

# My Project

Generated by Doxygen 1.9.8



<b>1 Class Index</b>	<b>1</b>
1.1 Class List	1
<b>2 File Index</b>	<b>3</b>
2.1 File List	3
<b>3 Class Documentation</b>	<b>5</b>
3.1 Adjacencia Struct Reference	5
3.1.1 Member Data Documentation	5
3.1.1.1 codVert	5
3.1.1.2 next	5
3.1.1.3 peso	5
3.1.1.4 visitado	5
3.2 Grafo Struct Reference	6
3.2.1 Member Data Documentation	6
3.2.1.1 inicio	6
3.2.1.2 numVert	6
3.3 Stack Struct Reference	6
3.3.1 Member Data Documentation	6
3.3.1.1 id	6
3.3.1.2 next	7
3.4 Vertice Struct Reference	7
3.4.1 Member Data Documentation	7
3.4.1.1 adjacencias	7
3.4.1.2 cod	7
3.4.1.3 next	7
3.4.1.4 numAdj	7
3.4.1.5 visitado	7
<b>4 File Documentation</b>	<b>9</b>
4.1 C:/Aulas/1º ano/2º semestre/Estruturas de Dados Avançadas/Trabalho EDA Fase 2/Src/Funcoes↵ LIB/Constantes.h File Reference	9
4.1.1 Detailed Description	9
4.1.2 Macro Definition Documentation	10
4.1.2.1 ERRO_AO_ABRIR_FICHEIRO	10
4.1.2.2 EXISTE_ADJACENCIA	10
4.1.2.3 GRAFO_NAO_EXISTE	10
4.1.2.4 LISTA_NAO_EXISTE	10
4.1.2.5 MAX	10
4.1.2.6 MAX_CARATERES	10
4.1.2.7 NAO_EXISTE_ADJACENCIA	10
4.1.2.8 PARAMETROS_INVALIDOS	10
4.1.2.9 SUCESSO	10

4.1.2.10 VERTICE_JA_EXISTE . . . . .	10
4.1.2.11 VERTICE_NAO_EXISTE . . . . .	11
4.2 Constantes.h . . . . .	11
4.3 Constantes.h File Reference . . . . .	11
4.3.1 Detailed Description . . . . .	11
4.3.2 Macro Definition Documentation . . . . .	12
4.3.2.1 ERRO_AO_ABRIR_FICHEIRO . . . . .	12
4.3.2.2 EXISTE_ADJACENCIA . . . . .	12
4.3.2.3 GRAFO_NAO_EXISTE . . . . .	12
4.3.2.4 LISTA_NAO_EXISTE . . . . .	12
4.3.2.5 MAX . . . . .	12
4.3.2.6 MAX_CARATERES . . . . .	12
4.3.2.7 NAO_EXISTE_ADJACENCIA . . . . .	12
4.3.2.8 PARAMETROS_INVALIDOS . . . . .	12
4.3.2.9 SUCESSO . . . . .	12
4.3.2.10 VERTICE_JA_EXISTE . . . . .	12
4.3.2.11 VERTICE_NAO_EXISTE . . . . .	13
4.4 Constantes.h . . . . .	13
4.5 C:/Aulas/1º ano/2º semestre/Estruturas de Dados Avançadas/Trabalho EDA Fase 2/Src/Funcoes↔ LIB/FuncoesProcura.c File Reference . . . . .	13
4.5.1 Function Documentation . . . . .	13
4.5.1.1 CalculoSomaEntreDoisVertices() . . . . .	13
4.5.1.2 DepthFirstSearchRec() . . . . .	14
4.5.1.3 DepthFirstTraversal() . . . . .	14
4.5.1.4 ProcessaVertice() . . . . .	15
4.6 C:/Aulas/1º ano/2º semestre/Estruturas de Dados Avançadas/Trabalho EDA Fase 2/Src/Funcoes↔ LIB/FuncoesProcura.h File Reference . . . . .	15
4.6.1 Detailed Description . . . . .	15
4.6.2 Function Documentation . . . . .	16
4.6.2.1 CalculoSomaEntreDoisVerticesz() . . . . .	16
4.6.2.2 DepthFirstSearchRec() . . . . .	16
4.6.2.3 DepthFirstTraversal() . . . . .	16
4.6.2.4 ProcessaVertice() . . . . .	17
4.7 FuncoesProcura.h . . . . .	17
4.8 FuncoesProcura.h File Reference . . . . .	17
4.8.1 Detailed Description . . . . .	18
4.8.2 Function Documentation . . . . .	18
4.8.2.1 CalculoSomaEntreDoisVertices() . . . . .	18
4.8.2.2 DepthFirstSearchRec() . . . . .	18
4.8.2.3 DepthFirstTraversal() . . . . .	19
4.8.2.4 ProcessaVertice() . . . . .	19
4.9 FuncoesProcura.h . . . . .	19

