

Trabalho de Grafos

1

Gerado por Doxygen 1.8.10

Sumário

1	Índice das Estruturas de Dados	1
1.1	Estruturas de Dados	1
2	Índice dos Arquivos	3
2.1	Lista de Arquivos	3
3	Estruturas	5
3.1	Referência da Estrutura Celula_priv	5
3.1.1	Descrição Detalhada	5
3.1.2	Campos	5
3.1.2.1	duracao	5
3.1.2.2	executada	5
3.1.2.3	id_externo	5
3.1.2.4	ini_min	6
3.1.2.5	nome	6
3.1.2.6	pre_req	6
3.1.2.7	reqs	6
3.2	Referência da Estrutura grafo_priv	6
3.2.1	Descrição Detalhada	7
3.2.2	Campos	7
3.2.2.1	origem	7
3.2.2.2	tabela	7
3.2.2.3	vert	7
3.3	Referência da Estrutura lista_aresta	7
3.3.1	Descrição Detalhada	8
3.3.2	Campos	8
3.3.2.1	destino	8
3.3.2.2	next	8
3.3.2.3	peso	8
3.4	Referência da Estrutura lista_origem	9
3.4.1	Descrição Detalhada	9
3.4.2	Campos	9

3.4.2.1	destino	9
3.4.2.2	next	9
3.5	Referência da Estrutura lista_vert	9
3.5.1	Descrição Detalhada	10
3.5.2	Campos	10
3.5.2.1	antecessores	10
3.5.2.2	id	10
3.5.2.3	id_externo	10
3.5.2.4	next	11
3.5.2.5	sucessores	11
3.6	Referência da Estrutura lista_vert_codigo	11
3.6.1	Descrição Detalhada	11
3.6.2	Campos	11
3.6.2.1	dado	11
3.6.2.2	id	12
3.6.2.3	next	12
4	Arquivos	13
4.1	Referência do Arquivo /home/rafael/ProjetoFinalMP/Fase2Interface/include/grafos.h	13
4.1.1	Descrição Detalhada	14
4.1.2	Definições dos tipos	14
4.1.2.1	Celula_priv_t	14
4.1.2.2	grafo_priv_t	15
4.1.2.3	resposta	15
4.1.3	Enumerações	15
4.1.3.1	resp	15
4.1.4	Funções	15
4.1.4.1	achar_celula(const grafo_priv_t *meu_grafos, int id_externo)	15
4.1.5	Descrição	15
4.1.6	Assertiva de entrada	15
4.1.6.1	achar_id(const grafo_priv_t *meu_grafos, int id_externo)	16
4.1.7	Descrição	16
4.1.8	Assertiva de entrada	16
4.1.8.1	cria_celula(int id_externo, int executada, int duracao, int ini_min, int pre_req, int *reqs, const char *nome)	17
4.1.9	Descrição	17
4.1.10	Assertiva de entrada	17
4.1.10.1	criaGrafoArq(char *nomeArq)	17
4.1.10.2	criar_grafos(void)	18
4.1.11	Descrição	18

4.1.12	Parâmetros	18
4.1.13	Assertiva de saída	18
4.1.13.1	deletar_grafo(grafo_priv_t *meu_grafo)	19
4.1.14	Descrição	19
4.1.15	Assertiva de saída	19
4.1.15.1	editar_celula(grafo_priv_t *meu_grafo, int ID)	20
4.1.15.2	eh_conexo(const grafo_priv_t *meu_grafo)	20
4.1.16	Descrição	21
4.1.17	Assertiva de entrada	22
4.1.17.1	existe_aresta(const grafo_priv_t *meu_grafo, int id_externo1, int id_externo2)	22
4.1.18	Descrição	22
4.1.19	Assertiva de entrada	23
4.1.19.1	existe_vert(const grafo_priv_t *meu_grafo, int id_externo)	23
4.1.20	Descrição	23
4.1.21	Assertiva de entrada	24
4.1.21.1	Grava_Arq(grafo_priv_t *meu_grafo, char *NomeArq)	24
4.1.21.2	Imprime_Tarefas(const grafo_priv_t *meu_grafo, int linha, int coluna)	24
4.1.21.3	inserir_aresta(grafo_priv_t *meu_grafo, int id_externo1, Celula_priv_t *celula2, int peso)	25
4.1.22	Descrição	25
4.1.23	Assertiva de entrada	25
4.1.23.1	inserir_vert(grafo_priv_t *meu_grafo, Celula_priv_t *celula)	26
4.1.24	Descrição	26
4.1.25	Assertiva de entrada	26
4.1.25.1	ja_feito(const grafo_priv_t *meu_grafo, int d)	27
4.1.25.2	Ler_Tarefas(grafo_priv_t *meu_grafo, Celula_priv_t *celula, const char *NomeArq)	28
4.1.25.3	maior_id(const grafo_priv_t *meu_grafo)	28
4.1.26	Descrição	28
4.1.27	Assertiva de entrada	28
4.1.27.1	menor_caminho(const grafo_priv_t *meu_grafo, int **dist)	29
4.1.28	Descrição	29
4.1.29	Assertiva de entrada	29
4.1.30	de saída	29
4.1.30.1	num_arestas(const grafo_priv_t *meu_grafo)	30
4.1.31	Descrição	30
4.1.32	Assertiva de entrada	30
4.1.32.1	num_vert(const grafo_priv_t *meu_grafo)	31
4.1.33	Descrição	31
4.1.34	Assertiva de entrada	31
4.1.34.1	remover_aresta(grafo_priv_t *meu_grafo, int id_externo1, int id_externo2)	31

