficheiro .txt e outra para o ficheiro .bin, em que guarda apenas as Antenas.

# 5. Conclusão

O trabalho desenvolvido consistiu na implementação de um sistema eficiente para manipulação de antenas e as suas zonas nefastas. As listas ligadas foram utilizadas para ordenar as Antenas e as Zonas Nefastas e o cálculo das zonas afetadas foi realizado com base nas interações entre antenas de frequências semelhantes. A solução foi implementada com sucesso, atendendo aos requisitos propostos no enunciado. Algumas melhorias podem ser feitas, designadamente ao nível da manipulação dos ficheiros.   
Tratou-se de um trabalho já de alguma exigência, ajudando a compreender melhor a manipulação de listas ligadas, a ler e a guardar em ficheiros.

Todo o trabalho está guardado no repositório GitHub com o link:

<https://github.com/Diogo-6731/Trabalho_Pratico>

# 6. Bibliografia

*Lista ligada – Wikipédia, a enciclopédia livre*. (n.d.).