

Instituto Politécnico do Cávado e do Ave

Escola Superior de Tecnologia

Estruturas de Dados Avançadas

Licenciatura em Engenharia de Sistemas Informáticos

Trabalho Prático

Fase 1

Fábio Alexandre Gomes Fernandes – a22996

março de 2025

Resumo

O desenvolvimento do sistema de gestão de antenas e efeitos nefastos, realizado no âmbito da disciplina de Estruturas de Dados Avançadas, representa uma aplicação prática dos conceitos adquiridos ao longo do curso. Com foco na eficiência e organização dos dados, o programa implementa uma abordagem baseada em listas ligadas para gerir antenas e calcular os efeitos nefastos causados por interferências de frequência.

A estrutura modular do sistema assegura uma clara separação entre os diferentes componentes, permitindo a manipulação eficiente das antenas e dos efeitos nefastos. A utilização de alocação dinâmica de memória e a organização do código em múltiplos ficheiros reforçam a escalabilidade e manutenção do programa. Além disso, a integração de leitura e escrita de ficheiros possibilita o carregamento e armazenamento da configuração das antenas, tornando o sistema mais versátil.

Ao longo deste relatório, são exploradas as funcionalidades e metodologias adotadas, evidenciando a aplicação dos conceitos de estruturas de dados dinâmicas e modularização. O projeto não só consolida os conhecimentos adquiridos, mas também demonstra a importância da organização e otimização na resolução de problemas computacionais.

Índice

1. Introdução.....	6
2. Pasta “ListaAntenas”.....	7
2.1 Código “antenas.h”.....	7
2.2 Código “antenas.c”.....	8
2.2.1 Criação de uma antena (criarAntena).....	9
2.2.2 Inserção de uma antena (inserirAntena).....	9
2.2.3 Remoção de uma antena (removerAntena).....	10
2.2.4 Exibição das antenas (mostrarAntenas).....	10
2.2.5 Limpeza da lista (limparAntenas).....	10
2.2.6 Carregamento de antenas a partir de um ficheiro (carregarFicheiro).....	11
3. Pasta “EfeitosNefastos”.....	11
3.1 Código “nefastos.h”.....	11
3.2 Código “nefastos.c”.....	13
3.2.1 Dedução dos Efeitos Nefastos (deduzirEfeitosNefastos).....	13
3.2.2 Exibição dos Efeitos Nefastos (mostrarEfeitosNefastos).....	14
3.2.3 Preenchimento da Matriz com Efeitos (preencherMatrizComEfeitos).....	14
3.2.4 Limpeza da Lista de Efeitos Nefastos (limparEfeitosNefastos).....	15
4. Ficheiro “estrutura.txt”.....	15
5. Manipulação das antenas e os efeitos de nefasto.....	16
5.1 Código “main.c”.....	16
5.1.1 Função mostrarMatriz.....	16
5.1.2 Função main.....	17
5.2 Comandos utilizado na linha de comandos.....	18
5.3 Testes.....	18
6. Repositório GitHub.....	20
7. Conclusão.....	21
8. Referências.....	22

Índice de Ilustrações

Figura 1: Estrutura de uma antena	7
Figura 2: Funções para gerir antenas	8
Figura 3: Função para criar uma antena	9
Figura 4: Função para inserir uma antena na lista	9
Figura 5: Função para remover antena da lista.....	10
Figura 6: Função para listar as antenas com as respetivas coordenadas	10
Figura 7: Função para libertar memória das antenas na lista	11
Figura 8: Função para carregar ficheiro	11
Figura 9: Estrutura de um efeito nefasto	12
Figura 10: Funções para gerir efeitos nefastos	13
Figura 11: Função para deduzir onde ficam os efeitos nefastos (uma parte da função).....	13
Figura 12: Função para listar as coordenadas dos efeitos nefastos	14
Figura 13: Função para inserir os efeitos nefastos na matriz	14
Figura 14: Função para libertar memória associada à lista de efeitos nefastos	15
Figura 15: Ficheiro "estrutura.txt"	15
Figura 16: Função para mostrar a matriz (antenas e nefastos)	16
Figura 17: Main (parte 1).....	17
Figura 18: Main (parte 2).....	18
Figura 19: Comandos para o terminal	18
Figura 20: Teste (parte 1)	18
Figura 21: Testes (parte 3).....	19
Figura 22: Testes (parte 4).....	19
Figura 23: Testes (parte 5).....	19
Figura 24: Testes (parte 6).....	19

1. Introdução

No âmbito da cadeira de Estruturas de Dados Avançadas (EDA), lecionada por Luís Ferreira no curso de Engenharia de Sistemas Informáticos, apresento este relatório sobre o desenvolvimento de um programa de gestão de antenas e efeitos nefastos. O gerenciamento eficiente das antenas e seus efeitos é um desafio que envolve o controle das localizações das antenas e a análise das interferências causadas por elas. Com o objetivo de simular e analisar esses efeitos, desenvolvi uma solução utilizando estruturas de dados dinâmicas em C.

