My Project

Generated by Doxygen 1.9.8

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 Adjacencia Struct Reference

`#include <Grafos.h>`

Public Attributes

- int codVert
- int peso
- bool visitado
- struct Adjacencia * next

3.1.1 Member Data Documentation

3.1.1.1 codVert

`int Adjacencia::codVert`

3.1.1.2 next

`struct Adjacencia * Adjacencia::next`

3.1.1.3 peso

`int Adjacencia::peso`

3.1.1.4 visitado

`bool Adjacencia::visitado`

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

- C:/Aulas/1º ano/2º semestre/Estruturas de Dados Avançadas/Trabalho EDA Fase 2/Src/FuncoesLIB/Grafos.h
- Grafos.h

3.2 Grafo Struct Reference

`#include <Grafos.h>`

Public Attributes

- Vertice * inicio
- int numVert

3.2.1 Member Data Documentation

3.2.1.1 inicio

`Vertice * Grafo::inicio`

3.2.1.2 numVert

`int Grafo::numVert`

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

- C:/Aulas/1º ano/2º semestre/Estruturas de Dados Avançadas/Trabalho EDA Fase 2/Src/FuncoesLIB/Grafos.h
- Grafos.h

3.3 Stack Struct Reference

`#include <Stack.h>`

Public Attributes

- int id
- struct Stack * next

3.3.1 Member Data Documentation

3.3.1.1 id

`int Stack::id`

3.3.1.2 next

```
struct Stack * Stack::next
```

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

- C:/Aulas/1º ano/2º semestre/Estruturas de Dados Avançadas/Trabalho EDA Fase 2/Src/FuncoesLIB/Stack.h
- Stack.h

3.4 Vertice Struct Reference

```
#include <Grafos.h>
```

Public Attributes

- int cod
- int numAdj
- bool visitado
- Adjacencia * adjacencias
- struct Vertice * next

3.4.1 Member Data Documentation

3.4.1.1 adjacencias

```
Adjacencia * Vertice::adjacencias
```

3.4.1.2 cod

```
int Vertice::cod
```

3.4.1.3 next

```
struct Vertice * Vertice::next
```

3.4.1.4 numAdj

```
int Vertice::numAdj
```

3.4.1.5 visitado

```
bool Vertice::visitado
```

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

- C:/Aulas/1º ano/2º semestre/Estruturas de Dados Avançadas/Trabalho EDA Fase 2/Src/FuncoesLIB/Grafos.h
- Grafos.h

Chapter 4

File Documentation

4.1 C:/Aulas/1º ano/2º semestre/Estruturas de Dados Avançadas/Trabalho EDA Fase 2/Src/FuncoesLIB/Constantes.h File Reference

Ficheiro .h com as contantes utilizadas no trabalho.

Macros

- #define MAX_CARATERES 256
- #define MAX 256
- #define ERRO_AO_ABRIR_FICHEIRO -1
- #define PARAMETROS_INVALIDOS -2
- #define GRAFO_NAO_EXISTE -3
- #define VERTICE_JA_EXISTE -4
- #define VERTICE_NAO_EXISTE -5
- #define LISTA_NAO_EXISTE -6
- #define EXISTE_ADJACENCIA -7
- #define NAO_EXISTE_ADJACENCIA -8
- #define SUCESSO 1

4.1.1 Detailed Description

Ficheiro .h com as contantes utilizadas no trabalho.

~

Author

Diogo Machado 26042

Date

20.05.2024

Copyright

4.1.2 Macro Definition Documentation

4.1.2.1 ERRO_AO_ABRIR_FICHEIRO

```
#define ERRO_AO_ABRIR_FICHEIRO -1
```

4.1.2.2 EXISTE_ADJACENCIA

```
#define EXISTE_ADJACENCIA -7
```

4.1.2.3 GRAFO_NAO_EXISTE

```
#define GRAFO_NAO_EXISTE -3
```

4.1.2.4 LISTA_NAO_EXISTE

```
#define LISTA_NAO_EXISTE -6
```

4.1.2.5 MAX

```
#define MAX 256
```

4.1.2.6 MAX_CARATERES

```
#define MAX_CARATERES 256
```

4.1.2.7 NAO_EXISTE_ADJACENCIA

```
#define NAO_EXISTE_ADJACENCIA -8
```

4.1.2.8 PARAMETROS_INVALIDOS

```
#define PARAMETROS_INVALIDOS -2
```

4.1.2.9 SUCESSO

```
#define SUCESSO 1
```

4.1.2.10 VERTICE_JA_EXISTE

```
#define VERTICE_JA_EXISTE -4
```

4.1.2.11 VERTICE_NAO_EXISTE

```
#define VERTICE_NAO_EXISTE -5
```

4.2 Constantes.h

Go to the documentation of this file.

```
00001
00011 #pragma once
00012
00013 #pragma region CONSTANTES
00014 #define MAX_CARATERES 256
00015 #define MAX 256
00016 #pragma endregion
00017
00018 #pragma region ERROS
00019 #define ERRO_AO_ABRIR_FICHEIRO -1
00020 #define PARAMETROS_INVALIDOS -2
00021 #define GRAFO_NAO_EXISTE -3
00022 #define VERTICE_JA_EXISTE -4
00023 #define VERTICE_NAO_EXISTE -5
00024 #define LISTA_NAO_EXISTE -6
00025 #define EXISTE_ADJACENCIA -7
00026 #define NAO_EXISTE_ADJACENCIA -8
00027 #define SUCESSO 1
00028 #pragma endregion
```

4.3 Constantes.h File Reference

Ficheiro .h com as contantes utilizadas no trabalho.

Macros

- #define MAX_CARATERES 256
- #define MAX 256
- #define ERRO_AO_ABRIR_FICHEIRO -1
- #define PARAMETROS_INVALIDOS -2
- #define GRAFO_NAO_EXISTE -3
- #define VERTICE_JA_EXISTE -4
- #define VERTICE_NAO_EXISTE -5
- #define LISTA_NAO_EXISTE -6
- #define EXISTE_ADJACENCIA -7
- #define NAO_EXISTE_ADJACENCIA -8
- #define SUCESSO 1

4.3.1 Detailed Description

Ficheiro .h com as contantes utilizadas no trabalho.

~

Author

Diogo Machado 26042

Date

20.05.2024

Copyright

4.3.2 Macro Definition Documentation

4.3.2.1 ERRO_AO_ABRIR_FICHEIRO

```
#define ERRO_AO_ABRIR_FICHEIRO -1
```

4.3.2.2 EXISTE_ADJACENCIA

```
#define EXISTE_ADJACENCIA -7
```

4.3.2.3 GRAFO_NAO_EXISTE

```
#define GRAFO_NAO_EXISTE -3
```

4.3.2.4 LISTA_NAO_EXISTE

```
#define LISTA_NAO_EXISTE -6
```

4.3.2.5 MAX

```
#define MAX 256
```

4.3.2.6 MAX_CARATERES

```
#define MAX_CARATERES 256
```

4.3.2.7 NAO_EXISTE_ADJACENCIA

```
#define NAO_EXISTE_ADJACENCIA -8
```

4.3.2.8 PARAMETROS_INVALIDOS

```
#define PARAMETROS_INVALIDOS -2
```

4.3.2.9 SUCESSO

```
#define SUCESSO 1
```

4.3.2.10 VERTICE_JA_EXISTE

```
#define VERTICE_JA_EXISTE -4
```

4.3.2.11 VERTICE_NAO_EXISTE

```
#define VERTICE_NAO_EXISTE -5
```

4.4 Constantes.h

Go to the documentation of this file.

```
00001
00011 #pragma once
00012
00013 #pragma region CONSTANTES
00014 #define MAX_CARATERES 256
00015 #define MAX 256
00016 #pragma endregion
00017
00018 #pragma region ERROS
00019 #define ERRO_AO_ABRIR_FICHEIRO -1
00020 #define PARAMETROS_INVALIDOS -2
00021 #define GRAFO_NAO_EXISTE -3
00022 #define VERTICE_JA_EXISTE -4
00023 #define VERTICE_NAO_EXISTE -5
00024 #define LISTA_NAO_EXISTE -6
00025 #define EXISTE_ADJACENCIA -7
00026 #define NAO_EXISTE_ADJACENCIA -8
00027 #define SUCESSO 1
00028 #pragma endregion
```