4.1.35	Descrição	31
4.1.36	Assertiva de entrada	32
4.1.36.1	remover_vert(<code>grafo_priv_t *meu_grafo</code> , <code>int id_externo</code>)	32
4.1.37	Descrição	32
4.1.38	Assertiva de entrada	33
4.1.38.1	tempo_minimo(<code>const grafo_priv_t *meu_grafo</code> , <code>int id_fim</code>)	34
4.2	Referência do Arquivo <code>/home/rafael/ProjetoFinalIMP/Fase2Interface/include/grafo_priv.h</code>	34
4.2.1	Definições dos tipos	36
4.2.1.1	<code>Celula_priv_t</code>	36
4.2.1.2	<code>grafo_priv_t</code>	36
4.2.1.3	<code>lista_aresta_t</code>	36
4.2.1.4	<code>lista_origem_t</code>	36
4.2.1.5	<code>lista_vert_codigo_t</code>	36
4.2.1.6	<code>lista_vert_t</code>	36
4.2.2	Funções	36
4.2.2.1	existe_origem(<code>const grafo_priv_t *meu_grafo</code> , <code>int id_externo</code>)	36
4.2.2.2	inserir_origem(<code>grafo_priv_t *meu_grafo</code> , <code>Celula_priv_t *celula</code>)	37
4.2.2.3	remover_origem(<code>grafo_priv_t *meu_grafo</code> , <code>int id_externo</code>)	37
4.3	Referência do Arquivo <code>grafo.cpp</code>	38
4.3.1	Definições e macros	39
4.3.1.1	DEBUG	39
4.3.2	Funções	39
4.3.2.1	achar_celula(<code>const grafo_priv_t *meu_grafo</code> , <code>int id_externo</code>)	39
4.3.3	Descrição	39
4.3.4	Assertiva de entrada	40
4.3.4.1	achar_id(<code>const grafo_priv_t *meu_grafo</code> , <code>int id_externo</code>)	40
4.3.5	Descrição	40
4.3.6	Assertiva de entrada	41
4.3.6.1	cria_celula(<code>int id_externo</code> , <code>int executada</code> , <code>int duracao</code> , <code>int ini_min</code> , <code>int pre_req</code> , <code>int *reqs</code> , <code>const char *nome</code>)	41
4.3.7	Descrição	41
4.3.8	Assertiva de entrada	42
4.3.8.1	criaGrafoArq(<code>char *nomeArq</code>)	42
4.3.8.2	criar_grafo(<code>void</code>)	42
4.3.9	Descrição	43
4.3.10	Parâmetros	43
4.3.11	Assertiva de saída	43
4.3.11.1	deletar_grafo(<code>grafo_priv_t *meu_grafo</code>)	43
4.3.12	Descrição	43
4.3.13	Assertiva de saída	44

