a22997_Projeto_EDA

1.0

Generated by Doxygen 1.13.2

Chapter 1

Data Structure Index

1.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

Chapter 3

Data Structure Documentation

3.1 ANT Struct Reference

Estrutura que representa uma antena.

`#include <funcao.h>`

Collaboration diagram for ANT:

Data Fields

- char freqAntena
- int x
- int y
- struct ANT * proxAntena

3.1.1 Detailed Description

Estrutura que representa uma antena.

Esta estrutura armazena a frequência da antena e suas coordenadas (x, y).

Definition at line 22 of file funcao.h.

3.1.2 Field Documentation

3.1.2.1 freqAntena

`char freqAntena`

Frequência da antena.

Definition at line 23 of file funcao.h.

3.1.2.2 proxAntena

```
struct ANT * proxAntena
```

Ponteiro para a próxima antena na lista.

Definition at line 26 of file funcao.h.

3.1.2.3 x

```
int x
```

Coordenada x da antena.

Definition at line 24 of file funcao.h.

3.1.2.4 y

```
int y
```

Coordenada y da antena.

Definition at line 25 of file funcao.h.

The documentation for this struct was generated from the following files:

- projetoeda/funcao.h
- projetoedalib/funcao.h

3.2 NEF Struct Reference

Estrutura que representa um nefasto.

```
#include <funcao.h>
```

Collaboration diagram for NEF:

Data Fields

- int x
- int y
- struct NEF * proxNef

3.2.1 Detailed Description

Estrutura que representa um nefasto.

Esta estrutura armazena as coordenadas (x, y) de um nefasto.

Definition at line 34 of file funcao.h.

3.2.2 Field Documentation

3.2.2.1 proxNef

```
struct NEF * proxNef
```

Ponteiro para o próximo nefasto na lista.

Definition at line 37 of file funcao.h.

3.2.2.2 x

```
int x
```

Coordenada x do nefasto.

Definition at line 35 of file funcao.h.

3.2.2.3 y

```
int y
```

Coordenada y do nefasto.

Definition at line 36 of file funcao.h.

The documentation for this struct was generated from the following files:

- projetoeda/funcao.h
- projetoedalib/funcao.h

Chapter 4

File Documentation

4.1 projetoeda/funcao.h File Reference

Definições de estruturas e protótipos de funções para manipulação de antenas e nefastos.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
```

Include dependency graph for funcao.h: This graph shows which files directly or indirectly include this file:

Data Structures

- struct ANT

  Estrutura que representa uma antena.
- struct NEF

  Estrutura que representa um nefasto.

Macros

- #define MAX_LINHAS 100
- #define MAX_COLUNAS 100

Typedefs

- typedef struct ANT ANT

  Estrutura que representa uma antena.
- typedef struct NEF NEF

  Estrutura que representa um nefasto.

Functions

- void inserirAntena (ANT **lista, char freq, int x, int y)

  Insere uma nova antena na lista de antenas.
- ANT * LerLista (const char *nomeFicheiro, char matriz[MAX_LINHAS][MAX_COLUNAS], int *linhas, int *colunas)

  Lê uma lista de antenas a partir de um arquivo e preenche uma matriz.
- void apresentarLista (ANT *lista)

  Apresenta a lista de antenas.
- void apresentarMatriz (char matriz[MAX_LINHAS][MAX_COLUNAS], int linhas, int colunas)

  Apresenta a matriz.
- void inserirNefasto (NEF **lista, int x, int y)

  Insere um nefasto na lista de nefastos.
- NEF * matrizNefastos (char matriz[MAX_LINHAS][MAX_COLUNAS], int linhas, int colunas)

  Detecta e insere nefastos na matriz.
- void apresentarListaNef (NEF *lista)

  Apresenta a lista de nefastos.
- void igualarMatrizes (char matrizNef[MAX_LINHAS][MAX_COLUNAS], char matriz[MAX_LINHAS][MAX_COLUNAS], int linhas, int colunas)

  Copia os dados de uma matriz para outra.
- void atualizarMatriz (ANT **lista, char freq, int x, int y, char matriz[MAX_LINHAS][MAX_COLUNAS])

  Atualiza a matriz e a lista de antenas com uma nova antena.
- void removerAntena (ANT **lista, int x, int y, char matriz[MAX_LINHAS][MAX_COLUNAS])

  Remove uma antena da lista e da matriz.

4.1.1 Detailed Description

Definições de estruturas e protótipos de funções para manipulação de antenas e nefastos.

Author

Fábio Rafael Gomes Costa @contact a22997@alunos.ipca.pt @course Engenharia Sistemas Informáticos

Definition in file funcao.h.

4.1.2 Macro Definition Documentation

4.1.2.1 MAX_COLUNAS