Este sistema não se limita a ser uma ferramenta técnica, mas sim uma aplicação prática dos conceitos que aprendi na disciplina, como listas ligadas e armazenamento em ficheiro. Ao longo deste relatório, vou explorar as principais funcionalidades do programa, desde o carregamento de antenas a partir de ficheiros até a dedução e exibição dos efeitos nefastos. Destacarei a forma como as antenas interagem dentro de uma matriz, gerando pontos afetados, e como a modularização e a documentação com Doxygen fundamentam a arquitetura do software. Não esquecendo, que no decorrer do trabalho também foi criado um repositório no Github.

Com este trabalho, busco não apenas demonstrar a aplicação dos conceitos de estruturas de dados dinâmicas, mas também fornecer uma solução eficaz para simulação de interferências de antenas, contribuindo para uma compreensão mais profunda dos desafios e soluções associadas ao desenvolvimento desse tipo de sistema.

2. Pasta “ListaAntenas”

2.1 Código “antenas.h”

O ficheiro “**antenas.h**” é o cabeçalho do módulo responsável pela gestão das antenas no sistema. Ele define a estrutura de dados que representa uma antena e declara as funções necessárias para a sua manipulação.

- **Estrutura de Dados**

A estrutura **Antena** é utilizada para armazenar informações sobre cada antena registada. Os campos presentes na estrutura são:

- **frequencia:** Um carácter que representa a frequência da antena.
- **x e y:** Coordenadas da antena na matriz.
- **proximo:** Um apontador para a próxima antena na lista ligada.

```
10 // Estrutura para representar uma antena
11 typedef struct Antena {
12     char frequencia; // Frequência da antena
13     int x, y; // Coordenadas da antena
14     struct Antena *proximo; // Apontador para a próxima antena (lista ligada)
15 } Antena;
```

Figura 1: Estrutura de uma antena

- **Funções Declaradas**

As funções declaradas em “antenas.h” permitem criar, inserir, remover, exibir e limpar antenas, bem como carregar uma lista de antenas a partir de um ficheiro:

1. **Antena* criarAntena(char frequencia, int x, int y);**

- Cria uma antena com os parâmetros fornecidos e retorna um apontador para ela.

2. ***void inserirAntena(Antena **lista, char frequencia, int x, int y);***
 - Insere uma nova antena na lista ligada, garantindo que as antenas sejam armazenadas de forma organizada.
3. ***void removerAntena(Antena **lista, int x, int y);***
 - Remove uma antena localizada nas coordenadas fornecidas, ajustando a ligação da lista ligada conforme necessário.
4. ***void mostrarAntenas(Antena *lista);***
 - Percorre a lista de antenas e imprime as informações de cada uma no terminal.
5. ***void limparAntenas(Antena **lista);***
 - Liberta a memória alocada para todas as antenas, esvaziando a lista.
6. ***void carregarFicheiro(Antena **lista, const char* nomeFicheiro);***
 - Lê um ficheiro e adiciona as antenas listadas nele à lista ligada.

```
17 // Funções para gerir antenas
18 Antena* criarAntena(char frequencia, int x, int y);
19 void inserirAntena(Antena **lista, char frequencia, int x, int y);
20 void removerAntena(Antena **lista, int x, int y);
21 void mostrarAntenas(Antena *lista);
22 void limparAntenas(Antena **lista);
23 void carregarFicheiro(Antena **lista, const char* nomeFicheiro);|
```

Figura 2: Funções para gerir antenas

2.2 Código “antenas.c”

O ficheiro “antenas.c” implementa as funções necessárias para gerir uma lista de antenas. Estas funções permitem criar, inserir, remover e visualizar antenas, bem como carregar dados a partir de um ficheiro.

