Trabalho Prático de EDA - Fase 1

Gerado por Doxygen 1.13.2

Capítulo 1

Índice dos componentes

1.1 Lista de componentes

Lista de classes, estruturas, uniões e interfaces com uma breve descrição:

Capítulo 2

Índice dos ficheiros

2.1 Lista de ficheiros

Lista de todos os ficheiros com uma breve descrição:

Capítulo 3

Documentação da classe

3.1 Referência à estrutura Antena

`#include <antenas.h>`

Atributos Públicos

- char frequencia
- int x
- int y
- struct Antena * proximo

3.1.1 Documentação dos dados membro

3.1.1.1 frequencia

`char Antena::frequencia`

3.1.1.2 proximo

`struct Antena* Antena::proximo`

3.1.1.3 x

`int Antena::x`

3.1.1.4 y

`int Antena::y`

A documentação para esta estrutura foi gerada a partir do seguinte ficheiro:

- U:/ESI/2 ano/2 semestre/eda (lufer)/Trabalhos/TP_EDA_F1_a22996/tp_eda/ListaAntenas/antenas.h

3.2 Referência à estrutura EfeitoNefasto

`#include <nefastos.h>`

Atributos Públicos

- int x
- int y
- struct EfeitoNefasto * proximo

3.2.1 Documentação dos dados membro

3.2.1.1 proximo

```
struct EfeitoNefasto* EfeitoNefasto::proximo
```

3.2.1.2 x

```
int EfeitoNefasto::x
```

3.2.1.3 y

```
int EfeitoNefasto::y
```

A documentação para esta estrutura foi gerada a partir do seguinte ficheiro:

- U:/ESI/2 ano/2 semestre/eda (lufer)/Trabalhos/TP_EDA_F1_a22996/tp_eda/EfeitosNefastos/nefastos.h

Capítulo 4

Documentação do ficheiro

4.1 Referência ao ficheiro U:/ESI/2 ano/2 semestre/eda (lufer)/↩ Trabalhos/TP_EDA_F1_a22996/tp_eda/EfeitosNefastos/nefastos.c

Implementação das funções para detetar, exibir e limpar efeitos nefastos causados por antenas.

```
#include <stdio.h>
#include <stdlib.h>
#include "nefastos.h"
```

Funções

- void deduzirEfeitosNefastos (Antena ∗lista, EfeitoNefasto ∗∗efeitos, int largura, int altura)

  Deduz e regista os efeitos nefastos causados pelas antenas.
- void mostrarEfeitosNefastos (EfeitoNefasto ∗lista)

  Mostra no terminal todos os efeitos nefastos detetados.
- void preencherMatrizComEfeitos (char ∗∗matriz, int largura, int altura, EfeitoNefasto ∗efeitos)

  Adiciona os efeitos nefastos à matriz, marcando-os com '#'.
- void limparEfeitosNefastos (EfeitoNefasto ∗∗lista)

  Liberta a memória associada à lista de efeitos nefastos.

4.1.1 Descrição detalhada

Implementação das funções para detetar, exibir e limpar efeitos nefastos causados por antenas.

Autor

Fábio Fernandes

Data

30/03/2025

O código analisa uma lista de antenas e determina os pontos onde ocorre o efeito nefasto com base nas condições especificadas: acontece quando duas antenas da mesma frequência estão alinhadas e uma está exatamente o dobro da distância da outra.

4.1.2 Documentação das funções

4.1.2.1 deduzirEfeitosNefastos()

```
void deduzirEfeitosNefastos (
            Antena * lista,
            EfeitoNefasto ** efeitos,
            int largura,
            int altura)
```

Deduz e regista os efeitos nefastos causados pelas antenas.