4.5 C:/Aulas/1º ano/2º semestre/Estruturas de Dados Avançadas/Trabalho EDA Fase 2/Src/FuncoesLIB/FuncoesProcura.c File Reference

```
#include "Grafos.h"
#include "Stack.h"
#include "Utilidades.h"
#include "Constantes.h"
#include <stdio.h>
```

Functions

- Stack * DepthFirstTraversal (Grafo *grafo, int codIni)

  Funcao Depth First Traversal que passa por todos os vertices do grafo em profundidade.
- void ProcessaVertice (int id)

  Funcao que imprime no ecra o vertice da Depth First search recursiva.
- bool DepthFirstSearchRec (Vertice *auxVert, int origem, int dest)

  Funcao Depth First Traversal que passa por todos os vertices do grafo em profundidade.
- int CalculoSomaEntreDoisVertices (Grafo *grafo, int codIni, int codDest)

  Calcula a soma dos pesos das adjacencias entre dois vertices.

4.5.1 Function Documentation

4.5.1.1 CalculoSomaEntreDoisVertices()

```
int CalculoSomaEntreDoisVertices (
            Grafo * grafo,
            int codIni,
            int codDest )
```

Calcula a soma dos pesos das adjacencias entre dois vertices.

Parameters

grafo	- grafo guardado em memoria
codIni	- codigo do vertice onde se comeca a pesquisa
codDest	- codigo do vertice onde se pretende chegar

Return values

-	valor da soma dos pesos

4.5.1.2 DepthFirstSearchRec()

```
bool DepthFirstSearchRec (
            Vertice * auxVert,
            int origem,
            int dest )
```

Funcao Depth First Traversal que passa por todos os vertices do grafo em profundidade.

Parameters

auxVert	- lista de vertices do grafo
origem	- onde comeca a procura
dest	- destino da procura

Return values

-	retorna sucesso ou insucesso de encontrar o vertice

4.5.1.3 DepthFirstTraversal()

```
Stack * DepthFirstTraversal (
            Grafo * grafo,
            int codIni )
```

Funcao Depth First Traversal que passa por todos os vertices do grafo em profundidade.

Parameters

grafo	- grafo guardado em memoria
codIni	- codigo do vertice onde comeca a procura

Return values

-	Stack com os registos onde se passou para a procura

4.5.1.4 ProcessaVertice()

```
void ProcessaVertice (
            int id )
```

Funcao que imprime no ecra o vertice da Depth First search recursiva.

Parameters

id	- id do vertice que se pretende imprimir

4.6 C:/Aulas/1º ano/2º semestre/Estruturas de Dados Avançadas/Trabalho EDA Fase 2/Src/FuncoesLIB/FuncoesProcura.h File Reference

Ficheiro .h com as funcoes de procura do trabalho.

```
#include "Grafos.h"
#include "Stack.h"
```

Functions

- Stack * DepthFirstTraversal (Grafo *grafo, int codIni)

  Funcao Depth First Traversal que passa por todos os vertices do grafo em profundidade.
- void ProcessaVertice (int id)

  Funcao que imprime no ecra o vertice da Depth First search recursiva.
- bool DepthFirstSearchRec (Vertice *auxVert, int origem, int dest)

  Funcao Depth First Traversal que passa por todos os vertices do grafo em profundidade.
- int CalculoSomaEntreDoisVerticesz (Grafo *grafo, int codIni, int codDest)

  Calcula a soma dos pesos das adjacencias entre dois vertices.

4.6.1 Detailed Description

Ficheiro .h com as funcoes de procura do trabalho.

~

Author

Diogo Machado 26042

Date

22.05.2024

Copyright

4.6.2 Function Documentation

4.6.2.1 CalculoSomaEntreDoisVerticesz()

```
int CalculoSomaEntreDoisVerticesz (
            Grafo * grafo,
            int codIni,
            int codDest )
```

Calcula a soma dos pesos das adjacencias entre dois vertices.

Parameters

grafo	- grafo guardado em memoria
codIni	- codigo do vertice onde se comeca a pesquisa
codDest	- codigo do vertice onde se pretende chegar

Return values

-	valor da soma dos pesos

4.6.2.2 DepthFirstSearchRec()

```
bool DepthFirstSearchRec (
            Vertice * auxVert,
            int origem,
            int dest )
```

Funcao Depth First Traversal que passa por todos os vertices do grafo em profundidade.

Parameters

auxVert	- lista de vertices do grafo
origem	- onde comeca a procura
dest	- destino da procura

Return values

-	retorna sucesso ou insucesso de encontrar o vertice

4.6.2.3 DepthFirstTraversal()

```
Stack * DepthFirstTraversal (
            Grafo * grafo,
            int codIni )
```

Funcao Depth First Traversal que passa por todos os vertices do grafo em profundidade.

Parameters

grafo	- grafo guardado em memoria
codIni	- codigo do vertice onde comeca a procura

Return values

-	Stack com os registos onde se passou para a procura

4.6.2.4 ProcessaVertice()

```
void ProcessaVertice (
            int id )
```

Funcao que imprime no ecra o vertice da Depth First search recursiva.

Parameters

id	- id do vertice que se pretende imprimir

4.7 FuncoesProcura.h

Go to the documentation of this file.

```
00001
00011 #pragma once
00012 #include "Grafos.h"
00013 #include "Stack.h"
00014
00021 Stack* DepthFirstTraversal(Grafo* grafo, int codIni);
00026 void ProcessaVertice(int id);
00034 bool DepthFirstSearchRec(Vertice* auxVert, int origem, int dest);
00042 int CalculoSomaEntreDoisVerticesz(Grafo* grafo, int codIni, int codDest);
```

4.8 FuncoesProcura.h File Reference

Ficheiro .h com as funcoes de procura do trabalho.

```
#include "Grafos.h"
#include "Stack.h"
```

Functions

- Stack ∗ DepthFirstTraversal (Grafo ∗grafo, int codIni)

  Funcao Depth First Traversal que passa por todos os vertices do grafo em profundidade.
- void ProcessaVertice (int id)

  Funcao que imprime no ecra o vertice da Depth First search recursiva.
- bool DepthFirstSearchRec (Vertice ∗auxVert, int origem, int dest)

  Funcao Depth First Traversal que passa por todos os vertices do grafo em profundidade.
- int CalculoSomaEntreDoisVertices (Grafo ∗grafo, int codIni, int codDest)

  Calcula a soma dos pesos das adjacencias entre dois vertices.

4.8.1 Detailed Description

Ficheiro .h com as funcoes de procura do trabalho.

~

Author

Diogo Machado 26042

Date

22.05.2024

Copyright

4.8.2 Function Documentation

4.8.2.1 CalculoSomaEntreDoisVertices()

```
int CalculoSomaEntreDoisVertices (
            Grafo * grafo,
            int codIni,
            int codDest )
```

Calcula a soma dos pesos das adjacencias entre dois vertices.

Parameters

grafo	- grafo guardado em memoria
codIni	- codigo do vertice onde se comeca a pesquisa
codDest	- codigo do vertice onde se pretende chegar

Return values

-	valor da soma dos pesos

4.8.2.2 DepthFirstSearchRec()

```
bool DepthFirstSearchRec (
            Vertice * auxVert,
            int origem,
            int dest )
```

Funcao Depth First Traversal que passa por todos os vertices do grafo em profundidade.

