

My Project

Generated by Doxygen 1.10.0

1 Data Structure Index	1
1.1 Data Structures	1
2 File Index	3
2.1 File List	3
3 Data Structure Documentation	5
3.1 Adjacent Struct Reference	5
3.1.1 Detailed Description	5
3.1.2 Field Documentation	5
3.1.2.1 dist	5
3.1.2.2 id	5
3.1.2.3 next	6
3.1.2.4 visited	6
3.2 Graph Struct Reference	6
3.2.1 Detailed Description	6
3.2.2 Field Documentation	6
3.2.2.1 inicioGraph	6
3.2.2.2 numVertices	6
3.2.2.3 totVertices	7
3.3 Vertex Struct Reference	7
3.3.1 Detailed Description	7
3.3.2 Field Documentation	7
3.3.2.1 id	7
3.3.2.2 nextAdjacent	7
3.3.2.3 nextVertex	7
3.3.2.4 visited	8
3.4 VertexDist Struct Reference	8
3.4.1 Detailed Description	8
3.4.2 Field Documentation	8
3.4.2.1 dist	8
3.4.2.2 id	8
4 File Documentation	9
4.1 Functions.c File Reference	9
4.1.1 Detailed Description	10
4.1.2 Macro Definition Documentation	10
4.1.2.1 INF	10
4.1.3 Function Documentation	10
4.1.3.1 addAdjacent()	10
4.1.3.2 Função: addAdjacent	10
4.1.3.3 addVertex()	11
4.1.3.4 Função: addVertex	11

4.1.3.5 compare()	11
4.1.3.6 Função: compare	11
4.1.3.7 createAdjacent()	11
4.1.3.8 Função: createAdjacent	12
4.1.3.9 createGraph()	12
4.1.3.10 Função: createGraph	12
4.1.3.11 createVertex()	12
4.1.3.12 Função: createVertex	12
4.1.3.13 DFS()	13
4.1.3.14 Função: DFS	13
4.1.3.15 DFSUtil()	13
4.1.3.16 Função: DFSUtil	13
4.1.3.17 findLongestPath()	14
4.1.3.18 Função: findLongestPath	14
4.1.3.19 findVertexById()	14
4.1.3.20 Função: findVertexById	14
4.1.3.21 readGraphFromFile()	15
4.1.3.22 Função: readGraphFromFile	15
4.1.3.23 removeAdjacent()	15
4.1.3.24 Função: removeAdjacent	15
4.1.3.25 removeVertex()	16
4.1.3.26 Função: removeVertex	16
4.1.3.27 showGraph()	16
4.1.3.28 Função: showGraph	16
4.1.3.29 updateAdjacentDistance()	17
4.1.3.30 Função: updateAdjacentDistance	17
4.2 Header.h File Reference	17
4.2.1 Detailed Description	18
4.2.2 Macro Definition Documentation	18
4.2.2.1 MAX_PATH_LENGTH	18
4.2.2.2 MAX_VERTICES	19
4.2.3 Typedef Documentation	19
4.2.3.1 Adjacent	19
4.2.3.2 Graph	19
4.2.3.3 Vertex	19
4.2.4 Function Documentation	19
4.2.4.1 addAdjacent()	19
4.2.4.2 Função: addAdjacent	19
4.2.4.3 addVertex()	20
4.2.4.4 Função: addVertex	20
4.2.4.5 compare()	20
4.2.4.6 Função: compare	20

4.2.4.7 createAdjacent()	20
4.2.4.8 Função: createAdjacent	21
4.2.4.9 createGraph()	21
4.2.4.10 Função: createGraph	21
4.2.4.11 createVertex()	21
4.2.4.12 Função: createVertex	21
4.2.4.13 DFS()	22
4.2.4.14 Função: DFS	22
4.2.4.15 DFSUtil()	22
4.2.4.16 Função: DFSUtil	22
4.2.4.17 findLongestPath()	23
4.2.4.18 Função: findLongestPath	23
4.2.4.19 findVertexById()	23
4.2.4.20 Função: findVertexById	23
4.2.4.21 readGraphFromFile()	24
4.2.4.22 Função: readGraphFromFile	24
4.2.4.23 removeAdjacent()	24
4.2.4.24 Função: removeAdjacent	24
4.2.4.25 removeVertex()	25
4.2.4.26 Função: removeVertex	25
4.2.4.27 showGraph()	25
4.2.4.28 Função: showGraph	25
4.2.4.29 updateAdjacentDistance()	26
4.2.4.30 Função: updateAdjacentDistance	26
4.3 Header.h	26
4.4 Main.c File Reference	27
4.4.1 Function Documentation	27
4.4.1.1 main()	27
Index	29

