RELATÓRIO

Thiago Abreu (nº31523)

Professor Luís Ferreira

IPCA
INSTITUTO POLITÉCNICO DO CÁVADO E DO AVE
ESCOLA SUPERIOR DE TECNOLOGIA

Conteúdo

Introdução

O estudo e a manipulação de estruturas de dados dinâmicas são fundamentais para o desenvolvimento de soluções eficientes e robustas na área da programação. A linguagem C, conhecida pela sua versatilidade e controlo sobre a gestão de memória, oferece um ambiente propício para o aprofundamento desses conceitos. Este projeto da Unidade Curricular de Estruturas de Dados Avançadas (EDA), integrado no 2.º semestre do 1.º ano, visa consolidar os conhecimentos adquiridos ao longo do semestre, através da implementação de uma solução que envolva a utilização de estruturas de dados dinâmicas, armazenamento em ficheiros e modularização do código.

A temática proposta envolve a modelagem de uma cidade com várias antenas, cada uma sintonizada numa frequência específica, e a análise do impacto que estas antenas causam em determinadas localizações da cidade, com base nas suas frequências de ressonância. O desafio consiste na identificação e mapeamento de locais afetados por efeitos nefastos gerados por antenas de mesma frequência, considerando distâncias específicas entre elas. O projeto busca aplicar os conceitos estudados de forma prática, por meio de uma abordagem que exige o uso de estruturas de dados dinâmicas para representar a cidade e calcular as interações entre as antenas de forma eficiente. A implementação e a documentação do código serão feitas com a utilização do Doxygen, visando uma estruturação clara e compreensível do software desenvolvido.

Neste trabalho, além de aprofundar os conceitos de estruturas de dados dinâmicas, o estudante será capaz de lidar com a complexidade do problema, estruturando uma solução que se mostre eficaz e bem documentada, seguindo as boas práticas de programação. O projeto reflete o desafio de transformar um problema real num problema computacional, utilizando as ferramentas adequadas para a sua resolução.

Ferramentas utilizadas

Neste trabalho, foram utilizadas diversas ferramentas para alcançar o resultado apresentado, sendo que cada uma delas desempenhou uma função específica, contribuindo de forma conjunta na realização do projeto. Entre estas ferramentas, destacam-se:

Linguagem C

A linguagem C foi a linguagem de programação utilizada para a implementação deste projeto. Considerada uma das linguagens mais fundamentais e poderosas no desenvolvimento de software, C oferece um controlo elevado sobre a gestão de memória, o que a torna especialmente adequada para a criação de estruturas de dados dinâmicas e a manipulação eficiente de recursos.

Visual Studio Code

O Visual Studio Code foi utilizado como o ambiente de desenvolvimento integrado (IDE) para a escrita, edição e debug do código-fonte do projeto. Com a sua interface leve e extensível, o VS Code permitiu a utilização de extensões essenciais, como o suporte para C/C++. Além disso, o terminal embutido facilitou a execução de comandos diretamente dentro da plataforma, assim permitindo um desenvolvimento mais eficiente e organizado.

Windows Subsystem for Linux (WSL)

O Windows Subsystem for Linux (WSL) foi utilizado neste projeto para fornecer um ambiente de desenvolvimento Linux dentro do Windows. O WSL permite a execução de ferramentas e utilitários nativos do Linux sem a necessidade de uma máquina virtual ou dual boot, garantindo maior compatibilidade com bibliotecas e compiladores frequentemente usdados no desenvolvimento em C. Neste projeto, o WSL foi essencial para compilar e executar o código utilizando o GCC (GNU Compiler Collection) e para utilizar o GDB (GNU Debugger) no debug do programa.

Multiple Cursor Case Preserve

O Multiple Cursor Case Preserve foi uma funcionalidade utilizada no Visual Studio Code para facilitar a edição de código. Esta ferramenta permite editar múltiplas ocorrências de um texto simultaneamente, mantendo a formatação original das palavras. Isto foi útil para a correção de erros em diferentes partes do código sem a necessidade de edições repetitivas.

Inteligencia Artificial – "ChatGPT"

A ferramenta de Inteligência Artificial – "ChatGPT" foi utilizada como um suporte no desenvolvimento do projeto, auxiliando com dúvidas, como o debug do código, explicação de conceitos complexos e na otimização do código.