Parâmetros

lista	Apontador para a lista de antenas.
efeitos	Apontador para a lista onde serão armazenados os efeitos nefastos.
largura	Largura do espaço disponível (matriz).
altura	Altura do espaço disponível (matriz).

```
00025 {
00026     // Percorrer todas as antenas existentes
00027     for (Antena* a1 = lista; a1 != NULL; a1 = a1->proximo)
00028     {
00029         for (Antena* a2 = a1->proximo; a2 != NULL; a2 = a2->proximo)
00030         {
00031             // Só interessa se forem da mesma frequência
00032             if (a1->frequencia == a2->frequencia)
00033             {
00034                 // Determina o deslocamento entre as duas antenas
00035                 int dx = a2->x - a1->x;
00036                 int dy = a2->y - a1->y;
00037
00038                 // Calcula as posições onde surgem os efeitos nefastos
00039                 int ex1_x = a1->x - dx;
00040                 int ex1_y = a1->y - dy;
00041                 int ex2_x = a2->x + dx;
00042                 int ex2_y = a2->y + dy;
00043
00044                 // Se o primeiro ponto calculado estiver dentro dos limites, adiciona à lista
00045                 if (ex1_x >= 0 && ex1_x < largura && ex1_y >= 0 && ex1_y < altura)
00046                 {
00047                     EfeitoNefasto* novo1 = (EfeitoNefasto*)malloc(sizeof(EfeitoNefasto));
00048                     if (!novo1)
00049                     {
00050                         printf("Erro ao alocar memória para efeito nefasto!\n");
00051                         return;
00052                     }
00053                     novo1->x = ex1_x;
00054                     novo1->y = ex1_y;
00055                     novo1->proximo = *efeitos;
00056                     *efeitos = novo1;
00057                 }
00058
00059                 // Se o segundo ponto calculado estiver dentro dos limites, adiciona à lista
00060                 if (ex2_x >= 0 && ex2_x < largura && ex2_y >= 0 && ex2_y < altura)
00061                 {
00062                     EfeitoNefasto* novo2 = (EfeitoNefasto*)malloc(sizeof(EfeitoNefasto));
00063                     if (!novo2)
00064                     {
00065                         printf("Erro ao alocar memória para efeito nefasto!\n");
00066                         return;
00067                     }
00068                     novo2->x = ex2_x;
00069                     novo2->y = ex2_y;
00070                     novo2->proximo = *efeitos;
00071                     *efeitos = novo2;
00072                 }
00073             }
00074         }
00075     }
00076 }
```

4.1.2.2 limparEfeitosNefastos()

```
void limparEfeitosNefastos (
            EfeitoNefasto ** lista)
```

Liberta a memória associada à lista de efeitos nefastos.

Parâmetros

lista	Apontador para o apontador da lista de efeitos nefastos.

```
00128 {
00129     // Percorre a lista e liberta cada nó
00130     while (*lista)
00131     {
00132         EfeitoNefasto* temp = *lista;
00133         *lista = (*lista)->proximo;
00134         free(temp);
00135     }
00136     *lista = NULL; // Garante que a lista fica vazia
00137 }
```

4.1.2.3 mostrarEfeitosNefastos()

```
void mostrarEfeitosNefastos (
            EfeitoNefasto * lista)
```

Mostra no terminal todos os efeitos nefastos detetados.

Parâmetros

lista	Apontador para a lista de efeitos nefastos.

```
00084 {
00085     if (!lista)
00086     {
00087         printf("Nenhum efeito nefasto detetado.\n");
00088         return;
00089     }
00090
00091     // Percorre a lista e imprime cada efeito encontrado
00092     while (lista)
00093     {
00094         printf("Efeito Nefasto em: (%d, %d)\n", lista->x, lista->y);
00095         lista = lista->proximo;
00096     }
00097 }
```

4.1.2.4 preencherMatrizComEfeitos()

```
void preencherMatrizComEfeitos (
            char ** matriz,
            int largura,
            int altura,
            EfeitoNefasto * efeitos)
```

Adiciona os efeitos nefastos à matriz, marcando-os com '#'.

Parâmetros

matriz	Matriz que representa o espaço.

largura	Largura da matriz.
altura	Altura da matriz.
efeitos	Apontador para a lista de efeitos nefastos.

```
00108 {
00109     // Percorre a lista de efeitos e marca-os na matriz
00110     for (EfeitoNefasto* aux = efeitos; aux; aux = aux->proximo)
00111     {
00112         if (aux->x >= 0 && aux->x < largura && aux->y >= 0 && aux->y < altura)
00113         {
00114             if (matriz[aux->y][aux->x] == '.')
00115             {  // Apenas marca como '#' se a posição ainda não tiver sido ocupada
00116                 matriz[aux->y][aux->x] = '#';
00117             }
00118         }
00119     }
00120 }
```

4.2 Referência ao ficheiro U:/ESI/2 ano/2 semestre/eda (lufer)/↩ Trabalhos/TP_EDA_F1_a22996/tp_eda/EfeitosNefastos/nefastos.h

Declarações das funções para gerir efeitos nefastos causados por antenas.

```
#include "../ListaAntenas/antenas.h"
```