Chapter 1

Data Structure Index

1.1 Data Structures

Here are the data structures with brief descriptions:

Adjacent	Estrutura para representar um vértice adjacente	5
Graph	Estrutura para representar um grafo	6
Vertex	Estrutura para representar um vértice	7
VertexDist	Estrutura para representar a distância de um vértice	8

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

Functions.c	Este ficheiro contém a implementação das funções do sistema de grafos	9
Header.h	Livraria que contém definições de tipos e estruturas para o sistema de grafos	17
Main.c	27

Chapter 3

Data Structure Documentation

3.1 Adjacent Struct Reference

Estrutura para representar um vértice adjacente.

```
#include <Header.h>
```

Data Fields

- int [id](#)
- float [dist](#)
- bool [visited](#)
- struct [Adjacent](#) * [next](#)

3.1.1 Detailed Description

Estrutura para representar um vértice adjacente.

A estrutura [Adjacent](#) representa um vértice adjacente de um grafo, ela tem o id que identifica a adjacência, a distancia que representa a distancia entre vértices, um campo visited para perceber se a adjacencia já foi visitada e um apontador para a proxima adjacencia.

3.1.2 Field Documentation

3.1.2.1 dist

```
float dist
```

3.1.2.2 id

```
int id
```

3.1.2.3 next

```
struct Adjacent* next
```

3.1.2.4 visited

```
bool visited
```

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

- [Header.h](#)

3.2 Graph Struct Reference

Estrutura para representar um grafo.

```
#include <Header.h>
```

Data Fields

- [Vertex](#) * [inicioGraph](#)
- int [numVertices](#)
- int [totVertices](#)

3.2.1 Detailed Description

Estrutura para representar um grafo.

A estrutura [Graph](#) representa um grafo, ela tem um apontador que aponta para o inicio da lista dos vértices, o número do momento de vértices e o total de vértices

3.2.2 Field Documentation

3.2.2.1 inicioGraph

```
Vertex* inicioGraph
```

3.2.2.2 numVertices

```
int numVertices
```

3.2.2.3 totVertices

```
int totVertices
```

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

- [Header.h](#)

3.3 Vertex Struct Reference

Estrutura para representar um vértice.

```
#include <Header.h>
```

Data Fields

- int [id](#)
- bool [visited](#)
- [Adjacent](#) * [nextAdjacent](#)
- struct [Vertex](#) * [nextVertex](#)

3.3.1 Detailed Description

Estrutura para representar um vértice.

A estrutura [Vertex](#) representa um vértice de um grafo, ela tem o id do vértice que identifica o vértice, um campo visited para ver se o vertice já foi visistado, um apontador para a lista de vértices adjacentes e um apontador para o próximo vértice da lista

3.3.2 Field Documentation

3.3.2.1 id

```
int id
```

3.3.2.2 nextAdjacent

```
Adjacent* nextAdjacent
```

3.3.2.3 nextVertex

```
struct Vertex* nextVertex
```

3.3.2.4 visited

```
bool visited
```

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

- [Header.h](#)

3.4 VertexDist Struct Reference

Estrutura para representar a distância de um vértice.

```
#include <Header.h>
```

Data Fields

- int [id](#)
- float [dist](#)

3.4.1 Detailed Description

Estrutura para representar a distância de um vértice.

A estrutura [VertexDist](#) é usada para armazenar o id de um vértice e a distância, facilitando a ordenação e manipulação em algoritmos que envolvem grafos.

3.4.2 Field Documentation

3.4.2.1 dist

```
float dist
```

3.4.2.2 id

```
int id
```

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

- [Header.h](#)