4.10 C:/Aulas/1º ano/2º semestre/Estruturas de Dados Avançadas/Trabalho EDA Fase 2/Src/Funcoes↔ LIB/GereGrafos.c File Reference	20
4.10.1 Detailed Description	20
4.10.2 Function Documentation	21
4.10.2.1 CriaAdjacencia()	21
4.10.2.2 CriaGrafo()	21
4.10.2.3 CriaVertice()	21
4.10.2.4 InsereAdjacencia()	22
4.10.2.5 InsereAdjacenciaVertice()	22
4.10.2.6 InsereVertice()	22
4.10.2.7 LeFicheiroGrafo()	23
4.10.2.8 LimpaGrafo()	23
4.10.2.9 MostraGrafo()	23
4.11 C:/Aulas/1º ano/2º semestre/Estruturas de Dados Avançadas/Trabalho EDA Fase 2/Src/Funcoes↔ LIB/GereGrafos.h File Reference	25
4.11.1 Detailed Description	25
4.11.2 Function Documentation	26
4.11.2.1 CriaAdjacencia()	26
4.11.2.2 CriaGrafo()	26
4.11.2.3 CriaVertice()	27
4.11.2.4 InsereAdjacencia()	27
4.11.2.5 InsereAdjacenciaVertice()	27
4.11.2.6 InsereVertice()	28
4.11.2.7 LeFicheiroGrafo()	28
4.11.2.8 LimpaGrafo()	28
4.11.2.9 MostraGrafo()	30
4.12 GereGrafos.h	30
4.13 GereGrafos.h File Reference	30
4.13.1 Detailed Description	31
4.13.2 Function Documentation	31
4.13.2.1 CriaAdjacencia()	31
4.13.2.2 CriaGrafo()	32
4.13.2.3 CriaVertice()	32
4.13.2.4 InsereAdjacencia()	32
4.13.2.5 InsereAdjacenciaVertice()	33
4.13.2.6 InsereVertice()	33
4.13.2.7 LeFicheiroGrafo()	34
4.13.2.8 LimpaGrafo()	34
4.13.2.9 MostraGrafo()	34
4.14 GereGrafos.h	35
4.15 C:/Aulas/1º ano/2º semestre/Estruturas de Dados Avançadas/Trabalho EDA Fase 2/Src/Funcoes↔ LIB/Grafos.h File Reference	35
4.15.1 Detailed Description	36

4.15.2 Typedef Documentation	36
4.15.2.1 Adjacencia	36
4.15.2.2 Grafo	36
4.15.2.3 Vertice	36
4.16 Grafos.h	36
4.17 Grafos.h File Reference	37
4.17.1 Detailed Description	37
4.17.2 Typedef Documentation	37
4.17.2.1 Adjacencia	37
4.17.2.2 Grafo	37
4.17.2.3 Vertice	38
4.18 Grafos.h	38
4.19 C:/Aulas/1º ano/2º semestre/Estruturas de Dados Avançadas/Trabalho EDA Fase 2/Src/Funcoes↔ LIB/Stack.c File Reference	38
4.19.1 Detailed Description	39
4.19.2 Function Documentation	39
4.19.2.1 CriaStackValor()	39
4.19.2.2 InsereNaStack()	39
4.19.2.3 IsStackEmpty()	40
4.19.2.4 OutputStack()	40
4.19.2.5 RemoveValorStack()	40
4.19.2.6 StackPeek()	41
4.20 C:/Aulas/1º ano/2º semestre/Estruturas de Dados Avançadas/Trabalho EDA Fase 2/Src/Funcoes↔ LIB/Stack.h File Reference	41
4.20.1 Detailed Description	42
4.20.2 Typedef Documentation	42
4.20.2.1 Stack	42
4.20.3 Function Documentation	42
4.20.3.1 CriaStackValor()	42
4.20.3.2 InsereNaStack()	42
4.20.3.3 IsStackEmpty()	44
4.20.3.4 OutputStack()	44
4.20.3.5 RemoveValorStack()	44
4.20.3.6 StackPeek()	45
4.21 Stack.h	45
4.22 Stack.h File Reference	45
4.22.1 Detailed Description	46
4.22.2 Typedef Documentation	46
4.22.2.1 Stack	46
4.22.3 Function Documentation	46
4.22.3.1 CriaStackValor()	46
4.22.3.2 InsereNaStack()	47
4.22.3.3 IsStackEmpty()	47