2.2.1 Criação de uma antena (criarAntena)

A função “criarAntena” aloca dinamicamente memória para uma nova antena e define os seus atributos:

- frequencia (tipo *char*)
- Coordenadas X e Y (tipo *int*)
- Apontador para a próxima antena (permitindo formar uma lista ligada) Se a alocação de memória falhar, a função retorna *NULL*.

```
23 Antena* criarAntena(char frequencia, int x, int y)
24 {
25     Antena* nova = (Antena*)malloc(sizeof(Antena));
26     if (!nova)
27     {
28         printf("Erro ao alocar memória para a antena!\n");
29         return NULL;
30     }
31     nova->frequencia = frequencia;
32     nova->x = x;
33     nova->y = y;
34     nova->proximo = NULL;
35     return nova;
36 }
```

Figura 3: Função para criar uma antena

2.2.2 Inserção de uma antena (inserirAntena)

Esta função permite adicionar uma nova antena à lista ligada. A inserção ocorre no início da lista para maior eficiência. A nova antena passa a ser o primeiro elemento, apontando para a lista previamente existente.

```
46 void inserirAntena(Antena** lista, char frequencia, int x, int y)
47 {
48     Antena* nova = criarAntena(frequencia, x, y);
49     if (!nova) return;
50
51     // Insere no início da lista
52     nova->proximo = *lista;
53     *lista = nova;
54 }
```

Figura 4: Função para inserir uma antena na lista

2.2.3 Remoção de uma antena (removerAntena)

Esta função percorre a lista até encontrar uma antena com as coordenadas especificadas. Se encontrada, é removida da lista e a memória correspondente é libertada. Se a antena a remover estiver no início da lista, o apontador principal é atualizado.

```
63 void removerAntena(Antena** lista, int x, int y)
64 {
65     Antena *atual = *lista, *anterior = NULL;
66
67     // Percorre a lista até encontrar a antena com as coordenadas especificadas
68     while (atual != NULL && (atual->x != x || atual->y != y))
69     {
70         anterior = atual;
71         atual = atual->proximo;
72     }
73     if (atual == NULL) return; // Se não encontrou, sai da função
74
75     // Se a antena a remover for a primeira da lista
76     if (anterior == NULL) *lista = atual->proximo;
77     else anterior->proximo = atual->proximo;
78
79     free(atual); // Liberta a memória da antena removida
80 }
```

Figura 5: Função para remover antena da lista

2.2.4 Exibição das antenas (mostrarAntenas)

Percorre a lista e imprime as informações de cada antena registada. Caso a lista esteja vazia, exibe uma mensagem apropriada.

```
87 void mostrarAntenas(Antena* lista)
88 {
89     if (!lista)
90     {
91         printf("Nenhuma antena registada.\n");
92         return;
93     }
94
95     // Percorre a lista e imprime cada antena
96     while (lista)
97     {
98         printf("Frequencia: %c | Coordenadas: (%d, %d)\n", lista->frequencia, lista->x, lista->y);
99         lista = lista->proximo;
100     }
101 }
```

Figura 6: Função para listar as antenas com as respetivas coordenadas

2.2.5 Limpeza da lista (limparAntenas)

Liberta a memória de todas as antenas na lista, garantindo que todos os recursos são corretamente desalocados. No final, o apontador principal é definido como *NULL* para evitar acessos indevidos.

```

108 void limparAntenas(Antena** lista)
109 {
110     // Percorre a lista e liberta cada antena
111     while (*lista)
112     {
113         Antena* temp = *lista;
114         *lista = (*lista)->proximo;
115         free(temp);
116     }
117
118     *lista = NULL; // Garante que a lista fica vazia
119 }

```

Figura 7: Função para libertar memória das antenas na lista

2.2.6 Carregamento de antenas a partir de um ficheiro (carregarFicheiro)

Esta função abre um ficheiro e processa cada linha para criar antenas com base no seu conteúdo. Cada carácter diferente de “.” é considerado uma antena e inserido na lista, sendo as coordenadas X e Y determinadas com base na posição no ficheiro.

```

127 void carregarFicheiro(Antena** lista, const char* nomeFicheiro)
128 {
129     FILE* ficheiro = fopen(nomeFicheiro, "r");
130     if (!ficheiro)
131     {
132         printf("Erro ao abrir o ficheiro %s!\n", nomeFicheiro);
133         return;
134     }
135
136     char linha[100];
137     int y = 0;
138
139     // Lê o ficheiro linha a linha
140     while (fgets(linha, sizeof(linha), ficheiro))
141     {
142         // Percorre cada carácter da linha
143         for (int x = 0; linha[x] != '\0' && linha[x] != '\n'; x++)
144         {
145             if (linha[x] != '.')
146             { // Assume que '.' significa "vazio"
147                 inserirAntena(lista, linha[x], x, y);
148             }
149         }
150         y++; // Incrementa a coordenada Y
151     }
152     fclose(ficheiro);
153 }

```

Figura 8: Função para carregar ficheiro

3. Pasta “EfeitosNefastos”

3.1 Código “nefastos.h”

O ficheiro "**nefastos.h**" é o cabeçalho do módulo responsável pela gestão dos efeitos nefastos causados pelas antenas no sistema. Ele define a estrutura de dados que representa um efeito nefasto e declara as funções necessárias para a sua manipulação.