Componentes

- struct EfeitoNefasto

Definições de tipos

- typedef struct EfeitoNefasto EfeitoNefasto

Funções

- void deduzirEfeitosNefastos (Antena ∗lista, EfeitoNefasto ∗∗efeitos, int largura, int altura)

  Deduz e regista os efeitos nefastos causados pelas antenas.
- void mostrarEfeitosNefastos (EfeitoNefasto ∗lista)

  Mostra no terminal todos os efeitos nefastos detetados.
- void preencherMatrizComEfeitos (char ∗∗matriz, int largura, int altura, EfeitoNefasto ∗efeitos)

  Adiciona os efeitos nefastos à matriz, marcando-os com '#'.
- void limparEfeitosNefastos (EfeitoNefasto ∗∗lista)

  Liberta a memória associada à lista de efeitos nefastos.

4.2.1 Descrição detalhada

Declarações das funções para gerir efeitos nefastos causados por antenas.

Autor

Fábio Fernandes

Data

30/03/2025

4.2.2 Documentação dos tipos

4.2.2.1 EfeitoNefasto

```
typedef struct EfeitoNefasto EfeitoNefasto
```

4.2.3 Documentação das funções

4.2.3.1 deduzirEfeitosNefastos()

```
void deduzirEfeitosNefastos (
            Antena * lista,
            EfeitoNefasto ** efeitos,
            int largura,
            int altura)
```

Deduz e regista os efeitos nefastos causados pelas antenas.

Parâmetros

lista	Apontador para a lista de antenas.
efeitos	Apontador para a lista onde serão armazenados os efeitos nefastos.
largura	Largura do espaço disponível (matriz).
altura	Altura do espaço disponível (matriz).

```
00025 {
00026     // Percorrer todas as antenas existentes
00027     for (Antena* a1 = lista; a1 != NULL; a1 = a1->proximo)
00028     {
00029         for (Antena* a2 = a1->proximo; a2 != NULL; a2 = a2->proximo)
00030         {
00031             // Só interessa se forem da mesma frequência
00032             if (a1->frequencia == a2->frequencia)
00033             {
00034                 // Determina o deslocamento entre as duas antenas
00035                 int dx = a2->x - a1->x;
00036                 int dy = a2->y - a1->y;
00037 
00038                 // Calcula as posições onde surgem os efeitos nefastos
00039                 int ex1_x = a1->x - dx;
00040                 int ex1_y = a1->y - dy;
00041                 int ex2_x = a2->x + dx;
00042                 int ex2_y = a2->y + dy;
00043 
00044                 // Se o primeiro ponto calculado estiver dentro dos limites, adiciona à lista
00045                 if (ex1_x >= 0 && ex1_x < largura && ex1_y >= 0 && ex1_y < altura)
00046                 {
00047                     EfeitoNefasto* novo1 = (EfeitoNefasto*)malloc(sizeof(EfeitoNefasto));
00048                     if (!novo1)
00049                     {
00050                         printf("Erro ao alocar memória para efeito nefasto!\n");
00051                         return;
00052                     }
00053                     novo1->x = ex1_x;
00054                     novo1->y = ex1_y;
00055                     novo1->proximo = *efeitos;
00056                     *efeitos = novo1;
00057                 }
00058 
00059                 // Se o segundo ponto calculado estiver dentro dos limites, adiciona à lista
00060                 if (ex2_x >= 0 && ex2_x < largura && ex2_y >= 0 && ex2_y < altura)
00061                 {
00062                     EfeitoNefasto* novo2 = (EfeitoNefasto*)malloc(sizeof(EfeitoNefasto));
00063                     if (!novo2)
00064                     {
00065                         printf("Erro ao alocar memória para efeito nefasto!\n");
00066                         return;
00067                     }
```

```
00068                     novo2->x = ex2_x;
00069                     novo2->y = ex2_y;
00070                     novo2->proximo = *efeitos;
00071                     *efeitos = novo2;
00072                 }
00073             }
00074         }
00075     }
00076 }
```

4.2.3.2 limparEfeitosNefastos()

```
void limparEfeitosNefastos (
            EfeitoNefasto ** lista)
```

Liberta a memória associada à lista de efeitos nefastos.

Parâmetros

lista	Apontador para o apontador da lista de efeitos nefastos.

```
00128 {
00129     // Percorre a lista e liberta cada nó
00130     while (*lista)
00131     {
00132         EfeitoNefasto* temp = *lista;
00133         *lista = (*lista)->proximo;
00134         free(temp);
00135     }
00136     *lista = NULL; // Garante que a lista fica vazia
00137 }
```

4.2.3.3 mostrarEfeitosNefastos()

```
void mostrarEfeitosNefastos (
            EfeitoNefasto * lista)
```

Mostra no terminal todos os efeitos nefastos detetados.