4.22.3.4	OutputStack()	47
4.22.3.5	RemoveValorStack()	48
4.22.3.6	StackPeek()	48
4.23	Stack.h	48
4.24	C:/Aulas/1º ano/2º semestre/Estruturas de Dados Avançadas/Trabalho EDA Fase 2/Src/Funcoes← LIB/Utilidades.c File Reference	49
4.24.1	Detailed Description	49
4.24.2	Function Documentation	50
4.24.2.1	AdicionaAdjacencia()	50
4.24.2.2	AdicionaVertice()	50
4.24.2.3	LimpaListaAdjacencias()	50
4.24.2.4	ProcuraVerticeCod()	51
4.24.2.5	RemoveAdjacencia()	51
4.24.2.6	RemoveVertice()	52
4.24.2.7	VerificaExisteAdjacencia()	52
4.24.2.8	VerificaExisteVertice()	52
4.25	C:/Aulas/1º ano/2º semestre/Estruturas de Dados Avançadas/Trabalho EDA Fase 2/Src/Funcoes← LIB/Utilidades.h File Reference	53
4.25.1	Detailed Description	53
4.25.2	Function Documentation	54
4.25.2.1	AdicionaAdjacencia()	54
4.25.2.2	AdicionaVertice()	54
4.25.2.3	LimpaListaAdjacencias()	54
4.25.2.4	ProcuraVerticeCod()	55
4.25.2.5	RemoveAdjacencia()	55
4.25.2.6	RemoveVertice()	56
4.25.2.7	VerificaExisteAdjacencia()	56
4.25.2.8	VerificaExisteVertice()	56
4.26	Utilidades.h	57
4.27	Utilidades.h File Reference	57
4.27.1	Detailed Description	58
4.27.2	Function Documentation	58
4.27.2.1	AdicionaAdjacencia()	58
4.27.2.2	AdicionaVertice()	58
4.27.2.3	LimpaListaAdjacencias()	59
4.27.2.4	ProcuraVerticeCod()	59
4.27.2.5	RemoveAdjacencia()	59
4.27.2.6	RemoveVertice()	60
4.27.2.7	VerificaExisteAdjacencia()	60
4.27.2.8	VerificaExisteVertice()	61
4.28	Utilidades.h	61
4.29	Main.c File Reference	61
4.29.1	Detailed Description	62

4.29.2 Function Documentation . . . . .	62
4.29.2.1 main() . . . . .	62
<b>Index</b>	<b>63</b>



# Chapter 1

## Class Index

### 1.1 Class List

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

<a href="#">Adjacencia</a>	5
<a href="#">Grafo</a>	6
<a href="#">Stack</a>	6
<a href="#">Vertice</a>	7



## Chapter 2

# File Index

### 2.1 File List

Here is a list of all files with brief descriptions:

C:/Aulas/1º ano/2º semestre/Estruturas de Dados Avançadas/Trabalho EDA Fase 2/Src/Funcoes↔ LIB/ <a href="#">Constantes.h</a> Ficheiro .h com as constantes utilizadas no trabalho . . . . .	9
C:/Aulas/1º ano/2º semestre/Estruturas de Dados Avançadas/Trabalho EDA Fase 2/Src/Funcoes↔ LIB/ <a href="#">FuncoesProcura.c</a> . . . . .	13
C:/Aulas/1º ano/2º semestre/Estruturas de Dados Avançadas/Trabalho EDA Fase 2/Src/Funcoes↔ LIB/ <a href="#">FuncoesProcura.h</a> Ficheiro .h com as funcoes de procura do trabalho . . . . .	15
C:/Aulas/1º ano/2º semestre/Estruturas de Dados Avançadas/Trabalho EDA Fase 2/Src/Funcoes↔ LIB/ <a href="#">GereGrafos.c</a> Ficheiro .c que contuns de gerir grafos . . . . .	20
C:/Aulas/1º ano/2º semestre/Estruturas de Dados Avançadas/Trabalho EDA Fase 2/Src/Funcoes↔ LIB/ <a href="#">GereGrafos.h</a> Ficheiro .h que contem funcoes de gerir grafos . . . . .	25
C:/Aulas/1º ano/2º semestre/Estruturas de Dados Avançadas/Trabalho EDA Fase 2/Src/Funcoes↔ LIB/ <a href="#">Grafos.h</a> Ficheiro .h com a estrutura de dados utilizada para o trabalho prco . . . . .	35
C:/Aulas/1º ano/2º semestre/Estruturas de Dados Avançadas/Trabalho EDA Fase 2/Src/Funcoes↔ LIB/ <a href="#">Stack.c</a> Ficheiro .c com as funcoes da stack utilizada nos algoritmos de procura . . . . .	38
C:/Aulas/1º ano/2º semestre/Estruturas de Dados Avançadas/Trabalho EDA Fase 2/Src/Funcoes↔ LIB/ <a href="#">Stack.h</a> Ficheiro .h com as funcoes da stack utilizada nos algoritmos de procura . . . . .	41
C:/Aulas/1º ano/2º semestre/Estruturas de Dados Avançadas/Trabalho EDA Fase 2/Src/Funcoes↔ LIB/ <a href="#">Utilidades.c</a> Ficheiro .c com algumas funcoes para manipular o grafo . . . . .	49
C:/Aulas/1º ano/2º semestre/Estruturas de Dados Avançadas/Trabalho EDA Fase 2/Src/Funcoes↔ LIB/ <a href="#">Utilidades.h</a> Ficheiro .h com algumas funcoes para manipular o grafo . . . . .	53
<a href="#">Constantes.h</a> Ficheiro .h com as constantes utilizadas no trabalho . . . . .	11
<a href="#">FuncoesProcura.h</a> Ficheiro .h com as funcoes de procura do trabalho . . . . .	17
<a href="#">GereGrafos.h</a> Ficheiro .h que contem funcoes de gerir grafos . . . . .	30

<a href="#">Grafos.h</a>	Ficheiro .h com a estrutura de dados utilizada para o trabalho prco . . . . .	37
<a href="#">Main.c</a>	Ficheiro main da segunda fase do trabalho de Estrutura de Dados Avanas . . . . .	61
<a href="#">Stack.h</a>	Ficheiro .h com as funcoes da stack utilizada nos algoritmos de procura . . . . .	45
<a href="#">Utilidades.h</a>	Ficheiro .h com algumas funcoes para manipular o grafo . . . . .	57

## 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

Diogo Machado, 2023. All right reserved.