Parameters

auxVert	- lista de vertices do grafo
origem	- onde comeca a procura
dest	- destino da procura

Return values

-	retorna sucesso ou insucesso de encontrar o vertice

4.8.2.3 DepthFirstTraversal()

```
Stack * DepthFirstTraversal (
            Grafo * grafo,
            int codIni )
```

Funcao Depth First Traversal que passa por todos os vertices do grafo em profundidade.

Parameters

grafo	- grafo guardado em memoria
codIni	- codigo do vertice onde comeca a procura

Return values

-	Stack com os registos onde se passou para a procura

4.8.2.4 ProcessaVertice()

```
void ProcessaVertice (
            int id )
```

Funcao que imprime no ecra o vertice da Depth First search recursiva.

Parameters

id	- id do vertice que se pretende imprimir

4.9 FuncoesProcura.h

Go to the documentation of this file.

```
00001
00011 #pragma once
00012 #include "Grafos.h"
00013 #include "Stack.h"
00014
```

```
00021 Stack* DepthFirstTraversal(Grafo* grafo, int codIni);
00026 void ProcessaVertice(int id);
00034 bool DepthFirstSearchRec(Vertice* auxVert, int origem, int dest);
00042 int CalculoSomaEntreDoisVertices(Grafo* grafo, int codIni, int codDest);
```

4.10 C:/Aulas/1º ano/2º semestre/Estruturas de Dados Avançadas/Trabalho EDA Fase 2/Src/FuncoesLIB/GereGrafos.c File Reference

Ficheiro .c que contuns de gerir grafos.

```
#include "Grafos.h"
#include "Constantes.h"
#include <stdio.h>
#include <stdbool.h>
#include <string.h>
#include <malloc.h>
```

Functions

- Vertice * CriaVertice (int cod)

  Cria espaco em memoria para um vertice e coloca valores nas suas variis.
- Vertice * InsereVertice (Vertice *inicio, Vertice *novo)

  Insere vertice novo na lista ligada dos vertices.
- Vertice * InsereAdjacenciaVertice (Vertice *vertice, Adjacencia *inicioAdj)

  Insere lista de adjacencias no vertice correspondente.
- Adjacencia * CriaAdjacencia (int cod, int peso)

  Cria espaco em memoria para uma adjacencia e coloca valores nas suas variaveis.
- Adjacencia * InsereAdjacencia (Adjacencia *inicio, Adjacencia *nova)

  Insere adjacencia nova na lista de adjacencias.
- Grafo * CriaGrafo (Vertice *inicioVert, int numVert)

  Cria memoria para o grafo e introduz os valores nas variaveis.
- int LimpaGrafo (Grafo *grafo)

  Limpa memoria.
- Grafo * LeFicheiroGrafo (char *filename)

  Le grafo do ficheiro CSV.
- int MostraGrafo (Grafo *grafo)

  Funcao que mostra grafo na linha de comandos.

4.10.1 Detailed Description

Ficheiro .c que contuns de gerir grafos.

~

Author

Diogo Machado 26042

Date

8.05.2024

Copyright

4.10.2 Function Documentation

4.10.2.1 CriaAdjacencia()

```
Adjacencia * CriaAdjacencia (
            int cod,
            int peso )
```

Cria espaco em memoria para uma adjacencia e coloca valores nas suas variaveis.

Parameters

cod	- numero de codigo do vertice adjacente
peso	- peso da adjacencia

Return values

-	adjacencia criada

4.10.2.2 CriaGrafo()

```
Grafo * CriaGrafo (
            Vertice * inicioVert,
            int numVert )
```

Cria memoria para o grafo e introduz os valores nas variaveis.

Parameters

inicioVert	- inicio da lista dos vertices
numVert	- numero de vertices do grafo

Return values

-	grafo com valores nas variaveis

4.10.2.3 CriaVertice()

```
Vertice * CriaVertice (
            int cod )
```

Cria espaco em memoria para um vertice e coloca valores nas suas variis.

Parameters

cod	- numero de codigo do vertice

Return values

-	vertice criado

4.10.2.4 InsereAdjacencia()

```
Adjacencia * InsereAdjacencia (
            Adjacencia * inicio,
            Adjacencia * nova )
```

Insere adjacencia nova na lista de adjacencias.

Parameters

inicio	- inicio da lista de adjacencias
nova	- nova adjacencia que se pretende colocar na lista

Return values

-	lista de adjacencias

4.10.2.5 InsereAdjacenciaVertice()

```
Vertice * InsereAdjacenciaVertice (
            Vertice * vertice,
            Adjacencia * inicioAdj )
```

Insere lista de adjacencias no vertice correspondente.

Parameters

vertice	- vertice do grafo que pretendemos colocar as adjacencias
inicioAdj	- lista de adjacentes do vertice

Return values

-	vertice com as adjacencias colocadas

4.10.2.6 InsereVertice()

```
Vertice * InsereVertice (
            Vertice * inicio,
            Vertice * novo )
```

Insere vertice novo na lista ligada dos vertices.

Parameters

inicio	- inicio da lista dos vertices
novo	- vertice novo que se pretende introduzir na lista

Return values

-	inicio da lista de vertices

4.10.2.7 LeFicheiroGrafo()

```
Grafo * LeFicheiroGrafo (
            char * filename )
```

Le grafo do ficheiro CSV.

Le ficheiro CSV com as informacoes do grafo.

Parameters

filename	- nome do ficheiro

Return values

-	grafo

4.10.2.8 LimpaGrafo()

```
int LimpaGrafo (
            Grafo * grafo )
```

Limpa memoria.

Parameters

grafo	- grafo guardado em memoria

Return values

-	Resultado da funcao

4.10.2.9 MostraGrafo()

```
int MostraGrafo (
            Grafo * grafo )
```

Funcao que mostra grafo na linha de comandos.

Parameters

grafo	- grafo guardado em memoria

Return values

-	Resultado da funcao

4.11 C:/Aulas/1º ano/2º semestre/Estruturas de Dados Avançadas/Trabalho EDA Fase 2/Src/FuncoesLIB/GereGrafos.h File Reference

Ficheiro .h que contem funcoes de gerir grafos.

```
#include "Grafos.h"
```

Functions

- Vertice * CriaVertice (int cod)

  Cria espaco em memoria para um vertice e coloca valores nas suas variis.
- Vertice * InsereVertice (Vertice *inicio, Vertice *novo)

  Insere vertice novo na lista ligada dos vertices.
- Adjacencia * CriaAdjacencia (int cod, int peso)

  Cria espaco em memoria para uma adjacencia e coloca valores nas suas variaveis.
- Adjacencia * InsereAdjacencia (Adjacencia *inicio, Adjacencia *nova)

  Insere adjacencia nova na lista de adjacencias.
- Vertice * InsereAdjacenciaVertice (Vertice *vertice, Adjacencia *inicioAdj)

  Insere lista de adjacencias no vertice correspondente.
- Grafo * CriaGrafo (Vertice *inicioVert, int numVert)

  Cria memoria para o grafo e introduz os valores nas variaveis.
- int LimpaGrafo (Grafo *grafo)

  Limpa memoria.
- Grafo * LeFicheiroGrafo (char *filename)

  Le ficheiro CSV com as informacoes do grafo.
- int MostraGrafo (Grafo *grafo)

  Funcao que mostra grafo na linha de comandos.

4.11.1 Detailed Description

Ficheiro .h que contem funcoes de gerir grafos.

~

Author

Diogo Machado 26042

Date

8.05.2024

Copyright

4.11.2 Function Documentation

4.11.2.1 CriaAdjacencia()

```
Adjacencia * CriaAdjacencia (
            int cod,
            int peso )
```

Cria espaco em memoria para uma adjacencia e coloca valores nas suas variaveis.