Parâmetros

lista	Apontador para a lista de efeitos nefastos.

```
00084 {
00085     if (!lista)
00086     {
00087         printf("Nenhum efeito nefasto detetado.\n");
00088         return;
00089     }
00090
00091     // Percorre a lista e imprime cada efeito encontrado
00092     while (lista)
00093     {
00094         printf("Efeito Nefasto em: (%d, %d)\n", lista->x, lista->y);
00095         lista = lista->proximo;
00096     }
00097 }
```

4.2.3.4 preencherMatrizComEfeitos()

```
void preencherMatrizComEfeitos (
            char ** matriz,
            int largura,
            int altura,
            EfeitoNefasto * efeitos)
```

Adiciona os efeitos nefastos à matriz, marcando-os com '#'.

Parâmetros

matriz	Matriz que representa o espaço.
largura	Largura da matriz.
altura	Altura da matriz.
efeitos	Apontador para a lista de efeitos nefastos.

```
00108 {
00109     // Percorre a lista de efeitos e marca-os na matriz
00110     for (EfeitoNefasto* aux = efeitos; aux; aux = aux->proximo)
00111     {
00112         if (aux->x >= 0 && aux->x < largura && aux->y >= 0 && aux->y < altura)
00113         {
00114             if (matriz[aux->y][aux->x] == '.')
00115             {  // Apenas marca como '#' se a posição ainda não tiver sido ocupada
00116                 matriz[aux->y][aux->x] = '#';
00117             }
00118         }
00119     }
00120 }
```

4.3 nefastos.h

Ir para a documentação deste ficheiro.

```
00001
00007
00008 #pragma once
00009
00010 #include "../ListaAntenas/antenas.h"
00011
00012 // Estrutura para representar um ponto com efeito nefasto
00013 typedef struct EfeitoNefasto {
00014     int x, y;                       // Coordenadas do efeito nefasto
00015     struct EfeitoNefasto *proximo;  // Apontador para o próximo efeito nefasto
00016 } EfeitoNefasto;
00017
00018 // Funções para gerir efeitos nefastos
00019 void deduzirEfeitosNefastos(Antena* lista, EfeitoNefasto** efeitos, int largura, int altura);
00020 void mostrarEfeitosNefastos(EfeitoNefasto *lista);
00021 void preencherMatrizComEfeitos(char** matriz, int largura, int altura, EfeitoNefasto* efeitos);
00022 void limparEfeitosNefastos(EfeitoNefasto **lista);
```

4.4 Referência ao ficheiro U:/ESI/2 ano/2 semestre/eda (lufer)/Trabalhos/TP_EDA_F1_a22996/tp_eda/ListaAntenas/antenas.c

Implementação das funções para gerir a lista de antenas.

```
#include "antenas.h"
#include <stdio.h>
#include <stdlib.h>
```

Funções

- Antena ∗ criarAntena (char frequencia, int x, int y)

  Cria uma nova antena e retorna um apontador para ela.
- void inserirAntena (Antena ∗∗lista, char frequencia, int x, int y)

  Insere uma nova antena na lista.
- void removerAntena (Antena ∗∗lista, int x, int y)

Remove uma antena com coordenadas específicas.

- void mostrarAntenas (Antena ∗lista)

  Mostra todas as antenas registadas.

- void limparAntenas (Antena ∗∗lista)

  Liberta a memória de todas as antenas da lista.

- void carregarFicheiro (Antena ∗∗lista, const char ∗nomeFicheiro)

  Carrega uma lista de antenas a partir de um ficheiro.

4.4.1 Descrição detalhada

Implementação das funções para gerir a lista de antenas.

Autor

Fábio Fernandes

Data

30/03/2025

Este módulo permite criar, inserir, remover e exibir antenas, além de carregar uma lista de antenas a partir de um ficheiro.

4.4.2 Documentação das funções

4.4.2.1 carregarFicheiro()

```
void carregarFicheiro (
            Antena ** lista,
            const char * nomeFicheiro)
```

Carrega uma lista de antenas a partir de um ficheiro.