## 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

Diogo Machado, 2023. All right reserved.

## 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 ecrã 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

<i>grafo</i>	- grafo guardado em memoria
<i>codIni</i>	- codigo do vertice onde se comeca a pesquisa
<i>codDest</i>	- 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

<i>auxVert</i>	- lista de vertices do grafo
<i>origem</i>	- onde comeca a procura
<i>dest</i>	- 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

<i>grafo</i>	- grafo guardado em memoria
<i>codIni</i>	- codigo do vertice onde comeca a procura

## Return values

-	<a href="#">Stack</a> com os registos onde se passou para a procura
---	---

#### 4.5.1.4 ProcessaVertice()

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

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

##### Parameters

<i>id</i>	- 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 ecrã 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

Diogo Machado, 2023. All right reserved.

## 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

<i>grafo</i>	- grafo guardado em memoria
<i>codIni</i>	- codigo do vertice onde se comeca a pesquisa
<i>codDest</i>	- 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

<i>auxVert</i>	- lista de vertices do grafo
<i>origem</i>	- onde comeca a procura
<i>dest</i>	- 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

<i>grafo</i>	- grafo guardado em memoria
<i>codIni</i>	- codigo do vertice onde comeca a procura

## Return values

-	<a href="#">Stack</a> com os registos onde se passou para a procura
---	---

## 4.6.2.4 ProcessaVertice()

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

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

## Parameters

<i>id</i>	- 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 ecrã 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

Diogo Machado, 2023. All right reserved.

### 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

<i>grafo</i>	- grafo guardado em memoria
<i>codIni</i>	- codigo do vertice onde se comeca a pesquisa
<i>codDest</i>	- 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

<i>auxVert</i>	- lista de vertices do grafo
<i>origem</i>	- onde comeca a procura
<i>dest</i>	- 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

<i>grafo</i>	- grafo guardado em memoria
<i>codIni</i>	- codigo do vertice onde comeca a procura

## Return values

-	<a href="#">Stack</a> com os registos onde se passou para a procura
---	---

## 4.8.2.4 ProcessaVertice()

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

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

## Parameters

<i>id</i>	- 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 espaço em memória 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 espaço em memória 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 memória para o grafo e introduz os valores nas variaveis.*
- int **LimpaGrafo** (Grafo \*grafo)  
*Limpa memória.*
- **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

Diogo Machado, 2023. All right reserved.

## 4.10.2 Function Documentation

### 4.10.2.1 CriaAdjacencia()

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

Cria espaço em memória para uma adjacência e coloca valores nas suas variáveis.

#### Parameters

<i>cod</i>	- número de código do vértice adjacente
<i>peso</i>	- peso da adjacência

#### Return values

-	adjacência criada
---	-------------------

### 4.10.2.2 CriaGrafo()

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

Cria memória para o grafo e introduz os valores nas variáveis.

#### Parameters

<i>inicioVert</i>	- início da lista dos vértices
<i>numVert</i>	- número de vértices do grafo

#### Return values

-	grafo com valores nas variáveis
---	---------------------------------

### 4.10.2.3 CriaVertice()

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

Cria espaço em memória para um vértice e coloca valores nas suas variáveis.

#### Parameters

<i>cod</i>	- número de código do vértice
------------	-------------------------------

## Return values

-	vertice criado
---	----------------

**4.10.2.4 InserirAdjacencia()**

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

Inserir adjacencia nova na lista de adjacencias.

## Parameters

<i>inicio</i>	- inicio da lista de adjacencias
<i>nova</i>	- nova adjacencia que se pretende colocar na lista

## Return values

-	lista de adjacencias
---	----------------------

**4.10.2.5 InserirAdjacenciaVertice()**

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

Inserir lista de adjacencias no vertice correspondente.

## Parameters

<i>vertice</i>	- vertice do grafo que pretendemos colocar as adjacencias
<i>inicioAdj</i>	- lista de adjacentes do vertice

## Return values

-	vertice com as adjacencias colocadas
---	--------------------------------------

**4.10.2.6 InserirVertice()**

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

Inserir vertice novo na lista ligada dos vertices.

#### Parameters

<i>inicio</i>	- inicio da lista dos vertices
<i>novo</i>	- 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

<i>filename</i>	- nome do ficheiro
-----------------	--------------------

#### Return values

-	grafo
---	-------

### 4.10.2.8 LimpaGrafo()

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

Limpa memoria.

#### Parameters

<i>grafo</i>	- 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

<i>grafo</i>	- 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](#) \* [InserereVertice](#) ([Vertice](#) \*inicio, [Vertice](#) \*novo)  
*Inserere 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](#) \* [InserereAdjacencia](#) ([Adjacencia](#) \*inicio, [Adjacencia](#) \*nova)  
*Inserere adjacencia nova na lista de adjacencias.*
- [Vertice](#) \* [InserereAdjacenciaVertice](#) ([Vertice](#) \*vertice, [Adjacencia](#) \*inicioAdj)  
*Inserere 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**

Diogo Machado, 2023. All right reserved.