Chapter 4

File Documentation

4.1 Functions.c File Reference

Este ficheiro contém a implementação das funções do sistema de grafos.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "Header.h"
#include "stdbool.h"
#include <float.h>
```

Macros

- `#define INF FLT_MAX`

Functions

- `Graph * createGraph ()`
- `Vertex * createVertex (int id)`
- `Adjacent * createAdjacent (int id, float dist)`
- `Graph * addVertex (Graph *graph, int id)`
- `Adjacent * addAdjacent (Vertex *vertex, int id, float dist)`
- `Graph * readGraphFromFile (const char *filename)`
- `bool removeAdjacent (Graph *graph, int vertexId, int adjId)`
- `bool removeVertex (Graph *graph, int vertexId)`
- `bool updateAdjacentDistance (Graph *graph, int vertexId, int adjId, float newDist)`
- `Vertex * findVertexById (Graph *graph, int id)`
- `int DFSUtil (Graph *graph, Vertex *vertex, int path[], int pathLen, float currentDistance, int destinationId)`
- `int DFS (Graph *graph, int startVertexId, int destinationId)`
- `int compare (const void *a, const void *b)`
- `int findLongestPath (Graph *graph, int startVertexId)`
- `void showGraph (Graph *graph)`

4.1.1 Detailed Description

Este ficheiro contém a implementação das funções do sistema de grafos.

Author

Alexandre Vilaça 26590

Date

25.05.2024

4.1.2 Macro Definition Documentation

4.1.2.1 INF

```
#define INF FLT_MAX
```

4.1.3 Function Documentation

4.1.3.1 addAdjacent()

```
Adjacent * addAdjacent (
    Vertex * vertex,
    int id,
    float dist )
```

4.1.3.2 Função: addAdjacent

Adiciona um novo vértice adjacente a um vértice já existente.

Parameters

<i>vertex</i>	Um apontador para a estrutura Vertex .
<i>id</i>	O id do vértice adjacente.
<i>dist</i>	A distância até ao vértice adjacente.

Returns

Um apontador para a estrutura [Adjacent](#) após a adição da adjacência

A função cria um novo vértice adjacente e insere-o no início da lista de adjacentes do vértice conforme o id passado por parâmetro.

4.1.3.3 addVertex()

```
Graph * addVertex (
    Graph * graph,
    int id )
```

4.1.3.4 Função: addVertex

Adiciona um novo vértice ao grafo.

Parameters

<i>graph</i>	Um apontador para a estrutura Graph .
<i>id</i>	O identificador do vértice a ser adicionado.

Returns

Um apontador para a estrutura [Graph](#) após a adição.

A função cria um novo vértice e insere-o no início da lista de vértices do grafo Incrementa os contadores de vértices.

4.1.3.5 compare()

```
int compare (
    const void * a,
    const void * b )
```

4.1.3.6 Função: compare

Função de comparação para ordenação de vértices pela distância.

Parameters

<i>a</i>	Um apontador para o primeiro elemento a ser comparado.
<i>b</i>	Um apontador para o segundo elemento a ser comparado.

Returns

O resultado da comparação.

A função compara duas estruturas [VertexDist](#) pela distância, para ser usada na ordenação em ordem decrescente.

4.1.3.7 createAdjacent()

```
Adjacent * createAdjacent (
    int id,
    float dist )
```

4.1.3.8 Função: createAdjacent

Cria e inicializa um novo vértice adjacente.

Parameters

<i>id</i>	O id do vértice adjacente.
<i>dist</i>	A distância até ao vértice adjacente.

Returns

Um apontador vertex para a estrutura de [Adjacent](#) criada.

A função aloca memória para um vértice adjacente, inicializa os campos e retorna o apontador para a estrutura.

4.1.3.9 createGraph()

```
Graph * createGraph ( )
```

4.1.3.10 Função: createGraph

Cria e inicializa um novo grafo.

Returns

Um apontador para a estrutura [Graph](#) criada.

A função aloca memória para uma estrutura [Graph](#), inicializa os apontadores e os contadores e retorna o apontador para a estrutura o local onde está o grafo.

4.1.3.11 createVertex()

```
Vertex * createVertex (
    int id )
```

4.1.3.12 Função: createVertex

Cria e inicializa um novo vértice.

Parameters

<i>id</i>	O id do vértice.
-----------	------------------

Returns

Um apontador para a estrutura [Vertex](#) recém-criada.

A função aloca memória para um vértice, inicializa os campos e retorna o apontador para a estrutura.

4.1.3.13 DFS()

```
int DFS (  
    Graph * graph,  
    int startVertexId,  
    int destinationId )
```

4.1.3.14 Função: DFS

Faz a busca em profundidade (DFS) no grafo.

Parameters

<i>graph</i>	Um apontador para a estrutura Graph .
<i>start↔ VertexId</i>	O id do vértice inicial.
<i>destination↔ Id</i>	O id do vértice de destino.

Returns

0 se não existirem erros, se existirem retorna 1.

A função inicializa a DFS a partir de um vértice inicial e chama a função utilitária DFSUtil para fazer a busca e encontrar o caminho até ao destino.

4.1.3.15 DFSUtil()