4.3.13.1	dfs(const grafo_priv_t *meu_grafo, lista_vert_t *atual, int *marc)	44
4.3.13.2	editar_celula(grafo_priv_t *meu_grafo, int ID)	45
4.3.13.3	eh_conexo(const grafo_priv_t *meu_grafo)	45
4.3.14	Descrição	45
4.3.15	Assertiva de entrada	45
4.3.15.1	existe_aresta(const grafo_priv_t *meu_grafo, int id_externo1, int id_externo2)	46
4.3.16	Descrição	46
4.3.17	Assertiva de entrada	46
4.3.17.1	existe_origem(const grafo_priv_t *meu_grafo, int id_externo)	47
4.3.17.2	existe_vert(const grafo_priv_t *meu_grafo, int id_externo)	48
4.3.18	Descrição	48
4.3.19	Assertiva de entrada	48
4.3.19.1	Grava_Arq(grafo_priv_t *meu_grafo, char *NomeArq)	49
4.3.19.2	Imprime_Tarefas(const grafo_priv_t *meu_grafo, int linha, int coluna)	49
4.3.19.3	inserir_aresta(grafo_priv_t *meu_grafo, int id_externo1, Celula_priv_t *celula2, int peso)	49
4.3.20	Descrição	49
4.3.21	Assertiva de entrada	50
4.3.21.1	inserir_origem(grafo_priv_t *meu_grafo, Celula_priv_t *celula)	51
4.3.21.2	inserir_vert(grafo_priv_t *meu_grafo, Celula_priv_t *celula)	51
4.3.22	Descrição	51
4.3.23	Assertiva de entrada	52
4.3.23.1	ja_feito(const grafo_priv_t *meu_grafo, int d)	52
4.3.23.2	maior_id(const grafo_priv_t *meu_grafo)	53
4.3.24	Descrição	53
4.3.25	Assertiva de entrada	53
4.3.25.1	menor_caminho(const grafo_priv_t *meu_grafo, int **dist)	54
4.3.26	Descrição	54
4.3.27	Assertiva de entrada	54
4.3.28	de saída	54
4.3.28.1	num_arestas(const grafo_priv_t *meu_grafo)	55
4.3.29	Descrição	55
4.3.30	Assertiva de entrada	55
4.3.30.1	num_vert(const grafo_priv_t *meu_grafo)	56
4.3.31	Descrição	56
4.3.32	Assertiva de entrada	56
4.3.32.1	remover_aresta(grafo_priv_t *meu_grafo, int id_externo1, int id_externo2)	56
4.3.33	Descrição	56
4.3.34	Assertiva de entrada	57
4.3.34.1	remover_origem(grafo_priv_t *meu_grafo, int id_externo)	57

4.3.34.2	<code>remover_vert(grafo_priv_t *meu_grafo, int id_externo)</code>	58
4.3.35	Descrição	58
4.3.36	Assertiva de entrada	58
4.3.36.1	<code>tempo_minimo(const grafo_priv_t *meu_grafo, int id_fim)</code>	59
4.4	Referência do Arquivo <code>interface.cpp</code>	60
4.4.1	Definições e macros	61
4.4.1.1	<code>NCURSES_CONST</code>	61
4.4.2	Funções	61
4.4.2.1	<code>interface_caminho_completo(const grafo_priv_t *meu_grafo)</code>	61
4.4.2.2	<code>interface_caminho_parcial(grafo_priv_t *meu_grafo)</code>	62
4.4.2.3	<code>interface_editar_tarefa(grafo_priv_t *meu_grafo)</code>	62
4.4.2.4	<code>interface_inserir_tarefa(grafo_priv_t *meu_grafo)</code>	63
4.4.2.5	<code>interface_remover_tarefa(grafo_priv_t *meu_grafo)</code>	63
4.4.2.6	<code>interface_vizualizar_determinada_tarefa(const grafo_priv_t *meu_grafo)</code>	64
4.4.2.7	<code>interface_vizualizar_tarefas(const grafo_priv_t *meu_grafo)</code>	65
4.4.2.8	<code>main()</code>	65
	Índice	67

Capítulo 1

Índice das Estruturas de Dados

1.1 Estruturas de Dados

Aqui estão as estruturas de dados, uniões e suas respectivas descrições:

Celula_priv	5
grafo_priv	6
lista_aresta	7
lista_origem	9
lista_vert	9
lista_vert_codigo	11

Capítulo 2

Índice dos Arquivos

2.1 Lista de Arquivos

Esta é a lista de todos os arquivos e suas respectivas descrições:

/home/rafael/ProjetoFinalMP/Fase2Interface/include/grafo.h	
Define funções usadas pelo usuario	13
/home/rafael/ProjetoFinalMP/Fase2Interface/include/grafo_priv.h	34
grafo.cpp	38
interface.cpp	60

Capítulo 3

Estruturas

3.1 Referência da Estrutura Celula_priv

```
#include <grafo_priv.h>
```

Campos de Dados

- int `id_externo`
- int `executada`
- int `duracao`
- int `ini_min`
- int `pre_req`
- int * `reqs`
- char `nome` [200]

3.1.1 Descrição Detalhada

Definição na linha 6 do arquivo `grafo_priv.h`.