```
#define MAX_COLUNAS 100
```

Número máximo de colunas na matriz.

Definition at line 15 of file funcao.h.

4.1.2.2 MAX_LINHAS

```
#define MAX_LINHAS 100
```

Número máximo de linhas na matriz.

Definition at line 14 of file funcao.h.

4.1.3 Typedef Documentation

4.1.3.1 ANT

```
typedef struct ANT ANT
```

Estrutura que representa uma antena.

Esta estrutura armazena a frequência da antena e suas coordenadas (x, y).

4.1.3.2 NEF

```
typedef struct NEF NEF
```

Estrutura que representa um nefasto.

Esta estrutura armazena as coordenadas (x, y) de um nefasto.

4.1.4 Function Documentation

4.1.4.1 apresentarLista()

```
void apresentarLista (
            ANT * lista)
```

Apresenta a lista de antenas.

Parameters

lista	Lista de antenas a ser apresentada.

Definition at line 96 of file funcao.c.

Here is the caller graph for this function:

4.1.4.2 apresentarListaNef()

```
void apresentarListaNef (
            NEF * lista)
```

Apresenta a lista de nefastos.

Parameters

lista	Lista de nefastos a ser apresentada.

Definition at line 243 of file funcao.c.

Here is the caller graph for this function:

4.1.4.3 apresentarMatriz()

```
void apresentarMatriz (
            char matriz[MAX_LINHAS][MAX_COLUNAS],
            int linhas,
            int colunas)
```

Apresenta a matriz.

Parameters

matriz	Matriz a ser apresentada.
linhas	Número de linhas da matriz.
colunas	Número de colunas da matriz.

Definition at line 115 of file funcao.c.

Here is the caller graph for this function:

4.1.4.4 atualizarMatriz()

```
void atualizarMatriz (
            ANT ** lista,
            char freq,
            int x,
            int y,
            char matriz[MAX_LINHAS][MAX_COLUNAS])
```

Atualiza a matriz e a lista de antenas com uma nova antena.

Parameters

lista	Ponteiro para a lista de antenas.
freq	Frequência da antena.
x	Coordenada x da antena.
y	Coordenada y da antena.
matriz	Matriz a ser atualizada.

Definition at line 265 of file funcao.c.

Here is the call graph for this function: Here is the caller graph for this function:

4.1.4.5 igualarMatrizes()

```
void igualarMatrizes (
            char matrizNef[MAX_LINHAS][MAX_COLUNAS],
            char matriz[MAX_LINHAS][MAX_COLUNAS],
            int linhas,
            int colunas)
```

Copia os dados de uma matriz para outra.

Parameters

matrizNef	Matriz de destino.
matriz	Matriz de origem.
linhas	Número de linhas da matriz.
colunas	Número de colunas da matriz.

Definition at line 132 of file funcao.c.

Here is the caller graph for this function:

4.1.4.6 inserirAntena()

```
void inserirAntena (
            ANT ** lista,
            char freq,
            int x,
            int y)
```

Insere uma nova antena na lista de antenas.

Parameters

lista	Ponteiro para a lista de antenas.
freq	Frequência da antena.
x	Coordenada x da antena.
y	Coordenada y da antena.

Definition at line 78 of file funcao.c.

4.1.4.7 inserirNefasto()

```
void inserirNefasto (
            NEF ** lista,
            int x,
            int y)
```

Insere um nefasto na lista de nefastos.

Parameters

lista	Ponteiro para a lista de nefastos.
x	Coordenada x do nefasto.
y	Coordenada y do nefasto.

Definition at line 225 of file funcao.c.

4.1.4.8 LerLista()

```
ANT * LerLista (
            const char * nomeFicheiro,
            char matriz[MAX_LINHAS][MAX_COLUNAS],
            int * linhas,
            int * colunas)
```

Lê uma lista de antenas a partir de um arquivo e preenche uma matriz.

Parameters

nomeFicheiro	Nome do arquivo a ser lido.
matriz	Matriz onde os dados serão armazenados.
linhas	Ponteiro para armazenar o número de linhas da matriz.
colunas	Ponteiro para armazenar o número de colunas da matriz.

Returns

ANT∗ Retorna a lista de antenas lida do arquivo.

Esta função lê um arquivo contendo dados de antenas, armazena os dados em uma matriz e cria uma lista encadeada de antenas.

Parameters

nomeFicheiro	Nome do arquivo a ser lido.
matriz	Matriz onde os dados serão armazenados.
linhas	Ponteiro para armazenar o número de linhas da matriz.
colunas	Ponteiro para armazenar o número de colunas da matriz.

Returns

ANT∗ Retorna a lista de antenas lida do arquivo.

Definition at line 27 of file funcao.c.

Here is the call graph for this function: Here is the caller graph for this function:

4.1.4.9 matrizNefastos()

