

# Reference Manual

Generated by Doxygen 1.6.1

Fri Mar 26 23:50:25 2010



# Contents

<b>1</b>	<b>Class Index</b>	<b>1</b>
1.1	Class List . . . . .	1
<b>2</b>	<b>File Index</b>	<b>3</b>
2.1	File List . . . . .	3
<b>3</b>	<b>Class Documentation</b>	<b>5</b>
3.1	adj Struct Reference . . . . .	5
3.1.1	Member Data Documentation . . . . .	5
3.1.1.1	id . . . . .	5
3.1.1.2	next . . . . .	5
3.1.1.3	nif . . . . .	5
3.1.1.4	peso . . . . .	5
3.2	grafo Struct Reference . . . . .	6
3.2.1	Member Data Documentation . . . . .	6
3.2.1.1	nodos . . . . .	6
3.2.1.2	ocupados . . . . .	6
3.3	lista_mensagens Struct Reference . . . . .	7
3.3.1	Member Data Documentation . . . . .	7
3.3.1.1	L_mensagens . . . . .	7
3.3.1.2	total_utilizadores . . . . .	7
3.4	mensagens Struct Reference . . . . .	8
3.4.1	Member Data Documentation . . . . .	8
3.4.1.1	mensagem . . . . .	8
3.4.1.2	next . . . . .	8
3.5	perfil Struct Reference . . . . .	9
3.5.1	Member Data Documentation . . . . .	9
3.5.1.1	apagado . . . . .	9
3.5.1.2	cidade . . . . .	9

3.5.1.3	email	9
3.5.1.4	estado_civil	9
3.5.1.5	id	9
3.5.1.6	nif	9
3.5.1.7	nome	9
3.6	remetente Struct Reference	10
3.6.1	Member Data Documentation	10
3.6.1.1	id	10
3.6.1.2	msg	10
3.6.1.3	n_mensagens	10
3.6.1.4	next	10
3.6.1.5	nif	10
3.7	user_hash Struct Reference	11
3.7.1	Member Data Documentation	11
3.7.1.1	next	11
3.7.1.2	user	11
<b>4</b>	<b>File Documentation</b>	<b>13</b>
4.1	define.h File Reference	13
4.1.1	Define Documentation	14
4.1.1.1	AMI	14
4.1.1.2	END	14
4.1.1.3	FAM	14
4.1.1.4	MAXAPAGADO	14
4.1.1.5	MAXCIDADE	14
4.1.1.6	MAXEMAIL	14
4.1.1.7	MAXESTADOCIV	14
4.1.1.8	MAXNIF	14
4.1.1.9	MAXNODES	14
4.1.1.10	MAXNOME	14
4.1.1.11	MAXUSERSTR	14
4.1.1.12	PRO	14
4.2	estruturas.h File Reference	15
4.2.1	Typedef Documentation	15
4.2.1.1	Perfil	15
4.2.1.2	User_h	15
4.2.2	Variable Documentation	15

4.2.2.1	hash_nif	15
4.2.2.2	hash_nome	15
4.3	main.c File Reference	16
4.3.1	Function Documentation	16
4.3.1.1	executa_comando	16
4.3.1.2	main	16
4.3.1.3	menu	16
4.3.1.4	menu_relacoes	16
4.4	mensagens.c File Reference	17
4.4.1	Function Documentation	17
4.4.1.1	grava_mensagem	17
4.4.1.2	print_mensagens	17
4.5	mensagens.h File Reference	18
4.5.1	Typedef Documentation	18
4.5.1.1	Lista_mensagens	18
4.5.1.2	Mensagens	18
4.5.1.3	Remetente	18
4.5.2	Function Documentation	18
4.5.2.1	grava_mensagem	18
4.5.2.2	print_mensagens	19
4.6	operacoes.c File Reference	20
4.6.1	Function Documentation	20
4.6.1.1	apagar_user	20
4.6.1.2	carregar_bd	20
4.6.1.3	check_input	21
4.6.1.4	converte_ll_str	21
4.6.1.5	F_hash_nif	21
4.6.1.6	F_hash_nome	21
4.6.1.7	findByName	22
4.6.1.8	findByNif	22
4.6.1.9	guarda_user	22
4.6.1.10	guardar_bd	22
4.6.1.11	guardar_str	23
4.6.1.12	inserir_user	23
4.6.1.13	ler_dados	23
4.6.1.14	ler_numero_int	23

4.6.1.15	ler_numero_ll	23
4.6.1.16	ler_parametro_f	23
4.6.1.17	ler_user_ficheiro	24
4.6.1.18	print_user	24
4.6.1.19	resposta	24
4.6.1.20	saveToHash	24
4.7	operacoes.h File Reference	25
4.7.1	Function Documentation	25
4.7.1.1	apagar_user	25
4.7.1.2	carregar_bd	26
4.7.1.3	check_input	26
4.7.1.4	converte_ll_str	26
4.7.1.5	F_hash_nif	26
4.7.1.6	F_hash_nome	26
4.7.1.7	findByName	27
4.7.1.8	findByNif	27
4.7.1.9	guarda_user	27
4.7.1.10	guardar_bd	27
4.7.1.11	inserir_user	28
4.7.1.12	ler_dados	28
4.7.1.13	ler_numero_int	28
4.7.1.14	ler_numero_ll	28
4.7.1.15	ler_parametro_f	29
4.7.1.16	ler_user_ficheiro	29
4.7.1.17	print_user	29
4.7.1.18	resposta	29
4.7.1.19	saveToHash	30
4.8	relacoes.c File Reference	31
4.8.1	Function Documentation	31
4.8.1.1	alterar_relacao	31
4.8.1.2	inserir_ligacao	32
4.8.1.3	relacoes_inserir	32
4.8.1.4	relacoes_remove	32
4.8.1.5	remove	33
4.8.1.6	ver_relacao	33
4.8.1.7	ver_relacoes	34

---

4.9	relacoes.h File Reference . . . . .	35
4.9.1	Typedef Documentation . . . . .	35
4.9.1.1	Adj . . . . .	35
4.9.1.2	Grafo . . . . .	35
4.9.2	Function Documentation . . . . .	35
4.9.2.1	alterar_relacao . . . . .	35
4.9.2.2	inserir_ligacao . . . . .	36
4.9.2.3	relacoes_inserir . . . . .	36
4.9.2.4	relacoes_remove . . . . .	37
4.9.2.5	remove . . . . .	37
4.9.2.6	ver_relacao . . . . .	37
4.9.2.7	ver_relacoes . . . . .	38





# Chapter 1

## Class Index

### 1.1 Class List

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

<a href="#">adj</a>	5
<a href="#">grafo</a>	6
<a href="#">lista_mensagens</a>	7
<a href="#">mensagens</a>	8
<a href="#">perfil</a>	9
<a href="#">remetente</a>	10
<a href="#">user_hash</a>	11



# Chapter 2

## File Index

### 2.1 File List

Here is a list of all files with brief descriptions:

<a href="#">define.h</a>	13
<a href="#">estruturas.h</a>	15
<a href="#">main.c</a>	16
<a href="#">mensagens.c</a>	17
<a href="#">mensagens.h</a>	18
<a href="#">operacoes.c</a>	20
<a href="#">operacoes.h</a>	25
<a href="#">relacoes.c</a>	31
<a href="#">relacoes.h</a>	35



# Chapter 3

## Class Documentation

### 3.1 adj Struct Reference

```
#include <relacoes.h>
```

#### Public Attributes

- long int [id](#)
- long long [nif](#)
- int [peso](#)
- struct [adj](#) \* [next](#)