Parameters

cod	- numero de codigo do vertice adjacente
peso	- peso da adjacencia

Return values

-	adjacencia criada

4.11.2.2 CriaGrafo()

```
Grafo * CriaGrafo (
            Vertice * inicioVert,
            int numVert )
```

Cria memoria para o grafo e introduz os valores nas variaveis.

Parameters

inicioVert	- inicio da lista dos vertices
numVert	- numero de vertices do grafo

Return values

-	grafo com valores nas variaveis

4.11.2.3 CriaVertice()

```
Vertice * CriaVertice (
            int cod )
```

Cria espaco em memoria para um vertice e coloca valores nas suas variis.

Parameters

cod	- numero de codigo do vertice

Return values

-	vertice criado

4.11.2.4 InsereAdjacencia()

```
Adjacencia * InsereAdjacencia (
            Adjacencia * inicio,
            Adjacencia * nova )
```

Insere adjacencia nova na lista de adjacencias.

Parameters

inicio	- inicio da lista de adjacencias
nova	- nova adjacencia que se pretende colocar na lista

Return values

-	lista de adjacencias

4.11.2.5 InsereAdjacenciaVertice()

```
Vertice * InsereAdjacenciaVertice (
            Vertice * vertice,
            Adjacencia * inicioAdj )
```

Insere lista de adjacencias no vertice correspondente.

Parameters

vertice	- vertice do grafo que pretendemos colocar as adjacencias
inicioAdj	- lista de adjacentes do vertice

Return values

-	vertice com as adjacencias colocadas

4.11.2.6 InsereVertice()

```
Vertice * InsereVertice (
            Vertice * inicio,
            Vertice * novo )
```

Insere vertice novo na lista ligada dos vertices.

Parameters

inicio	- inicio da lista dos vertices
novo	- vertice novo que se pretende introduzir na lista

Return values

-	inicio da lista de vertices

4.11.2.7 LeFicheiroGrafo()

```
Grafo * LeFicheiroGrafo (
            char * filename )
```

Le ficheiro CSV com as informacoes do grafo.

Parameters

filename	- nome do ficheiro

Return values

-	grafo com as informacoes lidas do ficheiro

Le ficheiro CSV com as informacoes do grafo.

Parameters

filename	- nome do ficheiro

Return values

-	grafo

4.11.2.8 LimpaGrafo()

```
int LimpaGrafo (
            Grafo * grafo )
```

Limpa memoria.

Parameters

grafo	- grafo guardado em memoria

Return values

-	Resultado da funcao

4.11.2.9 MostraGrafo()

```
int MostraGrafo (
            Grafo * grafo )
```

Funcao que mostra grafo na linha de comandos.

Parameters

grafo	- grafo guardado em memoria

Return values

-	Resultado da funcao

4.12 GereGrafos.h

Go to the documentation of this file.

```
00001
00011 #pragma once
00012 #include "Grafos.h"
00013
00014 #pragma region VERTICES
00020 Vertice* CriaVertice(int cod);
00027 Vertice* InsereVertice(Vertice* inicio, Vertice* novo);
00028 #pragma endregion
00029
00030 #pragma region ADJACENCIAS
00037 Adjacencia* CriaAdjacencia(int cod, int peso);
00044 Adjacencia* InsereAdjacencia(Adjacencia* inicio, Adjacencia* nova);
00045 #pragma endregion
00046
00047 #pragma region GRAFOS
00054 Vertice* InsereAdjacenciaVertice(Vertice* vertice, Adjacencia* inicioAdj);
00061 Grafo* CriaGrafo(Vertice* inicioVert, int numVert);
00067 int LimpaGrafo(Grafo* grafo);
00073 Grafo* LeFicheiroGrafo(char* filename);
00079 int MostraGrafo(Grafo* grafo);
00080 #pragma endregion
```

4.13 GereGrafos.h File Reference

Ficheiro .h que contem funcoes de gerir grafos.

```
#include "Grafos.h"
```

Functions

- Vertice * CriaVertice (int cod)

  Cria espaco em memoria para um vertice e coloca valores nas suas variis.
- Vertice * InsereVertice (Vertice *inicio, Vertice *novo)

  Insere vertice novo na lista ligada dos vertices.
- Adjacencia * CriaAdjacencia (int cod, int peso)

  Cria espaco em memoria para uma adjacencia e coloca valores nas suas variaveis.
- Adjacencia * InsereAdjacencia (Adjacencia *inicio, Adjacencia *nova)

  Insere adjacencia nova na lista de adjacencias.
- Vertice * InsereAdjacenciaVertice (Vertice *vertice, Adjacencia *inicioAdj)

  Insere lista de adjacencias no vertice correspondente.
- Grafo * CriaGrafo (Vertice *inicioVert, int numVert)

  Cria memoria para o grafo e introduz os valores nas variaveis.
- int LimpaGrafo (Grafo *grafo)

  Limpa memoria.
- Grafo * LeFicheiroGrafo (char *filename)

  Le ficheiro CSV com as informacoes do grafo.
- int MostraGrafo (Grafo *grafo)

  Funcao que mostra grafo na linha de comandos.

4.13.1 Detailed Description

Ficheiro .h que contem funcoes de gerir grafos.

~

Author

Diogo Machado 26042

Date

8.05.2024

Copyright

4.13.2 Function Documentation

4.13.2.1 CriaAdjacencia()

```
Adjacencia * CriaAdjacencia (
            int cod,
            int peso )
```

Cria espaco em memoria para uma adjacencia e coloca valores nas suas variaveis.

Parameters

cod	- numero de codigo do vertice adjacente
peso	- peso da adjacencia

Return values

-	adjacencia criada

4.13.2.2 CriaGrafo()

```
Grafo * CriaGrafo (
            Vertice * inicioVert,
            int numVert )
```

Cria memoria para o grafo e introduz os valores nas variaveis.

Parameters

inicioVert	- inicio da lista dos vertices
numVert	- numero de vertices do grafo

Return values

-	grafo com valores nas variaveis

4.13.2.3 CriaVertice()

```
Vertice * CriaVertice (
            int cod )
```

Cria espaco em memoria para um vertice e coloca valores nas suas variis.

Parameters

cod	- numero de codigo do vertice

Return values

-	vertice criado

4.13.2.4 InsereAdjacencia()

```
Adjacencia * InsereAdjacencia (
            Adjacencia * inicio,
            Adjacencia * nova )
```

Insere adjacencia nova na lista de adjacencias.

Parameters

inicio	- inicio da lista de adjacencias
nova	- nova adjacencia que se pretende colocar na lista

Return values

-	lista de adjacencias

4.13.2.5 InsereAdjacenciaVertice()

```
Vertice * InsereAdjacenciaVertice (
            Vertice * vertice,
            Adjacencia * inicioAdj )
```

Insere lista de adjacencias no vertice correspondente.

Parameters

vertice	- vertice do grafo que pretendemos colocar as adjacencias
inicioAdj	- lista de adjacentes do vertice

Return values

-	vertice com as adjacencias colocadas

4.13.2.6 InsereVertice()

```
Vertice * InsereVertice (
            Vertice * inicio,
            Vertice * novo )
```

Insere vertice novo na lista ligada dos vertices.

Parameters

inicio	- inicio da lista dos vertices
novo	- vertice novo que se pretende introduzir na lista

Return values

-	inicio da lista de vertices

4.13.2.7 LeFicheiroGrafo()

```
Grafo * LeFicheiroGrafo (
            char * filename )
```

Le ficheiro CSV com as informacoes do grafo.

Parameters

filename	- nome do ficheiro

Return values

-	grafo com as informacoes lidas do ficheiro

Le ficheiro CSV com as informacoes do grafo.

Parameters

filename	- nome do ficheiro

Return values

-	grafo

4.13.2.8 LimpaGrafo()

```
int LimpaGrafo (
            Grafo * grafo )
```

Limpa memoria.