```
NEF * matrizNefastos (
              char matriz[MAX_LINHAS][MAX_COLUNAS],
              int linhas,
              int colunas)
```

Detecta e insere nefastos na matriz.

Parameters

matriz	Matriz onde os nefastos serão inseridos.
linhas	Número de linhas da matriz.
colunas	Número de colunas da matriz.

Returns

NEF∗ Retorna a lista de nefastos detectados.

Esta função percorre a matriz e insere nefastos ('#') em posições específicas com base nas coordenadas das antenas.

Parameters

matriz	Matriz onde os nefastos serão inseridos.
linhas	Número de linhas da matriz.
colunas	Número de colunas da matriz.

Returns

NEF∗ Retorna a lista de nefastos detectados.

Definition at line 149 of file funcao.c.

Here is the call graph for this function: Here is the caller graph for this function:

4.1.4.10 removerAntena()

```
void removerAntena (
            ANT ** lista,
            int x,
            int y,
            char matriz[MAX_LINHAS][MAX_COLUNAS])
```

Remove uma antena da lista e da matriz.

Parameters

lista	Ponteiro para a lista de antenas.
x	Coordenada x da antena.
y	Coordenada y da antena.
matriz	Matriz a ser atualizada.

Definition at line 289 of file funcao.c.

Here is the caller graph for this function:

4.2 funcao.h

Go to the documentation of this file.

```
00001
00008
00009 #pragma once
00010 #include <stdio.h>
00011 #include <stdlib.h>
00012 #include <string.h>
00013
00014 #define MAX_LINHAS 100
00015 #define MAX_COLUNAS 100
00016
00022 typedef struct ANT {
00023     char freqAntena;
00024     int x;
00025     int y;
00026     struct ANT* proxAntena;
00027 } ANT;
00028
00034 typedef struct NEF {
```

```
00035     int x;
00036     int y;
00037     struct NEF* proxNef;
00038 } NEF;
00039
00040 // Protótipos das funções
00041
00050 void inserirAntena(ANT** lista, char freq, int x, int y);
00051
00061 ANT* LerLista(const char* nomeFicheiro, char matriz[MAX_LINHAS][MAX_COLUNAS], int* linhas, int*
      colunas);
00062
00068 void apresentarLista(ANT* lista);
00069
00077 void apresentarMatriz(char matriz[MAX_LINHAS][MAX_COLUNAS], int linhas, int colunas);
00078
00086 void inserirNefasto(NEF** lista, int x, int y);
00087
00096 NEF* matrizNefastos(char matriz[MAX_LINHAS][MAX_COLUNAS], int linhas, int colunas);
00097
00103 void apresentarListaNef(NEF* lista);
00104
00113 void igualarMatrizes(char matrizNef[MAX_LINHAS][MAX_COLUNAS], char matriz[MAX_LINHAS][MAX_COLUNAS],
      int linhas, int colunas);
00114
00124 void atualizarMatriz(ANT** lista, char freq, int x, int y, char matriz[MAX_LINHAS][MAX_COLUNAS]);
00125
00134 void removerAntena(ANT** lista, int x, int y, char matriz[MAX_LINHAS][MAX_COLUNAS]);
```

4.3 projetoedalib/funcao.h File Reference

Definições de estruturas e protótipos de funções para manipulação de antenas e nefastos.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
```

Include dependency graph for funcao.h: This graph shows which files directly or indirectly include this file:

Data Structures

- struct ANT

  Estrutura que representa uma antena.
- struct NEF

  Estrutura que representa um nefasto.

Macros

- #define MAX_LINHAS 100
- #define MAX_COLUNAS 100

Typedefs

- typedef struct ANT ANT

  Estrutura que representa uma antena.
- typedef struct NEF NEF

  Estrutura que representa um nefasto.

Functions

- void inserirAntena (ANT **lista, char freq, int x, int y)

  Insere uma nova antena na lista de antenas.
- ANT * LerLista (const char *nomeFicheiro, char matriz[MAX_LINHAS][MAX_COLUNAS], int *linhas, int *colunas)

  Lê uma lista de antenas a partir de um arquivo e preenche uma matriz.
- void apresentarLista (ANT *lista)

  Apresenta a lista de antenas.
- void apresentarMatriz (char matriz[MAX_LINHAS][MAX_COLUNAS], int linhas, int colunas)

  Apresenta a matriz.
- void inserirNefasto (NEF **lista, int x, int y)

  Insere um nefasto na lista de nefastos.
- NEF * matrizNefastos (char matriz[MAX_LINHAS][MAX_COLUNAS], int linhas, int colunas)