Parâmetros

lista	Apontador para o Apontador da lista de antenas.
nomeFicheiro	Nome do ficheiro a ser lido.

```
00128 {
00129     FILE* ficheiro = fopen(nomeFicheiro, "r");
00130     if (!ficheiro)
00131     {
00132         printf("Erro ao abrir o ficheiro %s!\n", nomeFicheiro);
00133         return;
00134     }
00135
00136     char linha[100];
00137     int y = 0;
00138
00139     // Lê o ficheiro linha a linha
00140     while (fgets(linha, sizeof(linha), ficheiro))
00141     {
00142         // Percorre cada carácter da linha
00143         for (int x = 0; linha[x] != '\0' && linha[x] != '\n'; x++)
00144         {
00145             if (linha[x] != '.')
00146             {  // Assume que '.' significa "vazio"
00147                 inserirAntena(lista, linha[x], x, y);
00148             }
00149         }
00150         y++; // Incrementa a coordenada Y
00151     }
00152     fclose(ficheiro);
00153 }
```

4.4.2.2 criarAntena()

```
Antena * criarAntena (
            char frequencia,
            int x,
            int y)
```

Cria uma nova antena e retorna um apontador para ela.

Parâmetros

frequencia	Carácter representando a frequência da antena.
x	Coordenada X da antena.
y	Coordenada Y da antena.

Retorna

Apontador para a nova antena ou NULL em caso de erro.

```
00024 {
00025     Antena* nova = (Antena*)malloc(sizeof(Antena));
00026     if (!nova)
00027     {
00028         printf("Erro ao alocar memória para a antena!\n");
00029         return NULL;
00030     }
00031     nova->frequencia = frequencia;
00032     nova->x = x;
00033     nova->y = y;
00034     nova->proximo = NULL;
00035     return nova;
00036 }
```

4.4.2.3 inserirAntena()

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

Insere uma nova antena na lista.

Parâmetros

lista	Apontador para o ponteiro da lista de antenas.
frequencia	Carácter representando a frequência da antena.
x	Coordenada X da antena.
y	Coordenada Y da antena.

```
00047 {
00048     Antena* nova = criarAntena(frequencia, x, y);
00049     if (!nova) return;
00050
00051     // Insere no início da lista
00052     nova->proximo = *lista;
00053     *lista = nova;
00054 }
```

4.4.2.4 limparAntenas()

```
void limparAntenas (
            Antena ** lista)
```

Liberta a memória de todas as antenas da lista.

Parâmetros

lista	Apontador para o Apontador da lista de antenas.

```
00109 {
00110     // Percorre a lista e liberta cada antena
00111     while (*lista)
00112     {
00113         Antena* temp = *lista;
00114         *lista = (*lista)->proximo;
00115         free(temp);
00116     }
00117
00118     *lista = NULL; // Garante que a lista fica vazia
00119 }
```

4.4.2.5 mostrarAntenas()

```
void mostrarAntenas (
            Antena * lista)
```

Mostra todas as antenas registadas.

Parâmetros

lista	Apontador para a lista de antenas.

```
00088 {
00089     if (!lista)
00090     {
00091         printf("Nenhuma antena registada.\n");
00092         return;
00093     }
00094
00095     // Percorre a lista e imprime cada antena
00096     while (lista)
00097     {
00098         printf("Frequencia: %c | Coordenadas: (%d, %d)\n", lista->frequencia, lista->x, lista->y);
00099         lista = lista->proximo;
00100     }
00101 }
```

4.4.2.6 removerAntena()

```
void removerAntena (
            Antena ** lista,
            int x,
            int y)
```

Remove uma antena com coordenadas específicas.

Parâmetros

lista	Apontador para o ponteiro da lista de antenas.
x	Coordenada X da antena a remover.
y	Coordenada Y da antena a remover.

```
00064 {
00065     Antena *atual = *lista, *anterior = NULL;
00066
00067     // Percorre a lista até encontrar a antena com as coordenadas especificadas
00068     while (atual != NULL && (atual->x != x || atual->y != y))
00069     {
00070         anterior = atual;
00071         atual = atual->proximo;
00072     }
00073     if (atual == NULL) return; // Se não encontrou, sai da função
00074
00075     // Se a antena a remover for a primeira da lista
00076     if (anterior == NULL) *lista = atual->proximo;
00077     else anterior->proximo = atual->proximo;
00078
00079     free(atual); // Liberta a memória da antena removida
00080 }
```

4.5 Referência ao ficheiro U:/ESI/2 ano/2 semestre/eda (lufer)/Trabalhos/TP_EDA_F1_a22996/tp_eda/ListaAntenas/antenas.h

Declarações das funções para gerir a lista de antenas.

