Reference Manual

Generated by Doxygen 1.6.1

Fri Mar 26 23:50:25 2010

Contents

Chapter 1

Class Index

1.1 Class List

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

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

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 mensagems 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 utlizado 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 mensagems 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 responsavel por imprimir todas as mensagens dos diferentes destinatarios de um utilizador.

Parameters:

nif campo correspondente ao nif que vai ser utlizado 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.

4.6 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*)

Acede 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 funçõa para ser guardada a referencia ao ultimo id atribuido.

Returns:

Retorna um interio que verefica se a operação correu bem.

4.6.1.3 void check_input (char * *s*)

Função auxiliar que verefica 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 responsavel por encontrar um perfil atraves 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 responsavel por encontrar um perfil atraves 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 prentede guardar.

Returns:

Retorna um interio que verefica 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 funçõa 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.

4.6.1.11 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, resposavel 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 vereficação.

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 vereficação 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 vereficação 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*)

Acede 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 funçõa para ser guardada a referencia ao ultimo id atribuido.

Returns:

Retorna um interio que verefica se a operação correu bem.

4.7.1.3 void check_input (char * *s*)

Função auxiliar que verefica 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 atraves desta string vai se gerar o indicie para a tabela de hash_nome

Returns:

retorna o indice para a tabela.

4.7.1.7 Perfil* findByName (char * *name*)

Função responsavel por encontrar um perfil atraves 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 responsavel por encontrar um perfil atraves de um numero 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 prentede guardar.

Returns:

Retorna um interio que verefica 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 funçõa 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, resposavel 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.

4.7.1.16 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 vereficação.

4.7.1.17 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

4.7.1.18 int resposta ()

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

Returns:

um interio de vereficação se a operação correu bem.

4.7.1.19 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 vereficação 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_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 *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_relação irá eleminar a relação entre dois utulizadores, apagando esta definitivamente chamando a função "remover" e cria uma nova entre os mesmo utilizadores usando a função inserir ligação. Nesta função é passado o tipo de relação que se quer alterar e que se pertende alterar, e as estruturas correspondetes 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 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 altera a ligação, este será util 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.

4.8.1.2 int inserir_ligacao (long long *nif1*, long long *nif2*, int *peso*, Grafo * *g*)

A função inserir_ligação 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.

4.8.1.3 int relacoes_inserir (long *nif1*, long *nif2*, int *peso*, char * *tipo*, Grafo * *f*, Grafo * *a*, Grafo * *p*)

A função relaçoes_inserir irá criar umaa relação entre dois utulizadores chamando a função "inserir". Nesta função é passado o tipo de relaçãoa 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 realção

nif2 campo correspondente ao nif de quem está 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.

4.8.1.4 int relacoes_remover (long *nif1*, long *nif2*, char * *tipo*, Grafo * *f*, Grafo * *a*, Grafo * *p*)

A função relaçoes_remover irá eleminar a relação entre dois utulizadores, apagando esta definitivamente chamando a função "remover". Nesta função é passado o tipo de relaçãoa a remover, e as estruturas correspondetes aos tres 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 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 quebrar 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 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(Proficional).

Returns:

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

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

A função remover irá eleminar a relação entre dois utulizadores, 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 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 quebrar a ligação, este será util 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_relação mostra os dados de um utilizador ligado a outro e os dadoas da relação. A função irá apenas mostrar uma relação especifica.

Parameters:

nif É campo correspondente ao nif de quem queremos ver uma relação, este será util 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 utlizada 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 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.

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_relação irá eleminar a relação entre dois utulizadores, apagando esta definitivamente chamando a função "remover" e cria uma nova entre os mesmo utilizadores usando a função inserir ligação. Nesta função é passado o tipo de relação que se quer alterar e que se pertende alterar, e as estruturas correspondetes 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 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 altera a ligação, este será util 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_ligação 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 relaçoes_inserir irá criar umaa relação entre dois utulizadores chamando a função "inserir". Nesta função é passado o tipo de relaçãoa 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 realção

nif2 campo correspondente ao nif de quem está 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_remover (long *nif1*, long *nif2*, char * *tipo*, Grafo * *f*, Grafo * *a*, Grafo * *p*)

A função relaçoes_remover irá eleminar a relação entre dois utulizadores, apagando esta definitivamente chamando a função "remover". Nesta função é passado o tipo de relaçãoa a remover, e as estruturas correspondetes aos tres 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 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 quebrar 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 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(Proficional).

Returns:

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

4.9.2.5 int remover (long *nif1*, long *nif2*, char * *relacao*, Grafo * *g*)

A função remover irá eleminar a relação entre dois utulizadores, 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 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 quebrar a ligação, este será util 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_relação mostra os dados de um utilizador ligado a outro e os dadoas da relação. A função irá apenas mostrar uma relação especifica.

Parameters:

nif É campo correspondente ao nif de quem queremos ver uma relação, este será util 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 utlizada 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