  Detecta e insere nefastos na matriz.
- void apresentarListaNef (NEF *lista)

  Apresenta a lista de nefastos.
- void igualarMatrizes (char matrizNef[MAX_LINHAS][MAX_COLUNAS], char matriz[MAX_LINHAS][MAX_COLUNAS], int linhas, int colunas)

  Copia os dados de uma matriz para outra.
- void atualizarMatriz (ANT **lista, char freq, int x, int y, char matriz[MAX_LINHAS][MAX_COLUNAS])

  Atualiza a matriz e a lista de antenas com uma nova antena.
- void removerAntena (ANT **lista, int x, int y, char matriz[MAX_LINHAS][MAX_COLUNAS])

  Remove uma antena da lista e da matriz.

4.3.1 Detailed Description

Definições de estruturas e protótipos de funções para manipulação de antenas e nefastos.

Author

Fábio Rafael Gomes Costa @contact a22997@alunos.ipca.pt @course Engenharia Sistemas Informáticos

Date

18/03/2025

Definition in file funcao.h.

4.3.2 Macro Definition Documentation

4.3.2.1 MAX_COLUNAS

```
#define MAX_COLUNAS 100
```

Número máximo de colunas na matriz.

Definition at line 16 of file funcao.h.

4.3.2.2 MAX_LINHAS

```
#define MAX_LINHAS 100
```

Número máximo de linhas na matriz.

Definition at line 15 of file funcao.h.

4.3.3 Typedef Documentation

4.3.3.1 ANT

```
typedef struct ANT ANT
```

Estrutura que representa uma antena.

Esta estrutura armazena a frequência da antena e suas coordenadas (x, y).

4.3.3.2 NEF

```
typedef struct NEF NEF
```

Estrutura que representa um nefasto.

Esta estrutura armazena as coordenadas (x, y) de um nefasto.

4.3.4 Function Documentation

4.3.4.1 apresentarLista()

```
void apresentarLista (
            ANT * lista)
```

Apresenta a lista de antenas.

Parameters

lista	Lista de antenas a ser apresentada.

Definition at line 96 of file funcao.c.

4.3.4.2 apresentarListaNef()

```
void apresentarListaNef (
            NEF * lista)
```

Apresenta a lista de nefastos.

Parameters

lista	Lista de nefastos a ser apresentada.

Definition at line 243 of file funcao.c.

4.3.4.3 apresentarMatriz()

```
void apresentarMatriz (
            char matriz[MAX_LINHAS][MAX_COLUNAS],
            int linhas,
            int colunas)
```

Apresenta a matriz.

Parameters

matriz	Matriz a ser apresentada.
linhas	Número de linhas da matriz.
colunas	Número de colunas da matriz.

Definition at line 115 of file funcao.c.

4.3.4.4 atualizarMatriz()

```
void atualizarMatriz (
            ANT ** lista,
            char freq,
            int x,
            int y,
            char matriz[MAX_LINHAS][MAX_COLUNAS])
```

Atualiza a matriz e a lista de antenas com uma nova antena.

Parameters

lista	Ponteiro para a lista de antenas.
freq	Frequência da antena.
x	Coordenada x da antena.
y	Coordenada y da antena.
matriz	Matriz a ser atualizada.

Definition at line 265 of file funcao.c.

4.3.4.5 igualarMatrizes()

```
void igualarMatrizes (
            char matrizNef[MAX_LINHAS][MAX_COLUNAS],
            char matriz[MAX_LINHAS][MAX_COLUNAS],
            int linhas,
            int colunas)
```

Copia os dados de uma matriz para outra.

Parameters

matrizNef	Matriz de destino.
matriz	Matriz de origem.
linhas	Número de linhas da matriz.
colunas	Número de colunas da matriz.

Definition at line 132 of file funcao.c.

4.3.4.6 inserirAntena()

```
void inserirAntena (
            ANT ** lista,
            char freq,
            int x,
            int y)
```

Insere uma nova antena na lista de antenas.

Parameters

lista	Ponteiro para a lista de antenas.
freq	Frequência da antena.
x	Coordenada x da antena.
y	Coordenada y da antena.

Definition at line 78 of file funcao.c.

Here is the caller graph for this function:

4.3.4.7 inserirNefasto()

```
void inserirNefasto (
            NEF ** lista,
            int x,
            int y)
```

Insere um nefasto na lista de nefastos.

Parameters

lista	Ponteiro para a lista de nefastos.
x	Coordenada x do nefasto.
y	Coordenada y do nefasto.

Definition at line 225 of file funcao.c.

