

# Relatório Técnico

Estrutura de Dados Avançada  
Sistema de Gerenciamento de Antenas



Engenharia de Sistemas Informáticos  
Instituto Politécnico do Cavado e Ave  
João Loureiro Nº31551

30 de março de 2025

## Conteúdo

<b>1</b>	<b>Introdução</b>	<b>1</b>
<b>2</b>	<b>Objetivos</b>	<b>1</b>
<b>3</b>	<b>Estrutura de Dados</b>	<b>1</b>
<b>4</b>	<b>Funcionalidades Principais</b>	<b>1</b>
4.1	Menu Principal . . . . .	1
4.2	Gerenciamento de Antenas . . . . .	3
<b>5</b>	<b>Detecção de Interferência</b>	<b>3</b>
<b>6</b>	<b>Conclusão</b>	<b>4</b>

## 1 Introdução

O Sistema de Gerenciamento de Antenas é uma aplicação desenvolvida em linguagem C para controle e monitoramento de antenas em um ambiente geográfico delimitado. O software permite:

- Inserção e remoção de antenas
- Visualização em formato de mapa em formato matriz.
- Detecção automática de pontos de interferência (efeito nefasto)
- Carregamento e armazenamento de configurações

## 2 Objetivos

O sistema foi desenvolvido com os seguintes objetivos:

- Fornecer uma ferramenta eficiente para gestão de antenas
- Identificar automaticamente áreas de interferência
- Facilitar o planejamento de instalações
- Oferecer visualização clara da disposição espacial

## 3 Estrutura de Dados

O sistema utiliza uma lista encadeada para armazenar as antenas:

```
1 typedef struct Antena {  
2     char freq;           // Frequencia (A-Z)  
3     int x, y;            // Coordenadas (0 a TAM_MAPA-1)  
4     struct Antena* next; // Ponteiro para proxima  
5 } Antena;
```

## 4 Funcionalidades Principais

### 4.1 Menu Principal

Interface de controle do sistema:

```
1 int main() {  
2     Antena* lista = NULL;  
3     int opcao;  
4     char nomeArquivo[50];  
5  
6     do {  
7         // Menu interativo  
8         printf("\n==== Menu =====\n");  
9         printf("1. Carregar Antenas de Arquivo\n");  
10        printf("2. Inserir Nova Antena\n");  
11        printf("3. Remover Antena\n");
```

```
12 printf("4. Listar Antenas\n");
13 printf("5. Deduzir Efeito Nefasto\n");
14 printf("6. Mostrar Mapa\n");
15 printf("7. Sair\n");
16
17 // Lógica de seleção
18 switch (opcao) {
19     case 1: {
20         // Carregar antenas a partir de um arquivo.
21         printf("Digite o nome do arquivo: ");
22         scanf("%s", nomeArquivo);
23         carregarAntenasDoArquivo(&lista, nomeArquivo);
24         break;
25     }
26     case 2: {
27         // Inserir uma nova antena.
28         char freq; ///< Frequência da antena.
29         int x, y; ///< Posições da antena (X, Y).
30         printf("Frequência da antena (letra): ");
31         scanf(" %c", &freq); ///< Lê a frequência da antena.
32         printf("Posição X: ");
33         scanf("%d", &x); ///< Lê a posição X.
34         printf("Posição Y: ");
35         scanf("%d", &y); ///< Lê a posição Y.
36
37         // Verifica se as posições estão dentro dos limites
do mapa.
38         if (x >= 0 && x < TAM_MAPA && y >= 0 && y < TAM_MAPA) {
39             inserirAntena(&lista, freq, x, y);
40             printf("Antena inserida!\n");
41         } else {
42             printf("Posição inválida!\n");
43         }
44         break;
45     }
46     case 3: {
47         // Remover uma antena da lista.
48         int x, y; ///< Posições da antena a ser removida.
49         printf("Posição X da antena a remover: ");
50         scanf("%d", &x); ///< Lê a posição X da antena.
51         printf("Posição Y da antena a remover: ");
52         scanf("%d", &y); ///< Lê a posição Y da antena.
53         removerAntena(&lista, x, y);
54         break;
55     }
56     case 4:
57         // Listar todas as antenas.
58         listarAntenas(lista);
59         break;
60     case 5:
61         // Deduzir as localizações com efeito nefasto.
62         deduzirEfeitoNefasto(lista);
63         break;
64     case 6:
65         // Mostrar o mapa com as antenas.
66         mostrarMapa(lista);
67         break;
```

```

68         case 7:
69             // Sair do programa.
70             printf("Saindo...\n");
71             break;
72         default:
73             // Caso o usu rio insira uma op o inv lida.
74             printf("Op o inv lida! Tente novamente.\n");
75     }
76     } while (opcao != 7);
77
78     liberarLista(lista);
79     return 0;
80 }