Componentes

- struct Antena

Definições de tipos

- typedef struct Antena Antena

Funções

- Antena * criarAntena (char frequencia, int x, int y)

  Cria uma nova antena e retorna um apontador para ela.
- void inserirAntena (Antena **lista, char frequencia, int x, int y)

  Insere uma nova antena na lista.
- void removerAntena (Antena **lista, int x, int y)

  Remove uma antena com coordenadas específicas.
- void mostrarAntenas (Antena *lista)

  Mostra todas as antenas registadas.
- void limparAntenas (Antena **lista)

  Liberta a memória de todas as antenas da lista.
- void carregarFicheiro (Antena **lista, const char *nomeFicheiro)

  Carrega uma lista de antenas a partir de um ficheiro.

4.5.1 Descrição detalhada

Declarações das funções para gerir a lista de antenas.

Autor

Fábio Fernandes

Data

30/03/2025

4.5.2 Documentação dos tipos

4.5.2.1 Antena

```
typedef struct Antena Antena
```

4.5.3 Documentação das funções

4.5.3.1 carregarFicheiro()

```
void carregarFicheiro (
            Antena ** lista,
            const char * nomeFicheiro)
```

Carrega uma lista de antenas a partir de um ficheiro.

Parâmetros

lista	Apontador para o Apontador da lista de antenas.
nomeFicheiro	Nome do ficheiro a ser lido.

```
00128 {
00129     FILE* ficheiro = fopen(nomeFicheiro, "r");
00130     if (!ficheiro)
00131     {
00132         printf("Erro ao abrir o ficheiro %s!\n", nomeFicheiro);
00133         return;
00134     }
00135
00136     char linha[100];
00137     int y = 0;
00138
00139     // Lê o ficheiro linha a linha
00140     while (fgets(linha, sizeof(linha), ficheiro))
00141     {
00142         // Percorre cada carácter da linha
00143         for (int x = 0; linha[x] != '\0' && linha[x] != '\n'; x++)
00144         {
00145             if (linha[x] != '.')
00146             {  // Assume que '.' significa "vazio"
00147                 inserirAntena(lista, linha[x], x, y);
00148             }
00149         }
00150         y++; // Incrementa a coordenada Y
00151     }
00152     fclose(ficheiro);
00153 }
```

4.5.3.2 criarAntena()

```
Antena * criarAntena (
            char frequencia,
            int x,
            int y)
```

Cria uma nova antena e retorna um apontador para ela.

Parâmetros

frequencia	Carácter representando a frequência da antena.
x	Coordenada X da antena.
y	Coordenada Y da antena.

Retorna

Apontador para a nova antena ou NULL em caso de erro.

```
00024 {
00025     Antena* nova = (Antena*)malloc(sizeof(Antena));
00026     if (!nova)
00027     {
00028         printf("Erro ao alocar memória para a antena!\n");
00029         return NULL;
00030     }
00031     nova->frequencia = frequencia;
00032     nova->x = x;
00033     nova->y = y;
00034     nova->proximo = NULL;
00035     return nova;
00036 }
```

4.5.3.3 inserirAntena()

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

Insere uma nova antena na lista.

Parâmetros

lista	Apontador para o ponteiro da lista de antenas.
frequencia	Carácter representando a frequência da antena.
x	Coordenada X da antena.
y	Coordenada Y da antena.

```
00047 {
00048     Antena* nova = criarAntena(frequencia, x, y);
00049     if (!nova) return;
00050
00051     // Insere no início da lista
00052     nova->proximo = *lista;
00053     *lista = nova;
00054 }
```

4.5.3.4 limparAntenas()

```
void limparAntenas (
            Antena ** lista)
```

Liberta a memória de todas as antenas da lista.

Parâmetros

lista	Apontador para o Apontador da lista de antenas.

```
00109 {
00110     // Percorre a lista e liberta cada antena
00111     while (*lista)
00112     {
00113         Antena* temp = *lista;
00114         *lista = (*lista)->proximo;
00115         free(temp);
00116     }
00117
00118     *lista = NULL; // Garante que a lista fica vazia
00119 }
```

4.5.3.5 mostrarAntenas()

```
void mostrarAntenas (
            Antena * lista)
```

Mostra todas as antenas registadas.