Doxygen

O Doxygen foi a ferramenta escolhida para gerar a documentação do projeto de forma automatizada. Ele permitiu a extração de comentário estruturados do código-fonte e a conversão destes em documentação bem organizada.

WinRAR

O WinRAR foi utilizado para a compactação de arquivos durante o desenvolvimento do projeto. Isto foi essencial visto que como eu, pessoalmente, vivo sozinho numa residência universitária, tenho de em alguns fins de semana visitar a família, assim não conseguindo levar o meu computador principal, a minha solução foi compactar os arquivos e assim enviar para o meu portátil para dar seguimento ao meu trabalho.

Microsoft Word

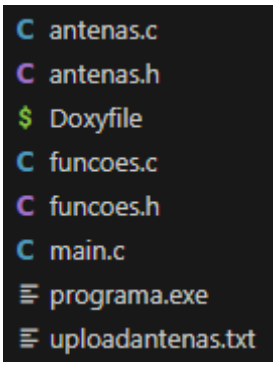
O Microsoft Word foi utilizado para a escrita e formatação do relatório do projeto. Através das suas ferramentas, consegui obter um relatório bem estruturado e organizado para entregar juntamente do meu projeto.

Inteligência Artificial – "MyMap"

A ferramenta de Inteligência Artificial – "MyMap" foi utilizada para organizar e estruturar as ideias do projeto por meio de mapas mentais. Esta abordagem ajudou a visualizar a lógica do código e ao compreender o código como um conjunto.

Estrutura do código

Primeiramente, ao abrir o nosso .rar com o projeto submetido, deparamo-nos com a seguinte estrutura:



(Imagem 1 – Estrutura)

Antenas.c

Local onde é feita a implementação das funções para manipulação da lista ligada de antenas.

Apresentamos a função *inserirAntena:

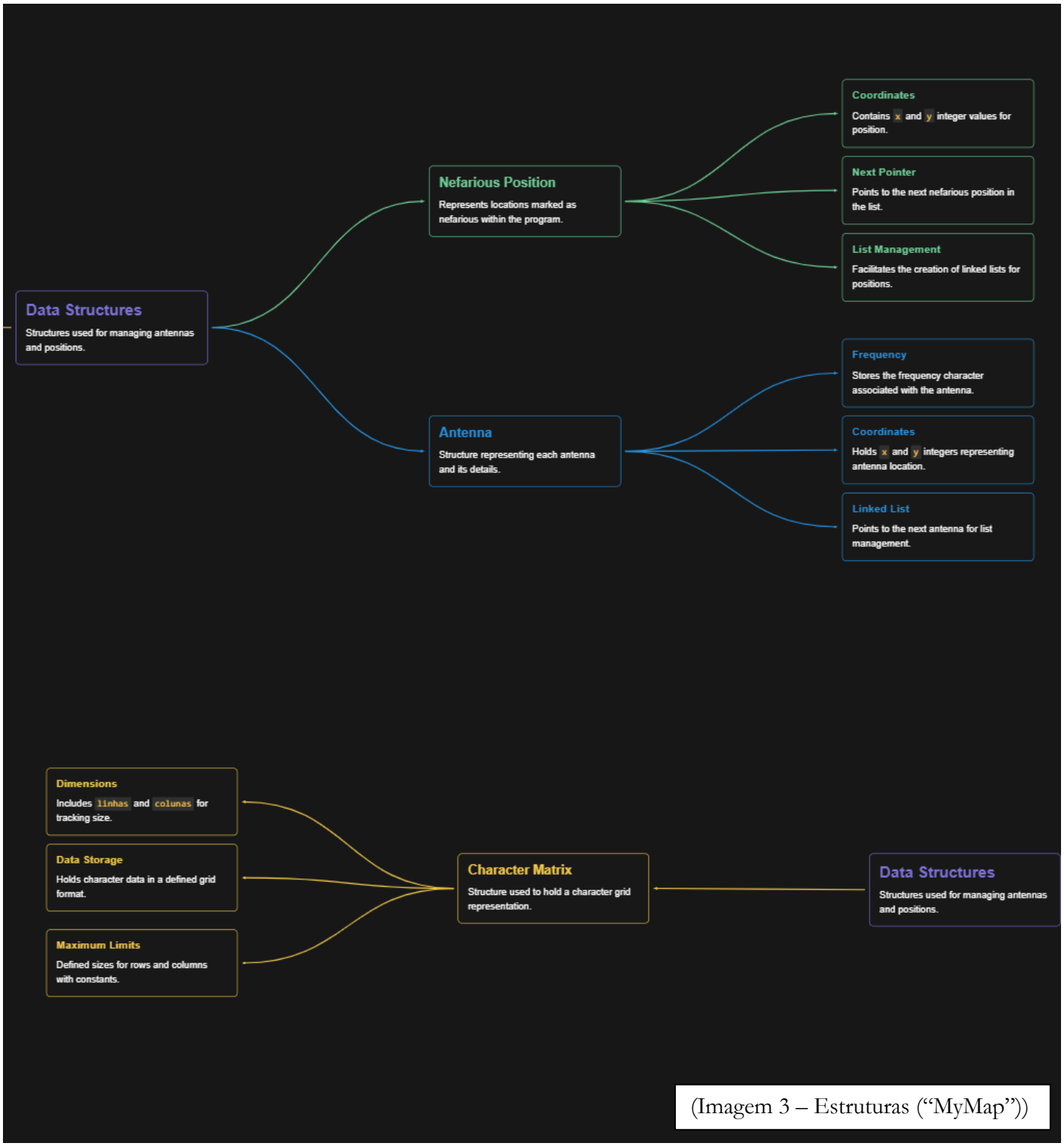
```
Antena *inserirAntena(Antena *lista, char frequencia, int x, int y)
```