Here is the caller graph for this function:

4.3.4.8 LerLista()

```
ANT * LerLista (
            const char * nomeFicheiro,
            char matriz[MAX_LINHAS][MAX_COLUNAS],
            int * linhas,
            int * colunas)
```

Lê uma lista de antenas a partir de um arquivo e preenche uma matriz.

Parameters

nomeFicheiro	Nome do arquivo a ser lido.
matriz	Matriz onde os dados serão armazenados.
linhas	Ponteiro para armazenar o número de linhas da matriz.
colunas	Ponteiro para armazenar o número de colunas da matriz.

Returns

ANT∗ Retorna a lista de antenas lida do arquivo.

Esta função lê um arquivo contendo dados de antenas, armazena os dados em uma matriz e cria uma lista encadeada de antenas.

Parameters

nomeFicheiro	Nome do arquivo a ser lido.
matriz	Matriz onde os dados serão armazenados.
linhas	Ponteiro para armazenar o número de linhas da matriz.
colunas	Ponteiro para armazenar o número de colunas da matriz.

Returns

ANT∗ Retorna a lista de antenas lida do arquivo.

Definition at line 27 of file funcao.c.

4.3.4.9 matrizNefastos()

```
NEF * matrizNefastos (
            char matriz[MAX_LINHAS][MAX_COLUNAS],
            int linhas,
            int colunas)
```

Detecta e insere nefastos na matriz.

Parameters

matriz	Matriz onde os nefastos serão inseridos.
linhas	Número de linhas da matriz.
colunas	Número de colunas da matriz.

Returns

NEF∗ Retorna a lista de nefastos detectados.

Esta função percorre a matriz e insere nefastos ('#') em posições específicas com base nas coordenadas das antenas.

Parameters

matriz	Matriz onde os nefastos serão inseridos.
linhas	Número de linhas da matriz.
colunas	Número de colunas da matriz.

Returns

NEF∗ Retorna a lista de nefastos detectados.

Definition at line 149 of file funcao.c.

4.3.4.10 removerAntena()

```
void removerAntena (
            ANT ** lista,
            int x,
            int y,
            char matriz[MAX_LINHAS][MAX_COLUNAS])
```

Remove uma antena da lista e da matriz.

Parameters

lista	Ponteiro para a lista de antenas.
x	Coordenada x da antena.
y	Coordenada y da antena.
matriz	Matriz a ser atualizada.

Definition at line 289 of file funcao.c.

4.4 funcao.h

Go to the documentation of this file.

```
00001
00009
00010 #pragma once
00011 #include <stdio.h>
00012 #include <stdlib.h>
00013 #include <string.h>
00014
00015 #define MAX_LINHAS 100
00016 #define MAX_COLUNAS 100
00017
00023 typedef struct ANT {
00024     char freqAntena;
00025     int x;
00026     int y;
00027     struct ANT* proxAntena;
00028 } ANT;
00029
00035 typedef struct NEF {
00036     int x;
00037     int y;
00038     struct NEF* proxNef;
00039 } NEF;
00040
00041 // Protótipos das funções
```

```
00042
00051 void inserirAntena(ANT** lista, char freq, int x, int y);
00052
00062 ANT* LerLista(const char* nomeFicheiro, char matriz[MAX_LINHAS][MAX_COLUNAS], int* linhas, int*
      colunas);
00063
00069 void apresentarLista(ANT* lista);
00070
00078 void apresentarMatriz(char matriz[MAX_LINHAS][MAX_COLUNAS], int linhas, int colunas);
00079
00087 void inserirNefasto(NEF** lista, int x, int y);
00088
00097 NEF* matrizNefastos(char matriz[MAX_LINHAS][MAX_COLUNAS], int linhas, int colunas);
00098
00104 void apresentarListaNef(NEF* lista);
00105
00114 void igualarMatrizes(char matrizNef[MAX_LINHAS][MAX_COLUNAS], char matriz[MAX_LINHAS][MAX_COLUNAS],
      int linhas, int colunas);
00115
00125 void atualizarMatriz(ANT** lista, char freq, int x, int y, char matriz[MAX_LINHAS][MAX_COLUNAS]);
00126
00135 void removerAntena(ANT** lista, int x, int y, char matriz[MAX_LINHAS][MAX_COLUNAS]);
```

4.5 projetoeda/main.c File Reference

Programa principal para manipulação de antenas e detecção de nefastos.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "funcao.h"
```

Include dependency graph for main.c:

Functions

- int main ()

  Função principal do programa.

4.5.1 Detailed Description

Programa principal para manipulação de antenas e detecção de nefastos.

Author

Fábio Rafael Gomes Costa @contact a22997@alunos.ipca.pt @course Engenharia Sistemas Informáticos

Date

18/03/2025

Definition in file main.c.

4.5.2 Function Documentation

4.5.2.1 main()