```

## 4.2 Gerenciamento de Antenas

Inserção e Remoção de Antenas:

```

1 void inserirAntena(Antena** head, char freq, int x, int y) {
2     if (x < 0 || x >= TAM_MAPA || y < 0 || y >= TAM_MAPA) {
3         printf("Posicao invalida!\n");
4         return;
5     }
6     Antena* nova = criarAntena(freq, x, y);
7     nova->next = *head;
8     *head = nova;
9 }
10
11 void removerAntena(Antena** head, int x, int y) {
12     Antena* temp = *head;
13     Antena* prev = NULL;
14
15     while (temp != NULL && (temp->x != x || temp->y != y)) {
16         prev = temp;
17         temp = temp->next;
18     }
19
20     if (temp == NULL) {
21         printf("Antena n o encontrada na posi o (%d, %d)!\n", x, y);
22         return;
23     }
24
25     if (prev == NULL) { // Primeira da lista
26         *head = temp->next;
27     } else {
28         prev->next = temp->next;
29     }
30
31     free(temp);
32     printf("Antena removida com sucesso!\n");
33 }

```

## 5 Detecção de Interferência

Algoritmo para identificar pontos de efeito nefasto:

```

1 void deduzirEfeitoNefasto(Antena* head) {
2     char mapa[TAM_MAPA][TAM_MAPA];
3
4     // Inicializa o mapa com '.'
5     for (int i = 0; i < TAM_MAPA; i++) {
6         for (int j = 0; j < TAM_MAPA; j++) {
7             mapa[i][j] = '.';
8         }
9     }
10
11     // Coloca as antenas no mapa
12     Antena* temp = head;
13     while (temp != NULL) {
14         mapa[temp->x][temp->y] = temp->freq;
15         temp = temp->next;
16     }
17
18     // Verifica pares de antenas para interferência
19     for (Antena* ant1 = head; ant1 != NULL; ant1 = ant1->next) {
20         for (Antena* ant2 = ant1->next; ant2 != NULL; ant2 = ant2->next) {
21             if (ant1->freq == ant2->freq) { // Mesma frequência
22                 int dx = ant2->x - ant1->x;
23                 int dy = ant2->y - ant1->y;
24
25                 // Confere alinhamento (horizontal, vertical, diagonal)
26                 if ((dx == 0 || dy == 0 || abs(dx) == abs(dy))) {
27                     int mx = (ant1->x + ant2->x) / 2;
28                     int my = (ant1->y + ant2->y) / 2;
29
30                     // Verifica se o ponto médio está dentro dos
31                     // limites do mapa e não é uma antena
32                     if (mx >= 0 && mx < TAM_MAPA && my >= 0 && my <
33                         TAM_MAPA && mapa[mx][my] == '.') {
34                         mapa[mx][my] = '#'; // Marca o efeito nefasto
35                     }
36                 }
37             }
38         }
39     }
40
41     // Mostra o mapa com efeito nefasto
42     printf("\nMapa com Localizações com Efeito Nefasto:\n");
43     for (int i = 0; i < TAM_MAPA; i++) {
44         for (int j = 0; j < TAM_MAPA; j++) {
45             printf("%c", mapa[i][j]);
46         }
47         printf("\n");
48     }
49 }

```

## 6 Conclusão

O sistema desenvolvido apresenta:

- Arquitetura modular e bem estruturada
- Algoritmos eficientes para detecção de interferências
- Interface intuitiva e fácil de usar
- Código otimizado e com boa documentação

Como trabalhos futuros, recomenda-se:

- Implementação de persistência em banco de dados
- Adição de interface gráfica
- Estudos de performance para grandes volumes de dados
- Repositório GitHub: [Clique aqui para acessar o repositório](#)