#### 3.1.1 Member Data Documentation

**3.1.1.1** long int `adj::id`

**3.1.1.2** struct `adj*` `adj::next` [`read`]

**3.1.1.3** long long `adj::nif`

**3.1.1.4** int `adj::peso`

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

- [relacoes.h](#)

## 3.2 grafo Struct Reference

```
#include <relacoes.h>
```

### Public Attributes

- `int ocupados`
- `Adj * nodos [MAXNODES]`

### 3.2.1 Member Data Documentation

#### 3.2.1.1 `Adj* grafo::nodos[MAXNODES]`

#### 3.2.1.2 `int grafo::ocupados`

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

- [relacoes.h](#)

## 3.3 lista\_mensagens Struct Reference

```
#include <mensagens.h>
```

### Public Attributes

- int [total\\_utilizadores](#)
- [Remetente](#) \* [L\\_mensagens](#) [MAXNODES]

### 3.3.1 Member Data Documentation

**3.3.1.1** [Remetente](#)\* [lista\\_mensagens::L\\_mensagens](#)[MAXNODES]

**3.3.1.2** int [lista\\_mensagens::total\\_utilizadores](#)

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

- [mensagens.h](#)

## 3.4 mensagens Struct Reference

```
#include <mensagens.h>
```

### Public Attributes

- char [mensagem](#) [1025]
- struct [mensagens](#) \* [next](#)

### 3.4.1 Member Data Documentation

#### 3.4.1.1 char mensagens::mensagem[1025]

#### 3.4.1.2 struct mensagens\* mensagens::next [read]

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

- [mensagens.h](#)



## 3.5 perfil Struct Reference

```
#include <estruturas.h>
```

### Public Attributes

- long long [id](#)
- char [nome](#) [MAXNOME]
- long long [nif](#)
- char [cidade](#) [MAXCIDADE]
- char [estado\\_civil](#) [MAXESTADOCIV]
- char [email](#) [MAXEMAIL]
- int [apagado](#)

### 3.5.1 Member Data Documentation

**3.5.1.1** int perfil::apagado

**3.5.1.2** char perfil::cidade[MAXCIDADE]

**3.5.1.3** char perfil::email[MAXEMAIL]

**3.5.1.4** char perfil::estado\_civil[MAXESTADOCIV]

**3.5.1.5** long long perfil::id

**3.5.1.6** long long perfil::nif

**3.5.1.7** char perfil::nome[MAXNOME]

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

- [estruturas.h](#)

## 3.6 remetente Struct Reference

```
#include <mensagens.h>
```

### Public Attributes

- int [id](#)
- int [n\\_mensagens](#)
- long long [nif](#)
- [Mensagens](#) \* [msg](#)
- struct [remetente](#) \* [next](#)

### 3.6.1 Member Data Documentation

**3.6.1.1** int [remetente::id](#)

**3.6.1.2** [Mensagens](#)\* [remetente::msg](#)

**3.6.1.3** int [remetente::n\\_mensagens](#)

**3.6.1.4** struct [remetente](#)\* [remetente::next](#) [[read](#)]

**3.6.1.5** long long [remetente::nif](#)

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

- [mensagens.h](#)

## 3.7 user\_hash Struct Reference

```
#include <estruturas.h>
```

### Public Attributes

- [Perfil](#) \* [user](#)
- struct [user\\_hash](#) \* [next](#)

### 3.7.1 Member Data Documentation

**3.7.1.1** struct [user\\_hash](#)\* [user\\_hash::next](#) [[read](#)]

**3.7.1.2** [Perfil](#)\* [user\\_hash::user](#)

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

- [estruturas.h](#)



# Chapter 4

## File Documentation

### 4.1 define.h File Reference

#### Defines

- #define [MAXNODES](#) 10000
- #define [MAXNIF](#) 12
- #define [MAXNOME](#) 51
- #define [MAXCIDADE](#) 21
- #define [MAXESTADOCIV](#) 12
- #define [MAXEMAIL](#) 51
- #define [MAXUSERSTR](#) 6
- #define [MAXAPAGADO](#) 2
- #define [END](#) 2
- #define [FAM](#) "fam"
- #define [AMI](#) "ami"
- #define [PRO](#) "pro"

## **4.1.1 Define Documentation**

**4.1.1.1 #define AMI "ami"**

**4.1.1.2 #define END 2**

**4.1.1.3 #define FAM "fam"**

**4.1.1.4 #define MAXAPAGADO 2**

**4.1.1.5 #define MAXCIDADE 21**

**4.1.1.6 #define MAXEMAIL 51**

**4.1.1.7 #define MAXESTADOCIV 12**

**4.1.1.8 #define MAXNIF 12**

**4.1.1.9 #define MAXNODES 10000**

**4.1.1.10 #define MAXNOME 51**

**4.1.1.11 #define MAXUSERSTR 6**

**4.1.1.12 #define PRO "pro"**

## 4.2 estruturas.h File Reference

```
#include "define.h"
```

### Classes

- struct [perfil](#)
- struct [user\\_hash](#)

### Typedefs

- typedef struct [perfil](#) [Perfil](#)
- typedef struct [user\\_hash](#) [User\\_h](#)

### Variables

- [User\\_h](#) \* [hash\\_nome](#) [MAXNODES]
- [User\\_h](#) \* [hash\\_nif](#) [MAXNODES]

#### 4.2.1 Typedef Documentation

4.2.1.1 typedef struct [perfil](#) [Perfil](#)

4.2.1.2 typedef struct [user\\_hash](#) [User\\_h](#)

#### 4.2.2 Variable Documentation

4.2.2.1 [User\\_h](#)\* [hash\\_nif](#)[MAXNODES]

4.2.2.2 [User\\_h](#)\* [hash\\_nome](#)[MAXNODES]

## 4.3 main.c File Reference

```
#include "operacoes.h"
#include "relacoes.h"
#include "mensagens.h"
#include <stdlib.h>
#include <string.h>
```

### Functions

- int [menu](#) ()
- char \* [menu\\_relacoes](#) ()
- int [executa\\_comando](#) (int comando, [Grafo](#) \*f, [Grafo](#) \*a, [Grafo](#) \*p, [Lista\\_mensagens](#) \*lm, long long \*idUser)
- int [main](#) ()

### 4.3.1 Function Documentation

**4.3.1.1** int [executa\\_comando](#) (int *comando*, [Grafo](#) \*f, [Grafo](#) \*a, [Grafo](#) \*p, [Lista\\_mensagens](#) \*lm, long long \*IdUser)

**4.3.1.2** int [main](#) ()

**4.3.1.3** int [menu](#) ()

**4.3.1.4** char\* [menu\\_relacoes](#) ()



## 4.4 mensagens.c File Reference

```
#include "mensagens.h"
#include "operacoes.h"
#include "relacoes.h"
#include <stdio.h>
#include <stdlib.h>
```

### Functions

- int [grava\\_mensagem](#) (long long *nif1*, long long *nif2*, char \**mensagem*, [Lista\\_mensagens](#) \**lm*)
- int [print\\_mensagens](#) (long long *nif*, [Lista\\_mensagens](#) \**lm*)

#### 4.4.1 Function Documentation

##### 4.4.1.1 int [grava\\_mensagem](#) (long long *nif1*, long long *nif2*, char \* *mensagem*, [Lista\\_mensagens](#) \* *lm*)

A função [grava\\_mensagem](#) irá guardar as mensagens de cada utilizador para um destinatario, criando uma lista de [mensagens](#) que cada utilizador recebeu.

