Trabalho Prático de EDA - Parte 2

Generated by Doxygen 1.9.1

1 Class Index 1

1.1 Class List . . . 1

2 File Index 3

2.1 File List . . . 3

3 Class Documentation 5

3.1 Adj Struct Reference . . . 5

3.1.1 Detailed Description . . . 5

3.2 Antena Struct Reference . . . 6

3.2.1 Detailed Description . . . 6

3.3 Caminho Struct Reference . . . 7

3.3.1 Detailed Description . . . 7

4 File Documentation 9

4.1 ecra.c File Reference . . . 9

4.1.1 Detailed Description . . . 10

4.1.2 Function Documentation . . . 10

4.1.2.1 ListarCaminhos() . . . 10

4.1.2.2 MostraGrafo() . . . 10

4.2 ecra.h File Reference . . . 11

4.2.1 Detailed Description . . . 12

4.2.2 Function Documentation . . . 12

4.2.2.1 ListarCaminhos() . . . 12

4.2.2.2 MostraGrafo() . . . 13

4.3 grafos.c File Reference . . . 13

4.3.1 Detailed Description . . . 15

4.3.2 Function Documentation . . . 15

4.3.2.1 CalcularArestas() . . . 15

4.3.2.2 CalcularCaminhosEntreAntenas() . . . 16

4.3.2.3 CarregarFicheiro() . . . 16

4.3.2.4 CriarAntena() . . . 17

4.3.2.5 CriarAresta() . . . 17

4.3.2.6 EncontrarAntena() . . . 18

4.3.2.7 GravarFicheiroBin() . . . 18

4.3.2.8 GuardarCaminho() . . . 19

4.3.2.9 InserirAntena() . . . 20

4.3.2.10 InserirAresta() . . . 20

4.3.2.11 LerFicheiroBin() . . . 21

4.3.2.12 LimparVisitados() . . . 21

4.3.2.13 ProcuraProfundidade() . . . 21

4.3.2.14 RemoverAntena() . . . 22

4.4 grafos.h File Reference . . . 22

4.4.1 Detailed Description . . . 25
4.4.2 Typedef Documentation . . . 25
4.4.2.1 Adj . . . 25
4.4.2.2 Caminho . . . 25
4.4.3 Function Documentation . . . 25
4.4.3.1 CalcularArestas() . . . 25
4.4.3.2 CalcularCaminhosEntreAntenas() . . . 26
4.4.3.3 CarregarFicheiro() . . . 27
4.4.3.4 CriarAntena() . . . 28
4.4.3.5 CriarAresta() . . . 29
4.4.3.6 EncontrarAntena() . . . 29
4.4.3.7 GravarFicheiroBin() . . . 30
4.4.3.8 GuardarCaminho() . . . 31
4.4.3.9 InserirAntena() . . . 32
4.4.3.10 InserirAresta() . . . 32
4.4.3.11 LerFicheiroBin() . . . 33
4.4.3.12 LimparVisitados() . . . 33
4.4.3.13 ProcuraProfundidade() . . . 34
4.4.3.14 RemoverAntena() . . . 35
4.5 main.c File Reference . . . 35
4.5.1 Detailed Description . . . 36
4.5.2 Function Documentation . . . 37
4.5.2.1 main() . . . 37

Index **39**

Chapter 1

Class Index

1.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Adj
: Representa uma aresta entre antenas no grafo 5

Antena
: Esta estrutura modela uma antena no grafo e contém informações sobre a sua frequência, posição no espaço (coordenadas x e y), ligações a outras antenas (adjacências), um ponteiro para a próxima antena na lista ligada e uma flag de visita para saber se esta ja foi visitada ou não 6

Caminho
: Representa um caminho composto por uma sequência de antenas 7

Chapter 2

File Index

2.1 File List

Here is a list of all documented files with brief descriptions:

ecra.c
Implementação das funções para mostrar no ecrã 9
ecra.h
Define as funções para apresentar no ecrã 11
grafos.c
Implementação de todas as funções 13
grafos.h
Define as funções e estruturas 22
main.c
Rograma principal que executa as funções 35

Chapter 3

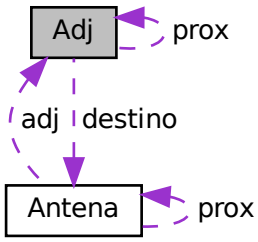
Class Documentation

3.1 Adj Struct Reference

Representa uma aresta entre antenas no grafo.

`#include <grafos.h>`

Collaboration diagram for Adj:



Public Attributes

- Antena * **destino**
- struct Adj * **prox**

3.1.1 Detailed Description

Representa uma aresta entre antenas no grafo.

Esta estrutura modela uma ligação entre duas antenas no grafo, onde cada uma representa uma adjacência (aresta) a partir de uma antena de origem para uma antena de destino

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

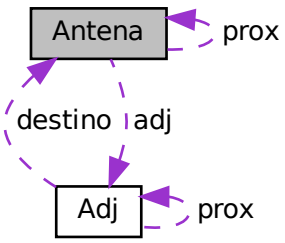
- grafos.h

3.2 Antena Struct Reference

Esta estrutura modela uma antena no grafo e contém informações sobre a sua frequência, posição no espaço (coordenadas x e y), ligações a outras antenas (adjacências), um ponteiro para a próxima antena na lista ligada e uma flag de visita para saber se esta ja foi visitada ou não.

`#include <grafos.h>`

Collaboration diagram for Antena:



Public Attributes

- char **freq**
- int **x**
- int **y**
- struct Adj * **adj**
- struct Antena * **prox**
- bool **visitado**

3.2.1 Detailed Description

Esta estrutura modela uma antena no grafo e contém informações sobre a sua frequência, posição no espaço (coordenadas x e y), ligações a outras antenas (adjacências), um ponteiro para a próxima antena na lista ligada e uma flag de visita para saber se esta ja foi visitada ou não.

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

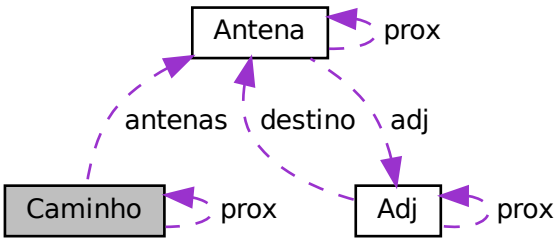
- grafos.h

3.3 Caminho Struct Reference

Representa um caminho composto por uma sequência de antenas.