```
int DFSUtil (  
    Graph * graph,  
    Vertex * vertex,  
    int path[],  
    int pathLen,  
    float currentDistance,  
    int destinationId )
```

4.1.3.16 Função: DFSUtil

Função utilitária/adjunta para a busca em profundidade (DFS).

Parameters

<i>graph</i>	Um apontador para a estrutura Graph .
--------------	---

Parameters

<i>vertex</i>	Um apontador para o vértice inicial.
<i>path</i>	Um array para armazenar o caminho atual.
<i>pathLen</i>	O comprimento atual do caminho.
<i>currentDistance</i>	A distância atual percorrida.
<i>destinationId</i>	O id do vértice de destino.

Returns

0 se não existirem erros, se existirem retorna 1

A função faz a busca em profundidade a partir de um vértice, marcando os vértices visitados e registra o caminho e a distância percorrida até ao destino.

4.1.3.17 findLongestPath()

```
int findLongestPath (
    Graph * graph,
    int startVertexId )
```

4.1.3.18 Função: findLongestPath

Encontra o caminho mais longo a partir de um vértice inicial.

Parameters

<i>graph</i>	Um apontador para a estrutura Graph .
<i>start↔ VertexId</i>	O id do vértice inicial.

Returns

A distância do caminho mais longo.

A função usa uma forma adaptada do algoritmo de Dijkstra para encontrar o caminho mais longo a partir de um vértice inicial, armazena e exibe o caminho e a distância.

4.1.3.19 findVertexById()

```
Vertex * findVertexById (
    Graph * graph,
    int id )
```

4.1.3.20 Função: findVertexById

Encontra um vértice no grafo pelo seu id.

Parameters

<i>graph</i>	Um apontador para a estrutura Graph .
<i>id</i>	O id do vértice.

Returns

Um apontador para o vertice cujo o id é igual ao id passado por parâmetro, senão encontrar retorna NULL.

A função percorre a lista de vértices do grafo e retorna o vértice que possui o id especificado.

4.1.3.21 readGraphFromFile()

```
Graph * readGraphFromFile (
    const char * filename )
```

4.1.3.22 Função: readGraphFromFile

Lê uma matriz do ficheiro distancias.txt e coloca-o na memória.

Parameters

<i>filename</i>	O nome do ficheiro de onde o grafo é lido.
-----------------	--

Returns

Um apontador para a estrutura [Graph](#) com os dados lidos do ficheiro.

A função abre o ficheiro especificado, lê a quantidade de vértices e as suas adjacências, cria as estruturas correspondentes e insere-as no grafo.

4.1.3.23 removeAdjacent()

```
bool removeAdjacent (
    Graph * graph,
    int vertexId,
    int adjId )
```

4.1.3.24 Função: removeAdjacent

Remove um vértice adjacente de um vértice especificado pelo o utilizador no grafo.

Parameters

<i>graph</i>	Um apontador para a estrutura Graph .
<i>vertexId</i>	O id do vértice principal.
<i>adjId</i>	O id do vértice adjacente a ser removido.

Returns

true se o vértice adjacente foi removido com sucesso, false caso contrário.

A função procura o vértice cujo id passado por parâmetro e, se encontrado, percorre a lista de adjacentes para encontrar e remover o vértice adjacente inserido pelo o utilizador.

4.1.3.25 removeVertex()

```
bool removeVertex (
    Graph * graph,
    int vertexId )
```

4.1.3.26 Função: removeVertex

Remove um vértice do grafo.

Parameters

<i>graph</i>	Um apontador para a estrutura Graph .
<i>vertexId</i>	O id do vértice a ser removido.

Returns

true se o vértice foi removido com sucesso, false caso contrário.

A função procura o vértice cujo id passado por parâmetro e, se encontrado, remove todas as suas adjacências.

4.1.3.27 showGraph()

```
void showGraph (
    Graph * graph )
```

4.1.3.28 Função: showGraph

Exibe os detalhes do grafo.

Parameters

<i>graph</i>	Um apontador para a estrutura Graph .
--------------	---

A função percorre a lista de vértices e adjacências do grafo e mostra o numero de vértices, os vértices e as suas respetivas adjacências presentes no grafo

4.1.3.29 updateAdjacentDistance()