##### Parameters:

- nif1* campo correspondente ao nif de quem vamos mandar a mensagem.  
*nif2* campo correspondente ao nif de quem está a mandar a mensagem.  
*mensagem* mensagem que se pretend enviar.  
*lm* estrutura de dados onde se guarda todas as [mensagens](#) de todos os utilizadores.

##### Returns:

Retorna um inteiro que verifica se conseguiu ou não efectuar a operação.

##### 4.4.1.2 int [print\\_mensagens](#) (long long *nif*, [Lista\\_mensagens](#) \* *lm*)

A função [print\\_mensagens](#) é responsavel por imprimir todas as [mensagens](#) dos diferentes destinatarios de um utilizador.

##### Parameters:

- nif* campo correspondente ao nif que vai ser utilizado para encontrar o id do utilizador atrves da função "findByNif"  
*lm* estrutura de dados onde vamos ver as [mensagens](#) de cada utilizador para depois podermos imprimilas.

##### Returns:

Retorna um inteiro que verifica se conseguiu ou não efectuar a operação.

## 4.5 mensagens.h File Reference

```
#include "define.h"
```

### Classes

- struct [mensagens](#)
- struct [remetente](#)
- struct [lista\\_mensagens](#)

### Typedefs

- typedef struct [mensagens](#) [Mensagens](#)
- typedef struct [remetente](#) [Remetente](#)
- typedef struct [lista\\_mensagens](#) [Lista\\_mensagens](#)

### Functions

- int [grava\\_mensagem](#) (long long *nif1*, long long *nif2*, char \**mensagem*, [Lista\\_mensagens](#) \**lm*)
- int [print\\_mensagens](#) (long long *nif*, [Lista\\_mensagens](#) \**lm*)

#### 4.5.1 Typedef Documentation

##### 4.5.1.1 typedef struct lista\_mensagens Lista\_mensagens

##### 4.5.1.2 typedef struct mensagens Mensagens

##### 4.5.1.3 typedef struct remetente Remetente

#### 4.5.2 Function Documentation

##### 4.5.2.1 int grava\_mensagem (long long *nif1*, long long *nif2*, char \* *mensagem*, [Lista\\_mensagens](#) \* *lm*)

A função `grava_mensagem` irá guardar as mensagens de cada utilizador para um destinatario, criando uma lista de [mensagens](#) que cada utilizador recebeu.

#### Parameters:

*nif1* campo correspondente ao nif de quem vamos mandar a mensagem.

*nif2* campo correspondente ao nif de quem está a mandar a mensagem.

*mensagem* mensagem que se pretend enviar.

*lm* estrutura de dados onde se guarda todas as [mensagens](#) de todos os utilizadores.

#### Returns:

Retorna um inteiro que verifica se conseguiu ou não efectuar a operação.

#### 4.5.2.2 int print\_mensagens (long long *nif*, Lista\_mensagens \* *lm*)

A função print\_mensagens é responsável por imprimir todas as mensagens dos diferentes destinatários de um utilizador.

**Parameters:**

*nif* campo correspondente ao nif que vai ser utilizado para encontrar o id do utilizador através da função "findByNif"

*lm* estrutura de dados onde vamos ver as mensagens de cada utilizador para depois podermos imprimilas.

**Returns:**

Retorna um inteiro que verifica se conseguiu ou não efectuar a operação.

## 4.6 operacoes.c File Reference

```
#include "operacoes.h"
```

### Functions

- long long [F\\_hash\\_nome](#) (char \*nome)
- long long [F\\_hash\\_nif](#) (long long nif)
- int [ler\\_numero\\_int](#) (char \*string)
- long long [ler\\_numero\\_ll](#) (char \*string)
- void [check\\_input](#) (char \*s)
- void [ler\\_dados](#) (char \*dado, char \*input, int dim)
- int [resposta](#) ()
- int [saveToHash](#) (long long IDnif, long long IDnome, [Perfil](#) \*userP)
- int [inserir\\_user](#) (long long \*IdUsers)
- int [apagar\\_user](#) (long long nif)
- int [print\\_user](#) ([Perfil](#) \*user)
- [Perfil](#) \* [findByNif](#) (long long nif)
- [Perfil](#) \* [findByName](#) (char \*name)
- void [ler\\_parametro\\_f](#) (FILE \*fp, int count, int dim, char \*parametro)
- int [ler\\_user\\_ficheiro](#) (FILE \*fp, [Perfil](#) \*\*ptr, int count, long long \*id)
- int [carregar\\_bd](#) (char \*input, long long \*id)
- void [guardar\\_str](#) (char \*str, int dim, FILE \*f)
- void [converte\\_ll\\_str](#) (long long numero, char \*numstr)
- void [guarda\\_user](#) (FILE \*fw, [Perfil](#) \*p)
- int [guardar\\_bd](#) (char \*output, long long \*id)

### 4.6.1 Function Documentation

#### 4.6.1.1 int [apagar\\_user](#) (long long *nif*)

Accede ao campo apagado de um [perfil](#) e marca com 1, que torna este utilizador apagado para as operações futuras.

##### Parameters:

*nif* Através do nif podemos chegar ao [perfil](#), através da função [findByNif](#) e alterar o mesmo

##### Returns:

o Um inteiro de verificação da operação

#### 4.6.1.2 int [carregar\\_bd](#) (char \* *input*, long long \* *id*)

Função que é invocada com o intuito de carregar toda a informação sobre os utilizadores para a estrutura de dados.

##### Parameters:

*input* nome do ficheiro onde se vai carregar a informação

*id* Este é chamado para a função para ser guardada a referencia ao ultimo id atribuido.

**Returns:**

Retorna um inteiro que verifica se a operação correu bem.

**4.6.1.3 void check\_input (char \* s)**

Função auxiliar que verifica se o input é correcto.

**4.6.1.4 void converte\_ll\_str (long long numero, char \* numstr)**

Função reponsavel por tranformar um numero em string

**Parameters:**

*numero* É o numero que se prentede tranforma em string

*numstr* string que vai ficar com o numero

**Returns:**

retorna um inteiro que verifica se correu bem.

**4.6.1.5 long long F\_hash\_nif (long long nif)**

Função reponsavel por gerar um indice para a tabela de hash\_nif O indice gerado esperace que seja o mais aleatorio possivel entre 0 e 10000 de forma a ter um bom desempenho na tabela de hash para isso a função tem varias condições de geração de numeros

**Parameters:**

*nif* gera-se o indice atraves deste long long

**Returns:**

retorna o indice para a tabela.

**4.6.1.6 long long F\_hash\_nome (char \* nome)**

Função reponsavel por gerar um indice para a tabela de hash\_nome O indice gerado esperace que seja o mais aleatorio possivel entre 0 e 10000 de forma a ter um bom desempenho na tabela de hash para isso a função tem varias condições de geração de numeros

**Parameters:**

*nome* atraves desta string vai se gerar o indicie para a tabela de hash\_nome

**Returns:**

retorna o indice para a tabela.

#### 4.6.1.7 Perfil\* findByName (char \* *name*)

Função responsável por encontrar um [perfil](#) através de um nome passado como argumento. Como existe a possibilidade de existir algumas pessoas com o mesmo nome é pedido uma confirmação de utilizador

**Parameters:**

*name* Nome do [perfil](#) que se pretende encontrar.

**Returns:**

o Perfil procurado

#### 4.6.1.8 Perfil\* findByNif (long long *nif*)

Função responsável por encontrar um [perfil](#) através de um numero passado como argumento.

**Parameters:**

*nif* Nif do [perfil](#) que se pretende encontrar.