`#include <grafos.h>`

Collaboration diagram for Caminho:



Public Attributes

- Antena * **antenas** [MAX_ANTENAS]
- int **nAntenas**
- struct Caminho * **prox**

3.3.1 Detailed Description

Representa um caminho composto por uma sequência de antenas.

Esta estrutura modela um caminho no grafo, contendo um array de ponteiros para antenas que constituem o percurso, o número total de antenas no caminho e um ponteiro para o próximo caminho na lista ligada

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

- grafos.h

Chapter 4

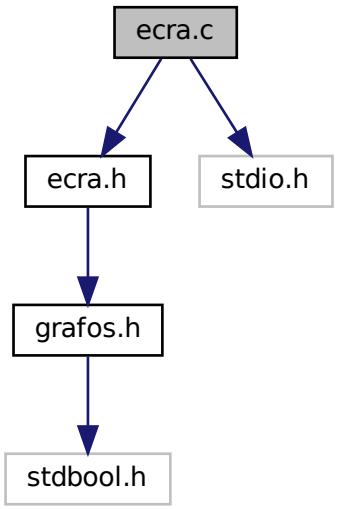
File Documentation

4.1 ecra.c File Reference

Implementação das funções para mostrar no ecrã

```
#include "ecra.h"
#include <stdio.h>
```

Include dependency graph for ecra.c:



Functions

- void MostraGrafo (Antena *inicio)

  Mostra as antenas e as respetivas adjacencias na consola.
- void ListarCaminhos (Caminho *caminhos)

  Lista todos os caminhos armazenados.

4.1.1 Detailed Description

Implementação das funções para mostrar no ecrã

Author

David Faria (a31517@alunos.ipca.pt)

Version

0.1

Date

2025-05-13

Copyright

Copyright (c) 2025

4.1.2 Function Documentation

4.1.2.1 ListarCaminhos()

```
void ListarCaminhos (
            Caminho * caminhos )
```

Lista todos os caminhos armazenados.

Lista todos os caminhos armazenados possiveis.

Esta função percorre a lista ligada de caminhos e apresenta para cada caminho as coordenadas das antenas que o compõem, na ordem em que são percorridas

Parameters

caminhos	Ponteiro para o início da lista de caminhos a listar

4.1.2.2 MostraGrafo()

```
void MostraGrafo (
            Antena * inicio )
```

Mostra as antenas e as respetivas adjacencias na consola.

Mostra o grafo de antenas na consola.

Esta função percorre a lista ligada de antenas e apresenta as coordenadas e frequência Para cada antena, também, lista as antenas adjacentes (arestas) associadas

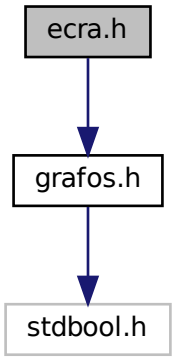
Parameters

inicio	Ponteiro para o início da lista de antenas

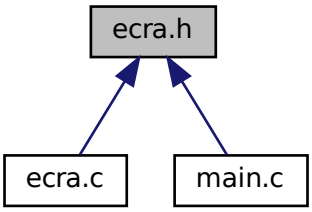
4.2 ecra.h File Reference

Define as funções para apresentar no ecrã

`#include "grafos.h"`
Include dependency graph for ecra.h:



This graph shows which files directly or indirectly include this file:



Functions

- void MostraGrafo (Antena *inicio)

  Mostra o grafo de antenas na consola.
- void ListarCaminhos (Caminho *caminhos)

  Lista todos os caminhos armazenados possiveis.

4.2.1 Detailed Description

Define as funções para apresentar no ecrã

Author

David Faria (a31517@alunos.ipca.pt)

Version

0.1

Date

2025-05-13

Copyright

Copyright (c) 2025

4.2.2 Function Documentation

4.2.2.1 ListarCaminhos()

```
void ListarCaminhos (
            Caminho * caminhos )
```

Lista todos os caminhos armazenados possiveis.

Parameters

caminhos	Ponteiro para o início da lista de caminhos a listar

Lista todos os caminhos armazenados possiveis.

Esta função percorre a lista ligada de caminhos e apresenta para cada caminho as coordenadas das antenas que o compõem, na ordem em que são percorridas

Parameters

caminhos	Ponteiro para o início da lista de caminhos a listar

4.2.2.2 MostraGrafo()

```
void MostraGrafo (
            Antena * inicio )
```

Mostra o grafo de antenas na consola.

Parameters

inicio	Ponteiro para o início da lista de antenas

Mostra o grafo de antenas na consola.

Esta função percorre a lista ligada de antenas e apresenta as coordenadas e frequência Para cada antena, também, lista as antenas adjacentes (arestas) associadas

Parameters

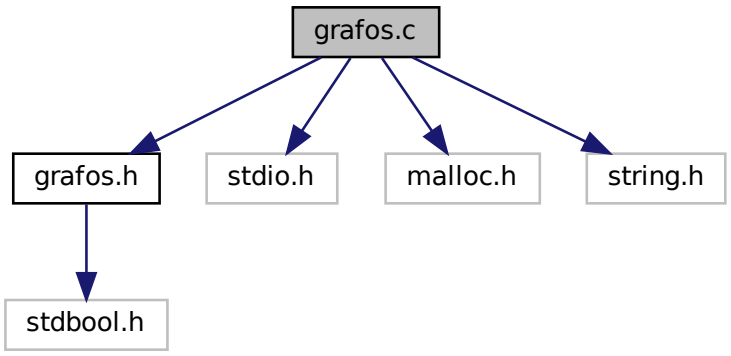
inicio	Ponteiro para o início da lista de antenas

4.3 grafos.c File Reference

Implementação de todas as funções.

```
#include "grafos.h"
#include <stdio.h>
#include <malloc.h>
#include <string.h>
```

Include dependency graph for grafos.c:



Functions

- Antena ∗ CriarAntena (char freq, int x, int y, bool ∗validar)

  Cria uma nova antena com os parâmetros especificados.
- Antena ∗ InserirAntena (Antena ∗inicio, Antena ∗nova, bool ∗validar)

  Insere uma nova antena numa lista ligada ordenada por coordenadas (y, x)
- Antena ∗ RemoverAntena (Antena ∗inicio, int x, int y, bool ∗validar)