```
bool updateAdjacentDistance (
    Graph * graph,
    int vertexId,
    int adjId,
    float newDist )
```

4.1.3.30 Função: updateAdjacentDistance

Atualiza a distância de um vértice adjacente.

Parameters

<i>graph</i>	Um apontador para a estrutura Graph .
<i>vertex↔ Id</i>	O id do vértice principal.
<i>adjId</i>	O id do vértice adjacente.
<i>newDist</i>	A nova distância para o vértice adjacente.

Returns

true se a distância foi atualizada com sucesso, false caso contrário.

A função procura o vértice com o id passado por parâmetro e a adjacencia igual, e se encontrados, atualiza a distância do vértice adjacente.

4.2 Header.h File Reference

Livraria que contém definições de tipos e estruturas para o sistema de grafos.

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

Data Structures

- struct [Adjacent](#)
Estrutura para representar um vértice adjacente.
- struct [Vertex](#)
Estrutura para representar um vértice.
- struct [Graph](#)
Estrutura para representar um grafo.
- struct [VertexDist](#)
Estrutura para representar a distância de um vértice.

Macros

- `#define MAX_VERTICES 100`
- `#define MAX_PATH_LENGTH 100`

Typedefs

- `typedef struct Adjacent Adjacent`
- `typedef struct Vertex Vertex`
- `typedef struct Graph Graph`

Functions

- `Graph * createGraph ()`
- `Vertex * createVertex (int id)`
- `Adjacent * createAdjacent (int id, float dist)`
- `Graph * addVertex (Graph *graph, int id)`
- `Adjacent * addAdjacent (Vertex *vertex, int id, float dist)`
- `Graph * readGraphFromFile (const char *filename)`
- `bool removeAdjacent (Graph *graph, int vertexId, int adjId)`
- `bool removeVertex (Graph *graph, int vertexId)`
- `bool updateAdjacentDistance (Graph *graph, int vertexId, int adjId, float newDist)`
- `Vertex * findVertexById (Graph *graph, int id)`
- `int DFSUtil (Graph *graph, Vertex *vertex, int path[], int pathLen, float currentDistance, int destinationId)`
- `int DFS (Graph *graph, int startVertexId, int destinationId)`
- `int compare (const void *a, const void *b)`
- `int findLongestPath (Graph *graph, int startVertexId)`
- `void showGraph (Graph *graph)`

4.2.1 Detailed Description

Livraria que contém definições de tipos e estruturas para o sistema de grafos.

Author

Alexandre Vilaça 26590

Date

25.05.2024

Este ficheiro é uma livraria que possui as estruturas utilizadas em código na implementação do grafo, este ficheiro também ajuda na reutilização do código.

4.2.2 Macro Definition Documentation

4.2.2.1 MAX_PATH_LENGTH

```
#define MAX_PATH_LENGTH 100
```


4.2.2.2 MAX_VERTICES

```
#define MAX_VERTICES 100
```

4.2.3 Typedef Documentation

4.2.3.1 Adjacent

```
typedef struct Adjacent Adjacent
```

4.2.3.2 Graph

```
typedef struct Graph Graph
```

4.2.3.3 Vertex

```
typedef struct Vertex Vertex
```

4.2.4 Function Documentation

4.2.4.1 addAdjacent()

```
Adjacent * addAdjacent (
    Vertex * vertex,
    int id,
    float dist )
```

4.2.4.2 Função: addAdjacent

Adiciona um novo vértice adjacente a um vértice já existente.

Parameters

<i>vertex</i>	Um apontador para a estrutura Vertex .
<i>id</i>	O id do vértice adjacente.
<i>dist</i>	A distância até ao vértice adjacente.

Returns

Um apontador para a estrutura [Adjacent](#) após a adição da adjacência

A função cria um novo vértice adjacente e insere-o no início da lista de adjacentes do vértice conforme o id passado por parâmetro.

4.2.4.3 addVertex()

```
Graph * addVertex (
    Graph * graph,
    int id )
```

4.2.4.4 Função: addVertex

Adiciona um novo vértice ao grafo.

Parameters

<i>graph</i>	Um apontador para a estrutura Graph .
<i>id</i>	O identificador do vértice a ser adicionado.

Returns

Um apontador para a estrutura [Graph](#) após a adição.

A função cria um novo vértice e insere-o no início da lista de vértices do grafo Incrementa os contadores de vértices.

4.2.4.5 compare()

```
int compare (
    const void * a,
    const void * b )
```

4.2.4.6 Função: compare

Função de comparação para ordenação de vértices pela distância.

Parameters

<i>a</i>	Um apontador para o primeiro elemento a ser comparado.
<i>b</i>	Um apontador para o segundo elemento a ser comparado.

Returns

O resultado da comparação.

A função compara duas estruturas [VertexDist](#) pela distância, para ser usada na ordenação em ordem decrescente.

4.2.4.7 createAdjacent()