## 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**

<i>cod</i>	- numero de codigo do vertice adjacente
<i>peso</i>	- 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**

<i>inicioVert</i>	- inicio da lista dos vertices
<i>numVert</i>	- numero de vertices do grafo

**Return values**

-	grafo com valores nas variaveis
---	---------------------------------

#### 4.11.2.3 CriaVertice()

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

Cria espaço em memória para um vértice e coloca valores nas suas variáveis.

##### Parameters

<i>cod</i>	- número de código do vértice
------------	-------------------------------

##### Return values

-	vértice criado
---	----------------

#### 4.11.2.4 InsereAdjacencia()

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

Insere adjacência nova na lista de adjacências.

##### Parameters

<i>inicio</i>	- início da lista de adjacências
<i>nova</i>	- nova adjacência que se pretende colocar na lista

##### Return values

-	lista de adjacências
---	----------------------

#### 4.11.2.5 InsereAdjacenciaVertice()

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

Insere lista de adjacências no vértice correspondente.

##### Parameters

<i>vertice</i>	- vértice do grafo que pretendemos colocar as adjacências
<i>inicioAdj</i>	- lista de adjacentes do vértice

##### Return values

-	vértice com as adjacências colocadas
---	--------------------------------------

#### 4.11.2.6 InserirVertice()

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

Inserir vertice novo na lista ligada dos vertices.

##### Parameters

<i>inicio</i>	- inicio da lista dos vertices
<i>novo</i>	- vertice novo que se pretende introduzir na lista

##### Return values

-	inicio da lista de vertices
---	-----------------------------

#### 4.11.2.7 LerFicheiroGrafo()

```
Grafo * LerFicheiroGrafo (
    char * filename )
```

Ler ficheiro CSV com as informacoes do grafo.

##### Parameters

<i>filename</i>	- nome do ficheiro
-----------------	--------------------

##### Return values

-	grafo com as informacoes lidas do ficheiro
---	--

Ler ficheiro CSV com as informacoes do grafo.

##### Parameters

<i>filename</i>	- nome do ficheiro
-----------------	--------------------

##### Return values

-	grafo
---	-------

#### 4.11.2.8 LimparGrafo()

```
int LimparGrafo (
    Grafo * grafo )
```

Limpa memoria.

**Parameters**

<i>grafo</i>	- 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**

<i>grafo</i>	- 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

Diogo Machado, 2023. All right reserved.

### 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**

<i>cod</i>	- numero de codigo do vertice adjacente
<i>peso</i>	- 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**

<i>inicioVert</i>	- inicio da lista dos vertices
<i>numVert</i>	- 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**

<i>cod</i>	- 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

<i>inicio</i>	- inicio da lista de adjacencias
<i>nova</i>	- 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

<i>vertice</i>	- vertice do grafo que pretendemos colocar as adjacencias
<i>inicioAdj</i>	- 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

<i>inicio</i>	- inicio da lista dos vertices
<i>novo</i>	- 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

<i>filename</i>	- nome do ficheiro
-----------------	--------------------

##### Return values

-	grafo com as informacoes lidas do ficheiro
---	--

Le ficheiro CSV com as informacoes do grafo.

##### Parameters

<i>filename</i>	- nome do ficheiro
-----------------	--------------------

##### Return values

-	grafo
---	-------

#### 4.13.2.8 LimpaGrafo()

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

Limpa memoria.