Parameters

grafo	- grafo guardado em memoria

Return values

-	Resultado da funcao

4.13.2.9 MostraGrafo()

```
int MostraGrafo (
            Grafo * grafo )
```

Funcao que mostra grafo na linha de comandos.

Parameters

grafo	- grafo guardado em memoria

Return values

-	Resultado da funcao

4.14 GereGrafos.h

Go to the documentation of this file.

```
00001
00011 #pragma once
00012 #include "Grafos.h"
00013
00014 #pragma region VERTICES
00020 Vertice* CriaVertice(int cod);
00027 Vertice* InsereVertice(Vertice* inicio, Vertice* novo);
00028 #pragma endregion
00029
00030 #pragma region ADJACENCIAS
00037 Adjacencia* CriaAdjacencia(int cod, int peso);
00044 Adjacencia* InsereAdjacencia(Adjacencia* inicio, Adjacencia* nova);
00045 #pragma endregion
00046
00047 #pragma region GRAFOS
00054 Vertice* InsereAdjacenciaVertice(Vertice* vertice, Adjacencia* inicioAdj);
00061 Grafo* CriaGrafo(Vertice* inicioVert, int numVert);
00067 int LimpaGrafo(Grafo* grafo);
00073 Grafo* LeFicheiroGrafo(char* filename);
00079 int MostraGrafo(Grafo* grafo);
00080 #pragma endregion
```

4.15 C:/Aulas/1º ano/2º semestre/Estruturas de Dados Avançadas/Trabalho EDA Fase 2/Src/FuncoesLIB/Grafos.h File Reference

Ficheiro .h com a estrutura de dados utilizada para o trabalho prco.

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

Classes

- struct Adjacencia
- struct Vertice
- struct Grafo

Typedefs

- typedef struct Adjacencia Adjacencia
- typedef struct Vertice Vertice
- typedef struct Grafo Grafo

4.15.1 Detailed Description

Ficheiro .h com a estrutura de dados utilizada para o trabalho prco.

~

Author

Diogo Machado 26042

Date

8.05.2024

Copyright

4.15.2 Typedef Documentation

4.15.2.1 Adjacencia

```
typedef struct Adjacencia Adjacencia
```

4.15.2.2 Grafo

```
typedef struct Grafo Grafo
```

4.15.2.3 Vertice

```
typedef struct Vertice Vertice
```

4.16 Grafos.h

Go to the documentation of this file.

```
00001
00011 #pragma once
00012 #include <stdbool.h>
00013
00014 #pragma region ESTRUTURA_GRAFO
00015 typedef struct Adjacencia {
00016     int codVert;                //Codigo do vertice adjacente
00017     int peso;                   //Peso da adjacencia
00018     bool visitado;
00019     struct Adjacencia* next;    //Apontador para outra adjacencia
00020 }Adjacencia;
00021
00022 typedef struct Vertice {
00023     int cod;                    //Codigo do vertice
00024     int numAdj;                 //Numero de adjacencias do vertice
00025     bool visitado;
00026     Adjacencia* adjacencias;    //Lista de adjacencias do vertice
00027     struct Vertice* next;       //Apontador para o vertice seguinte
00028 }Vertice;
00029
00030 typedef struct Grafo {
00031     Vertice* inicio;            //Inicio da lista dos vertices do grafo
00032     int numVert;                //Numero de vertices do grafo
00033 }Grafo;
00034 #pragma endregion
```

4.17 Grafos.h File Reference

Ficheiro .h com a estrutura de dados utilizada para o trabalho prco.

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

Classes

- struct Adjacencia
- struct Vertice
- struct Grafo

Typedefs

- typedef struct Adjacencia Adjacencia
- typedef struct Vertice Vertice
- typedef struct Grafo Grafo

4.17.1 Detailed Description

Ficheiro .h com a estrutura de dados utilizada para o trabalho prco.

~

Author

Diogo Machado 26042

Date

8.05.2024

Copyright

4.17.2 Typedef Documentation

4.17.2.1 Adjacencia

```
typedef struct Adjacencia Adjacencia
```

4.17.2.2 Grafo

```
typedef struct Grafo Grafo
```

4.17.2.3 Vertice

```
typedef struct Vertice Vertice
```

4.18 Grafos.h

Go to the documentation of this file.

```
00001
00011 #pragma once
00012 #include <stdbool.h>
00013
00014 #pragma region ESTRUTURA_GRAFO
00015 typedef struct Adjacencia {
00016     int codVert;                //Codigo do vertice adjacente
00017     int peso;                   //Peso da adjacencia
00018     bool visitado;
00019     struct Adjacencia* next;    //Apontador para outra adjacencia
00020 }Adjacencia;
00021
00022 typedef struct Vertice {
00023     int cod;                    //Codigo do vertice
00024     int numAdj;                 //Numero de adjacencias do vertice
00025     bool visitado;
00026     Adjacencia* adjacencias;    //Lista de adjacencias do vertice
00027     struct Vertice* next;       //Apontador para o vertice seguinte
00028 }Vertice;
00029
00030 typedef struct Grafo {
00031     Vertice* inicio;            //Inicio da lista dos vertices do grafo
00032     int numVert;                //Numero de vertices do grafo
00033 }Grafo;
00034 #pragma endregion
```

4.19 C:/Aulas/1º ano/2º semestre/Estruturas de Dados Avançadas/Trabalho EDA Fase 2/Src/FuncoesLIB/Stack.c File Reference

Ficheiro .c com as funcoes da stack utilizada nos algoritmos de procura.

```
#include "GereGrafos.h"
#include "Grafos.h"
#include "Constantes.h"
#include "Stack.h"
#include <stdio.h>
#include <stdlib.h>
```

Functions

- Stack * CriaStackValor (int cod)

  Funcao que cria um valor da stack em memoria.
- Stack * InsereNaStack (Stack *inicio, Stack *novo)

  Funcao que insere novo valor no fim da stack.
- Stack * RemoveValorStack (Stack *inicio)

  Funcao que Remove o valor de topo da stack.
- void OutputStack (Stack *stack)

  Funcao que da output aos valores guardados numa stack.
- int StackPeek (Stack *stack)

  Funcao que procura pelo codigo do valor de topo da stack.
- int IsStackEmpty (Stack *stack)

  Funcao que verifica se a stack esta vazia.

4.19.1 Detailed Description

Ficheiro .c com as funcoes da stack utilizada nos algoritmos de procura.

~

Author

Diogo Machado 26042

Date

21.05.2024

Copyright

4.19.2 Function Documentation

4.19.2.1 CriaStackValor()

```
Stack * CriaStackValor (
            int cod )
```

Funcao que cria um valor da stack em memoria.

Parameters

cod	- codigo do valor da stack

Return values

-	valor da stack criado em memoria

4.19.2.2 InsereNaStack()

```
Stack * InsereNaStack (
            Stack * inicio,
            Stack * novo )
```

Funcao que insere novo valor no fim da stack.

Parameters

inicio	- stack guardada em memoria
novo	- novo valor da stack que se pretende inserir na stack guardada em memoria

Return values

-	stack guarada em memoria

4.19.2.3 IsStackEmpty()

```
int IsStackEmpty (
            Stack * stack )
```

Funcao que verifica se a stack esta vazia.

Parameters

stack	- stack guardada em memoria

Return values

-	1 se estiver vazia e 0 se nao estiver

4.19.2.4 OutputStack()

```
void OutputStack (
            Stack * stack )
```

Funcao que da output aos valores guardados numa stack.

Parameters

stack	- stack guardada em memoria

4.19.2.5 RemoveValorStack()

```
Stack * RemoveValorStack (
            Stack * inicio )
```

Funcao que Remove o valor de topo da stack.