```
Adjacent * createAdjacent (
    int id,
    float dist )
```

4.2.4.8 Função: createAdjacent

Cria e inicializa um novo vértice adjacente.

Parameters

<i>id</i>	O id do vértice adjacente.
<i>dist</i>	A distância até ao vértice adjacente.

Returns

Um apontador vertex para a estrutura de [Adjacent](#) criada.

A função aloca memória para um vértice adjacente, inicializa os campos e retorna o apontador para a estrutura.

4.2.4.9 createGraph()

```
Graph * createGraph ( )
```

4.2.4.10 Função: createGraph

Cria e inicializa um novo grafo.

Returns

Um apontador para a estrutura [Graph](#) criada.

A função aloca memória para uma estrutura [Graph](#), inicializa os apontadores e os contadores e retorna o apontador para a estrutura o local onde está o grafo.

4.2.4.11 createVertex()

```
Vertex * createVertex (
    int id )
```

4.2.4.12 Função: createVertex

Cria e inicializa um novo vértice.

Parameters

<i>id</i>	O id do vértice.
-----------	------------------

Returns

Um apontador para a estrutura [Vertex](#) recém-criada.

A função aloca memória para um vértice, inicializa os campos e retorna o apontador para a estrutura.

4.2.4.13 DFS()

```
int DFS (
    Graph * graph,
    int startVertexId,
    int destinationId )
```

4.2.4.14 Função: DFS

Faz a busca em profundidade (DFS) no grafo.

Parameters

<i>graph</i>	Um apontador para a estrutura Graph .
<i>start↔ VertexId</i>	O id do vértice inicial.
<i>destination↔ Id</i>	O id do vértice de destino.

Returns

0 se não existirem erros, se existirem retorna 1.

A função inicializa a DFS a partir de um vértice inicial e chama a função utilitária DFSUtil para fazer a busca e encontrar o caminho até ao destino.

4.2.4.15 DFSUtil()

```
int DFSUtil (
    Graph * graph,
    Vertex * vertex,
    int path[],
    int pathLen,
    float currentDistance,
    int destinationId )
```

4.2.4.16 Função: DFSUtil

Função utilitária/adjunta para a busca em profundidade (DFS).

Parameters

<i>graph</i>	Um apontador para a estrutura Graph .
--------------	---

Parameters

<i>vertex</i>	Um apontador para o vértice inicial.
<i>path</i>	Um array para armazenar o caminho atual.
<i>pathLen</i>	O comprimento atual do caminho.
<i>currentDistance</i>	A distância atual percorrida.
<i>destinationId</i>	O id do vértice de destino.

Returns

0 se não existirem erros, se existirem retorna 1

A função faz a busca em profundidade a partir de um vértice, marcando os vértices visitados e resgista o caminho e a distância percorrida até ao destino.

4.2.4.17 findLongestPath()

```
int findLongestPath (
    Graph * graph,
    int startVertexId )
```

4.2.4.18 Função: findLongestPath

Encontra o caminho mais longo a partir de um vértice inicial.

Parameters

<i>graph</i>	Um apontador para a estrutura Graph .
<i>start↔ VertexId</i>	O id do vértice inicial.

Returns

A distância do caminho mais longo.

A função usa uma forma adaptada do algoritmo de Dijkstra para encontrar o caminho mais longo a partir de um vértice inicial, armazena e exibe o caminho e a distância.

4.2.4.19 findVertexById()

```
Vertex * findVertexById (
    Graph * graph,
    int id )
```

4.2.4.20 Função: findVertexById

Encontra um vértice no grafo pelo seu id.

Parameters

<i>graph</i>	Um apontador para a estrutura Graph .
<i>id</i>	O id do vértice.

Returns

Um apontador para o vertice cujo o id é igual ao id passado por parâmetro, senão encontrar retorna NULL.

A função percorre a lista de vértices do grafo e retorna o vértice que possui o id especificado.

4.2.4.21 readGraphFromFile()

```
Graph * readGraphFromFile (
    const char * filename )
```

4.2.4.22 Função: readGraphFromFile

Lê uma matriz do ficheiro distancias.txt e coloca-o na memória.

Parameters

<i>filename</i>	O nome do ficheiro de onde o grafo é lido.
-----------------	--

Returns

Um apontador para a estrutura [Graph](#) com os dados lidos do ficheiro.

A função abre o ficheiro especificado, lê a quantidade de vértices e as suas adjacências, cria as estruturas correspondentes e insere-as no grafo.

4.2.4.23 removeAdjacent()