  Remove uma antena da lista ligada com coordenadas específicas (x, y)
- Antena ∗ EncontrarAntena (Antena ∗inicio, int x, int y)

  Procura uma antena na lista ligada com coordenadas específicas (x, y)
- Adj ∗ CriarAresta (Antena ∗destino, bool ∗validar)

  Cria uma nova aresta que liga uma antena de origem a uma antena de destino.
- Adj ∗ InserirAresta (Antena ∗inicio, Adj ∗aresta, bool ∗validar)

  Insere uma nova aresta na lista de adjacências de uma antena.
- Adj ∗ CalcularArestas (Antena ∗inicio, bool ∗validar)

  Insere uma nova aresta na lista de adjacências de uma antena.
- Antena ∗ CarregarFicheiro (char ∗nomeficheiro, bool ∗validar)

  Carrega um ficheiro de texto e cria uma lista encadeada de antenas.
- bool GravarFicheiroBin (char ∗nomeficheiro, Antena ∗inicio)

  Grava a lista de antenas num ficheiro binário.
- Antena ∗ LerFicheiroBin (char ∗nomeficheiro, bool ∗validar)

  Lê uma lista de antenas e as respetivas adjacências de um ficheiro binário.
- void LimparVisitados (Antena ∗inicio, bool ∗validar)

  Limpa o estado de visita de todas as antenas na lista.
- void ProcuraProfundidade (Antena ∗antena, Antena ∗visitados[ ], int ∗nVisitados, bool ∗validar)

  Realiza uma busca em profundidade a partir de uma antena.
- Caminho ∗ GuardarCaminho (Antena ∗caminho[ ], int posicao, Caminho ∗caminhos)

  Armazena um caminho percorrido por um vetor de antenas numa lista ligada de caminhos.
- Caminho ∗ CalcularCaminhosEntreAntenas (Antena ∗origem, Antena ∗destino, Antena ∗caminho[ ], int posicao, Caminho ∗caminhos, bool ∗validar)

  Calcula todos os caminhos entre duas antenas com a mesma frequência.

4.3.1 Detailed Description

Implementação de todas as funções.

Author

David Faria (a31517@alunos.ipca.pt)

Version

0.1

Date

2025-05-12

Copyright

4.3.2 Function Documentation

4.3.2.1 CalcularArestas()

```
Adj* CalcularArestas (
            Antena * inicio,
            bool * validar )
```

Insere uma nova aresta na lista de adjacências de uma antena.

Calcula todas as arestas de adjacência no grafo de antenas.

Esta função adiciona a aresta `aresta` à lista de adjacências da antena `inicio` Se a lista de adjacências estiver vazia, a aresta torna-se o primeiro elemento Caso contrário, a aresta é inserida no final da lista Se ocorrer uma falha na alocação de memória ou se a lista de antenas estiver vazia, a operação é considerada inválida e o parâmetro `validar` é definido como `false`

Parameters

inicio	Ponteiro para a antena cuja lista de adjacências será atualizada
validar	Ponteiro para a nova aresta a ser inserida

Returns

Adj* Ponteiro para variável booleana onde será armazenado o resultado da operação

4.3.2.2 CalcularCaminhosEntreAntenas()

```
Caminho* CalcularCaminhosEntreAntenas (
            Antena * origem,
            Antena * destino,
            Antena * caminho[],
            int posicao,
            Caminho * caminhos,
            bool * validar )
```

Calcula todos os caminhos entre duas antenas com a mesma frequência.

Calcula todos os caminhos possíveis entre duas antenas no grafo

Esta função calcula todos caminhos possiveis da antena de origem até à antena de destino, armazenando todos os caminhos encontrados num vetor de antenas. Apenas são consideradas antenas com a mesma frequência da antena de origem e que ainda não tenham sido visitadas

Parameters

origem	Ponteiro para a antena de origem da busca
destino	Ponteiro para a antena de destino da busca
caminho	Vetor de ponteiros para antenas que representa o caminho percorrido até ao momento
posicao	Índice da próxima posição disponível no vetor `caminho`
caminhos	Ponteiro para a lista ligada de caminhos onde serão armazenados os caminhos encontrados
validar	Ponteiro para variável booleana que será setada a `true` se a operação for bem-sucedida

Returns

Caminho∗ Ponteiro para a lista ligada de caminhos atualizada com os novos caminhos encontrados

4.3.2.3 CarregarFicheiro()

```
Antena* CarregarFicheiro (
            char * nomeficheiro,
            bool * validar )
```

Carrega um ficheiro de texto e cria uma lista encadeada de antenas.

Carrega uma lista de antenas a partir de um ficheiro de texto.

Esta função lê um ficheiro de texto especificado por `nomeficheiro`, linha por linha, e cria uma lista encadeada de antenas representadas por caracteres não pontuais ('.') Cada linha do ficheiro representa uma linha de antenas, com cada carácter representando uma antena numa posição específica. As antenas são inseridas na lista de acordo com a sua posição (x, y) na matriz representada pelo ficheiro

Parameters

nomeficheiro	Nome do ficheiro de texto a ser lido
validar	Ponteiro para variável booleana onde será armazenado o resultado da operação

Returns

Antena* Ponteiro para o início da lista encadeada de antenas, ou `NULL` se ocorrer um erro

4.3.2.4 CriarAntena()

```
Antena* CriarAntena (
            char freq,
            int x,
            int y,
            bool * validar )
```

Cria uma nova antena com os parâmetros especificados.

Esta função aloca dinamicamente memória para uma nova estrutura `Antena`, inicializa os seus campos com os valores fornecidos e define o campo `visitado` como `false`. Caso a alocação de memória falhe, o parâmetro `validar` é atualizado para `false`; caso contrário, é atualizado para `true`.

Parameters

freq	Frequência de ressonância da antena
x	Coordenada X da posição da antena
y	Coordenada Y da posição da antena
validar	Ponteiro para uma variável booleana que indica se a operação foi bem-sucedida

Returns

Antena* Ponteiro para a nova antena criada, ou `NULL` se ocorrer um erro durante a alocação

4.3.2.5 CriarAresta()

```
Adj* CriarAresta (
            Antena * destino,
            bool * validar )
```

Cria uma nova aresta que liga uma antena de origem a uma antena de destino.