- **Estrutura de Dados**

A estrutura **EfeitoNefasto** é utilizada para armazenar informações sobre os pontos do sistema afetados negativamente pelas antenas. Os campos presentes na estrutura são:

- **x e y:** Coordenadas do efeito nefasto na matriz.
- **proximo:** Um apontador para o próximo efeito nefasto na lista ligada.

```
12 // Estrutura para representar um ponto com efeito nefasto
13 typedef struct EfeitoNefasto {
14     int x, y; // Coordenadas do efeito nefasto
15     struct EfeitoNefasto *proximo; // Apontador para o próximo efeito nefasto
16 } EfeitoNefasto;
```

Figura 9: Estrutura de um efeito nefasto

- **Funções Declaradas**

As funções declaradas em "**nefastos.h**" permitem deduzir os efeitos nefastos das antenas, exibi-los, preencher a matriz do sistema com esses efeitos e limpar a lista quando necessário:

1. ***void deduzirEfeitosNefastos(Antena* lista, EfeitoNefasto** efeitos, int largura, int altura);***
 - Analisa a lista de antenas e determina quais posições da matriz sofrem efeitos nefastos, armazenando essas posições numa lista ligada.
2. ***void mostrarEfeitosNefastos(EfeitoNefasto *lista);***
 - Percorre a lista de efeitos nefastos e exibe as suas coordenadas no terminal.
3. ***void preencherMatrizComEfeitos(char** matriz, int largura, int altura, EfeitoNefasto* efeitos);***
 - Insere os efeitos nefastos na matriz principal do sistema, alterando os valores das posições afetadas.
4. ***void limparEfeitosNefastos(EfeitoNefasto ** lista);***
 - Liberta a memória alocada para a lista de efeitos nefastos, garantindo que a lista fica vazia.

```

18 // Funções para gerir efeitos nefastos
19 void deduzirEfeitosNefastos(Antena* lista, EfeitoNefasto** efeitos, int largura, int altura);
20 void mostrarEfeitosNefastos(EfeitoNefasto *lista);
21 void preencherMatrizComEfeitos(char** matriz, int largura, int altura, EfeitoNefasto* efeitos);
22 void limparEfeitosNefastos(EfeitoNefasto **lista);

```

Figura 10: Funções para gerir efeitos nefastos

3.2 Código “nefastos.c”

O ficheiro “**nefastos.c**” implementa as funções responsáveis por detetar e gerir os efeitos nefastos causados pelas antenas. O sistema analisa a posição das antenas e determina os pontos onde ocorrem interferências, armazenando esses pontos numa lista ligada.

3.2.1 Dedução dos Efeitos Nefastos (deduzirEfeitosNefastos)

Esta função percorre a lista de antenas e verifica quais delas geram efeitos nefastos. A lógica utilizada é a seguinte:

- Se duas antenas têm a mesma frequência e estão alinhadas (horizontalmente, verticalmente ou diagonalmente), calcula-se a posição dos pontos de interferência.
- O efeito nefasto ocorre quando uma antena está ao dobro da distância da outra em relação a um ponto intermédio.
- Os pontos identificados como afetados são adicionados a uma lista ligada de efeitos nefastos, desde que estejam dentro dos limites do espaço definido.

```

24 void deduzirEfeitosNefastos(Antena* lista, EfeitoNefasto** efeitos, int largura, int altura)
25 {
26     // Percorrer todas as antenas existentes
27     for (Antena* a1 = lista; a1 != NULL; a1 = a1->proximo)
28     {
29         for (Antena* a2 = a1->proximo; a2 != NULL; a2 = a2->proximo)
30         {
31             // Só interessa se forem da mesma frequência
32             if (a1->frequencia == a2->frequencia)
33             {
34                 // Determina o deslocamento entre as duas antenas
35                 int dx = a2->x - a1->x;
36                 int dy = a2->y - a1->y;
37
38                 // Calcula as posições onde surgem os efeitos nefastos
39                 int ex1_x = a1->x - dx;
40                 int ex1_y = a1->y - dy;

```

Figura 11: Função para deduzir onde ficam os efeitos nefastos (uma parte da função)

3.2.2 Exibição dos Efeitos Nefastos (mostrarEfeitosNefastos)

Percorre a lista de efeitos nefastos e imprime as suas coordenadas no terminal.