3.1.2 Campos

3.1.2.1 int Celula_priv::duracao

Definição na linha 7 do arquivo `grafo_priv.h`.

Referenciado por `cria_celula()`, `criaGrafoArq()`, `editar_celula()`, `Grava_Arq()`, `Imprime_Tarefas()`, `interface_inserir_tarefa()` e `interface_vizualizar_determinada_tarefa()`.

3.1.2.2 int Celula_priv::executada

Definição na linha 7 do arquivo `grafo_priv.h`.

Referenciado por `cria_celula()`, `criaGrafoArq()`, `criar_grafo()`, `editar_celula()`, `Grava_Arq()`, `Imprime_Tarefas()`, `interface_inserir_tarefa()`, `interface_vizualizar_determinada_tarefa()` e `menor_caminho()`.

3.1.2.3 int Celula_priv::id_externo

Definição na linha 7 do arquivo `grafo_priv.h`.

Referenciado por `achar_celula()`, `achar_id()`, `cria_celula()`, `criaGrafoArq()`, `criar_grafo()`, `deletar_grafo()`, `editar_celula()`, `existe_vert()`, `Grava_Arq()`, `Imprime_Tarefas()`, `inserir_aresta()`, `inserir_origem()`, `inserir_vert()`, `interface_caminho_completo()`, `interface_inserir_tarefa()` e `interface_vizualizar_determinada_tarefa()`.

3.1.2.4 `int Celula_priv::ini_min`

Definição na linha 8 do arquivo `grafo_priv.h`.

Referenciado por `cria_celula()`, `criaGrafoArq()`, `criar_grafo()`, `editar_celula()`, `Grava_Arq()`, `Imprime_Tarefas()`, `inserir_vert()`, `interface_inserir_tarefa()`, `interface_vizualizar_determinada_tarefa()` e `menor_caminho()`.

3.1.2.5 `char Celula_priv::nome[200]`

Definição na linha 10 do arquivo `grafo_priv.h`.

Referenciado por `cria_celula()`, `criaGrafoArq()`, `criar_grafo()`, `editar_celula()`, `Grava_Arq()`, `Imprime_Tarefas()`, `inserir_origem()`, `inserir_vert()`, `interface_inserir_tarefa()` e `interface_vizualizar_determinada_tarefa()`.

3.1.2.6 `int Celula_priv::pre_req`

Definição na linha 8 do arquivo `grafo_priv.h`.

Referenciado por `cria_celula()`, `criaGrafoArq()`, `criar_grafo()`, `editar_celula()`, `Grava_Arq()`, `Imprime_Tarefas()`, `inserir_vert()`, `interface_inserir_tarefa()` e `interface_vizualizar_determinada_tarefa()`.

3.1.2.7 `int* Celula_priv::reqs`

Definição na linha 9 do arquivo `grafo_priv.h`.

Referenciado por `cria_celula()`, `criaGrafoArq()`, `editar_celula()`, `Grava_Arq()`, `Imprime_Tarefas()`, `interface_inserir_tarefa()` e `interface_vizualizar_determinada_tarefa()`.