Cria uma nova aresta (ligação) entre antenas.

Esta função aloca dinamicamente memória para uma nova estrutura `Adj`, inicializa o campo `destino` com o ponteiro para a antena de destino fornecida e define o campo `prox` como `NULL`. Caso a alocação de memória falhe, o parâmetro `validar` é atualizado para `false`; caso contrário, é atualizado para `true`

Parameters

destino	Ponteiro para a antena de destino da aresta
validar	Ponteiro para uma variável booleana que indica se a operação foi bem-sucedida

Returns

Adj∗ Ponteiro para a nova aresta criada, ou `NULL` se ocorrer um erro durante a alocação

4.3.2.6 EncontrarAntena()

```
Antena* EncontrarAntena (
            Antena * inicio,
            int x,
            int y )
```

Procura uma antena na lista ligada com coordenadas específicas (x, y)

Procura uma antena na lista ligada com base nas suas coordenadas.

Esta função percorre a lista ligada de antenas `inicio` e retorna o ponteiro para a primeira antena que tenha as coordenadas (x, y) Se não encontrar nenhuma antena com essas coordenadas, a função retorna `NULL`

Parameters

inicio	Ponteiro para o início da lista de antenas
x	Coordenada x da antena a procurar
y	Coordenada y da antena a procurar

Returns

Antena∗ Ponteiro para a antena encontrada, ou `NULL` se não for encontrada

4.3.2.7 GravarFicheiroBin()

```
bool GravarFicheiroBin (
            char * nomeficheiro,
            Antena * inicio )
```

Grava a lista de antenas num ficheiro binário.

Esta função grava os dados da lista ligada de antenas e respetivas adjacências num ficheiro binário especificado por `nomeficheiro`

O formato do ficheiro é:

- Número total de antenas (int)
- Para cada antena:
  - Frequência (char)
  - Coordenada x (int)
  - Coordenada y (int)
  - Número de adjacências (int)
  - Para cada adjacência:
    - Coordenada x do destino (int)
    - Coordenada y do destino (int)

Parameters

nomeficheiro	Nome do ficheiro onde os dados serão gravados
inicio	Ponteiro para o início da lista de antenas

Returns

true se a gravação foi bem sucedida

false caso contrário

4.3.2.8 GuardarCaminho()

```
Caminho* GuardarCaminho (
            Antena * caminho[],
            int posicao,
            Caminho * caminhos )
```

Armazena um caminho percorrido por um vetor de antenas numa lista ligada de caminhos.

Guarda um caminho percorrido pelas antenas na lista de caminhos.

Esta função cria um novo nó na lista ligada de caminhos, copiando o vetor de antenas até a posição especificada, e insere-o no início da lista

Parameters

caminho	Vetor de ponteiros para antenas que representam o caminho percorrido
posicao	Índice da última antena válida no vetor `caminho`
caminhos	Ponteiro para o início da lista ligada de caminhos

Returns

Caminho∗ Ponteiro para o novo início da lista ligada de caminhos

4.3.2.9 InserirAntena()

```
Antena* InserirAntena (
            Antena * inicio,
            Antena * nova,
            bool * validar )
```

Insere uma nova antena numa lista ligada ordenada por coordenadas (y, x)

Insere uma nova antena na lista ligada de antenas.

Esta função insere a antena `nova` na lista ligada de antenas `inicio`, mantendo a ordem crescente primeiro pela coordenada `y` e, em caso de empate, pela coordenada `x`. Caso já exista uma antena com as mesmas coordenadas, a inserção não é realizada e `validar` é definido como falso

Parameters

inicio	
nova	
validar	

Returns

Antena∗

4.3.2.10 InserirAresta()

```
Adj* InserirAresta (
            Antena * inicio,
            Adj * aresta,
            bool * validar )
```

Insere uma nova aresta na lista de adjacências de uma antena.

Parameters

inicio	Ponteiro para a lista de antenas
aresta	Ponteiro para a aresta a ser inserida
validar	Ponteiro para uma variável booleana que indica se a inserção foi bem-sucedida

Returns

Adj∗ Ponteiro para a lista de antenas atualizada

4.3.2.11 LerFicheiroBin()

```
Antena* LerFicheiroBin (
            char * nomeficheiro,
            bool * validar )
```

Lê uma lista de antenas e as respetivas adjacências de um ficheiro binário.

Lê um ficheiro binário e carrega os dados das antenas.

Esta função carrega a informação de antenas e as suas adjacências a partir de um ficheiro binário especificado por `nomeficheiro` e recria a lista ligada de antenas

Parameters

nomeficheiro	Nome do ficheiro binário a ler
validar	Apontador para booleano que será setado a true se a leitura for bem sucedida

Returns

Antena* Ponteiro para o início da lista ligada de antenas, ou NULL em caso de erro

4.3.2.12 LimparVisitados()

```
void LimparVisitados (
            Antena * inicio,
            bool * validar )
```

Limpa o estado de visita de todas as antenas na lista.

Limpa o estado de visitação de todas as antenas no grafo.

Esta função percorre a lista ligada de antenas, começando pelo ponteiro `inicio`, e define o campo `visitado` de cada antena como `false`. É útil para reiniciar o estado de visita antes de realizar uma nova busca ou algoritmo que dependa desse campo, como a busca em profundidade e em largura etc...

Parameters

inicio	Ponteiro para o início da lista ligada de antenas
validar	Ponteiro para variável booleana onde será armazenado o resultado da operação

4.3.2.13 ProcuraProfundidade()

```
void ProcuraProfundidade (
            Antena * antena,
```

```
            Antena * visitados[],
            int * nVisitados,
            bool * validar )
```

Realiza uma busca em profundidade a partir de uma antena.

Esta função percorre recursivamente as antenas adjacentes à antena fornecida, marcando-as como visitadas e armazenando-as no vetor `visitados`. A busca é limitada às antenas que ainda não foram visitadas e que partilham a mesma frequência da antena de origem.

Parameters

antena	Ponteiro para a antena de origem da busca
visitados	Vetor onde serão armazenadas as antenas visitadas durante a busca
nVisitados	Ponteiro para o número de antenas visitadas até o momento
validar	Ponteiro para variável booleana que será setada a `true` se a operação for bem-sucedida

4.3.2.14 RemoverAntena()

```
Antena* RemoverAntena (
            Antena * inicio,
            int x,
            int y,
            bool * validar )
```

Remove uma antena da lista ligada com coordenadas específicas (x, y)

Remove uma antena da lista ligada com base nas suas coordenadas.