Parameters

inicio	- stack guardada em memoria

Return values

-	stack guardada em memoria depois de remover o valor de topo

4.19.2.6 StackPeek()

```
int StackPeek (
            Stack * stack )
```

Funcao que procura pelo codigo do valor de topo da stack.

Parameters

stack	- stack guardada em memoria

Return values

-	valor inteiro do codigo do valor de topo da stack

4.20 C:/Aulas/1º ano/2º semestre/Estruturas de Dados Avançadas/Trabalho EDA Fase 2/Src/FuncoesLIB/Stack.h File Reference

Ficheiro .h com as funcoes da stack utilizada nos algoritmos de procura.

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

Classes

- struct Stack

Typedefs

- typedef struct Stack Stack

Functions

- Stack * CriaStackValor (int cod)
  Funcao que cria um valor da stack em memoria.
- Stack * InsereNaStack (Stack *inicio, Stack *novo)
  Funcao que insere novo valor no fim da stack.
- Stack * RemoveValorStack (Stack *inicio)
  Funcao que Remove o valor de topo da stack.
- void OutputStack (Stack *stack)
  Funcao que da output aos valores guardados numa stack.
- int StackPeek (Stack *stack)
  Funcao que procura pelo codigo do valor de topo da stack.
- int IsStackEmpty (Stack *stack)
  Funcao que verifica se a stack esta vazia.

4.20.1 Detailed Description

Ficheiro .h com as funcoes da stack utilizada nos algoritmos de procura.

~

Author

Diogo Machado 26042

Date

22.05.2024

Copyright

4.20.2 Typedef Documentation

4.20.2.1 Stack

```
typedef struct Stack Stack
```

4.20.3 Function Documentation

4.20.3.1 CriaStackValor()

```
Stack * CriaStackValor (
            int cod )
```

Funcao que cria um valor da stack em memoria.

Parameters

cod	- codigo do valor da stack

Return values

-	valor da stack criado em memoria

4.20.3.2 InsereNaStack()

```
Stack * InsereNaStack (
            Stack * inicio,
            Stack * novo )
```

Funcao que insere novo valor no fim da stack.

Parameters

inicio	- stack guardada em memoria
novo	- novo valor da stack que se pretende inserir na stack guardada em memoria

Return values

-	stack guarada em memoria

4.20.3.3 IsStackEmpty()

```
int IsStackEmpty (
            Stack * stack )
```

Funcao que verifica se a stack esta vazia.

Parameters

stack	- stack guardada em memoria

Return values

-	1 se estiver vazia e 0 se nao estiver

4.20.3.4 OutputStack()

```
void OutputStack (
            Stack * stack )
```

Funcao que da output aos valores guardados numa stack.

Parameters

stack	- stack guardada em memoria

4.20.3.5 RemoveValorStack()

```
Stack * RemoveValorStack (
            Stack * inicio )
```

Funcao que Remove o valor de topo da stack.

Parameters

inicio	- stack guardada em memoria

Return values

-	stack guardada em memoria depois de remover o valor de topo

4.20.3.6 StackPeek()

```
int StackPeek (
            Stack * stack )
```

Funcao que procura pelo codigo do valor de topo da stack.

Parameters

stack	- stack guardada em memoria

Return values

-	valor inteiro do codigo do valor de topo da stack

4.21 Stack.h

Go to the documentation of this file.

```
00001
00011 #pragma once
00012 #include <stdbool.h>
00013 typedef struct Stack {
00014     int id;
00015     struct Stack* next;
00016 }Stack;
00017
00023 Stack* CriaStackValor(int cod);
00030 Stack* InsereNaStack(Stack* inicio, Stack* novo);
00036 Stack* RemoveValorStack(Stack* inicio);
00041 void OutputStack(Stack* stack);
00047 int StackPeek(Stack* stack);
00053 int IsStackEmpty(Stack* stack);
```

4.22 Stack.h File Reference

Ficheiro .h com as funcoes da stack utilizada nos algoritmos de procura.

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

Classes

- struct Stack

Typedefs

- typedef struct Stack Stack

Functions

- Stack * CriaStackValor (int cod)

  Funcao que cria um valor da stack em memoria.
- Stack * InsereNaStack (Stack *inicio, Stack *novo)

  Funcao que insere novo valor no fim da stack.
- Stack * RemoveValorStack (Stack *inicio)

  Funcao que Remove o valor de topo da stack.
- void OutputStack (Stack *stack)

  Funcao que da output aos valores guardados numa stack.
- int StackPeek (Stack *stack)

  Funcao que procura pelo codigo do valor de topo da stack.
- int IsStackEmpty (Stack *stack)

  Funcao que verifica se a stack esta vazia.

4.22.1 Detailed Description

Ficheiro .h com as funcoes da stack utilizada nos algoritmos de procura.

~

Author

Diogo Machado 26042

Date

22.05.2024

Copyright

4.22.2 Typedef Documentation

4.22.2.1 Stack

```
typedef struct Stack Stack
```

4.22.3 Function Documentation

4.22.3.1 CriaStackValor()

```
Stack * CriaStackValor (
            int cod )
```

Funcao que cria um valor da stack em memoria.

Parameters

cod	- codigo do valor da stack

Return values

-	valor da stack criado em memoria

4.22.3.2 InsereNaStack()

```
Stack * InsereNaStack (
            Stack * inicio,
            Stack * novo )
```

Funcao que insere novo valor no fim da stack.

Parameters

inicio	- stack guardada em memoria
novo	- novo valor da stack que se pretende inserir na stack guardada em memoria

Return values

-	stack guarada em memoria

4.22.3.3 IsStackEmpty()

```
int IsStackEmpty (
            Stack * stack )
```

Funcao que verifica se a stack esta vazia.

Parameters

stack	- stack guardada em memoria

Return values

-	1 se estiver vazia e 0 se nao estiver

4.22.3.4 OutputStack()

```
void OutputStack (
            Stack * stack )
```

Funcao que da output aos valores guardados numa stack.

Parameters

stack	- stack guardada em memoria

4.22.3.5 RemoveValorStack()

```
Stack * RemoveValorStack (
            Stack * inicio )
```

Funcao que Remove o valor de topo da stack.

Parameters

inicio	- stack guardada em memoria

Return values

-	stack guardada em memoria depois de remover o valor de topo

4.22.3.6 StackPeek()

```
int StackPeek (
            Stack * stack )
```

Funcao que procura pelo codigo do valor de topo da stack.

Parameters

stack	- stack guardada em memoria

Return values

-	valor inteiro do codigo do valor de topo da stack

4.23 Stack.h

Go to the documentation of this file.

```
00001
00011 #pragma once
00012 #include <stdbool.h>
00013 typedef struct Stack {
00014     int id;
00015     struct Stack* next;
00016 }Stack;
00017
00023 Stack* CriaStackValor(int cod);
00030 Stack* InsereNaStack(Stack* inicio, Stack* novo);
00036 Stack* RemoveValorStack(Stack* inicio);
00041 void OutputStack(Stack* stack);
00047 int StackPeek(Stack* stack);
00053 int IsStackEmpty(Stack* stack);
```

4.24 C:/Aulas/1º ano/2º semestre/Estruturas de Dados Avançadas/Trabalho EDA Fase 2/Src/FuncoesLIB/Utilidades.c File Reference

Ficheiro .c com algumas funcoes para manipular o grafo.

```
#include "Utilidades.h"
#include "Grafos.h"
#include "Constantes.h"
#include "GereGrafos.h"
#include <stdio.h>
#include <stdlib.h>
```

Functions

- int VerificaExisteVertice (Vertice *listaVert, int codVert)

  Funcao que verifica se o vertice com um determinado codigo ja existe ou nao.
- Grafo * AdicionaVertice (Grafo *grafo, int codVert)

  Funcao que adiciona vertice ao grafo.
- Grafo * RemoveVertice (Grafo *grafo, int codVert)

  Funcao que remove vertice do grafo.
- int LimpaListaAdjacencias (Vertice *vert)

  Funcao que limpa lista de adjacencias de um determinado vertice.
- int VerificaExisteAdjacencia (Adjacencia *listaAdj, int codDest)

  Funcao que verifica se existe uma determinada adjacencia na lista de adjacencias.
- Grafo * RemoveAdjacencia (Grafo *grafo, int codVert, int codAdj)

  Funcao que remove uma adjacencia de um determinado vertice.
- Grafo * AdicionaAdjacencia (Grafo *grafo, int codVert, int peso, int codDest)

  Funcao que adiciona uma determinada adjacencia a vertice destino.
- Vertice * ProcuraVerticeCod (Vertice *auxVert, int cod)

  Funcao que procura vertice com um determinado codigo.