```
int main ()
```

Função principal do programa.

Esta função é responsável por:

- Ler a lista de antenas de um arquivo.
- Apresentar a matriz e a lista de antenas.
- Detetar nefastos na matriz.
- Inserir e remover antenas da matriz.
- Liberar a memória alocada para as listas de antenas e nefastos.

Returns

int Retorna 0 se o programa for executado com sucesso.

< Matriz para armazenar dados do arquivo

< Variáveis para armazenar o número de linhas e colunas da matriz

< Lista de antenas lida do arquivo

< Apresenta a matriz lida do arquivo

< Apresenta a lista de antenas

< Matriz para armazenar dados com nefastos

< Copia os dados da matriz original para a matriz de nefastos

< Insere nefastos na matriz e devolve a lista de nefastos

< Apresenta a matriz com nefastos

< Apresenta a lista de nefastos

< Insere uma antena na matriz e na lista de antenas

< Apresenta a matriz atualizada

< Apresenta a lista de antenas atualizada

< Insere outra antena na matriz e na lista de antenas

< Apresenta a matriz atualizada

< Apresenta a lista de antenas atualizada

< Matriz para armazenar dados com nefastos

< Copia os dados da matriz original para a matriz de nefastos

< Insere nefastos na matriz e devolve a lista de nefastos

< Apresenta a matriz com nefastos

< Apresenta a lista de nefastos

< Remove uma antena da matriz e da lista de antenas

< Apresenta a matriz atualizada

< Apresenta a lista de antenas atualizada

< Libera a memória alocada para a lista de antenas

< Libera a memória alocada para a lista de nefastos

< Libera a memória alocada para a lista de nefastos

< Retorna 0 indicando que o programa foi executado com sucesso

Definition at line 27 of file main.c.

Here is the call graph for this function:

4.6 main.c

Go to the documentation of this file.

```
00001
00009
00010 #include <stdio.h>
00011 #include <stdlib.h>
00012 #include <string.h>
00013 #include "funcao.h"
00014
00027 int main() {
00028     char matriz[MAX_LINHAS][MAX_COLUNAS];
00029     int linhas, colunas;
00030
00031     // Ler Lista de antenas do arquivo
00032     ANT* lista = LerLista("antenas.txt", matriz, &linhas, &colunas);
00033
00034     //-----------------------------------------------
00035     // Apresentar a matriz e a lista de antenas
00036     printf("Matriz Lida do Arquivo:\n");
00037     apresentarMatriz(matriz, linhas, colunas);
00038     apresentarLista(lista);
00039     printf("--------------------\n\n\n");
00040
00041     //-----------------------------------------------
00042     // Detetar Nefastos na matriz
00043     char matrizNef[MAX_LINHAS][MAX_COLUNAS];
00044     igualarMatrizes(matrizNef, matriz, linhas, colunas);
00045     printf("Matriz Com Nefastos:\n");
00046     NEF* listaNef = matrizNefastos(matrizNef, linhas, colunas);
00047     apresentarMatriz(matrizNef, linhas, colunas);
00048     apresentarListaNef(listaNef);
00049     printf("--------------------\n\n\n");
00050
00051     //-----------------------------------------------
00052     // Inserir Antena na matriz e na lista de antenas
00053     printf("Inserir Antena:\tFrequencia: \'C\'\tCoordenadas: (2,2)\n");
00054     atualizarMatriz(&lista, 'C', 2, 2, matriz);
00055     apresentarMatriz(matriz, linhas, colunas);
00056     apresentarLista(lista);
00057
00058     printf("\n\n\Inserir Antena:\tFrequencia: \'C\'\tCoordenadas: (4,6)\n");
00059     atualizarMatriz(&lista, 'C', 4, 6, matriz);
```