Esta função procura e remove a antena na lista ligada `inicio` que tenha as coordenadas (x, y) Se a antena for encontrada, é removida da lista, a memória é libertada, e o ponteiro para o início da lista é atualizado Caso contrário, a lista permanece inalterada e `validar` é definido como falso.

Parameters

inicio	Ponteiro para o início da lista de antenas
x	Coordenada x da antena a remover
y	Coordenada y da antena a remover
validar	Ponteiro para variável booleana onde será armazenado o resultado da operação

Returns

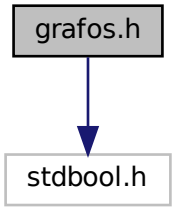
Antena∗ Ponteiro para o início atualizado da lista de antenas

4.4 grafos.h File Reference

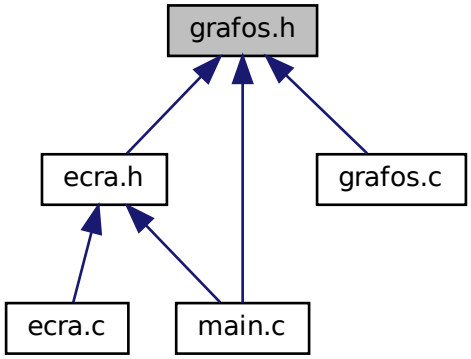
Define as funções e estruturas.

```
#include <stdbool.h>
```

Include dependency graph for grafos.h:



This graph shows which files directly or indirectly include this file:



Classes

- struct Antena

  Esta estrutura modela uma antena no grafo e contém informações sobre a sua frequência, posição no espaço (coordenadas x e y), ligações a outras antenas (adjacências), um ponteiro para a próxima antena na lista ligada e uma flag de visita para saber se esta ja foi visitada ou não.
- struct Adj

  Representa uma aresta entre antenas no grafo.
- struct Caminho

  Representa um caminho composto por uma sequência de antenas.

Macros

- #define **TAM_MAX_LINHA** 200
- #define **MAX_ANTENAS** 100

Typedefs

- typedef struct Antena Antena

  Esta estrutura modela uma antena no grafo e contém informações sobre a sua frequência, posição no espaço (coordenadas x e y), ligações a outras antenas (adjacências), um ponteiro para a próxima antena na lista ligada e uma flag de visita para saber se esta ja foi visitada ou não.
- typedef struct Adj Adj

  Representa uma aresta entre antenas no grafo.
- typedef struct Caminho Caminho

  Representa um caminho composto por uma sequência de antenas.

Functions

- Antena ∗ CriarAntena (char freq, int x, int y, bool ∗validar)

  Cria uma nova antena com os parâmetros especificados.
- Antena ∗ InserirAntena (Antena ∗inicio, Antena ∗nova, bool ∗validar)

  Insere uma nova antena na lista ligada de antenas.
- Antena ∗ RemoverAntena (Antena ∗inicio, int x, int y, bool ∗validar)

  Remove uma antena da lista ligada com base nas suas coordenadas.
- Antena ∗ EncontrarAntena (Antena ∗inicio, int x, int y)

  Procura uma antena na lista ligada com base nas suas coordenadas.
- Adj ∗ CriarAresta (Antena ∗destino, bool ∗validar)

  Cria uma nova aresta (ligação) entre antenas.
- Adj ∗ InserirAresta (Antena ∗inicio, Adj ∗aresta, bool ∗validar)

  Insere uma nova aresta na lista de adjacências de uma antena.
- Adj ∗ CalcularArestas (Antena ∗inicio, bool ∗validar)

  Calcula todas as arestas de adjacência no grafo de antenas.
- Antena ∗ CarregarFicheiro (char ∗nomeficheiro, bool ∗validar)

  Carrega uma lista de antenas a partir de um ficheiro de texto.
- bool GravarFicheiroBin (char ∗nomeficheiro, Antena ∗inicio)

  Grava a lista de antenas num ficheiro binário.
- Antena ∗ LerFicheiroBin (char ∗nomeficheiro, bool ∗validar)

  Lê um ficheiro binário e carrega os dados das antenas.
- void LimparVisitados (Antena ∗inicio, bool ∗validar)

  Limpa o estado de visitação de todas as antenas no grafo.
- void ProcuraProfundidade (Antena ∗antena, Antena ∗visitados[ ], int ∗nVisitados, bool ∗validar)

  Realiza uma busca em profundidade a partir de uma antena.
- Caminho ∗ GuardarCaminho (Antena ∗caminho[ ], int posicao, Caminho ∗caminhos)

  Guarda um caminho percorrido pelas antenas na lista de caminhos.
- Caminho ∗ CalcularCaminhosEntreAntenas (Antena ∗origem, Antena ∗destino, Antena ∗caminho[ ], int posicao, Caminho ∗caminhos, bool ∗validar)

  Calcula todos os caminhos possíveis entre duas antenas no grafo

4.4.1 Detailed Description

Define as funções e estruturas.

Author

David Faria (a31517@alunos.ipca.pt)

Version

0.1

Date

2025-05-12

Copyright

4.4.2 Typedef Documentation

4.4.2.1 Adj

```
typedef struct Adj Adj
```

Representa uma aresta entre antenas no grafo.

Esta estrutura modela uma ligação entre duas antenas no grafo, onde cada uma representa uma adjacência (aresta) a partir de uma antena de origem para uma antena de destino

4.4.2.2 Caminho

```
typedef struct Caminho Caminho
```

Representa um caminho composto por uma sequência de antenas.

Esta estrutura modela um caminho no grafo, contendo um array de ponteiros para antenas que constituem o percurso, o número total de antenas no caminho e um ponteiro para o próximo caminho na lista ligada

4.4.3 Function Documentation

4.4.3.1 CalcularArestas()

```
Adj* CalcularArestas (
            Antena * inicio,
            bool * validar )
```

Calcula todas as arestas de adjacência no grafo de antenas.

Parameters

inicio	Ponteiro para o início da lista de antenas
validar	Ponteiro para uma variável booleana que indica se a operação foi bem-sucedida.