- Se existirem efeitos registados, são apresentados no formato (x, y).
- Se não houver efeitos detetados, uma mensagem apropriada é exibida.

```

83 void mostrarEfeitosNefastos(EfeitoNefasto* lista)
84 {
85     if (!lista)
86     {
87         printf("Nenhum efeito nefasto detetado.\n");
88         return;
89     }
90
91     // Percorre a lista e imprime cada efeito encontrado
92     while (lista)
93     {
94         printf("Efeito Nefasto em: (%d, %d)\n", lista->x, lista->y);
95         lista = lista->proximo;
96     }
97 }

```

Figura 12: Função para listar as coordenadas dos efeitos nefastos

3.2.3 Preenchimento da Matriz com Efeitos (preencherMatrizComEfeitos)

Esta função insere os efeitos nefastos detetados na matriz do sistema, marcando os pontos afetados com o símbolo “#”.

- A função percorre a lista de efeitos nefastos e verifica se cada ponto está dentro dos limites da matriz.
- Se a posição estiver disponível (representada por “.”), o efeito é registado na matriz.

```

107 void preencherMatrizComEfeitos(char** matriz, int largura, int altura, EfeitoNefasto* efeitos)
108 {
109     // Percorre a lista de efeitos e marca-os na matriz
110     for (EfeitoNefasto* aux = efeitos; aux; aux = aux->proximo)
111     {
112         if (aux->x >= 0 && aux->x < largura && aux->y >= 0 && aux->y < altura)
113         {
114             if (matriz[aux->y][aux->x] == '.')
115             { // Apenas marca como '#' se a posição ainda não tiver sido ocupada
116                 matriz[aux->y][aux->x] = '#';
117             }
118         }
119     }
120 }

```

Figura 13: Função para inserir os efeitos nefastos na matriz

3.2.4 Limpeza da Lista de Efeitos Nefastos (limparEfeitosNefastos)

Liberta a memória alocada para a lista de efeitos nefastos.

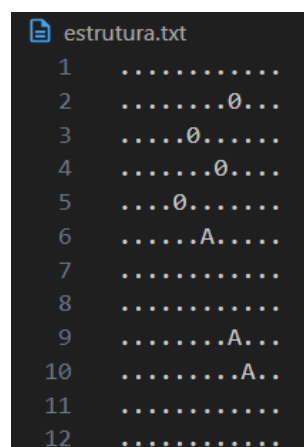
- A função percorre a lista e desaloca cada nó individualmente.
- No final, o apontador principal é definido como *NULL* para evitar acessos inválidos.

```
127 void limparEfeitosNefastos(EfeitoNefasto** lista)
128 {
129     // Percorre a lista e liberta cada nó
130     while (*lista)
131     {
132         EfeitoNefasto* temp = *lista;
133         *lista = (*lista)->proximo;
134         free(temp);
135     }
136     *lista = NULL; // Garante que a lista fica vazia
137 }
```

Figura 14: Função para libertar memória associada à lista de efeitos nefastos

4. Ficheiro “estrutura.txt”

O ficheiro "estrutura.txt" contém a representação da disposição das antenas num espaço bidimensional. Este ficheiro é utilizado pelo programa para carregar a configuração inicial das antenas e determinar a sua distribuição na matriz.



```
estrutura.txt
1  .....
2  .....0...
3  .....0...
4  .....0...
5  .....0...
6  .....A...
7  .....
8  .....
9  .....A...
10 .....A...
11 .....
12 .....
```

Figura 15: Ficheiro "estrutura.txt"

5. Manipulação das antenas e os efeitos nefasto

5.1 Código “main.c”

O ficheiro “**main.c**” contém o programa principal responsável por gerir a lista de antenas e os efeitos nefastos. Ele realiza a leitura da configuração inicial, processa as antenas e os seus efeitos, e apresenta os resultados ao utilizador.

5.1.1 Função `mostrarMatriz`

Esta função gera e exhibe uma matriz que representa a disposição das antenas e dos efeitos nefastos. O processo ocorre em várias etapas:

- **Alocação de memória:** A matriz é criada dinamicamente com base na largura e altura definidas.
- **Inicialização da matriz:** Todas as posições são preenchidas com “.”, representando áreas vazias.
- **Inserção das antenas:** Percorre a lista de antenas e adiciona os caracteres correspondentes na matriz.
- **Adição dos efeitos nefastos:** A função “`preencherMatrizComEfeitos`” é chamada para marcar os pontos de interferência com “#”.
- **Exibição da matriz:** A matriz é impressa no terminal, linha por linha.
- **Libertação de memória:** Após a exibição, a memória alocada para a matriz é desalocada.

```
22 void mostrarMatriz(Antena* lista, EfeitoNefasto* efeitos, int largura, int altura)
23 {
24     // Aloca memória para a matriz (altura linhas)
25     char** matriz = (char**)malloc(altura * sizeof(char*));
26     for (int i = 0; i < altura; i++)
27     {
28         // Para cada linha, aloca memória para armazenar a largura, colunas
29         matriz[i] = (char*)malloc(largura * sizeof(char));
30     }
31
32     // Inicializa a matriz com o caracter '.' para representar áreas vazias
33     for (int i = 0; i < altura; i++)
34     {
35         for (int j = 0; j < largura; j++)
```