(Imagem 2 – inserirAntena ("MyMap"))

Antenas.h e funcoes.h

Local onde são definidas as estruturas e constantes para a manipulação de antenas (antenas.h) e local onde são feitas as declarações de funções auxiliares para a manipulação de antenas e locais nefastos (funções.h).

Nestes ficheiros temos a definição das nossas estruturas:



(Imagem 3 – Estruturas ("MyMap"))

Tradução:

Data Structures (Estruturas de Dados) – Estruturas utilizadas para gerir antenas e posições.

Nefarious Position (Posição Nefasta) – Estrutura que representa localizações marcadas como nefastas dentro do programa.

Antenna (Antena) – Estrutura que representa cada antena e os seus detalhes.

Character Matrix (Matriz de Caracteres) – Estrutura utilizada para guardar a matriz do mapa.

Coordinates (Coordenadas) [Posição Nefasta] – Contem as integrais x e y que equivalem as posições.

Next Pointer (Proximo Apontador) – Aponta para a próxima posição nefasta da lista.

List Management (Gestão da Lista) – Facilita a criação de listas ligadas para as posições.

Frequency (Frequência) – Guarda o caractere da frequência associada com a antena.

Coordinates (Coordenadas) [Antena] - Contem as integrais x e y que equivalem a localização das antenas.

Linked List (Lista Ligada) – Aponta para a próxima antena para a gestão de listas.

Dimensions (Dimensões) – Inclui as variáveis linhas e colunas para acompanhar o tamanho.