```
bool removeAdjacent (
    Graph * graph,
    int vertexId,
    int adjId )
```

4.2.4.24 Função: removeAdjacent

Remove um vértice adjacente de um vértice especificado pelo o utilizador no grafo.

Parameters

<i>graph</i>	Um apontador para a estrutura Graph .
<i>vertexId</i>	O id do vértice principal.
<i>adjId</i>	O id do vértice adjacente a ser removido.

Returns

true se o vértice adjacente foi removido com sucesso, false caso contrário.

A função procura o vértice cujo id passado por parâmetro e, se encontrado, percorre a lista de adjacentes para encontrar e remover o vértice adjacente inserido pelo o utilizador.

4.2.4.25 removeVertex()

```
bool removeVertex (
    Graph * graph,
    int vertexId )
```

4.2.4.26 Função: removeVertex

Remove um vértice do grafo.

Parameters

<i>graph</i>	Um apontador para a estrutura Graph .
<i>vertexId</i>	O id do vértice a ser removido.

Returns

true se o vértice foi removido com sucesso, false caso contrário.

A função procura o vértice cujo id passado por parâmetro e, se encontrado, remove todas as suas adjacências.

4.2.4.27 showGraph()

```
void showGraph (
    Graph * graph )
```

4.2.4.28 Função: showGraph

Exibe os detalhes do grafo.

Parameters

<i>graph</i>	Um apontador para a estrutura Graph .
--------------	---

A função percorre a lista de vértices e adjacências do grafo e mostra o numero de vértices, os vértices e as suas respetivas adjacências presentes no grafo

4.2.4.29 updateAdjacentDistance()

```
bool updateAdjacentDistance (
    Graph * graph,
    int vertexId,
    int adjId,
    float newDist )
```

4.2.4.30 Função: updateAdjacentDistance

Atualiza a distância de um vértice adjacente.

Parameters

<i>graph</i>	Um apontador para a estrutura Graph .
<i>vertex↔ Id</i>	O id do vértice principal.
<i>adjId</i>	O id do vértice adjacente.
<i>newDist</i>	A nova distância para o vértice adjacente.

Returns

true se a distância foi atualizada com sucesso, false caso contrário.

A função procura o vértice com o id passado por parâmetro e a adjacência igual, e se encontrados, atualiza a distância do vértice adjacente.

4.3 Header.h

[Go to the documentation of this file.](#)

```
00001 #include <stdio.h>
00002 #include <stdlib.h>
00003 #include <stdbool.h>
00004 #define MAX_VERTICES 100
00005 #define MAX_PATH_LENGTH 100
00022 typedef struct Adjacent {
00023     int id;
00024     float dist;
00025     bool visited;
00026     struct Adjacent *next;
00027 }Adjacent;
00035 typedef struct Vertex {
00036     int id;
00037     bool visited;
00038     Adjacent *nextAdjacent;
00039     struct Vertex *nextVertex;
00040 } Vertex;
00041
00048 typedef struct Graph {
00049     Vertex* inicioGraph;
00050     int numVertices;
00051     int totVertices;
00052 }Graph;
00060 typedef struct {
00061     int id;
00062     float dist;
00063 } VertexDist;
00064
00075 Graph* createGraph();
00087 Vertex* createVertex(int id);
00100 Adjacent* createAdjacent(int id, float dist);
```



```
00113 Graph* addVertex(Graph *graph, int id);
00127 Adjacent* addAdjacent(Vertex *vertex, int id, float dist);
00139 Graph* readGraphFromFile(const char *filename);
00153 bool removeAdjacent(Graph *graph, int vertexId, int adjId);
00166 bool removeVertex(Graph *graph, int vertexId);
00181 bool updateAdjacentDistance(Graph *graph, int vertexId, int adjId, float newDist);
00193 Vertex *findVertexById(Graph *graph, int id);
00210 int DFSUtil(Graph *graph, Vertex *vertex, int path[], int pathLen, float currentDistance, int
    destinationId);
00224 int DFS(Graph *graph, int startVertexId, int destinationId);
00237 int compare(const void *a, const void *b);
00250 int findLongestPath(Graph *graph, int startVertexId);
00260 void showGraph(Graph *graph);
```

4.4 Main.c File Reference