Parâmetros

lista	Apontador para a lista de antenas.

```
00088 {
00089     if (!lista)
00090     {
00091         printf("Nenhuma antena registada.\n");
00092         return;
00093     }
00094
00095     // Percorre a lista e imprime cada antena
00096     while (lista)
00097     {
00098         printf("Frequencia: %c | Coordenadas: (%d, %d)\n", lista->frequencia, lista->x, lista->y);
00099         lista = lista->proximo;
00100     }
00101 }
```

4.5.3.6 removerAntena()

```
void removerAntena (
            Antena ** lista,
            int x,
            int y)
```

Remove uma antena com coordenadas específicas.

Parâmetros

lista	Apontador para o ponteiro da lista de antenas.
x	Coordenada X da antena a remover.
y	Coordenada Y da antena a remover.

```
00064 {
00065     Antena *atual = *lista, *anterior = NULL;
00066
00067     // Percorre a lista até encontrar a antena com as coordenadas especificadas
00068     while (atual != NULL && (atual->x != x || atual->y != y))
00069     {
00070         anterior = atual;
00071         atual = atual->proximo;
00072     }
00073     if (atual == NULL) return; // Se não encontrou, sai da função
00074
00075     // Se a antena a remover for a primeira da lista
00076     if (anterior == NULL) *lista = atual->proximo;
00077     else anterior->proximo = atual->proximo;
00078
00079     free(atual); // Liberta a memória da antena removida
00080 }
```

4.6 antenas.h

Ir para a documentação deste ficheiro.

```
00001
00007
00008 #pragma once
00009
00010 // Estrutura para representar uma antena
00011 typedef struct Antena {
00012     char frequencia;   // Frequência da antena
00013     int x, y;          // Coordenadas da antena
00014     struct Antena *proximo; // Apontador para a próxima antena (lista ligada)
00015 } Antena;
00016
00017 // Funções para gerir antenas
00018 Antena* criarAntena(char frequencia, int x, int y);
00019 void inserirAntena(Antena **lista, char frequencia, int x, int y);
00020 void removerAntena(Antena **lista, int x, int y);
00021 void mostrarAntenas(Antena *lista);
00022 void limparAntenas(Antena **lista);
00023 void carregarFicheiro(Antena **lista, const char* nomeFicheiro);
```

4.7 Referência ao ficheiro U:/ESI/2 ano/2 semestre/eda (lufer)/Trabalhos/TP_EDA_F1_a22996/tp_eda/main.c

Programa principal para gerir antenas e efeitos nefastos.

```
#include <stdio.h>
#include <stdlib.h>
#include "ListaAntenas/antenas.h"
#include "EfeitosNefastos/nefastos.h"
```

Funções

- void mostrarMatriz (Antena *lista, EfeitoNefasto *efeitos, int largura, int altura)

  Mostra a matriz no terminal, preenchendo-a com os efeitos nefastos ou somente com as antenas.
- int main ()

4.7.1 Descrição detalhada

Programa principal para gerir antenas e efeitos nefastos.