**Returns:**

o Perfil procurado

#### 4.6.1.9 void guarda\_user (FILE \* *fw*, Perfil \* *p*)

Função que é invocada com o intuito de guardar a informação relativa a um utilizador A informação vai ser guardada num ficheiro que irá servir de base de dados

**Parameters:**

*fw* ficheiro onde se vai guardar a informação

*p* Perfil que se pretende guardar.

**Returns:**

Retorna um inteiro que verifica se a operação correu bem.

#### 4.6.1.10 int guardar\_bd (char \* *output*, long long \* *id*)

Função que é invocada com o intuito de guardar toda a informação sobre os utilizadores para que possa ser utilizada novamente. A informação vai ser guardada num ficheiro que irá servir de base de dados

**Parameters:**

*output* nome do ficheiro onde se vai guardar a informação

*id* Este é chamada para a função para ser guardado no ficheiro como um elemento à parte, pois será utilizado como referencia ao ultimo id atribuido.

**Returns:**

Retorna um inteiro que verifica se a operação correu bem.

**4.6.1.11 void guardar\_str (char \* *str*, int *dim*, FILE \* *f*)****4.6.1.12 int inserir\_user (long long \* *IdUsers*)**

Esta função recolhe os dados do utilizador que pretende criar o seu [perfil](#) na estrutura. Esta função usa a `saveToHash` para inserir o [perfil](#) nas tabelas de hash.

**Parameters:**

*IdUsers* Recebe o id do ultimo registo e incrementa-o

**Returns:**

retorna um inteiro de controlo da operação.

**4.6.1.13 void ler\_dados (char \* *dado*, char \* *input*, int *dim*)**

Função auxiliar que le os dados introduzidos pelo utilizador.

**4.6.1.14 int ler\_numero\_int (char \* *string*)**

Função reponsavel por tranformar uma string em numero

**Parameters:**

*string* String que contem o numero

**Returns:**

retorna o numero como int.

**4.6.1.15 long long ler\_numero\_ll (char \* *string*)**

Função reponsavel por tranformar uma string em numero

**Parameters:**

*string* String que contem o numero

**Returns:**

retorna o numero como long long.

**4.6.1.16 void ler\_parametro\_f (FILE \* *fp*, int *count*, int *dim*, char \* *parametro*)**

Função auxiliar, responsavel por ler os varios parametros do ficheiro de base de dados.

**Parameters:**

*fp* ficheiro da base de dados , de onde vão ser lidos os dados.

*dim* dimensao do parametro a ler

*parametro* a ler

*count* controlo do numero de parametros.

#### 4.6.1.17 `int ler_user_ficheiro (FILE * fp, Perfil ** ptr, int count, long long * id)`

Função que le os dados todos de um utilizador e os coloca num [perfil](#) de um user

##### Parameters:

*fp* ficheiro de base de dados de onde é retirado os dados

*ptr* [perfil](#) que recebe os parametros

*id* id a atribuir ao novos registos

*count* controlo de de leitura

##### Returns:

Interio de verificação.

#### 4.6.1.18 `int print_user (Perfil * user)`

Função que imprime os dados do [perfil](#) de um utilizador.

##### Parameters:

*user* Contem o [perfil](#) a imprimir

##### Returns:

o Um inteiro de verificação da operação

#### 4.6.1.19 `int resposta ()`

Função que recebe uma resposta do utilizador com yes ou no.

##### Returns:

um interio de verificação se a operação correu bem.

#### 4.6.1.20 `int saveToHash (long long IDnif, long long IDnome, Perfil * userP)`

A função responsavel por colocar o [perfil](#) do utilizador criado na estrutura de dados(na tabela de hash\_nome e tabela de hash\_nif). Ela regista o [perfil](#) nas posições que lhe são fornecidas através do IDnif e do IDnome.

##### Parameters:

*IDnif* indice para a tabela de hash\_nif

*IDnome* indice para a tabela de hash\_nome

*userP* [perfil](#) de tilizador a guardar

##### Returns:

um interio de verificação se a operação correu bem.



## 4.7 operacoes.h File Reference

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

### Functions

- long long [F\\_hash\\_nome](#) (char \*nome)
- long long [F\\_hash\\_nif](#) (long long nif)
- int [ler\\_numero\\_int](#) (char \*string)
- long long [ler\\_numero\\_ll](#) (char \*string)
- void [check\\_input](#) (char \*s)
- void [ler\\_dados](#) (char \*dado, char \*input, int dim)
- int [resposta](#) ()
- int [saveToHash](#) (long long IDnif, long long IDnome, [Perfil](#) \*userP)
- int [inserir\\_user](#) (long long \*IdUsers)
- int [apagar\\_user](#) (long long nif)
- int [print\\_user](#) ([Perfil](#) \*user)
- [Perfil](#) \* [findByNif](#) (long long nif)
- [Perfil](#) \* [findByName](#) (char \*name)
- void [ler\\_parametro\\_f](#) (FILE \*fp, int count, int dim, char \*parametro)
- int [ler\\_user\\_ficheiro](#) (FILE \*fp, [Perfil](#) \*\*ptr, int count, long long \*id)
- int [carregar\\_bd](#) (char \*input, long long \*id)
- void [converte\\_ll\\_str](#) (long long numero, char \*numstr)
- void [guarda\\_user](#) (FILE \*fw, [Perfil](#) \*p)
- int [guardar\\_bd](#) (char \*output, long long \*id)

### 4.7.1 Function Documentation

#### 4.7.1.1 int [apagar\\_user](#) (long long *nif*)

Accede ao campo apagado de um [perfil](#) e marca com 1, que torna este utilizador apagado para as operações futuras.

##### Parameters:

*nif* Através do nif podemos chegar ao [perfil](#), através da função [findByNif](#) e alterar o mesmo

##### Returns:

o Um inteiro de verificação da operação

#### 4.7.1.2 int carregar\_bd (char \* *input*, long long \* *id*)

Função que é invocada com o intuito de carregar toda a informação sobre os utilizadores para a estrutura de dados.

##### Parameters:

*input* nome do ficheiro onde se vai carregar a informação

*id* Este é chamado para a função para ser guardada a referencia ao ultimo id atribuido.

##### Returns:

Retorna um inteiro que verifica se a operação correu bem.

#### 4.7.1.3 void check\_input (char \* *s*)

Função auxiliar que verifica se o input é correcto.

#### 4.7.1.4 void converte\_ll\_str (long long *numero*, char \* *numstr*)

Função reponsavel por tranformar um numero em string

##### Parameters:

*numero* É o numero que se prentede tranforma em string

*numstr* string que vai ficar com o numero

##### Returns:

retorna um inteiro que verifica se correu bem.

#### 4.7.1.5 long long F\_hash\_nif (long long *nif*)

Função reponsavel por gerar um indice para a tabela de hash\_nif O indice gerado esperace que seja o mais aleatorio possivel entre 0 e 10000 de forma a ter um bom desempenho na tabela de hash para isso a função tem varias condições de geração de numeros

##### Parameters:

*nif* gera-se o indice atraves deste long long

##### Returns:

retorna o indice para a tabela.

#### 4.7.1.6 long long F\_hash\_nome (char \* *nome*)

Função reponsavel por gerar um indice para a tabela de hash\_nome O indice gerado esperace que seja o mais aleatorio possivel entre 0 e 10000 de forma a ter um bom desempenho na tabela de hash para isso a função tem varias condições de geração de numeros

**Parameters:**

*nome* através desta string vai se gerar o índice para a tabela de hash\_nome

**Returns:**

retorna o índice para a tabela.