```
#include "Header.h"
#include <stdio.h>
#include <stdbool.h>
```

Functions

- int `main` ()
Função principal que gerencia a execução do programa.

4.4.1 Function Documentation

4.4.1.1 `main()`

```
int main ( )
```

Função principal que gerencia a execução do programa.

Author

Alexandre Vilaça 26590

Date

25.05.2024

Returns

Retorna 0 se a execução foi concluída com sucesso, caso contrário, retorna 1.

Index

- addAdjacent
 - Functions.c, [10](#)
 - Header.h, [19](#)
- addVertex
 - Functions.c, [10](#)
 - Header.h, [19](#)
- Adjacent, [5](#)
 - dist, [5](#)
 - Header.h, [19](#)
 - id, [5](#)
 - next, [5](#)
 - visited, [6](#)
- compare
 - Functions.c, [11](#)
 - Header.h, [20](#)
- createAdjacent
 - Functions.c, [11](#)
 - Header.h, [20](#)
- createGraph
 - Functions.c, [12](#)
 - Header.h, [21](#)
- createVertex
 - Functions.c, [12](#)
 - Header.h, [21](#)
- DFS
 - Functions.c, [13](#)
 - Header.h, [22](#)
- DFSUtil
 - Functions.c, [13](#)
 - Header.h, [22](#)
- dist
 - Adjacent, [5](#)
 - VertexDist, [8](#)
- findLongestPath
 - Functions.c, [14](#)
 - Header.h, [23](#)
- findVertexById
 - Functions.c, [14](#)
 - Header.h, [23](#)
- Functions.c, [9](#)
 - addAdjacent, [10](#)
 - addVertex, [10](#)
 - compare, [11](#)
 - createAdjacent, [11](#)
 - createGraph, [12](#)
 - createVertex, [12](#)
 - DFS, [13](#)
- DFSUtil, [13](#)
- findLongestPath, [14](#)
- findVertexById, [14](#)
- INF, [10](#)
- readGraphFromFile, [15](#)
- removeAdjacent, [15](#)
- removeVertex, [16](#)
- showGraph, [16](#)
- updateAdjacentDistance, [16](#)

- Graph, [6](#)
 - Header.h, [19](#)
 - inicioGraph, [6](#)
 - numVertices, [6](#)
 - totVertices, [6](#)
- Header.h, [17, 26](#)
 - addAdjacent, [19](#)
 - addVertex, [19](#)
 - Adjacent, [19](#)
 - compare, [20](#)
 - createAdjacent, [20](#)
 - createGraph, [21](#)
 - createVertex, [21](#)
 - DFS, [22](#)
 - DFSUtil, [22](#)
 - findLongestPath, [23](#)
 - findVertexById, [23](#)
 - Graph, [19](#)
 - MAX_PATH_LENGTH, [18](#)
 - MAX_VERTICES, [18](#)
 - readGraphFromFile, [24](#)
 - removeAdjacent, [24](#)
 - removeVertex, [25](#)
 - showGraph, [25](#)
 - updateAdjacentDistance, [25](#)
 - Vertex, [19](#)
- id
 - Adjacent, [5](#)
 - Vertex, [7](#)
 - VertexDist, [8](#)
- INF
 - Functions.c, [10](#)
- inicioGraph
 - Graph, [6](#)
- main
 - Main.c, [27](#)
- Main.c, [27](#)

- main, [27](#)
- MAX_PATH_LENGTH
 - Header.h, [18](#)
- MAX_VERTICES
 - Header.h, [18](#)
- next
 - Adjacent, [5](#)
- nextAdjacent
 - Vertex, [7](#)
- nextVertex
 - Vertex, [7](#)
- numVertices
 - Graph, [6](#)
- readGraphFromFile
 - Functions.c, [15](#)
 - Header.h, [24](#)
- removeAdjacent
 - Functions.c, [15](#)
 - Header.h, [24](#)
- removeVertex
 - Functions.c, [16](#)
 - Header.h, [25](#)
- showGraph
 - Functions.c, [16](#)
 - Header.h, [25](#)
- totVertices
 - Graph, [6](#)
- updateAdjacentDistance
 - Functions.c, [16](#)
 - Header.h, [25](#)
- Vertex, [7](#)
 - Header.h, [19](#)
 - id, [7](#)
 - nextAdjacent, [7](#)
 - nextVertex, [7](#)
 - visited, [7](#)
- VertexDist, [8](#)
 - dist, [8](#)
 - id, [8](#)
- visited
 - Adjacent, [6](#)
 - Vertex, [7](#)