##### Parameters

<i>grafo</i>	- 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

<i>grafo</i>	- 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 proco.

```
#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

Diogo Machado, 2023. All right reserved.

### 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

Diogo Machado, 2023. All right reserved.

### 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

Diogo Machado, 2023. All right reserved.

### 4.19.2 Function Documentation

#### 4.19.2.1 CriaStackValor()

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

Funcao que cria um valor da stack em memoria.

#### Parameters

<i>cod</i>	- 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

<i>inicio</i>	- stack guardada em memoria
<i>novo</i>	- novo valor da stack que se pretende inserir na stack guardada em memoria

## Return values

-	stack guardada em memoria
---	---------------------------

#### 4.19.2.3 IsStackEmpty()

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

Funcao que verifica se a stack esta vazia.

## Parameters

<i>stack</i>	- 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

<i>stack</i>	- stack guardada em memoria
--------------	-----------------------------

#### 4.19.2.5 RemoveValorStack()

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

Funcao que Remove o valor de topo da stack.

## Parameters

<i>inicio</i>	- 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

<code>stack</code>	- 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

Diogo Machado, 2023. All right reserved.

### 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

<i>cod</i>	- 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**

<i>inicio</i>	- stack guardada em memoria
<i>novo</i>	- 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**

<i>stack</i>	- 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**

<i>stack</i>	- stack guardada em memoria
--------------	-----------------------------

**4.20.3.5 RemoveValorStack()**

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

Funcao que Remove o valor de topo da stack.

**Parameters**

<i>inicio</i>	- 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

Diogo Machado, 2023. All right reserved.

### 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

<i>cod</i>	- codigo do valor da stack
------------	----------------------------

## Return values

-	valor da stack criado em memoria
---	----------------------------------

### 4.22.3.2 InereNaStack()

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

Funcao que insere novo valor no fim da stack.

## Parameters

<i>inicio</i>	- stack guardada em memoria
<i>novo</i>	- 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

<i>stack</i>	- 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

<i>stack</i>	- stack guardada em memoria
--------------	-----------------------------

**4.22.3.5 RemoveValorStack()**

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

Funcao que Remove o valor de topo da stack.

## Parameters

<i>inicio</i>	- 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

<i>stack</i>	- 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

Diogo Machado, 2023. All right reserved.

## 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

<i>grafo</i>	- grafo guardado em memoria
<i>codVert</i>	- codigo do vertice de partida
<i>peso</i>	- peso da adjacencia
<i>codDest</i>	- 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

<i>grafo</i>	- grafo guardado em memoria
<i>codVert</i>	- 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

<i>vert</i>	- 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

<i>auxVert</i>	- lista dos vertices
<i>cod</i>	- 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

<i>grafo</i>	- grafo guardado em memoria
<i>codVert</i>	- codigo do vertice que se pretende remover a adjacencia
<i>codAdj</i>	- 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

<i>grafo</i>	- grafo guardado em memoria
<i>codVert</i>	- 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

<i>listaAdj</i>	- lista de adjacencias
<i>codDest</i>	- 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

<i>listaVert</i>	- lista de vertices do grafo
<i>codVert</i>	- 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

Diogo Machado, 2023. All right reserved.

## 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

<i>grafo</i>	- grafo guardado em memoria
<i>codVert</i>	- codigo do vertice de partida
<i>peso</i>	- peso da adjacencia
<i>codDest</i>	- 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

<i>grafo</i>	- grafo guardado em memoria
<i>codVert</i>	- 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

<i>vert</i>	- 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

<i>auxVert</i>	- lista dos vertices
<i>cod</i>	- 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

<i>grafo</i>	- grafo guardado em memoria
<i>codVert</i>	- codigo do vertice que se pretende remover a adjacencia
<i>codAdj</i>	- 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

<i>grafo</i>	- grafo guardado em memoria
<i>codVert</i>	- 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

<i>listaAdj</i>	- lista de adjacencias
<i>codDest</i>	- 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

<i>listaVert</i>	- lista de vertices do grafo
<i>codVert</i>	- 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(Vertex* listaVert, int codVert);
00028 Grafo* AdicionaVertice(Grafo* grafo, int codVert);
00035 Grafo* RemoveVertice(Grafo* grafo, int codVert);
00042 Vertex* ProcuraVerticeCod(Vertex* auxVert, int cod);
00043 #pragma endregion
00044
00045 #pragma region ADJACENCIAS
00051 int LimpaListaAdjacencias(Vertex* 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](#) ([Vertex](#) \*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.*
- [Vertex](#) \* [ProcuraVerticeCod](#) ([Vertex](#) \*auxVert, int cod)  
*Funcao que procura vertice com um determinado codigo.*
- int [LimpaListaAdjacencias](#) ([Vertex](#) \*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

Diogo Machado, 2023. All right reserved.

### 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

<i>grafo</i>	- grafo guardado em memoria
<i>codVert</i>	- codigo do vertice de partida
<i>peso</i>	- peso da adjacencia
<i>codDest</i>	- 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

<i>grafo</i>	- grafo guardado em memoria
<i>codVert</i>	- 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

<i>vert</i>	- 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

<i>auxVert</i>	- lista dos vertices
<i>cod</i>	- 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

<i>grafo</i>	- grafo guardado em memoria
<i>codVert</i>	- codigo do vertice que se pretende remover a adjacencia
<i>codAdj</i>	- 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

<i>grafo</i>	- grafo guardado em memoria
<i>codVert</i>	- 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

<i>listaAdj</i>	- lista de adjacencias
<i>codDest</i>	- 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

<i>listaVert</i>	- lista de vertices do grafo
<i>codVert</i>	- 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

#### Copyright

Diogo Machado, 2023. All right reserved.

### 4.29.2 Function Documentation

#### 4.29.2.1 main()