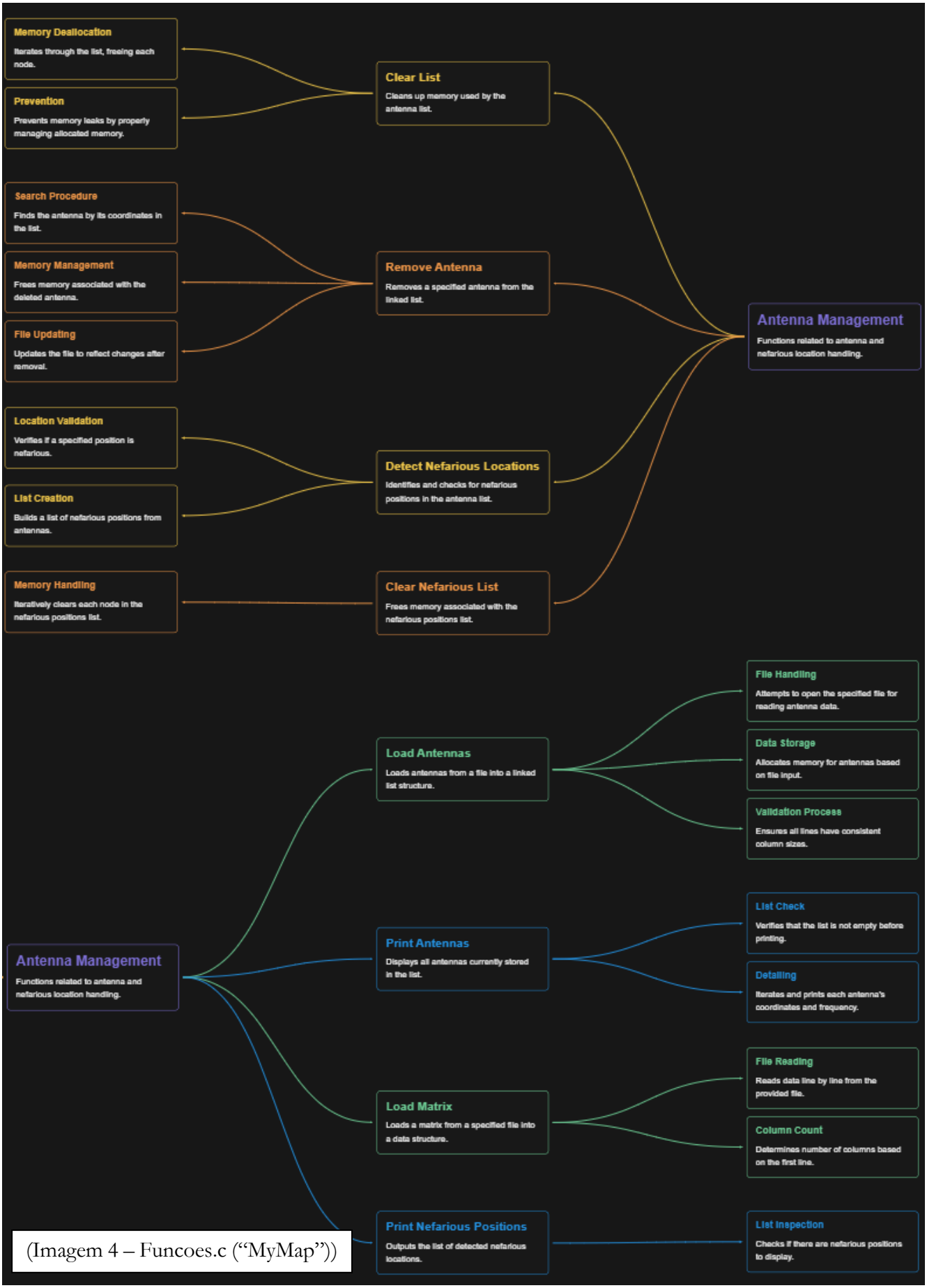
Data Storage (Armazenamento de Dados) – Guarda os dados dos caracteres em um formato de tabela.

Maximum Limits (Limites Máximos) – Tamanho definido para as linhas e colunas com constantes.

Funcoes.c

Local onde são implementadas as funções auxiliares para a manipulação de antenas e locais nefastos.

Neste ficheiro temos as funções:



(Imagem 4 – Funcoes.c (“MyMap”))

Tradução:

Antenna Management (Gestão de Antenas) – Funções relacionadas com gestão de antenas e localizações nefárias.