Autor

Fábio Fernandes

Data

30/03/2025

4.7.2 Documentação das funções

4.7.2.1 main()

```
int main ()
00072 {
00073     Antena* listaAntenas = NULL;
00074     EfeitoNefasto* listaEfeitos = NULL;
00075
00076     // Carregar a matriz original a partir do ficheiro
00077     carregarFicheiro(&listaAntenas, "estrutura.txt");
00078
00079     // Exibir a matriz original
00080     printf("\nMATRIZ ORIGINAL:\n");
00081     mostrarMatriz(listaAntenas, listaEfeitos, 12, 12);
00082
00083     // Exibir as coordenadas das antenas
00084     printf("\nLISTA DE ANTENAS:\n");
00085     mostrarAntenas(listaAntenas);
00086
00087     // Deduzir e exibir os efeitos nefastos iniciais
00088     printf("\nLISTA DE EFEITOS NEFASTOS:\n");
00089     deduzirEfeitosNefastos(listaAntenas, &listaEfeitos, 12, 12);
00090     mostrarEfeitosNefastos(listaEfeitos);
00091
00092     // Mostrar matriz com os efeitos nefastos
00093     printf("\nMATRIZ COM OS EFEITOS NEFASTOS:\n");
00094     mostrarMatriz(listaAntenas, listaEfeitos, 12, 12);
00095
00096     // Inserir novas antenas, B(3, 3) e B(5, 5), e mostrar a matriz
00097     printf("\nMATRIZ APOS INSERIR NOVAS ANTENAS:\n");
00098     inserirAntena(&listaAntenas, 'B', 3, 3);
00099     inserirAntena(&listaAntenas, 'B', 5, 5);
00100     limparEfeitosNefastos(&listaEfeitos);
00101     mostrarMatriz(listaAntenas, listaEfeitos, 12, 12);
00102
00103     //Mostrar lista de antenas após inserção, recalcular os efeitos nefastos, e mostrar a matriz com
      os efeitos nefastos e sem os efeitos nefastos
00104     printf("\nLISTA DE ANTENAS:\n");
00105     mostrarAntenas(listaAntenas);
00106     printf("\nLISTA DE EFEITOS NEFASTOS APOS INSERCAO:\n");
00107     deduzirEfeitosNefastos(listaAntenas, &listaEfeitos, 12, 12);
00108     mostrarEfeitosNefastos(listaEfeitos);
00109     printf("\nMATRIZ COM OS EFEITOS NEFASTOS APOS INSERCAO:\n");
00110     mostrarMatriz(listaAntenas, listaEfeitos, 12, 12);
00111
00112     // Remover as antenas B(3, 3) e B(5, 5)
00113     printf("\nMATRIZ APOS REMOVER AS ANTENAS INSERIDAS:\n");
00114     removerAntena(&listaAntenas, 3, 3);
00115     removerAntena(&listaAntenas, 5, 5);
00116     limparEfeitosNefastos(&listaEfeitos);
00117     mostrarMatriz(listaAntenas, listaEfeitos, 12, 12);
00118
00119     // Recalcular os efeitos nefastos após remover as antenas B(3, 3) e B(5, 5)
00120     printf("\nMATRIZ DE EFEITOS NEFASTOS APOS REMOVER AS ANTENAS INSERIDAS:\n");
00121     deduzirEfeitosNefastos(listaAntenas, &listaEfeitos, 12, 12);
00122     mostrarMatriz(listaAntenas, listaEfeitos, 12, 12);
00123
00124     // Liberar memória alocada
00125     limparEfeitosNefastos(&listaEfeitos);
00126     limparAntenas(&listaAntenas);
00127
00128     return 0;
00129 }
```

4.7.2.2 mostrarMatriz()

```
void mostrarMatriz (
            Antena * lista,
            EfeitoNefasto * efeitos,
            int largura,
            int altura)
```

Mostra a matriz no terminal, preenchendo-a com os efeitos nefastos ou somente com as antenas.

Parâmetros

lista	Apontador para a lista de antenas.
efeitos	Apontador para a lista de efeitos nefastos.
largura	Largura da matriz.
altura	Altura da matriz.

```
00023 {
00024     // Aloca memória para a matriz (altura linhas)
00025     char** matriz = (char**)malloc(altura * sizeof(char*));
00026     for (int i = 0; i < altura; i++)
00027     {
00028         // Para cada linha, aloca memória para armazenar a largura, colunas
00029         matriz[i] = (char*)malloc(largura * sizeof(char));
00030     }
00031
00032     // Inicializa a matriz com o caracter '.' para representar áreas vazias
00033     for (int i = 0; i < altura; i++)
00034     {
00035         for (int j = 0; j < largura; j++)
00036         {
00037             matriz[i][j] = '.';
00038         }
00039     }
00040
00041     // Percorre a lista de antenas e marca suas posições na matriz
00042     for (Antena* aux = lista; aux; aux = aux->proximo)
00043     {
00044         // Insere a frequência da antena na posição correspondente
00045         // Supondo que 'x' e 'y' estão dentro dos limites da matriz
00046         matriz[aux->y][aux->x] = aux->frequencia;
00047     }
00048
00049     // Chama a função que preenche a matriz com os efeitos nefastos
00050     preencherMatrizComEfeitos(matriz, largura, altura, efeitos);
00051
00052     // Imprime a matriz na tela
00053     for (int i = 0; i < altura; i++)
00054     {
00055         for (int j = 0; j < largura; j++)
00056         {
00057             printf("%c ", matriz[i][j]); // Exibe cada elemento seguido de um espaço
00058         }
00059         printf("\n"); // Nova linha após imprimir cada linha da matriz
00060     }
00061
00062     // Liberta a memória alocada para a matriz
00063     for (int i = 0; i < altura; i++)
00064     {
00065         free(matriz[i]); // Liberta cada linha individualmente
00066     }
00067     free(matriz); // Liberta o apontador principal da matriz
00068 }
```

Índice