```
00060     apresentarMatriz(matriz, linhas, colunas);
00061     apresentarLista(lista);
00062     printf("--------------------\n\n\n");
00063 
00064     //--------------------------------------------
00065     // Detetar Nefastos na matriz
00066     char matrizNef2[MAX_LINHAS][MAX_COLUNAS];
00067     igualarMatrizes(matrizNef2, matriz, linhas, colunas);
00068     printf("Matriz Com Nefastos com a adicao da frequencia C:\n");
00069     NEF* listaNef2 = matrizNefastos(matrizNef2, linhas, colunas);
00070     apresentarMatriz(matrizNef2, linhas, colunas);
00071     apresentarListaNef(listaNef2);
00072     printf("--------------------\n\n\n");
00073 
00074     //--------------------------------------------
00075     // Remover Antena da matriz e da lista de antenas
00076     printf("Remover Antena:\tCoordenadas: (2,2)\n");
00077     removerAntena(&lista, 2, 2, matriz);
00078     apresentarMatriz(matriz, linhas, colunas);
00079     apresentarLista(lista);
00080 
00081     // Liberar memória alocada para as listas de antenas e nefastos
00082     ANT* temp;
00083     while (lista != NULL) {
00084         temp = lista;
00085         lista = lista->proxAntena;
00086         free(temp);
00087     }
00088 
00089     NEF* tempNef;
00090     while (listaNef != NULL) {
00091         tempNef = listaNef;
00092         listaNef = listaNef->proxNef;
00093         free(tempNef);
00094     }
00095 
00096     NEF* tempNef2;
00097     while (listaNef2 != NULL) {
00098         tempNef2 = listaNef2;
00099         listaNef2 = listaNef2->proxNef;
00100         free(tempNef2);
00101     }
00102 
00103     return 0;
00104 }
```

4.7 projetoedalib/funcao.c File Reference

Implementação de funções para manipulação de antenas e nefastos.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "funcao.h"
```

Include dependency graph for funcao.c:

4.8 funcao.c

Go to the documentation of this file.

```
00001 
00009 
00010 #include <stdio.h>
00011 #include <stdlib.h>
00012 #include <string.h>
00013 #include "funcao.h"
00014 
00027 ANT* LerLista(const char* nomeFicheiro, char matriz[MAX_LINHAS][MAX_COLUNAS], int* linhas, int*
      colunas) {
00028     FILE* ficheiro = fopen(nomeFicheiro, "r");
00029     if (ficheiro == NULL) {
00030         perror("Erro ao abrir o ficheiro");
00031         return NULL;
```

```
00032     }
00033
00034     ANT* lista = NULL;
00035     char linha[MAX_COLUNAS];
00036     int y = 0;
00037     int maxColunas = 0;
00038
00039     while (fgets(linha, sizeof(linha), ficheiro)) {
00040         int tamLinha = strlen(linha);
00041         if (linha[tamLinha - 1] == '\n') {
00042             linha[tamLinha - 1] = '\0'; // Remover quebra de linha
00043             tamLinha--;
00044         }
00045
00046         if (tamLinha > maxColunas) {
00047             maxColunas = tamLinha;
00048         }
00049
00050         for (int x = 0; x < tamLinha; x++) {
00051             matriz[y][x] = linha[x]; // Armazena na matriz
00052             if (linha[x] != '.') {
00053                 inserirAntena(&lista, linha[x], x, y);
00054             }
00055         }
00056
00057         // Preencher com '.' os espaços não usados
00058         for (int x = tamLinha; x < MAX_COLUNAS; x++) {
00059             matriz[y][x] = '.';
00060         }
00061         y++;
00062     }
00063
00064     *linhas = y;
00065     *colunas = maxColunas;
00066     fclose(ficheiro);
00067     return lista;
00068 }
00069
00078 void inserirAntena(ANT** lista, char freq, int x, int y) {
00079     ANT* novaAntena = (ANT*)malloc(sizeof(ANT));
00080     if (novaAntena == NULL) {
00081         perror("Erro ao alocar memória");
00082         return;
00083     }
00084     novaAntena->freqAntena = freq;
00085     novaAntena->x = x;
00086     novaAntena->y = y;
00087     novaAntena->proxAntena = *lista;
00088     *lista = novaAntena;
00089 }
00090
00096 void apresentarLista(ANT* lista) {
00097     if (lista == NULL) {
00098         printf("Lista vazia\n");
00099         return;
00100     }
00101     printf("\nLista de Antenas:\n");
00102     while (lista != NULL) {
00103         printf("Antena: %c | Coordenadas: (%d, %d)\n", lista->freqAntena, lista->x, lista->y);
00104         lista = lista->proxAntena;
00105     }
00106 }
00107
00115 void apresentarMatriz(char matriz[MAX_LINHAS][MAX_COLUNAS], int linhas, int colunas) {
00116     for (int i = 0; i < linhas; i++) {
00117         for (int j = 0; j < colunas; j++) {
00118             printf("%c ", matriz[i][j]);
00119         }
00120         printf("\n");
00121     }
00122 }
00123
00132 void igualarMatrizes(char matrizNef[MAX_LINHAS][MAX_COLUNAS], char matriz[MAX_LINHAS][MAX_COLUNAS],
    int linhas, int colunas) {
00133     for (int i = 0; i < linhas; i++) {
00134         memcpy(matrizNef[i], matriz[i], colunas * sizeof(char));
00135     }
00136 }
00137
00149 NEF* matrizNefastos(char matriz[MAX_LINHAS][MAX_COLUNAS], int linhas, int colunas) {
00150     int menorx, menory, maiorx, maiory, difx, dify;
00151     NEF* lista = NULL;
00152     for (int y = 0; y < linhas; y++) {
00153         for (int x = 0; x < colunas; x++) {
00154             if (matriz[y][x] != '.' && matriz[y][x] != '#') {
00155                 for (int y2 = 0; y2 < linhas; y2++) {
00156                     for (int x2 = 0; x2 < colunas; x2++) {
```

```
00157                         if (matriz[y][x] == matriz[y2][x2] && (y != y2 || x != x2)) {
00158
00159                             if (x > x2)
00160                             {
00161                                 difx = x - x2;
00162                                 menorx = x2 - difx;
00163                                 maiorx = x + difx;
00164                             }
00165                             else
00166                             {
00167                                 difx = x2 - x;
00168                                 menorx = x - difx;
00169                                 maiorx = x2 + difx;
00170                             }
00171                             if (y > y2)
00172                             {
00173                                 dify = y - y2;
00174                                 menory = y2 - dify;
00175                                 maiory = y + dify;
00176                             }
00177                             else
00178                             {
00179                                 dify = y2 - y;
00180                                 menory = y - dify;
00181                                 maiory = y2 + dify;
00182                             }
00183                             if (x > x2 && y > y2 || x < x2 && y < y2)
00184                             {
00185                                 if (matriz[menory][menorx] == '.' && menorx >= 0 && menory >= 0)
00186                                 {
00187                                     matriz[menory][menorx] = '#';
00188                                     inserirNefasto(&lista, menorx, menory);
00189                                 }
00190                                 if (matriz[maiory][maiorx] == '.' && maiorx <= colunas && maiory <=
      linhas)
00191                                 {
00192                                     matriz[maiory][maiorx] = '#';
00193                                     inserirNefasto(&lista, maiorx, maiory);
00194                                 }
00195                             }
00196                             else if (x > x2 && y < y2 || x<x2 && y>y2)
00197                             {
00198                                 if (matriz[menory][maiorx] == '.' && menorx >= 0 && menory >= 0)
00199                                 {
00200                                     matriz[menory][maiorx] = '#';
00201                                     inserirNefasto(&lista, maiorx, menory);
00202                                 }
00203                                 if (matriz[maiory][menorx] == '.' && maiorx <= colunas && maiory <=
      linhas)
00204                                 {
00205                                     matriz[maiory][menorx] = '#';
00206                                     inserirNefasto(&lista, menorx, maiory);
00207                                 }
00208                             }
00209                         }
00210                     }
00211                 }
00212             }
00213         }
00214     }
00215     return lista;
00216 }
00217
00225 void inserirNefasto(NEF** lista, int x, int y) {
00226     NEF* novoNefasto = (NEF*)malloc(sizeof(NEF));
00227     if (novoNefasto == NULL) {
00228         perror("Erro ao alocar memória");
00229         return;
00230     }
00231
00232     novoNefasto->x = x;
00233     novoNefasto->y = y;
00234     novoNefasto->proxNef = *lista;
00235     *lista = novoNefasto;
00236 }
00237
00243 void apresentarListaNef(NEF* lista) {
00244     if (lista == NULL) {
00245         printf("Lista vazia\n");
00246         return;
00247     }
00248
00249     printf("\nLista de Nefastos:\n");
00250     while (lista != NULL) {
00251         printf("Nefastos Coordenadas: (%d, %d)\n", lista->x, lista->y);
00252         lista = lista->proxNef;
00253     }
```

```
00254 }
00255
00265 void atualizarMatriz(ANT** lista, char freq, int x, int y, char matriz[MAX_LINHAS][MAX_COLUNAS])
00266 {
00267     ANT* temp = *lista;
00268     while (temp != NULL)
00269     {
00270         if (temp->x == x && temp->y == y)
00271         {
00272             matriz[y][x] = freq;
00273             return;
00274         }
00275         temp = temp->proxAntena;
00276     }
00277     inserirAntena(lista, freq, x, y);
00278     matriz[y][x] = freq;
00279 }
00280
00289 void removerAntena(ANT** lista, int x, int y, char matriz[MAX_LINHAS][MAX_COLUNAS])
00290 {
00291     ANT* temp = *lista;
00292     ANT* anterior = NULL;
00293     while (temp != NULL)
00294     {
00295         if (temp->x == x && temp->y == y)
00296         {
00297             if (anterior == NULL)
00298             {
00299                 *lista = temp->proxAntena;
00300             }
00301             else
00302             {
00303                 anterior->proxAntena = temp->proxAntena;
00304             }
00305             matriz[y][x] = '.';
00306             free(temp);
00307             return;
00308         }
00309         anterior = temp;
00310         temp = temp->proxAntena;
00311     }
00312 }
```

Index