Returns

Adj∗ Ponteiro para a lista de arestas calculadas

Calcula todas as arestas de adjacência no grafo de antenas.

Esta função adiciona a aresta `aresta` à lista de adjacências da antena `inicio` Se a lista de adjacências estiver vazia, a aresta torna-se o primeiro elemento Caso contrário, a aresta é inserida no final da lista Se ocorrer uma falha na alocação de memória ou se a lista de antenas estiver vazia, a operação é considerada inválida e o parâmetro `validar` é definido como `false`

Parameters

inicio	Ponteiro para a antena cuja lista de adjacências será atualizada
validar	Ponteiro para a nova aresta a ser inserida

Returns

Adj∗ Ponteiro para variável booleana onde será armazenado o resultado da operação

4.4.3.2 CalcularCaminhosEntreAntenas()

```
Caminho* CalcularCaminhosEntreAntenas (
            Antena * origem,
            Antena * destino,
            Antena * caminho[],
            int posicao,
            Caminho * caminhos,
            bool * validar )
```

Calcula todos os caminhos possíveis entre duas antenas no grafo

Parameters

origem	Ponteiro para a antena de origem do caminho
destino	Ponteiro para a antena de destino do caminho
caminho	Vetor que armazena o caminho atual durante a busca
posicao	Índice que representa a posição atual no vetor `caminho`
caminhos	Ponteiro para a lista de caminhos onde os caminhos encontrados serão armazenados
validar	Ponteiro para uma variável booleana que indica se a operação foi bem-sucedida

Returns

Caminho* Ponteiro para a lista de caminhos atualizada com os novos caminhos encontrados

Calcula todos os caminhos possíveis entre duas antenas no grafo

Esta função calcula todos caminhos possiveis da antena de origem até à antena de destino, armazenando todos os caminhos encontrados num vetor de antenas. Apenas são consideradas antenas com a mesma frequência da antena de origem e que ainda não tenham sido visitadas

Parameters

origem	Ponteiro para a antena de origem da busca
destino	Ponteiro para a antena de destino da busca
caminho	Vetor de ponteiros para antenas que representa o caminho percorrido até ao momento
posicao	Índice da próxima posição disponível no vetor `caminho`
caminhos	Ponteiro para a lista ligada de caminhos onde serão armazenados os caminhos encontrados
validar	Ponteiro para variável booleana que será setada a `true` se a operação for bem-sucedida

Returns

Caminho* Ponteiro para a lista ligada de caminhos atualizada com os novos caminhos encontrados

4.4.3.3 CarregarFicheiro()

```
Antena* CarregarFicheiro (
            char * nomeficheiro,
            bool * validar )
```

Carrega uma lista de antenas a partir de um ficheiro de texto.

Parameters

nomeficheiro	Nome do ficheiro a ser carregado
validar	Ponteiro para uma variável booleana que indica se a operação foi bem-sucedida

Returns

Antena* Ponteiro para a lista de antenas carregada, ou NULL se ocorrer um erro durante a leitura

Carrega uma lista de antenas a partir de um ficheiro de texto.

Esta função lê um ficheiro de texto especificado por `nomeficheiro`, linha por linha, e cria uma lista encadeada de antenas representadas por caracteres não pontuais ('.') Cada linha do ficheiro representa uma linha de antenas, com cada carácter representando uma antena numa posição específica. As antenas são inseridas na lista de acordo com a sua posição (x, y) na matriz representada pelo ficheiro

Parameters

nomeficheiro	Nome do ficheiro de texto a ser lido
validar	Ponteiro para variável booleana onde será armazenado o resultado da operação

Returns

Antena∗ Ponteiro para o início da lista encadeada de antenas, ou `NULL` se ocorrer um erro

4.4.3.4 CriarAntena()

```
Antena* CriarAntena (
            char freq,
            int x,
            int y,
            bool * validar )
```

Cria uma nova antena com os parâmetros especificados.

Parameters

freq	Carácter representando a frequência de ressonância da antena
x	Coordenada horizontal (x) da antena
y	Coordenada vertical (y) da antena
validar	Ponteiro para uma variável booleana que indica se a criação foi bem-sucedida

Returns

Antena∗ Ponteiro para a nova antena criada, ou NULL se ocorrer um erro durante a alocação

Esta função aloca dinamicamente memória para uma nova estrutura `Antena`, inicializa os seus campos com os valores fornecidos e define o campo `visitado` como `false`. Caso a alocação de memória falhe, o parâmetro `validar` é atualizado para `false`; caso contrário, é atualizado para `true`.

Parameters

freq	Frequência de ressonância da antena
x	Coordenada X da posição da antena
y	Coordenada Y da posição da antena
validar	Ponteiro para uma variável booleana que indica se a operação foi bem-sucedida

Returns

Antena∗ Ponteiro para a nova antena criada, ou `NULL` se ocorrer um erro durante a alocação

4.4.3.5 CriarAresta()

```
Adj* CriarAresta (
            Antena * destino,
            bool * validar )
```

Cria uma nova aresta (ligação) entre antenas.

Parameters

destino	Ponteiro para a antena de destino da aresta
validar	Ponteiro para uma variável booleana que indica se a criação foi bem-sucedida

Returns

Adj* Ponteiro para a nova aresta criada, ou NULL se ocorrer um erro durante a alocação

Cria uma nova aresta (ligação) entre antenas.

Esta função aloca dinamicamente memória para uma nova estrutura `Adj`, inicializa o campo `destino` com o ponteiro para a antena de destino fornecida e define o campo `prox` como `NULL`. Caso a alocação de memória falhe, o parâmetro `validar` é atualizado para `false`; caso contrário, é atualizado para `true`

Parameters

destino	Ponteiro para a antena de destino da aresta
validar	Ponteiro para uma variável booleana que indica se a operação foi bem-sucedida

Returns

Adj* Ponteiro para a nova aresta criada, ou `NULL` se ocorrer um erro durante a alocação

4.4.3.6 EncontrarAntena()

```
Antena* EncontrarAntena (
            Antena * inicio,
            int x,
            int y )
```

Procura uma antena na lista ligada com base nas suas coordenadas.

Parameters

inicio	Ponteiro para o início da lista de antenas
x	Coordenada horizontal (x) da antena a procurar
y	Coordenada vertical (y) da antena a procurar

Returns

Antena* Ponteiro para a antena encontrada, ou NULL se não encontrar nenhuma

Procura uma antena na lista ligada com base nas suas coordenadas.