```
int main ( )
```

# Index

AdicionaAdjacencia  
    Utilidades.c, 50  
    Utilidades.h, 54, 58  
AdicionaVertice  
    Utilidades.c, 50  
    Utilidades.h, 54, 58  
Adjacencia, 5  
    codVert, 5  
    Grafos.h, 36, 37  
    next, 5  
    peso, 5  
    visitado, 5  
adjacencias  
    Vertice, 7  
C:/Aulas/1º ano/2º semestre/Estruturas de Dados  
    Avançadas/Trabalho EDA Fase 2/Src/FuncoesLIB/Constantes.h, 9, 11  
C:/Aulas/1º ano/2º semestre/Estruturas de Dados  
    Avançadas/Trabalho EDA Fase 2/Src/FuncoesLIB/FuncoesProcura.c, 13  
C:/Aulas/1º ano/2º semestre/Estruturas de Dados  
    Avançadas/Trabalho EDA Fase 2/Src/FuncoesLIB/FuncoesProcura.h, 15, 17  
C:/Aulas/1º ano/2º semestre/Estruturas de Dados  
    Avançadas/Trabalho EDA Fase 2/Src/FuncoesLIB/GereGrafos.c, 20  
C:/Aulas/1º ano/2º semestre/Estruturas de Dados  
    Avançadas/Trabalho EDA Fase 2/Src/FuncoesLIB/GereGrafos.h, 25, 30  
C:/Aulas/1º ano/2º semestre/Estruturas de Dados  
    Avançadas/Trabalho EDA Fase 2/Src/FuncoesLIB/Grafos.h, 35, 36  
C:/Aulas/1º ano/2º semestre/Estruturas de Dados  
    Avançadas/Trabalho EDA Fase 2/Src/FuncoesLIB/Stack.c, 38  
C:/Aulas/1º ano/2º semestre/Estruturas de Dados  
    Avançadas/Trabalho EDA Fase 2/Src/FuncoesLIB/Stack.h, 41, 45  
C:/Aulas/1º ano/2º semestre/Estruturas de Dados  
    Avançadas/Trabalho EDA Fase 2/Src/FuncoesLIB/Utilidades.c, 49  
C:/Aulas/1º ano/2º semestre/Estruturas de Dados  
    Avançadas/Trabalho EDA Fase 2/Src/FuncoesLIB/Utilidades.h, 53, 57  
CalculoSomaEntreDoisVertices  
    FuncoesProcura.c, 13  
    FuncoesProcura.h, 18  
CalculoSomaEntreDoisVerticesz  
    FuncoesProcura.h, 16  
cod  
    Vertice, 7  
codVert  
    Adjacencia, 5  
Constantes.h, 11  
    ERRO\_AO\_ABRIR\_FICHEIRO, 10, 12  
    EXISTE\_ADJACENCIA, 10, 12  
    GRAFO\_NAO\_EXISTE, 10, 12  
    LISTA\_NAO\_EXISTE, 10, 12  
    MAX, 10, 12  
    MAX\_CARATERES, 10, 12  
    NAO\_EXISTE\_ADJACENCIA, 10, 12  
    PARAMETROS\_INVALIDOS, 10, 12  
    SUCESSO, 10, 12  
    VERTICE\_JA\_EXISTE, 10, 12  
    VERTICE\_NAO\_EXISTE, 10, 12  
CriaAdjacencia  
    GereGrafos.c, 21  
CriaGrafo  
    GereGrafos.h, 26, 31  
CriaProcura  
    FuncoesProcura.c, 13  
    GereGrafos.c, 21  
    GereGrafos.h, 26, 32  
CriaStackValor  
    FuncoesProcura.h, 15  
    Stack.c, 39  
    Stack.h, 42, 46  
CriaVertice  
    GereGrafos.c, 21  
    GereGrafos.h, 26, 32  
CriaVerticez  
    GereGrafos.h, 25  
DepthFirstSearchRec  
    FuncoesProcura.c, 14  
FuncoesProcura.c, 16, 18  
FuncoesProcura.h, 16, 18  
DepthFirstTraversal  
    FuncoesProcura.c, 14  
    FuncoesProcura.h, 16, 19  
ERRO\_AO\_ABRIR\_FICHEIRO  
    Constantes.h, 10, 12  
EXISTE\_ADJACENCIA  
    Constantes.h, 10, 12  
FuncoesProcura.c  
    CalculoSomaEntreDoisVertices, 13  
    DepthFirstSearchRec, 14  
    DepthFirstTraversal, 14  
    ProcessaVertice, 14  
FuncoesProcura.h, 17  
    CalculoSomaEntreDoisVertices, 18  
    CalculoSomaEntreDoisVerticesz, 16  
    DepthFirstSearchRec, 16, 18

- DepthFirstTraversal, [16](#), [19](#)
- ProcessaVertice, [17](#), [19](#)
- GereGrafos.c
  - CriaAdjacencia, [21](#)
  - CriaGrafo, [21](#)
  - CriaVertice, [21](#)
  - InsererAdjacencia, [22](#)
  - InsererAdjacenciaVertice, [22](#)
  - InsererVertice, [22](#)
  - LeFicheiroGrafo, [23](#)
  - LimpaGrafo, [23](#)
  - MostraGrafo, [23](#)
- GereGrafos.h, [30](#)
  - CriaAdjacencia, [26](#), [31](#)
  - CriaGrafo, [26](#), [32](#)
  - CriaVertice, [26](#), [32](#)
  - InsererAdjacencia, [27](#), [32](#)