Figura 16: Função para mostrar a matriz (antenas e nefastos)

5.1.2 Função main

A função principal **coordena a execução do programa**, realizando as seguintes ações:

1. **Levantamento das antenas:**

- O programa lê o ficheiro "estrutura.txt" e carrega as antenas na lista ligada.
- Exibe a matriz inicial e a lista de antenas carregadas.

2. **Deteção e exibição dos efeitos nefastos:**

- Os efeitos nefastos são calculados e apresentados no terminal.
- A matriz é exibida novamente, agora incluindo os efeitos nefastos.

3. **Inserção de novas antenas:**

- Duas novas antenas da frequência 'B' são adicionadas.
- A matriz é exibida novamente sem efeitos nefastos recalculados.

4. **Atualização dos efeitos nefastos:**

- Os efeitos são recalculados após a inserção das novas antenas.
- A matriz atualizada com os novos efeitos é apresentada.

5. **Remoção de antenas e novo processamento:**

- As antenas inseridas são removidas.
- Os efeitos nefastos são recalculados.
- A matriz final, sem as antenas adicionadas e com os efeitos ajustados, é exibida.

6. **Libertação de memória:**

- Antes de terminar, o programa liberta toda a memória alocada dinamicamente, garantindo que não há fugas de memória.

```
71 int main()
72 {
73     Antena* listaAntenas = NULL;
74     EfeitoNefasto* listaEfeitos = NULL;
75
76     // Carregar a matriz original a partir do ficheiro
77     carregarFicheiro(&listaAntenas, "estrutura.txt");
78
79     // Exibir a matriz original
80     printf("\nMATRIZ ORIGINAL:\n");
81     mostrarMatriz(listaAntenas, listaEfeitos, 12, 12);
82
83     // Exibir as coordenadas das antenas
84     printf("\nLISTA DE ANTENAS:\n");
85     mostrarAntenas(listaAntenas);
86
87     // Deduzir e exibir os efeitos nefastos iniciais
88     printf("\nLISTA DE EFEITOS NEFASTOS:\n");
89     deduzirEfeitosNefastos(listaAntenas, &listaEfeitos, 12, 12);
```

Figura 17: Main (parte 1)

```

96 // Inserir novas antenas, B(3, 3) e B(5, 5), e mostrar a matriz
97 printf("\nMATRIZ APOS INSERIR NOVAS ANTENAS:\n");
98 inserirAntena(&listaAntenas, 'B', 3, 3);
99 inserirAntena(&listaAntenas, 'B', 5, 5);
100 limparEfeitosNefastos(&listaEfeitos);
101 mostrarMatriz(listaAntenas, listaEfeitos, 12, 12);
102
103 //Mostrar lista de antenas após inserção, recalculer os efeitos nefastos, e
104 printf("\nLISTA DE ANTENAS:\n");
105 mostrarAntenas(listaAntenas);
106 printf("\nLISTA DE EFEITOS NEFASTOS APOS INSERCAO:\n");
107 deduzirEfeitosNefastos(listaAntenas, &listaEfeitos, 12, 12);
108 mostrarEfeitosNefastos(listaEfeitos);
109 printf("\nMATRIZ COM OS EFEITOS NEFASTOS APOS INSERCAO:\n");
110 mostrarMatriz(listaAntenas, listaEfeitos, 12, 12);
111
112 // Remover as antenas B(3, 3) e B(5, 5)
113 printf("\nMATRIZ APOS REMOVER AS ANTENAS INSERIDAS:\n");
114 removerAntena(&listaAntenas, 3, 3);
115 removerAntena(&listaAntenas, 5, 5);
116 limparEfeitosNefastos(&listaEfeitos);
117 mostrarMatriz(listaAntenas, listaEfeitos, 12, 12);
118
119 // Recalculer os efeitos nefastos após remover as antenas B(3, 3) e B(5, 5)
120 printf("\nMATRIZ DE EFEITOS NEFASTOS APOS REMOVER AS ANTENAS INSERIDAS:\n");
121 deduzirEfeitosNefastos(listaAntenas, &listaEfeitos, 12, 12);
122 mostrarMatriz(listaAntenas, listaEfeitos, 12, 12);
123
124 // Liberar memória alocada
125 limparEfeitosNefastos(&listaEfeitos);
126 limparAntenas(&listaAntenas);
127
128 return 0;