**4.7.1.7 Perfil\* findByName (char \* name)**

Função responsável por encontrar um [perfil](#) através de um nome passado como argumento. Como existe a possibilidade de existir algumas pessoas com o mesmo nome é pedido uma confirmação de utilizador

**Parameters:**

*name* Nome do [perfil](#) que se pretende encontrar.

**Returns:**

o Perfil procurado

**4.7.1.8 Perfil\* findByNif (long long nif)**

Função responsável por encontrar um [perfil](#) através de um número passado como argumento.

**Parameters:**

*nif* Nif do [perfil](#) que se pretende encontrar.

**Returns:**

o Perfil procurado

**4.7.1.9 void guarda\_user (FILE \* fw, Perfil \* p)**

Função que é invocada com o intuito de guardar a informação relativa a um utilizador. A informação vai ser guardada num ficheiro que irá servir de base de dados

**Parameters:**

*fw* ficheiro onde se vai guardar a informação

*p* Perfil que se pretende guardar.

**Returns:**

Retorna um inteiro que verifica se a operação correu bem.

**4.7.1.10 int guardar\_bd (char \* output, long long \* id)**

Função que é invocada com o intuito de guardar toda a informação sobre os utilizadores para que possa ser utilizada novamente. A informação vai ser guardada num ficheiro que irá servir de base de dados

**Parameters:**

*output* nome do ficheiro onde se vai guardar a informação

*id* Este é chamada para a função para ser guardado no ficheiro como um elemento á parte, pois será utilizado como referencia ao ultimo id atribuido.

**Returns:**

Retorna um interio que verefica se a operação correu bem.

**4.7.1.11 int inserir\_user (long long \* *IdUsers*)**

Esta função recolhe os dados do utilizador que pretende criar o seu [perfil](#) na estrutura. Esta função usa a saveToHash para inserir o [perfil](#) nas tabelas de hash.

**Parameters:**

*IdUsers* Recebe o id do ultimo registo e incrementa-o

**Returns:**

retorna um inteiro de controlo da operação.

**4.7.1.12 void ler\_dados (char \* *dado*, char \* *input*, int *dim*)**

Função auxiliar que le os dados introduzidos pelo utilizador.

**4.7.1.13 int ler\_numero\_int (char \* *string*)**

Função reponsavel por tranformar uma string em numero

**Parameters:**

*string* String que contem o numero

**Returns:**

retorna o numero como int.

**4.7.1.14 long long ler\_numero\_ll (char \* *string*)**

Função reponsavel por tranformar uma string em numero

**Parameters:**

*string* String que contem o numero

**Returns:**

retorna o numero como long long.

**4.7.1.15 void ler\_parametro\_f (FILE \**fp*, int *count*, int *dim*, char \**parametro*)**

Função auxiliar, responsável por ler os vários parâmetros do ficheiro de base de dados.

**Parameters:**

*fp* ficheiro da base de dados , de onde vão ser lidos os dados.

*dim* dimensão do parâmetro a ler

*parametro* a ler

*count* controlo do número de parâmetros.

**4.7.1.16 int ler\_user\_ficheiro (FILE \**fp*, Perfil \*\**ptr*, int *count*, long long \**id*)**

Função que lê os dados todos de um utilizador e os coloca num [perfil](#) de um user

**Parameters:**

*fp* ficheiro de base de dados de onde é retirado os dados

*ptr* [perfil](#) que recebe os parâmetros

*id* id a atribuir aos novos registos

*count* controlo de leitura

**Returns:**

Inteiro de verificação.

**4.7.1.17 int print\_user (Perfil \**user*)**

Função que imprime os dados do [perfil](#) de um utilizador.

**Parameters:**

*user* Contém o [perfil](#) a imprimir

**Returns:**

o Um inteiro de verificação da operação

**4.7.1.18 int resposta ()**

Função que recebe uma resposta do utilizador com yes ou no.

**Returns:**

um inteiro de verificação se a operação correu bem.

**4.7.1.19 int saveToHash (long long *IDnif*, long long *IDnome*, Perfil \* *userP*)**

A função responsável por colocar o [perfil](#) do utilizador criado na estrutura de dados (na tabela de hash\_nome e tabela de hash\_nif). Ela regista o [perfil](#) nas posições que lhe são fornecidas através do *IDnif* e do *IDnome*.

**Parameters:**

*IDnif* índice para a tabela de hash\_nif

*IDnome* índice para a tabela de hash\_nome

*userP* [perfil](#) de utilizador a guardar

**Returns:**

um inteiro de verificação se a operação correu bem.

## 4.8 relacoes.c File Reference

```
#include "relacoes.h"
#include "operacoes.h"
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
```

### Functions

- int [inserir\\_ligacao](#) (long long nif1, long long nif2, int peso, [Grafo](#) \*g)
- int [remover](#) (long nif1, long nif2, char \*relacao, [Grafo](#) \*g)
- int [relacoes\\_remove](#) (long nif1, long nif2, char \*tipo, [Grafo](#) \*f, [Grafo](#) \*a, [Grafo](#) \*p)
- int [relacoes\\_inserir](#) (long nif1, long nif2, int peso, char \*tipo, [Grafo](#) \*f, [Grafo](#) \*a, [Grafo](#) \*p)
- int [alterar\\_relacao](#) (long nif1, long nif2, char \*tipo1, char \*tipo2, int peso, [Grafo](#) \*a, [Grafo](#) \*f, [Grafo](#) \*p)
- int [ver\\_relacao](#) (long long nif, [Grafo](#) \*g)
- int [ver\\_relacoes](#) (long nif, char \*tipo, [Grafo](#) \*f, [Grafo](#) \*a, [Grafo](#) \*p)

### 4.8.1 Function Documentation

#### 4.8.1.1 int [alterar\\_relacao](#) (long *nif1*, long *nif2*, char \* *tipo1*, char \* *tipo2*, int *peso*, [Grafo](#) \* *a*, [Grafo](#) \* *f*, [Grafo](#) \* *p*)

A função `alterar_relacao` irá eliminar a relação entre dois utilizadores, apagando esta definitivamente chamando a função "remove" e cria uma nova entre os mesmo utilizadores usando a função `inserir_ligacao`. Nesta função é passado o tipo de relação que se quer alterar e que se pretende alterar, e as estruturas correspondentes aos tres tipos de dados para se poderem efectuar as operações. A função vai ir decidir em qual das estruturas vai apagar a relação e criar a nova relação.

#### Parameters:

- nif1* campo correspondente ao nif de quem queremos alterar uma relação, este será útil para podermos ter o id do utilizador e aceder imediatamente ao campo correspondente na estrutura de dados.
- nif2* campo correspondente ao nif de quem está a alterar a ligação, este será útil para podermos ter o id do utilizador e aceder imediatamente ao campo correspondente na estrutura de dados.
- tipo1* É o tipo de relação que se vai eliminar.
- tipo2* É o tipo de relação que se vai criar.
- peso* É o novo peso da relação.
- f* estrutura de dados onde se guarda todas as relações de um mesmo tipo(familia).
- a* estrutura de dados onde se guarda todas as relações de um mesmo tipo(amigos).
- p* estrutura de dados onde se guarda todas as relações de um mesmo tipo(Profissional).

#### Returns:

Retorna um inteiro que verifica se conseguiu ou não efectuar a operação.

#### 4.8.1.2 int inserir\_ligacao (long long *nif1*, long long *nif2*, int *peso*, Grafo \* *g*)

A função `inserir_ligacao` irá criar uma ligação entre dois utilizadores, num determinado `grafo` ("estrutura de dados escolhida") que poderá corresponder a família, amigos ou profissional. Será então criada uma ligação entre dois utilizadores e será atribuído um determinado peso a essa ligação.