4.24.1 Detailed Description

Ficheiro .c com algumas funcoes para manipular o grafo.

~

Author

Diogo Machado 26042

Date

21.05.2024

Copyright

4.24.2 Function Documentation

4.24.2.1 AdicionaAdjacencia()

```
Grafo * AdicionaAdjacencia (
            Grafo * grafo,
            int codVert,
            int peso,
            int codDest )
```

Funcao que adiciona uma determinada adjacencia a vertice destino.

Parameters

grafo	- grafo guardado em memoria
codVert	- codigo do vertice de partida
peso	- peso da adjacencia
codDest	- codigo do vertice destino

Return values

-	grafo guardado em memoria

4.24.2.2 AdicionaVertice()

```
Grafo * AdicionaVertice (
            Grafo * grafo,
            int codVert )
```

Funcao que adiciona vertice ao grafo.

Parameters

grafo	- grafo guardado em memoria
codVert	- codigo do vertice que pretendemos adicionar

Return values

-	grafo guardado em memoria

4.24.2.3 LimpaListaAdjacencias()

```
int LimpaListaAdjacencias (
            Vertice * vert )
```

Funcao que limpa lista de adjacencias de um determinado vertice.

Parameters

vert	- vertice que se pretende eliminar a lista de adjacencias

Return values

-	resultado da funcao

4.24.2.4 ProcuraVerticeCod()

```
Vertice * ProcuraVerticeCod (
            Vertice * auxVert,
            int cod )
```

Funcao que procura vertice com um determinado codigo.

Parameters

auxVert	- lista dos vertices
cod	- codigo que se pretende encontrar

Return values

-	vertice com o codigo pretendido

4.24.2.5 RemoveAdjacencia()

```
Grafo * RemoveAdjacencia (
            Grafo * grafo,
            int codVert,
            int codAdj )
```

Funcao que remove uma adjacencia de um determinado vertice.

Parameters

grafo	- grafo guardado em memoria
codVert	- codigo do vertice que se pretende remover a adjacencia
codAdj	- codigo da adjacencia que se pretende remover

Return values

-	grafo guardado em memoria

4.24.2.6 RemoveVertice()

```
Grafo * RemoveVertice (
            Grafo * grafo,
            int codVert )
```

Funcao que remove vertice do grafo.

Parameters

grafo	- grafo guardado em memoria
codVert	- codigo do vertice que se pretende adicionar

Return values

-	grafo guardado em memoria

4.24.2.7 VerificaExisteAdjacencia()

```
int VerificaExisteAdjacencia (
            Adjacencia * listaAdj,
            int codDest )
```

Funcao que verifica se existe uma determinada adjacencia na lista de adjacencias.

Parameters

listaAdj	- lista de adjacencias
codDest	- codigo da adjacencia que se pretende eliminar

Return values

-	resultado da funcao

4.24.2.8 VerificaExisteVertice()

```
int VerificaExisteVertice (
            Vertice * listaVert,
            int codVert )
```

Funcao que verifica se o vertice com um determinado codigo ja existe ou nao.

Parameters

listaVert	- lista de vertices do grafo
codVert	- codigo do vertice que pretendemos adicionar

Return values

-	se ja existe o vertice ou nao

4.25 C:/Aulas/1º ano/2º semestre/Estruturas de Dados Avançadas/Trabalho EDA Fase 2/Src/FuncoesLIB/Utilidades.h File Reference

Ficheiro .h com algumas funcoes para manipular o grafo.

```
#include "Grafos.h"
```

Functions

- int VerificaExisteVertice (Vertice *listaVert, int codVert)
  Funcao que verifica se o vertice com um determinado codigo ja existe ou nao.
- Grafo * AdicionaVertice (Grafo *grafo, int codVert)
  Funcao que adiciona vertice ao grafo.
- Grafo * RemoveVertice (Grafo *grafo, int codVert)
  Funcao que remove vertice do grafo.
- Vertice * ProcuraVerticeCod (Vertice *auxVert, int cod)
  Funcao que procura vertice com um determinado codigo.
- int LimpaListaAdjacencias (Vertice *vert)
  Funcao que limpa lista de adjacencias de um determinado vertice.
- int VerificaExisteAdjacencia (Adjacencia *listaAdj, int codDest)
  Funcao que verifica se existe uma determinada adjacencia na lista de adjacencias.
- Grafo * RemoveAdjacencia (Grafo *grafo, int codVert, int codAdj)
  Funcao que remove uma adjacencia de um determinado vertice.
- Grafo * AdicionaAdjacencia (Grafo *grafo, int codVert, int peso, int codDest)
  Funcao que adiciona uma determinada adjacencia a vertice destino.

4.25.1 Detailed Description

Ficheiro .h com algumas funcoes para manipular o grafo.

~

Author

Diogo Machado 26042

Date

21.05.2024

Copyright

4.25.2 Function Documentation

4.25.2.1 AdicionaAdjacencia()

```
Grafo * AdicionaAdjacencia (
            Grafo * grafo,
            int codVert,
            int peso,
            int codDest )
```

Funcao que adiciona uma determinada adjacencia a vertice destino.

Parameters

grafo	- grafo guardado em memoria
codVert	- codigo do vertice de partida
peso	- peso da adjacencia
codDest	- codigo do vertice destino

Return values

-	grafo guardado em memoria

4.25.2.2 AdicionaVertice()

```
Grafo * AdicionaVertice (
            Grafo * grafo,
            int codVert )
```

Funcao que adiciona vertice ao grafo.

Parameters

grafo	- grafo guardado em memoria
codVert	- codigo do vertice que pretendemos adicionar

Return values

-	grafo guardado em memoria

4.25.2.3 LimpaListaAdjacencias()

```
int LimpaListaAdjacencias (
            Vertice * vert )
```

Funcao que limpa lista de adjacencias de um determinado vertice.

Parameters

vert	- vertice que se pretende eliminar a lista de adjacencias

Return values

-	resultado da funcao

4.25.2.4 ProcuraVerticeCod()

```
Vertice * ProcuraVerticeCod (
            Vertice * auxVert,
            int cod )
```

Funcao que procura vertice com um determinado codigo.

Parameters

auxVert	- lista dos vertices
cod	- codigo que se pretende encontrar

Return values

-	vertice com o codigo pretendido

4.25.2.5 RemoveAdjacencia()

```
Grafo * RemoveAdjacencia (
            Grafo * grafo,
            int codVert,
            int codAdj )
```

Funcao que remove uma adjacencia de um determinado vertice.

Parameters

grafo	- grafo guardado em memoria
codVert	- codigo do vertice que se pretende remover a adjacencia
codAdj	- codigo da adjacencia que se pretende remover

Return values

-	grafo guardado em memoria

4.25.2.6 RemoveVertice()

```
Grafo * RemoveVertice (
            Grafo * grafo,
            int codVert )
```

Funcao que remove vertice do grafo.