  - InsererAdjacenciaVertice, [27](#), [33](#)
  - InsererVertice, [28](#), [33](#)
  - LeFicheiroGrafo, [28](#), [33](#)
  - LimpaGrafo, [28](#), [34](#)
  - MostraGrafo, [30](#), [34](#)
- Grafo, [6](#)
  - Grafos.h, [36](#), [37](#)
  - inicio, [6](#)
  - numVert, [6](#)
- GRAFO\_NAO\_EXISTE
  - Constantes.h, [10](#), [12](#)
- Grafos.h, [37](#)
  - Adjacencia, [36](#), [37](#)
  - Grafo, [36](#), [37](#)
  - Vertice, [36](#), [37](#)
- id
  - Stack, [6](#)
- inicio
  - Grafo, [6](#)
- InsererAdjacencia
  - GereGrafos.c, [22](#)
  - GereGrafos.h, [27](#), [32](#)
- InsererAdjacenciaVertice
  - GereGrafos.c, [22](#)
  - GereGrafos.h, [27](#), [33](#)
- InsererNaStack
  - Stack.c, [39](#)
  - Stack.h, [42](#), [47](#)
- InsererVertice
  - GereGrafos.c, [22](#)
  - GereGrafos.h, [28](#), [33](#)
- IsStackEmpty
  - Stack.c, [40](#)
  - Stack.h, [44](#), [47](#)
- LeFicheiroGrafo
  - GereGrafos.c, [23](#)
  - GereGrafos.h, [28](#), [33](#)
- LimpaGrafo
  - GereGrafos.c, [23](#)
- GereGrafos.h, [28](#), [34](#)
- LimpaListaAdjacencias
  - Utilidades.c, [50](#)
  - Utilidades.h, [54](#), [59](#)
- LISTA\_NAO\_EXISTE
  - Constantes.h, [10](#), [12](#)
- main
  - Main.c, [62](#)
- Main.c, [61](#)
  - main, [62](#)
- MAX
  - Constantes.h, [10](#), [12](#)
- MAX\_CARATERES
  - Constantes.h, [10](#), [12](#)
- MostraGrafo
  - GereGrafos.c, [23](#)
  - GereGrafos.h, [30](#), [34](#)
- NAO\_EXISTE\_ADJACENCIA
  - Constantes.h, [10](#), [12](#)
- next
  - Adjacencia, [5](#)
  - Stack, [6](#)
  - Vertice, [7](#)
- numAdj
  - Vertice, [7](#)
- numVert
  - Grafo, [6](#)
- OutputStack
  - Stack.c, [40](#)
  - Stack.h, [44](#), [47](#)
- PARAMETROS\_INVALIDOS
  - Constantes.h, [10](#), [12](#)
- peso
  - Adjacencia, [5](#)
- ProcessaVertice
  - FuncoesProcura.c, [14](#)
  - FuncoesProcura.h, [17](#), [19](#)
- ProcuraVerticeCod
  - Utilidades.c, [51](#)
  - Utilidades.h, [55](#), [59](#)
- RemoveAdjacencia
  - Utilidades.c, [51](#)
  - Utilidades.h, [55](#), [59](#)
- RemoveValorStack
  - Stack.c, [40](#)
  - Stack.h, [44](#), [48](#)
- RemoveVertice
  - Utilidades.c, [51](#)
  - Utilidades.h, [55](#), [60](#)
- Stack, [6](#)
  - id, [6](#)
  - next, [6](#)
  - Stack.h, [42](#), [46](#)
- Stack.c



- CriaStackValor, 39
- InserirNaStack, 39
- IsStackEmpty, 40
- OutputStack, 40
- RemoveValorStack, 40
- StackPeek, 40
- Stack.h, 45
  - CriaStackValor, 42, 46
  - InserirNaStack, 42, 47
  - IsStackEmpty, 44, 47
  - OutputStack, 44, 47
  - RemoveValorStack, 44, 48
  - Stack, 42, 46
  - StackPeek, 45, 48
- StackPeek
  - Stack.c, 40
  - Stack.h, 45, 48
- SUCESSO
  - Constantes.h, 10, 12
- Utilidades.c
  - AdicionaAdjacencia, 50
  - AdicionaVertice, 50
  - LimpaListaAdjacencias, 50
  - ProcuraVerticeCod, 51
  - RemoveAdjacencia, 51
  - RemoveVertice, 51
  - VerificaExisteAdjacencia, 52
  - VerificaExisteVertice, 52
- Utilidades.h, 57
  - AdicionaAdjacencia, 54, 58
  - AdicionaVertice, 54, 58
  - LimpaListaAdjacencias, 54, 59
  - ProcuraVerticeCod, 55, 59
  - RemoveAdjacencia, 55, 59
  - RemoveVertice, 55, 60
  - VerificaExisteAdjacencia, 56, 60
  - VerificaExisteVertice, 56, 61
- VerificaExisteAdjacencia
  - Utilidades.c, 52
  - Utilidades.h, 56, 60
- VerificaExisteVertice
  - Utilidades.c, 52
  - Utilidades.h, 56, 61
- Vertice, 7
  - adjacencias, 7
  - cod, 7
  - Grafos.h, 36, 37
  - next, 7
  - numAdj, 7
  - visitado, 7
- VERTICE\_JA\_EXISTE
  - Constantes.h, 10, 12
- VERTICE\_NAO\_EXISTE
  - Constantes.h, 10, 12
- visitado
  - Adjacencia, 5
  - Vertice, 7