Esta função percorre a lista ligada de antenas `inicio` e retorna o ponteiro para a primeira antena que tenha as coordenadas (x, y) Se não encontrar nenhuma antena com essas coordenadas, a função retorna `NULL`

Parameters

inicio	Ponteiro para o início da lista de antenas
x	Coordenada x da antena a procurar
y	Coordenada y da antena a procurar

Returns

Antena* Ponteiro para a antena encontrada, ou `NULL` se não for encontrada

4.4.3.7 GravarFicheiroBin()

```
bool GravarFicheiroBin (
            char * nomeficheiro,
            Antena * inicio )
```

Grava a lista de antenas num ficheiro binário.

Parameters

nomeficheiro	Nome do ficheiro binário onde as antenas serão gravadas
inicio	Ponteiro para o início da lista de antenas a ser gravada

Returns

true se todas as antenas forem gravadas com sucesso

false em caso de erro

Esta função grava os dados da lista ligada de antenas e respetivas adjacências num ficheiro binário especificado por `nomeficheiro`

O formato do ficheiro é:

- Número total de antenas (int)
- Para cada antena:
  - Frequência (char)
  - Coordenada x (int)
  - Coordenada y (int)

- Número de adjacências (int)
- Para cada adjacência:
  - Coordenada x do destino (int)
  - Coordenada y do destino (int)

Parameters

nomeficheiro	Nome do ficheiro onde os dados serão gravados
inicio	Ponteiro para o início da lista de antenas

Returns

true se a gravação foi bem sucedida

false caso contrário

4.4.3.8 GuardarCaminho()

```
Caminho* GuardarCaminho (
            Antena * caminho[],
            int posicao,
            Caminho * caminhos )
```

Guarda um caminho percorrido pelas antenas na lista de caminhos.

Parameters

caminho	Vetor de ponteiros para as antenas que compõem o caminho
posicao	Índice que representa a posição final do caminho no vetor
caminhos	Ponteiro para a lista de caminhos onde o novo caminho será inserido

Returns

Caminho∗ Ponteiro para a lista de caminhos atualizada com o novo caminho

Guarda um caminho percorrido pelas antenas na lista de caminhos.

Esta função cria um novo nó na lista ligada de caminhos, copiando o vetor de antenas até a posição especificada, e insere-o no início da lista

Parameters

caminho	Vetor de ponteiros para antenas que representam o caminho percorrido
posicao	Índice da última antena válida no vetor `caminho`
caminhos	Ponteiro para o início da lista ligada de caminhos

Returns

Caminho* Ponteiro para o novo início da lista ligada de caminhos

4.4.3.9 InserirAntena()

```
Antena* InserirAntena (
            Antena * inicio,
            Antena * nova,
            bool * validar )
```

Insere uma nova antena na lista ligada de antenas.

Parameters

inicio	Ponteiro para o início da lista de antenas
nova	Ponteiro para a nova antena a ser inserida
validar	Ponteiro para uma variável booleana que indica se a inserção foi bem-sucedida

Returns

Antena* Ponteiro para o início atualizado da lista de antenas

Insere uma nova antena na lista ligada de antenas.

Esta função insere a antena `nova` na lista ligada de antenas `inicio`, mantendo a ordem crescente primeiro pela coordenada `y` e, em caso de empate, pela coordenada `x`. Caso já exista uma antena com as mesmas coordenadas, a inserção não é realizada e `validar` é definido como falso

Parameters

inicio	
nova	
validar	

Returns

Antena*

4.4.3.10 InserirAresta()

```
Adj* InserirAresta (
            Antena * inicio,
            Adj * aresta,
            bool * validar )
```

Insere uma nova aresta na lista de adjacências de uma antena.

Parameters

inicio	Ponteiro para a lista de antenas
aresta	Ponteiro para a aresta a ser inserida
validar	Ponteiro para uma variável booleana que indica se a inserção foi bem-sucedida

Returns

Adj∗ Ponteiro para a lista de antenas atualizada

4.4.3.11 LerFicheiroBin()

```
Antena* LerFicheiroBin (
            char * nomeficheiro,
            bool * validar )
```

Lê um ficheiro binário e carrega os dados das antenas.

Parameters

nomeficheiro	Nome do ficheiro binário a ser lido
validar	Ponteiro para uma variável booleana que indica se a operação foi bem-sucedida

Returns

Antena∗ Ponteiro para o início da lista de antenas carregada, ou NULL se ocorrer um erro durante a leitura

Lê um ficheiro binário e carrega os dados das antenas.

Esta função carrega a informação de antenas e as suas adjacências a partir de um ficheiro binário especificado por `nomeficheiro` e recria a lista ligada de antenas

Parameters

nomeficheiro	Nome do ficheiro binário a ler
validar	Apontador para booleano que será setado a true se a leitura for bem sucedida

Returns

Antena∗ Ponteiro para o início da lista ligada de antenas, ou NULL em caso de erro

4.4.3.12 LimparVisitados()

```
void LimparVisitados (
            Antena * inicio,
            bool * validar )
```

Limpa o estado de visitação de todas as antenas no grafo.

Parameters

inicio	Ponteiro para o início da lista de antenas
validar	Ponteiro para uma variável booleana que indica se a operação foi bem-sucedida

Limpa o estado de visitação de todas as antenas no grafo.

Esta função percorre a lista ligada de antenas, começando pelo ponteiro `inicio`, e define o campo `visitado` de cada antena como `false`. É útil para reiniciar o estado de visita antes de realizar uma nova busca ou algoritmo que dependa desse campo, como a busca em profundidade e em largura etc...

Parameters

inicio	Ponteiro para o início da lista ligada de antenas
validar	Ponteiro para variável booleana onde será armazenado o resultado da operação

4.4.3.13 ProcuraProfundidade()

```
void ProcuraProfundidade (
            Antena * antena,
            Antena * visitados[],
            int * nVisitados,
            bool * validar )
```

Realiza uma busca em profundidade a partir de uma antena.

Parameters

antena	Ponteiro para a antena de início da busca
visitados	Vetor de ponteiros para antenas já visitadas
nVisitados	Ponteiro para o número de antenas visitadas
validar	Ponteiro para uma variável booleana que indica se a operação foi bem-sucedida

Esta função percorre recursivamente as antenas adjacentes à antena fornecida, marcando-as como visitadas e armazenando-as no vetor `visitados`. A busca é limitada às antenas que ainda não foram visitadas e que partilham a mesma frequência da antena de origem.