Parameters

grafo	- grafo guardado em memoria
codVert	- codigo do vertice que se pretende adicionar

Return values

-	grafo guardado em memoria

4.25.2.7 VerificaExisteAdjacencia()

```
int VerificaExisteAdjacencia (
            Adjacencia * listaAdj,
            int codDest )
```

Funcao que verifica se existe uma determinada adjacencia na lista de adjacencias.

Parameters

listaAdj	- lista de adjacencias
codDest	- codigo da adjacencia que se pretende eliminar

Return values

-	resultado da funcao

4.25.2.8 VerificaExisteVertice()

```
int VerificaExisteVertice (
            Vertice * listaVert,
            int codVert )
```

Funcao que verifica se o vertice com um determinado codigo ja existe ou nao.

Parameters

listaVert	- lista de vertices do grafo
codVert	- codigo do vertice que pretendemos adicionar

Return values

-	se ja existe o vertice ou nao

4.26 Utilidades.h

Go to the documentation of this file.

```
00001
00011 #pragma once
00012 #include "Grafos.h"
00013
00014 #pragma region VERTICES
00021 int VerificaExisteVertice(Vertice* listaVert, int codVert);
00028 Grafo* AdicionaVertice(Grafo* grafo, int codVert);
00035 Grafo* RemoveVertice(Grafo* grafo, int codVert);
00042 Vertice* ProcuraVerticeCod(Vertice* auxVert, int cod);
00043 #pragma endregion
00044
00045 #pragma region ADJACENCIAS
00051 int LimpaListaAdjacencias(Vertice* vert);
00058 int VerificaExisteAdjacencia(Adjacencia* listaAdj, int codDest);
00066 Grafo* RemoveAdjacencia(Grafo* grafo, int codVert, int codAdj);
00075 Grafo* AdicionaAdjacencia(Grafo* grafo, int codVert, int peso, int codDest);
00076 #pragma endregion
```

4.27 Utilidades.h File Reference

Ficheiro .h com algumas funcoes para manipular o grafo.

```
#include "Grafos.h"
```

Functions

- int VerificaExisteVertice (Vertice *listaVert, int codVert)

  Funcao que verifica se o vertice com um determinado codigo ja existe ou nao.
- Grafo * AdicionaVertice (Grafo *grafo, int codVert)

  Funcao que adiciona vertice ao grafo.
- Grafo * RemoveVertice (Grafo *grafo, int codVert)

  Funcao que remove vertice do grafo.
- Vertice * ProcuraVerticeCod (Vertice *auxVert, int cod)

  Funcao que procura vertice com um determinado codigo.
- int LimpaListaAdjacencias (Vertice *vert)

  Funcao que limpa lista de adjacencias de um determinado vertice.
- int VerificaExisteAdjacencia (Adjacencia *listaAdj, int codDest)

  Funcao que verifica se existe uma determinada adjacencia na lista de adjacencias.
- Grafo * RemoveAdjacencia (Grafo *grafo, int codVert, int codAdj)

  Funcao que remove uma adjacencia de um determinado vertice.
- Grafo * AdicionaAdjacencia (Grafo *grafo, int codVert, int peso, int codDest)

  Funcao que adiciona uma determinada adjacencia a vertice destino.

4.27.1 Detailed Description

Ficheiro .h com algumas funcoes para manipular o grafo.

~

Author

Diogo Machado 26042

Date

21.05.2024

Copyright

4.27.2 Function Documentation

4.27.2.1 AdicionaAdjacencia()

```
Grafo * AdicionaAdjacencia (
            Grafo * grafo,
            int codVert,
            int peso,
            int codDest )
```

Funcao que adiciona uma determinada adjacencia a vertice destino.

Parameters

grafo	- grafo guardado em memoria
codVert	- codigo do vertice de partida
peso	- peso da adjacencia
codDest	- codigo do vertice destino

Return values

-	grafo guardado em memoria

4.27.2.2 AdicionaVertice()

```
Grafo * AdicionaVertice (
            Grafo * grafo,
            int codVert )
```

Funcao que adiciona vertice ao grafo.

Parameters

grafo	- grafo guardado em memoria
codVert	- codigo do vertice que pretendemos adicionar

Return values

-	grafo guardado em memoria

4.27.2.3 LimpaListaAdjacencias()

```
int LimpaListaAdjacencias (
            Vertice * vert )
```

Funcao que limpa lista de adjacencias de um determinado vertice.

Parameters

vert	- vertice que se pretende eliminar a lista de adjacencias

Return values

-	resultado da funcao

4.27.2.4 ProcuraVerticeCod()

```
Vertice * ProcuraVerticeCod (
            Vertice * auxVert,
            int cod )
```

Funcao que procura vertice com um determinado codigo.

Parameters

auxVert	- lista dos vertices
cod	- codigo que se pretende encontrar

Return values

-	vertice com o codigo pretendido

4.27.2.5 RemoveAdjacencia()

```
Grafo * RemoveAdjacencia (
            Grafo * grafo,
```

```
            int codVert,
            int codAdj )
```

Funcao que remove uma adjacencia de um determinado vertice.

Parameters

grafo	- grafo guardado em memoria
codVert	- codigo do vertice que se pretende remover a adjacencia
codAdj	- codigo da adjacencia que se pretende remover

Return values

-	grafo guardado em memoria

4.27.2.6 RemoveVertice()

```
Grafo * RemoveVertice (
            Grafo * grafo,
            int codVert )
```

Funcao que remove vertice do grafo.

Parameters

grafo	- grafo guardado em memoria
codVert	- codigo do vertice que se pretende adicionar

Return values

-	grafo guardado em memoria

4.27.2.7 VerificaExisteAdjacencia()

```
int VerificaExisteAdjacencia (
            Adjacencia * listaAdj,
            int codDest )
```

Funcao que verifica se existe uma determinada adjacencia na lista de adjacencias.

Parameters

listaAdj	- lista de adjacencias
codDest	- codigo da adjacencia que se pretende eliminar

Return values

-	resultado da funcao

4.27.2.8 VerificaExisteVertice()

```
int VerificaExisteVertice (
            Vertice * listaVert,
            int codVert )
```

Funcao que verifica se o vertice com um determinado codigo ja existe ou nao.

Parameters

listaVert	- lista de vertices do grafo
codVert	- codigo do vertice que pretendemos adicionar

Return values

-	se ja existe o vertice ou nao

4.28 Utilidades.h

Go to the documentation of this file.

```
00001
00011 #pragma once
00012 #include "Grafos.h"
00013
00014 #pragma region VERTICES
00021 int VerificaExisteVertice(Vertice* listaVert, int codVert);
00028 Grafo* AdicionaVertice(Grafo* grafo, int codVert);
00035 Grafo* RemoveVertice(Grafo* grafo, int codVert);
00042 Vertice* ProcuraVerticeCod(Vertice* auxVert, int cod);
00043 #pragma endregion
00044
00045 #pragma region ADJACENCIAS
00051 int LimpaListaAdjacencias(Vertice* vert);
00058 int VerificaExisteAdjacencia(Adjacencia* listaAdj, int codDest);
00066 Grafo* RemoveAdjacencia(Grafo* grafo, int codVert, int codAdj);
00075 Grafo* AdicionaAdjacencia(Grafo* grafo, int codVert, int peso, int codDest);
00076 #pragma endregion
```

4.29 Main.c File Reference

Ficheiro main da segunda fase do trabalho de Estrutura de Dados Avanas.

```
#include "GereGrafos.h"
#include "Utilidades.h"
#include "FuncoesProcura.h"
#include "Stack.h"
```

Functions

- int main ()

4.29.1 Detailed Description

Ficheiro main da segunda fase do trabalho de Estrutura de Dados Avanas.

~

Author

Diogo Machado 26042

Date

6.05.2024

4.29.2 Function Documentation

4.29.2.1 main()

```
int main ( )
```

Index