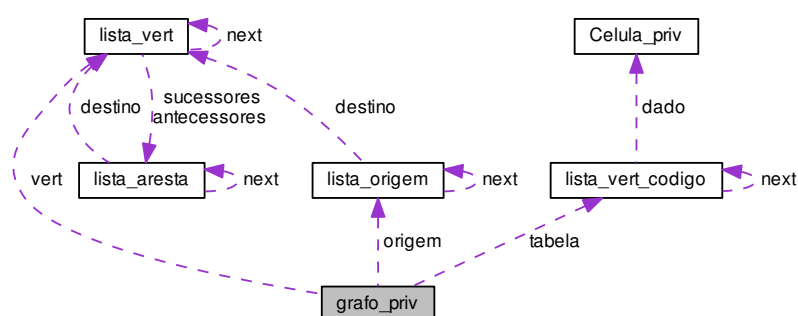
A documentação para esta estrutura foi gerada a partir do seguinte arquivo:

- `/home/rafael/ProjetoFinalIMP/Fase2Interface/include/grafo_priv.h`

3.2 Referência da Estrutura `grafo_priv`

```
#include <grafo_priv.h>
```

Diagrama de colaboração para `grafo_priv`:



Campos de Dados

- [lista_vert_codigo_t * tabela](#)
- [lista_vert_t * vert](#)
- [lista_origem_t * origem](#)

3.2.1 Descrição Detalhada

Definição na linha 41 do arquivo grafo_priv.h.

3.2.2 Campos

3.2.2.1 lista_origem_t* grafo_priv::origem

Definição na linha 44 do arquivo grafo_priv.h.

Referenciado por `criar_grafo()`, `eh_conexo()`, `existe_origem()`, `inserir_origem()` e `remover_origem()`.

3.2.2.2 lista_vert_codigo_t* grafo_priv::tabela

Definição na linha 42 do arquivo grafo_priv.h.

Referenciado por `achar_celula()`, `achar_id()`, `criar_grafo()`, `deletar_grafo()`, `existe_vert()`, `Grava_Arq()`, `Imprime_↔` `Tarefas()`, `inserir_vert()`, `interface_caminho_completo()`, `ja_feito()`, `maior_id()` e `remover_vert()`.

3.2.2.3 lista_vert_t* grafo_priv::vert

Definição na linha 43 do arquivo grafo_priv.h.

Referenciado por `criar_grafo()`, `existe_aresta()`, `inserir_aresta()`, `inserir_origem()`, `inserir_vert()`, `menor_caminho()`, `num_arestas()`, `num_vert()`, `remover_aresta()` e `remover_vert()`.

A documentação para esta estrutura foi gerada a partir do seguinte arquivo:

- [/home/rafael/ProjetoFinalMP/Fase2Interface/include/grafo_priv.h](#)

3.3 Referência da Estrutura lista_aresta

```
#include <grafo_priv.h>
```

Diagrama de colaboração para lista_aresta:



Campos de Dados

- struct `lista_vert` * `destino`
- int `peso`
- struct `lista_aresta` * `next`

3.3.1 Descrição Detalhada

Definição na linha 22 do arquivo `grafo_priv.h`.

3.3.2 Campos

3.3.2.1 struct `lista_vert`* `lista_aresta::destino`

Definição na linha 23 do arquivo `grafo_priv.h`.

Referenciado por `dfs()`, `existe_aresta()`, `inserir_aresta()`, `menor_caminho()` e `remover_vert()`.

3.3.2.2 struct `lista_aresta`* `lista_aresta::next`

Definição na linha 25 do arquivo `grafo_priv.h`.

Referenciado por `dfs()`, `existe_aresta()`, `inserir_aresta()`, `menor_caminho()`, `num_arestas()` e `remover_aresta()`.

3.3.2.3 int `lista_aresta::peso`

Definição na linha 24 do arquivo `grafo_priv.h`.

Referenciado por `inserir_aresta()` e `menor_caminho()`.

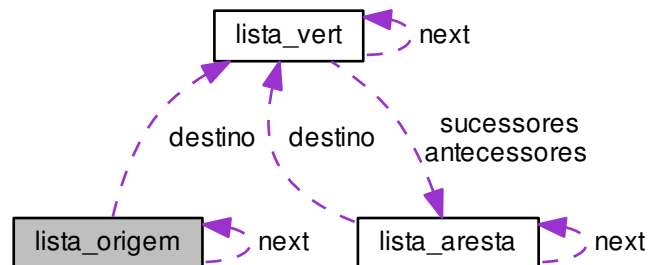
A documentação para esta estrutura foi gerada a partir do seguinte arquivo:

- `/home/rafael/ProjetoFinalMP/Fase2Interface/include/grafo_priv.h`

3.4 Referência da Estrutura lista_origem

```
#include <grafo_priv.h>
```

Diagrama de colaboração para lista_origem:



Campos de Dados

- `lista_vert_t * destino`
- `struct lista_origem * next`

3.4.1 Descrição Detalhada

Definição na linha 36 do arquivo `grafo_priv.h`.