Parameters

antena	Ponteiro para a antena de origem da busca
visitados	Vetor onde serão armazenadas as antenas visitadas durante a busca
nVisitados	Ponteiro para o número de antenas visitadas até o momento
validar	Ponteiro para variável booleana que será setada a `true` se a operação for bem-sucedida

4.4.3.14 RemoverAntena()

```
Antena* RemoverAntena (
            Antena * inicio,
            int x,
            int y,
            bool * validar )
```

Remove uma antena da lista ligada com base nas suas coordenadas.

Parameters

inicio	Ponteiro para o início da lista de antenas
x	Coordenada horizontal (x) da antena a remover
y	Coordenada vertical (y) da antena a remover
validar	Ponteiro para uma variável booleana que indica se a remoção foi bem-sucedida

Returns

Antena∗ Ponteiro para o início atualizado da lista de antenas

Remove uma antena da lista ligada com base nas suas coordenadas.

Esta função procura e remove a antena na lista ligada `inicio` que tenha as coordenadas (x, y) Se a antena for encontrada, é removida da lista, a memória é libertada, e o ponteiro para o início da lista é atualizado Caso contrário, a lista permanece inalterada e `validar` é definido como falso.

Parameters

inicio	Ponteiro para o início da lista de antenas
x	Coordenada x da antena a remover
y	Coordenada y da antena a remover
validar	Ponteiro para variável booleana onde será armazenado o resultado da operação

Returns

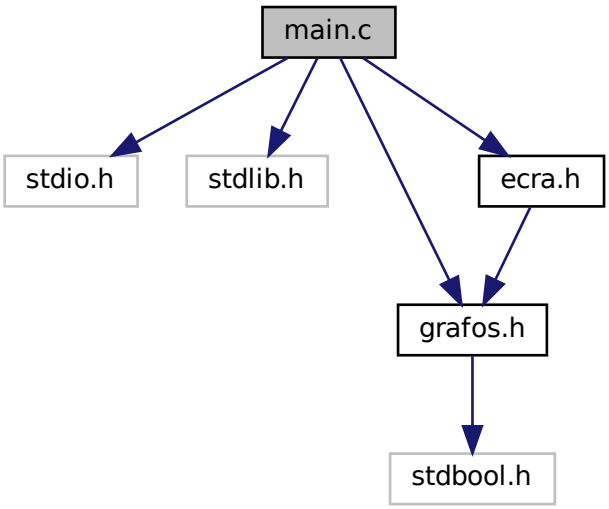
Antena∗ Ponteiro para o início atualizado da lista de antenas

4.5 main.c File Reference

rograma principal que executa as funções

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

```
#include "ecra.h"
```
Include dependency graph for main.c:



Functions

- int main ()

  Função principal do programa.

4.5.1 Detailed Description

rograma principal que executa as funções

Author

David Faria (a31517@alunos.ipca.pt)

Version

0.1

Date

2025-05-12

Copyright

Copyright (c) 2025

4.5.2 Function Documentation

4.5.2.1 main()

```
int main ( )
```

Função principal do programa.

Esta função é responsável por coordenar a execução principal do programa, que envolve:

- Carregar o grafo de antenas a partir de um ficheiro de texto
- Calcular as adjacências entre antenas
- Mostrar o grafo e as suas adjacências
- Gravar o grafo num ficheiro binário
- Ler o grafo a partir do ficheiro binário
- Limpar os marcadores de "visitado" antes de uma procura em profundidade
- Executar uma procura em profundidade no grafo
- Calcular todos os caminhos possíveis entre duas antenas especificadas pelo utilizador

Returns

int Retorna 0 no fim da execução

Index

Adj, 5
    grafos.h, 25
Antena, 6

CalcularArestas
    grafos.c, 15
    grafos.h, 25
CalcularCaminhosEntreAntenas
    grafos.c, 15
    grafos.h, 26
Caminho, 7
    grafos.h, 25
CarregarFicheiro
    grafos.c, 16
    grafos.h, 27
CriarAntena
    grafos.c, 17
    grafos.h, 28
CriarAresta
    grafos.c, 17
    grafos.h, 28

ecra.c, 9
    ListarCaminhos, 10
    MostraGrafo, 10
ecra.h, 11
    ListarCaminhos, 12
    MostraGrafo, 13
EncontrarAntena
    grafos.c, 18
    grafos.h, 29

grafos.c, 13
    CalcularArestas, 15
    CalcularCaminhosEntreAntenas, 15
    CarregarFicheiro, 16
    CriarAntena, 17
    CriarAresta, 17
    EncontrarAntena, 18
    GravarFicheiroBin, 18
    GuardarCaminho, 19
    InserirAntena, 19
    InserirAresta, 20
    LerFicheiroBin, 20
    LimparVisitados, 21
    ProcuraProfundidade, 21
    RemoverAntena, 22
grafos.h, 22
    Adj, 25
    CalcularArestas, 25
    CalcularCaminhosEntreAntenas, 26
    Caminho, 25
    CarregarFicheiro, 27
    CriarAntena, 28
    CriarAresta, 28
    EncontrarAntena, 29
    GravarFicheiroBin, 30
    GuardarCaminho, 31
    InserirAntena, 32
    InserirAresta, 32
    LerFicheiroBin, 33
    LimparVisitados, 33
    ProcuraProfundidade, 34
    RemoverAntena, 34
GravarFicheiroBin
    grafos.c, 18
    grafos.h, 30
GuardarCaminho
    grafos.c, 19
    grafos.h, 31

InserirAntena
    grafos.c, 19
    grafos.h, 32
InserirAresta
    grafos.c, 20
    grafos.h, 32

LerFicheiroBin
    grafos.c, 20
    grafos.h, 33
LimparVisitados
    grafos.c, 21
    grafos.h, 33
ListarCaminhos
    ecra.c, 10
    ecra.h, 12

main
    main.c, 37
main.c, 35
    main, 37
MostraGrafo
    ecra.c, 10
    ecra.h, 13

ProcuraProfundidade
    grafos.c, 21
    grafos.h, 34

RemoverAntena

grafos.c, 22
grafos.h, 34