Clear List (Lista de Limpeza) – Limpa a memória utilizada pela lista de antenas.
Remove Antenna (Remoção de Antena) – Remove uma antena específica da lista ligada.
Detect Nefarious Locations (Deteção de Localizações Nefárias) – Identifica e verifica por posições nefárias na lista de antenas.
Clear Nefarious List (Limpar Lista Nefária) – Liberta a memória associada com a lista de posições nefárias.
Load Antennas (Carregar Antenas) – Carrega Antenas de um ficheiro (uploadantenas.txt) para a estrutura de lista ligada.
Print Antenas (Imprimir Antenas) – Mostra todas as antenas atualmente guardadas na lista.
Load Matrix (Carregar Matriz) – Carrega a matriz de um ficheiro específico para a estrutura de dados.
Print Nefarious Positions (Imprimir Posições Nefárias) – Dispõe a lista de localizações nefárias detetadas.
Memory Deallocation (Desalocação de Memória) – Percorre a lista, libertando cada elemento.
Prevention (Prevenção) – Evita fugas de memória ao gerir apropriadamente a memória alocada.
Search Procedure (Processo de busca) – Encontra a antena pelas suas coordenadas na lista.
Memory Management (Gestão de Memória) [Remoção de Antena] – Liberta memória associada com a antena apagada.
File Updating (Atualização de Ficheiro) – Atualiza o ficheiro para refletir as mudanças depois da remoção.
Location Validation (Validação de Localização) – Verifica se uma posição específica é nefária.
List Creation (Criação de Lista) – Constrói uma lista de posições nefárias de antenas.
Memory Handling (Gestão de Memória) [Limpar Lista Nefária] – Percorre a lista limpando cada elemento na lista nefária de posições.

File Handling (Gestão de Ficheiros) – Abre o ficheiro específico para ler os dados das antenas.

Data Storage (Armazenamento de Dados) – Aloca memória para as antenas baseado no ficheiro submetido.

Validation Process (Processo de Validação) – Verifica se todas as linhas têm tamanhos consistentes de colunas.

List Check (Verificação da lista) – Verifica se a lista não está vazia antes de imprimir.

Detailing (Detalhes) – Percorre a lista e imprime cada coordenada e frequência das antenas.

File Reading (Leitura de Ficheiro) – Lê todos os dados linha por linha do ficheiro providenciado.

Column Count (Contagem de Colunas) – Determina o número de colunas baseado na primeira linha.

List Inspection (Inspeção da Lista) – Verifica se existem posições nefárias para mostrar.

Desafios encontrados

A maior dificuldade sentida ao longo do desenvolvimento deste projeto até agora foi a interpretação do enunciado, especialmente na parte relativa à identificação das posições nefastas causadas pelas antenas. A definição das regras para determinar estas posições exigiu uma análise detalhada e cuidadosa para garantir a correta implementação.

Além disso, o planeamento do trabalho revelou-se um desafio, uma vez que foi necessário estruturar a abordagem de forma eficiente.

No entanto, com uma análise mais aprofundada e a divisão do problema em partes mais simples, foi possível superar estas dificuldades e desenvolver uma solução funcional.

O que pode ser melhorado

Como esta é apenas a primeira parte do trabalho, é natural que ainda faltem alguns elementos que serão desenvolvidos mais adiante. No entanto, no projeto atual, acredito que uma das principais melhorias seria a organização do ficheiro main.c de uma forma mais estruturada, garantindo uma melhor separação das funcionalidades.

Uma abordagem mais modular, com uma interface, permitiria que as funções fossem chamadas dinamicamente com base nas decisões do utilizador ao executar o programa.

Conclusão

Este trabalho permitiu compreender a importância das estruturas de dados dinâmicas e a sua aplicação na resolução de problemas computacionais mais complexos. Através da implementação do projeto, foi possível explorar conceitos fundamentais da linguagem C, como a gestão eficiente de memória, a modularização do código e a manipulação de estruturas de dados.

Ao longo do desenvolvimento, enfrentei desafios, especialmente na interpretação do enunciado e no planeamento da solução, mas consegui ultrapassá-los com uma análise mais detalhada do problema. A implementação das funcionalidades propostas permitiu consolidar conhecimentos e aplicar boas práticas de programação, de forma a garantir um código mais organizado e eficiente, como por exemplo ao documentar.

Além disso, reconhece-se que há aspetos que podem ser melhorados, como a estruturação do main.c de forma a criar uma interface própria. Como esta é apenas a primeira parte do trabalho, é esperado que ainda existam aspetos a aprimorar em funcionalidades a acrescentar na próxima fase.

No geral, este projeto revelou-se um desafio, permitindo não só reforçar o domínio sobre estruturas de dados dinâmicas, mas também desenvolver competências essenciais para a resolução estruturada e eficiente de problemas em programação.

Link Github: https://github.com/notunked/Projeto-EDA-Parte-1