3.4.2 Campos

3.4.2.1 `lista_vert_t * lista_origem::destino`

Definição na linha 37 do arquivo `grafo_priv.h`.

Referenciado por `eh_conexo()`, `existe_origem()`, `inserir_origem()` e `remover_origem()`.

3.4.2.2 `struct lista_origem * lista_origem::next`

Definição na linha 38 do arquivo `grafo_priv.h`.

Referenciado por `eh_conexo()`, `existe_origem()`, `inserir_origem()` e `remover_origem()`.

A documentação para esta estrutura foi gerada a partir do seguinte arquivo:

- `/home/rafael/ProjetoFinalMP/Fase2Interface/include/grafo_priv.h`

3.5 Referência da Estrutura lista_vert

```
#include <grafo_priv.h>
```

Diagrama de colaboração para lista_vert:



Campos de Dados

- int `id`
- int `id_externo`
- `lista_aresta_t` * `antecessores`
- `lista_aresta_t` * `sucessores`
- struct `lista_vert` * `next`

3.5.1 Descrição Detalhada

Definição na linha 28 do arquivo `grafo_priv.h`.

3.5.2 Campos

3.5.2.1 `lista_aresta_t`* `lista_vert::antecessores`

Definição na linha 31 do arquivo `grafo_priv.h`.

Referenciado por `inserir_vert()` e `remover_aresta()`.

3.5.2.2 `int lista_vert::id`

Definição na linha 29 do arquivo `grafo_priv.h`.

Referenciado por `dfs()`, `eh_conexo()`, `inserir_origem()`, `inserir_vert()`, `menor_caminho()`, `remover_aresta()`, `remover_origem()` e `remover_vert()`.

3.5.2.3 `int lista_vert::id_externo`

Definição na linha 30 do arquivo `grafo_priv.h`.

Referenciado por `existe_aresta()`, `existe_origem()`, `inserir_vert()`, `menor_caminho()` e `remover_vert()`.

3.5.2.4 struct lista_vert* lista_vert::next

Definição na linha 33 do arquivo grafo_priv.h.

Referenciado por existe_aresta(), inserir_aresta(), inserir_origem(), inserir_vert(), menor_caminho(), num_arestas(), num_vert(), remover_aresta() e remover_vert().

3.5.2.5 lista_aresta_t* lista_vert::sucessores

Definição na linha 32 do arquivo grafo_priv.h.

Referenciado por dfs(), existe_aresta(), inserir_aresta(), inserir_vert(), menor_caminho(), num_arestas() e remover_aresta().

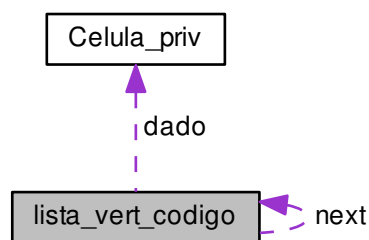
A documentação para esta estrutura foi gerada a partir do seguinte arquivo:

- [/home/rafael/ProjetoFinalMP/Fase2Interface/include/grafo_priv.h](#)

3.6 Referência da Estrutura lista_vert_codigo

```
#include <grafo_priv.h>
```

Diagrama de colaboração para lista_vert_codigo:



Campos de Dados

- [Celula_priv_t](#) dado
- int id
- struct [lista_vert_codigo](#) * next

3.6.1 Descrição Detalhada

Definição na linha 16 do arquivo grafo_priv.h.

3.6.2 Campos

3.6.2.1 Celula_priv_t lista_vert_codigo::dado

Definição na linha 17 do arquivo grafo_priv.h.

Referenciado por `achar_celula()`, `achar_id()`, `deletar_grafo()`, `existe_vert()`, `Grava_Arq()`, `Imprime_Tarefas()` e `interface_caminho_completo()`.

3.6.2.2 `int lista_vert_codigo::id`

Definição na linha 18 do arquivo `grafo_priv.h`.

Referenciado por `achar_id()`, `inserir_vert()`, `ja_feito()`, `maior_id()` e `remover_vert()`.

3.6.2.3 `struct lista_vert_codigo* lista_vert_codigo::next`

Definição na linha 19 do arquivo `grafo_priv.h`.