##### Parameters:

*nif1* campo correspondente ao nif de quem queremos criar uma relação, este será útil para podermos ter o id do utilizador e aceder imediatamente ao campo correspondente na estrutura de dados.

*nif2* campo correspondente ao nif de quem está a criar a ligação, este será útil para podermos ter o id do utilizador e aceder imediatamente ao campo correspondente na estrutura de dados.

*peso* É um avaliador da qualidade da relação.

*g* estrutura de dados onde se guarda todas as relações de um mesmo tipo.

##### Returns:

Retorna um inteiro que verifica se conseguiu ou não efectuar a operação.

#### 4.8.1.3 int relacoes\_inserir (long *nif1*, long *nif2*, int *peso*, char \* *tipo*, Grafo \* *f*, Grafo \* *a*, Grafo \* *p*)

A função `relacoes_inserir` irá criar uma relação entre dois utilizadores chamando a função "inserir". Nesta função é passado o tipo de relação a criar, e as estruturas correspondentes aos três tipos de dados. A função irá então decidir em qual das estruturas se vai criar a relação.

##### Parameters:

*nif1* campo correspondente ao nif de quem queremos criar uma relação, este será útil para podermos ter o id do utilizador e aceder imediatamente ao campo correspondente na estrutura de dados.

*peso* O novo peso da relação

*nif2* campo correspondente ao nif de quem está a criar a ligação, este será útil para podermos ter o id do utilizador e aceder imediatamente ao campo correspondente na estrutura de dados.

*tipo* É o tipo de relação que se vai criar.

*f* estrutura de dados onde se guarda todas as relações de um mesmo tipo(família).

*a* estrutura de dados onde se guarda todas as relações de um mesmo tipo(amigos).

*p* estrutura de dados onde se guarda todas as relações de um mesmo tipo(Profissional).

##### Returns:

Retorna um inteiro que verifica se conseguiu ou não efectuar a operação.

#### 4.8.1.4 int relacoes\_remove (long *nif1*, long *nif2*, char \* *tipo*, Grafo \* *f*, Grafo \* *a*, Grafo \* *p*)

A função `relacoes_remove` irá eliminar a relação entre dois utilizadores, apagando esta definitivamente chamando a função "remove". Nesta função é passado o tipo de relação a remover, e as estruturas correspondentes aos três tipos de dados. A função vai decidir em qual das estruturas vai apagar a relação.



**Parameters:**

*nif1* campo correspondente ao nif de quem queremos quebrar uma relação, este será útil para podermos ter o id do utilizador e aceder imediatamente ao campo correspondente na estrutura de dados.

*nif2* campo correspondente ao nif de quem está a quebrar a ligação, este será útil para podermos ter o id do utilizador e aceder imediatamente ao campo correspondente na estrutura de dados.

*tipo* É o tipo de relação que se vai eliminar.

*f* estrutura de dados onde se guarda todas as relações de um mesmo tipo(familia).

*a* estrutura de dados onde se guarda todas as relações de um mesmo tipo(amigos).

*p* estrutura de dados onde se guarda todas as relações de um mesmo tipo(Profissional).

**Returns:**

Retorna um inteiro que verifica se conseguiu ou não efectuar a operação.

**4.8.1.5 int remove (long *nif1*, long *nif2*, char \* *relacao*, Grafo \* *g*)**

A função remove irá eliminar a relação entre dois utilizadores, apagando esta definitivamente, Desta forma os dois utilizadores deixam de tar ligados numa determinada relação.

**Parameters:**

*nif1* campo correspondente ao nif de quem queremos quebrar uma relação, este será útil para podermos ter o id do utilizador e aceder imediatamente ao campo correspondente na estrutura de dados.

*nif2* campo correspondente ao nif de quem está a quebrar a ligação, este será útil para podermos ter o id do utilizador e aceder imediatamente ao campo correspondente na estrutura de dados.

*relacao* É o tipo de relação que se vai eliminar.

*g* estrutura de dados onde se guarda todas as relações de um mesmo tipo.

**Returns:**

Retorna um inteiro que verifica se conseguiu ou não efectuar a operação.

**4.8.1.6 int ver\_relacao (long long *nif*, Grafo \* *a*)**

A função ver\_relacao mostra os dados de um utilizador ligado a outro e os dados da relação. A função irá apenas mostrar uma relação específica.

**Parameters:**

*nif* É campo correspondente ao nif de quem queremos ver uma relação, este será útil para podermos ter o id do utilizador e aceder imediatamente ao campo correspondente na estrutura de dados.

*a* Estrutura de dados onde se guarda todas as relações de um mesmo tipo e que vai ser utilizada para consultarmos a relação.

**Returns:**

Retorna um inteiro que verifica se conseguiu ou não efectuar a operação.

**4.8.1.7 int ver\_relacoes (long *nif1*, char \* *tipo*, Grafo \* *f*, Grafo \* *a*, Grafo \* *p*)**

A função `ver_relacoes` mostra todos os utilizadores ligados a um outro, para isso ela vai invocando a função `ver_relacao`. A função irá mostrar todas as pessoas ligas a um utilizador num tipo especifico de relação.

**Parameters:**

*nif1* É campo correspondente ao nif de quem queremos ver as relações, este será útil para podermos ter o id do utilizador e aceder imediatamente ao campo correspondente na estrutura de dados.

*tipo* É o tipo de relação que se pretende ver as pessoas ligadas ao utilizador escolhido.

*f* estrutura de dados onde se guarda todas as relações de um mesmo tipo(familia).

*a* estrutura de dados onde se guarda todas as relações de um mesmo tipo(amigos).

*p* estrutura de dados onde se guarda todas as relações de um mesmo tipo(Proficional).

**Returns:**

Retorna um inteiro que verifica se conseguiu ou não efectuar a operação.

## 4.9 relacoes.h File Reference

```
#include "define.h"
```

### Classes

- struct [adj](#)
- struct [grafo](#)

### Typedefs

- typedef struct [adj](#) [Adj](#)
- typedef struct [grafo](#) [Grafo](#)

### Functions

- int [inserir\\_ligacao](#) (long long *nif1*, long long *nif2*, int *peso*, [Grafo](#) \**g*)
- int [remover](#) (long *nif1*, long *nif2*, char \**relacao*, [Grafo](#) \**g*)
- int [relacoes\\_remover](#) (long *nif1*, long *nif2*, char \**tipo*, [Grafo](#) \**f*, [Grafo](#) \**a*, [Grafo](#) \**p*)
- int [relacoes\\_inserir](#) (long *nif1*, long *nif2*, int *peso*, char \**tipo*, [Grafo](#) \**f*, [Grafo](#) \**a*, [Grafo](#) \**p*)
- int [alterar\\_relacao](#) (long *nif1*, long *nif2*, char \**tipo1*, char \**tipo2*, int *peso*, [Grafo](#) \**a*, [Grafo](#) \**f*, [Grafo](#) \**p*)
- int [ver\\_relacao](#) (long long *nif*, [Grafo](#) \**a*)
- int [ver\\_relacoes](#) (long *nif1*, char \**tipo*, [Grafo](#) \**f*, [Grafo](#) \**a*, [Grafo](#) \**p*)

### 4.9.1 Typedef Documentation

#### 4.9.1.1 typedef struct adj Adj

#### 4.9.1.2 typedef struct grafo Grafo

### 4.9.2 Function Documentation

#### 4.9.2.1 int alterar\_relacao (long *nif1*, long *nif2*, char \* *tipo1*, char \* *tipo2*, int *peso*, [Grafo](#) \* *a*, [Grafo](#) \* *f*, [Grafo](#) \* *p*)

A função `alterar_relacao` irá eliminar a relação entre dois utilizadores, apagando esta definitivamente chamando a função "remover" e cria uma nova entre os mesmo utilizadores usando a função `inserir_ligacao`. Nesta função é passado o tipo de relação que se quer alterar e que se pretende alterar, e as estruturas correspondentes aos tres tipos de dados para se poderem efectuar as operações. A função vai irá decidir em qual das estruturas vai apagar a relação e criar a nova relação.

