Projeto de Gerenciamento de Antenas

Sistema de Controle e Detecção de Interferências

Alexandre Barbosa

30 de março de 2025

Conteúdo

Listings

Capítulo 1

Introdução

Este documento descreve um sistema completo de gerenciamento de antenas implementado em C, com todas as funcionalidades de inserção, remoção, visualização e detecção de interferências.

Capítulo 2

Implementação

2.1 Estruturas de Dados

```
typedef struct Antena {
    char frequencia;  // Frequ ncia (A-Z)
    int x, y;         // Coordenadas
    struct Antena *prox; // Pr xima antena
} Antena;
```

Listing 2.1: Definição da estrutura Antena

2.2 Funções Principais

2.2.1 Criação de Antena

```
Antena* criarAntena(char frequencia, int x, int y) {
    Antena* nova = (Antena*)malloc(sizeof(Antena));
    nova->frequencia = frequencia;
    nova->x = x;
    nova->y = y;
    nova->prox = NULL;
    return nova;
}
```

Listing 2.2: Função criarAntena

2.2.2 Inserção

```
void inserirAntena(Antena **lista, char frequencia, int x, int y) {
    Antena *nova = criarAntena(frequencia, x, y);
    nova->prox = *lista;
    *lista = nova;
    printf("Antena inserida com sucesso!\n");
}
```

Listing 2.3: Função inserirAntena

2.2.3 Remoção

```
void removerAntena(Antena **lista, int x, int y) {
    Antena *atual = *lista, *anterior = NULL;
    while (atual != NULL && (atual->x != x || atual->y != y)) {
        anterior = atual;
        atual = atual->prox;
    }
    if (atual == NULL) {
        printf("Antena nao encontrada!\n");
        return;
    }
    if (anterior == NULL) *lista = atual->prox;
    else anterior->prox = atual->prox;
    free(atual);
    printf("Antena removida com sucesso!\n");
}
```

Listing 2.4: Função removerAntena

2.2.4 Visualização do Mapa

```
void exibirMapa(Antena *lista) {
    char mapa[MAP_SIZE][MAP_SIZE];
    for (int i = 0; i < MAP_SIZE; i++) {
        for (int j = 0; j < MAP_SIZE; j++) {
            mapa[i][j] = '.';
        }
    }

    Antena *atual = lista;
    while (atual != NULL) {
        if (atual->x >= 0 && atual->x < MAP_SIZE &&
            atual->y >= 0 && atual->y < MAP_SIZE) {
            mapa[atual->y][atual->x] = atual->frequencia;
        }
        atual = atual->prox;
    }

    printf("\nMapa de Antenas:\n");
    for (int i = 0; i < MAP_SIZE; i++) {
        for (int j = 0; j < MAP_SIZE; j++) {
            printf("%c ", mapa[i][j]);
        }
        printf("\n");
    }
}
```

Listing 2.5: Função exibirMapa

2.2.5 Detecção de Interferências

```
void detectarInterferencia(Antena *lista) {
    Antena *a1, *a2;
    int interferencia = 0;
```

```
    printf("\nAntenas com interferencia:\n");
    for (a1 = lista; a1 != NULL; a1 = a1->prox) {
        for (a2 = a1->prox; a2 != NULL; a2 = a2->prox) {
            if (a1->frequencia == a2->frequencia) {
                int distancia = abs(a1->x - a2->x) + abs(a1->y - a2->y);
                if (distancia <= 2) {
                    printf("Antenas em (%d, %d) e (%d, %d) com
   frequencia '%c'\n",
                           a1->x, a1->y, a2->x, a2->y, a1->frequencia);
                    interferencia = 1;
                }
            }
        }
    }
    if (!interferencia) printf("Nenhuma interferencia detectada.\n");
}
```

Listing 2.6: Função detectarInterferencia

2.3 Função Principal

```
int main() {
    Antena *lista = NULL;
    int opcao, x, y;
    char freq;

    do {
        printf("\nMenu:\n");
        printf("1 - Inserir Antena\n");
        printf("2 - Remover Antena\n");
        printf("3 - Listar Antenas\n");
        printf("4 - Exibir Mapa\n");
        printf("5 - Detectar Interferencias\n");
        printf("0 - Sair\n");
        printf("Escolha: ");
        scanf("%d", &opcao);

        switch (opcao) {
            case 1:
                printf("Frequencia (A-Z): ");
                scanf(" %c", &freq);
                printf("Coordenadas (x y): ");
                scanf("%d %d", &x, &y);
                inserirAntena(&lista, freq, x, y);
                break;
            case 2:
                printf("Coordenadas para remover (x y): ");
                scanf("%d %d", &x, &y);
                removerAntena(&lista, x, y);
                break;
            case 3:
                imprimirAntenas(lista);
                break;
            case 4:
                exibirMapa(lista);
                break;
```

```
36              case 5:
37                  detectarInterferencia(lista);
38                  break;
39              case 0:
40                  printf("Saindo...\n");
41                  break;
42              default:
43                  printf("Opcao invalida!\n");
44          }
45      } while (opcao != 0);
46
47      return 0;
48  }
```

Listing 2.7: Função main

Capítulo 3

Conclusão

O sistema implementa todas as funcionalidades requeridas de forma eficiente, utilizando estruturas de dados adequadas e algoritmos otimizados para a detecção de interferências.