```

Figura 18: Main (parte 2)

5.2 Comandos utilizado na linha de comandos

```

PS U:\ESI\2 ano\2 semestre\eda (lufer)\Trabalhos\tp_eda> cd EfeitosNefastos
PS U:\ESI\2 ano\2 semestre\eda (lufer)\Trabalhos\tp_eda\EfeitosNefastos> gcc -c nefastos.c
PS U:\ESI\2 ano\2 semestre\eda (lufer)\Trabalhos\tp_eda\EfeitosNefastos> cd ..
PS U:\ESI\2 ano\2 semestre\eda (lufer)\Trabalhos\tp_eda> cd ListaAntenas
PS U:\ESI\2 ano\2 semestre\eda (lufer)\Trabalhos\tp_eda\ListaAntenas> gcc -c antenas.c
PS U:\ESI\2 ano\2 semestre\eda (lufer)\Trabalhos\tp_eda\ListaAntenas> cd ..
PS U:\ESI\2 ano\2 semestre\eda (lufer)\Trabalhos\tp_eda> gcc main.c ListaAntenas/antenas.c EfeitosNefastos/nefastos.c -o main
PS U:\ESI\2 ano\2 semestre\eda (lufer)\Trabalhos\tp_eda> ./main

```

Figura 19: Comandos para o terminal

5.3 Testes

```

MATRIZ ORIGINAL:
. . . . .
. . . . . 0 . . . .
. . . . . 0 . . . .
. . . . . 0 . . . .
. . . . . A . . . .
. . . . . . . . . .
. . . . . A . . . .
. . . . . A . . . .
. . . . . . . . . .
. . . . . . . . . .

LISTA DE ANTENAS:
Frequencia: A | Coordenadas: (9, 9)
Frequencia: A | Coordenadas: (8, 8)
Frequencia: A | Coordenadas: (6, 5)
Frequencia: 0 | Coordenadas: (4, 4)
Frequencia: 0 | Coordenadas: (7, 3)
Frequencia: 0 | Coordenadas: (5, 2)

```

Figura 20: Teste (parte 1)

```

LISTA DE EFEITOS NEFASTOS:
Efeito Nefasto em: (11, 0)
Efeito Nefasto em: (2, 3)
Efeito Nefasto em: (6, 5)
Efeito Nefasto em: (3, 1)
Efeito Nefasto em: (9, 4)
Efeito Nefasto em: (0, 7)
Efeito Nefasto em: (6, 0)
Efeito Nefasto em: (3, 6)
Efeito Nefasto em: (10, 2)
Efeito Nefasto em: (1, 5)
Efeito Nefasto em: (4, 2)
Efeito Nefasto em: (10, 11)
Efeito Nefasto em: (3, 1)
Efeito Nefasto em: (7, 7)
Efeito Nefasto em: (10, 10)

MATRIZ COM OS EFEITOS NEFASTOS:
. . . . . # . . . . . #
. . . . . # . . . . . 0 . . . .
. . . . . # 0 . . . . . # . .
. . # . . . . . 0 . . . . .
. . . . . 0 . . . . . # . . .
. # . . . . . A . . . . .
. . . . . # . . . . .
# . . . . . # . . . . .
. . . . . . . . . . A . . . .
. . . . . . . . . . A . . . .
. . . . . . . . . . # . . . .
. . . . . . . . . . # . . . .

```

Figura 21: Testes (parte 3)

```

MATRIZ APOS INSERIR NOVAS ANTENAS:
. . . . . . . . . . .
. . . . . . . . . . 0 . . .
. . . . . 0 . . . . .
. . . . . B . . . . 0 . . . .
. . . . . 0 . . . . .
. . . . . B A . . . . .
. . . . . . . . . . .
. . . . . . . . . . .
. . . . . . . . . . A . . .
. . . . . . . . . . A . . .
. . . . . . . . . . .
. . . . . . . . . . .

LISTA DE ANTENAS:
Frequencia: B | Coordenadas: (5, 5)
Frequencia: B | Coordenadas: (3, 3)
Frequencia: A | Coordenadas: (9, 9)
Frequencia: A | Coordenadas: (8, 8)
Frequencia: A | Coordenadas: (6, 5)
Frequencia: 0 | Coordenadas: (4, 4)
Frequencia: 0 | Coordenadas: (7, 3)
Frequencia: 0 | Coordenadas: (5, 2)
Frequencia: 0 | Coordenadas: (8, 1)