#### Parameters:

- nif1* campo correspondente ao nif de quem queremos alterar uma relação, este será útil para podermos ter o id do utilizador e aceder imediatamente ao campo correspondente na estrutura de dados.
- nif2* campo correspondente ao nif de quem está a alterar a ligação, este será útil para podermos ter o id do utilizador e aceder imediatamente ao campo correspondente na estrutura de dados.
- tipo1* É o tipo de relação que se vai eliminar.

*tipo2* É o tipo de relação que se vai criar.

*peso* É o novo peso da relação.

*f* estrutura de dados onde se guarda todas as relações de um mesmo tipo(familia).

*a* estrutura de dados onde se guarda todas as relações de um mesmo tipo(amigos).

*p* estrutura de dados onde se guarda todas as relações de um mesmo tipo(Proficional).

#### Returns:

Retorna um inteiro que verifica se conseguiu ou não efectuar a operação.

#### 4.9.2.2 int inserir\_ligacao (long long *nif1*, long long *nif2*, int *peso*, Grafo \* *g*)

A função `inserir_ligacao` irá criar uma ligação entre dois utilizadores, num determinado `grafo`("estrutura de dados escolhida") que poderá corresponder a familia, amigos ou profissional. Será então criada uma ligação entre dois utilizadores e será atribuido um determinado peso a essa ligação.

#### Parameters:

*nif1* campo correspondente ao nif de quem queremos criar uma relação, este será util para podermos ter o id do utilizador e aceder imediatamente ao campo correspondente na estrutura de dados.

*nif2* campo correspondente ao nif de quem está a criar a ligação, este será util para podermos ter o id do utilizador e aceder imediatamente ao campo correspondente na estrutura de dados.

*peso* É um avaliador da qualidade da relação.

*g* estrutura de dados onde se guarda todas as relações de um mesmo tipo.

#### Returns:

Retorna um inteiro que verifica se conseguiu ou não efectuar a operação.

#### 4.9.2.3 int relacoes\_inserir (long *nif1*, long *nif2*, int *peso*, char \* *tipo*, Grafo \* *f*, Grafo \* *a*, Grafo \* *p*)

A função `relacoes_inserir` irá criar uma relação entre dois utulizadores chamando a função "inserir". Nesta função é passado o tipo de relação a criar, e as estruturas correspondetes aos tres tipos de dados. A função irá então decidir em qual das estruturas se vai criar a relação.

#### Parameters:

*nif1* campo correspondente ao nif de quem queremos criar uma relação, este será util para podermos ter o id do utilizador e aceder imediatamente ao campo correspondente na estrutura de dados.

*peso* O novo peso da relação

*nif2* campo correspondente ao nif de quem está a criar a ligação, este será util para podermos ter o id do utilizador e aceder imediatamente ao campo correspondente na estrutura de dados.

*tipo* É o tipo de relação que se vai criar.

*f* estrutura de dados onde se guarda todas as relações de um mesmo tipo(familia).

*a* estrutura de dados onde se guarda todas as relações de um mesmo tipo(amigos).

*p* estrutura de dados onde se guarda todas as relações de um mesmo tipo(Proficional).

#### Returns:

Retorna um inteiro que verifica se conseguiu ou não efectuar a operação.

#### 4.9.2.4 int relacoes\_remove (long *nif1*, long *nif2*, char \* *tipo*, Grafo \* *f*, Grafo \* *a*, Grafo \* *p*)

A função `relacoes_remove` irá eliminar a relação entre dois utilizadores, apagando esta definitivamente chamando a função "remove". Nesta função é passado o tipo de relação a remover, e as estruturas correspondentes aos três tipos de dados. A função vai decidir em qual das estruturas vai apagar a relação.

##### Parameters:

*nif1* campo correspondente ao nif de quem queremos quebrar uma relação, este será útil para podermos ter o id do utilizador e aceder imediatamente ao campo correspondente na estrutura de dados.

*nif2* campo correspondente ao nif de quem está a quebrar a ligação, este será útil para podermos ter o id do utilizador e aceder imediatamente ao campo correspondente na estrutura de dados.

*tipo* É o tipo de relação que se vai eliminar.

*f* estrutura de dados onde se guarda todas as relações de um mesmo tipo(familia).

*a* estrutura de dados onde se guarda todas as relações de um mesmo tipo(amigos).

*p* estrutura de dados onde se guarda todas as relações de um mesmo tipo(Profissional).

##### Returns:

Retorna um inteiro que verifica se conseguiu ou não efectuar a operação.

#### 4.9.2.5 int remove (long *nif1*, long *nif2*, char \* *relacao*, Grafo \* *g*)

A função `remove` irá eliminar a relação entre dois utilizadores, apagando esta definitivamente, Desta forma os dois utilizadores deixam de estar ligados numa determinada relação.

##### Parameters:

*nif1* campo correspondente ao nif de quem queremos quebrar uma relação, este será útil para podermos ter o id do utilizador e aceder imediatamente ao campo correspondente na estrutura de dados.

*nif2* campo correspondente ao nif de quem está a quebrar a ligação, este será útil para podermos ter o id do utilizador e aceder imediatamente ao campo correspondente na estrutura de dados.

*relacao* É o tipo de relação que se vai eliminar.

*g* estrutura de dados onde se guarda todas as relações de um mesmo tipo.

##### Returns:

Retorna um inteiro que verifica se conseguiu ou não efectuar a operação.

#### 4.9.2.6 int ver\_relacao (long long *nif*, Grafo \* *a*)

A função `ver_relacao` mostra os dados de um utilizador ligado a outro e os dados da relação. A função irá apenas mostrar uma relação específica.

##### Parameters:

*nif* É campo correspondente ao nif de quem queremos ver uma relação, este será útil para podermos ter o id do utilizador e aceder imediatamente ao campo correspondente na estrutura de dados.

*a* Estrutura de dados onde se guarda todas as relações de um mesmo tipo e que vai ser utilizada para consultarmos a relação.

**Returns:**

Retorna um inteiro que verifica se conseguiu ou não efectuar a operação.

**4.9.2.7 int ver\_relacoes (long *nif1*, char \* *tipo*, Grafo \* *f*, Grafo \* *a*, Grafo \* *p*)**

A função `ver_relacoes` mostra todos os utilizadores ligados a um outro, para isso ela vai invocando a função `ver_relacao`. A função irá mostrar todas as pessoas ligas a um utilizador num tipo especifico de relação.

**Parameters:**

*nif1* É campo correspondente ao nif de quem queremos ver as relações, este será util para podermos ter o id do utilizador e aceder imediatamente ao campo correspondente na estrutura de dados.

*tipo* É o tipo de relação que se pretende ver as pessoas ligadas ao utilizador escolhido.

*f* estrutura de dados onde se guarda todas as relações de um mesmo tipo(familia).

*a* estrutura de dados onde se guarda todas as relações de um mesmo tipo(amigos).

*p* estrutura de dados onde se guarda todas as relações de um mesmo tipo(Proficional).

**Returns:**

Retorna um inteiro que verifica se conseguiu ou não efectuar a operação.

# Index

- Adj
  - relacoes.h, 35
- adj, 5
  - id, 5
  - next, 5
  - nif, 5
  - peso, 5
- alterar\_relacao
  - relacoes.c, 31
  - relacoes.h, 35
- AMI
  - define.h, 14
- apagado
  - perfil, 9
- apagar\_user
  - operacoes.c, 20
  - operacoes.h, 25
- carregar\_bd
  - operacoes.c, 20
  - operacoes.h, 25
- check\_input
  - operacoes.c, 21
  - operacoes.h, 26
- cidade
  - perfil, 9
- converte\_ll\_str
  - operacoes.c, 21
  - operacoes.h, 26
- define.h, 13
  - AMI, 14
  - END, 14
  - FAM, 14
  - MAXAPAGADO, 14
  - MAXCIDADE, 14
  - MAXEMAIL, 14
  - MAXESTADOCIV, 14
  - MAXNIF, 14
  - MAXNODES, 14
  - MAXNOME, 14
  - MAXUSERSTR, 14
  - PRO, 14
  - perfil, 9
- END
  - define.h, 14
- estado\_civil
  - perfil, 9
- estruturas.h, 15
  - hash\_nif, 15
  - hash\_nome, 15
  - Perfil, 15
  - User\_h, 15
- executa\_comando
  - main.c, 16
- F\_hash\_nif
  - operacoes.c, 21
  - operacoes.h, 26
- F\_hash\_nome
  - operacoes.c, 21
  - operacoes.h, 26
- FAM
  - define.h, 14
- findByName
  - operacoes.c, 21
  - operacoes.h, 27
- findByNif
  - operacoes.c, 22
  - operacoes.h, 27
- Grafo
  - relacoes.h, 35
- grafo, 6
  - nodos, 6
  - ocupados, 6
- grava\_mensagem
  - mensagens.c, 17
  - mensagens.h, 18
- guarda\_user
  - operacoes.c, 22
  - operacoes.h, 27
- guardar\_bd
  - operacoes.c, 22
  - operacoes.h, 27
- guardar\_str
  - operacoes.c, 22
- hash\_nif

- estruturas.h, 15
- hash\_nome
  - estruturas.h, 15
- id
  - adj, 5
  - perfil, 9
  - remetente, 10
- inserir\_ligacao
  - relacoes.c, 31
  - relacoes.h, 36
- inserir\_user
  - operacoes.c, 23
  - operacoes.h, 28
- L\_mensagens
  - lista\_mensagens, 7
- ler\_dados
  - operacoes.c, 23
  - operacoes.h, 28
- ler\_numero\_int
  - operacoes.c, 23
  - operacoes.h, 28
- ler\_numero\_ll
  - operacoes.c, 23
  - operacoes.h, 28
- ler\_parametro\_f
  - operacoes.c, 23
  - operacoes.h, 28
- ler\_user\_ficheiro
  - operacoes.c, 23
  - operacoes.h, 29
- Lista\_mensagens
  - mensagens.h, 18
- lista\_mensagens, 7
  - L\_mensagens, 7
  - total\_utilizadores, 7
- main
  - main.c, 16
- main.c, 16
  - executa\_comando, 16
  - main, 16
  - menu, 16
  - menu\_relacoes, 16
- MAXAPAGADO
  - define.h, 14
- MAXCIDADE
  - define.h, 14
- MAXEMAIL
  - define.h, 14
- MAXESTADOCIV
  - define.h, 14
- MAXNIF
  - define.h, 14
- MAXNODES
  - define.h, 14
- MAXNOME
  - define.h, 14
- MAXUSERSTR
  - define.h, 14
- mensagem
  - mensagens, 8
- Mensagens
  - mensagens.h, 18
- mensagens, 8
  - mensagem, 8
  - next, 8
- mensagens.c, 17
  - grava\_mensagem, 17
  - print\_mensagens, 17
- mensagens.h, 18
  - grava\_mensagem, 18
  - Lista\_mensagens, 18
  - Mensagens, 18
  - print\_mensagens, 18
  - Remetente, 18
- menu
  - main.c, 16
- menu\_relacoes
  - main.c, 16
- msg
  - remetente, 10
- n\_mensagens
  - remetente, 10
- next
  - adj, 5
  - mensagens, 8
  - remetente, 10
  - user\_hash, 11
- nif
  - adj, 5
  - perfil, 9
  - remetente, 10
- nodos
  - grafo, 6
- nome
  - perfil, 9
- ocupados
  - grafo, 6
- operacoes.c, 20
  - apagar\_user, 20
  - carregar\_bd, 20
  - check\_input, 21
  - converte\_ll\_str, 21
  - F\_hash\_nif, 21



- F\_hash\_nome, 21
- findByName, 21
- findByNif, 22
- guarda\_user, 22
- guardar\_bd, 22
- guardar\_str, 22
- inserir\_user, 23
- ler\_dados, 23
- ler\_numero\_int, 23
- ler\_numero\_ll, 23
- ler\_parametro\_f, 23
- ler\_user\_ficheiro, 23
- print\_user, 24
- resposta, 24
- saveToHash, 24
- operacoes.h, 25
  - apagar\_user, 25
  - carregar\_bd, 25
  - check\_input, 26
  - converte\_ll\_str, 26
  - F\_hash\_nif, 26
  - F\_hash\_nome, 26
  - findByName, 27
  - findByNif, 27
  - guarda\_user, 27
  - guardar\_bd, 27
  - inserir\_user, 28
  - ler\_dados, 28
  - ler\_numero\_int, 28
  - ler\_numero\_ll, 28
  - ler\_parametro\_f, 28
  - ler\_user\_ficheiro, 29
  - print\_user, 29
  - resposta, 29
  - saveToHash, 29
- Perfil
  - estruturas.h, 15
- perfil, 9
  - apagado, 9
  - cidade, 9
  - email, 9
  - estado\_civil, 9
  - id, 9
  - nif, 9
  - nome, 9
- peso
  - adj, 5
- print\_mensagens
  - mensagens.c, 17
  - mensagens.h, 18
- print\_user
  - operacoes.c, 24
  - operacoes.h, 29
- PRO
  - define.h, 14
- relacoes.c, 31
  - alterar\_relacao, 31
  - inserir\_ligacao, 31
  - relacoes\_inserir, 32
  - relacoes\_remove, 32
  - remove, 33
  - ver\_relacao, 33
  - ver\_relacoes, 33
- relacoes.h, 35
  - Adj, 35
  - alterar\_relacao, 35
  - Grafo, 35
  - inserir\_ligacao, 36
  - relacoes\_inserir, 36
  - relacoes\_remove, 36
  - remove, 37
  - ver\_relacao, 37
  - ver\_relacoes, 38
- relacoes\_inserir
  - relacoes.c, 32
  - relacoes.h, 36
- relacoes\_remove
  - relacoes.c, 32
  - relacoes.h, 36
- Remetente
  - mensagens.h, 18
- remetente, 10
  - id, 10
  - msg, 10
  - n\_mensagens, 10
  - next, 10
  - nif, 10
- remove
  - relacoes.c, 33
  - relacoes.h, 37
- resposta
  - operacoes.c, 24
  - operacoes.h, 29
- saveToHash
  - operacoes.c, 24
  - operacoes.h, 29
- total\_utilizadores
  - lista\_mensagens, 7
- user
  - user\_hash, 11
- User\_h
  - estruturas.h, 15
- user\_hash, 11

next, [11](#)

user, [11](#)

ver\_relacao

relacoes.c, [33](#)

relacoes.h, [37](#)

ver\_relacoes

relacoes.c, [33](#)

relacoes.h, [38](#)