```

Figura 22: Testes (parte 4)

```

LISTA DE EFEITOS NEFASTOS APOS INSERCAO:
Efeito Nefasto em: (11, 0)
Efeito Nefasto em: (2, 3)
Efeito Nefasto em: (6, 5)
Efeito Nefasto em: (3, 1)
Efeito Nefasto em: (9, 4)
Efeito Nefasto em: (0, 7)
Efeito Nefasto em: (6, 0)
Efeito Nefasto em: (3, 6)
Efeito Nefasto em: (10, 2)
Efeito Nefasto em: (1, 5)
Efeito Nefasto em: (4, 2)
Efeito Nefasto em: (10, 11)
Efeito Nefasto em: (3, 1)
Efeito Nefasto em: (7, 7)
Efeito Nefasto em: (10, 10)
Efeito Nefasto em: (1, 1)
Efeito Nefasto em: (7, 7)

MATRIZ COM OS EFEITOS NEFASTOS APOS INSERCAO:
. . . . . # . . . . . #
. # . # . . . . . 0 . . . .
. . . . . # 0 . . . . . # . .
. . # B . . . . . 0 . . . . .
. . . . . 0 . . . . . # . . .
. # . . . . . B A . . . . .
. . . . . # . . . . .
# . . . . . # . . . . .
. . . . . . . . . . A . . . .
. . . . . . . . . . A . . . .
. . . . . . . . . . # . . . .
. . . . . . . . . . # . . . .

```

Figura 23: Testes (parte 5)

```

MATRIZ APOS REMOVER AS ANTENAS INSERIDAS:
. . . . . . . . . . .
. . . . . . . . . . 0 . . .
. . . . . 0 . . . . .
. . . . . 0 . . . . .
. . . . . 0 . . . . .
. . . . . A . . . . .
. . . . . . . . . . .
. . . . . . . . . . A . . .
. . . . . . . . . . A . . .
. . . . . . . . . . .
. . . . . . . . . . .

MATRIZ DE EFEITOS NEFASTOS APOS REMOVER AS ANTENAS INSERIDAS:
. . . . . # . . . . . #
. . . . . # . . . . . 0 . . .
. . . . . # 0 . . . . . # . .
. . # . . . . . 0 . . . . .
. . . . . 0 . . . . . # . . .
. # . . . . . A . . . . .
. . . . . # . . . . .
# . . . . . # . . . . .
. . . . . . . . . . A . . . .
. . . . . . . . . . A . . . .
. . . . . . . . . . # . . . .
. . . . . . . . . . # . . . .

```

Figura 24: Testes (parte 6)

6. Repositório GitHub

O repositório GitHub foi criado e utilizado para armazenar não apenas o código-fonte do projeto, mas também toda a documentação associada, incluindo os ficheiros gerados pelo Doxygen e o relatório do trabalho. Endereço:

https://github.com/fabiofernandes6/TP_EDA_A22996.git

7. Conclusão

Este projeto teve como objetivo a implementação de um sistema para a gestão de antenas e a simulação dos efeitos nefastos causados por interferências entre elas. Utilizando estruturas de dados dinâmicas e programação em C, consegui desenvolver uma solução robusta que permite a inserção, remoção e visualização das antenas, além de calcular e exibir os efeitos nefastos na matriz de coordenadas.

Através da lista ligada para o armazenamento das antenas e efeitos nefastos, o sistema permite uma manipulação eficiente e flexível dos dados. O processo de carga de informações a partir de ficheiros e a atualização das antenas e seus efeitos foram implementados de forma a garantir a simplicidade e a modularidade do código, respeitando os princípios da boa programação.

Além disso, o programa foi estruturado de maneira a lidar com todos os aspetos do problema proposto, incluindo a análise e visualização dos efeitos de interferência, o que torna a solução prática e aplicável a cenários reais de gestão de antenas.

Com este trabalho, consegui consolidar os conceitos de estruturas de dados dinâmicas e modularização, além de explorar a manipulação de memória dinâmica e a leitura/escrita de ficheiros em C, componentes essenciais para o desenvolvimento de soluções eficientes em sistemas computacionais. A implementação da solução permite não só demonstrar os conhecimentos adquiridos ao longo da disciplina, mas também aplicar esses conceitos de forma concreta em um projeto desafiador e de relevância prática.

8. Referências

- [1] “Doxygen download.” Accessed: Mar. 30, 2025. [Online]. Available: <https://www.doxygen.nl/download.html>
- [2] “latex (2) - Online LaTeX Editor Overleaf.” Accessed: Mar. 30, 2025. [Online]. Available: <https://www.overleaf.com/project/67e987ef9fecaabac97c54fc>