Referenciado por `achar_celula()`, `achar_id()`, `existe_vert()`, `Grava_Arq()`, `Imprime_Tarefas()`, `inserir_vert()`, `interface_caminho_completo()`, `ja_feito()`, `maior_id()` e `remover_vert()`.

A documentação para esta estrutura foi gerada a partir do seguinte arquivo:

- `/home/rafael/ProjetoFinalMP/Fase2Interface/include/grafo_priv.h`

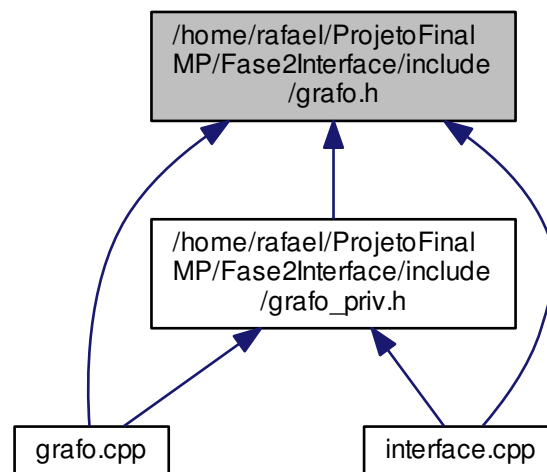
Capítulo 4

Arquivos

4.1 Referência do Arquivo /home/rafael/ProjetoFinalMP/Fase2Interface/include/graf.h

Define funções usadas pelo usuario.

Este grafo mostra quais arquivos estão direta ou indiretamente relacionados com este arquivo:



Definições de Tipos

- typedef enum `resp` `resposta`
- typedef struct `grafo_priv` `grafo_priv_t`
- typedef struct `Celula_priv` `Celula_priv_t`

Enumerações

- enum `resp` { `FALSE_T`, `TRUE_T` }

Funções

- `grafo_priv_t * criar_grafo` (void)
Cria grafo.
- `grafo_priv_t * deletar_grafo` (`grafo_priv_t *meu_grafo`)
Deleta grafo.
- `resposta existe_vert` (const `grafo_priv_t *meu_grafo`, int id_externo)
Existe vértice nome?
- `Celula_priv_t * cria_celula` (int id_externo, int executada, int duracao, int ini_min, int pre_req, int *reqs, const char *nome)
Criar nova célula.
- `resposta existe_aresta` (const `grafo_priv_t *meu_grafo`, int id_externo1, int id_externo2)
Existe aresta?
- int `achar_id` (const `grafo_priv_t *meu_grafo`, int id_externo)
Achar id.
- `Celula_priv_t * achar_celula` (const `grafo_priv_t *meu_grafo`, int id_externo)
Achar célula.
- void `inserir_vert` (`grafo_priv_t *meu_grafo`, `Celula_priv_t *celula`)
Inserir vértice.
- void `inserir_aresta` (`grafo_priv_t *meu_grafo`, int id_externo1, `Celula_priv_t *celula2`, int peso)
Inserir aresta.
- void `remover_vert` (`grafo_priv_t *meu_grafo`, int id_externo)
Remover vértice.
- void `remover_aresta` (`grafo_priv_t *meu_grafo`, int id_externo1, int id_externo2)
Remover aresta.
- int `maior_id` (const `grafo_priv_t *meu_grafo`)
Maior id.
- int `num_vert` (const `grafo_priv_t *meu_grafo`)
Número de vértices.
- int `num_arestas` (const `grafo_priv_t *meu_grafo`)
Número de arestas.
- int `menor_caminho` (const `grafo_priv_t *meu_grafo`, int **dist)
Menor caminho.
- `resposta eh_conexo` (const `grafo_priv_t *meu_grafo`)
É conexo?
- int `tempo_minimo` (const `grafo_priv_t *meu_grafo`, int id_fim)
- void `ja_feito` (const `grafo_priv_t *meu_grafo`, int d)
- void `Ler_Tarefas` (`grafo_priv_t *meu_grafo`, `Celula_priv_t *celula`, const char *NomeArq)
- void `editar_celula` (`grafo_priv_t *meu_grafo`, int ID)
- void `Imprime_Tarefas` (const `grafo_priv_t *meu_grafo`, int linha, int coluna)
- void `Grava_Arq` (`grafo_priv_t *meu_grafo`, char *NomeArq)
- `grafo_priv_t * criaGrafoArq` (char *nomeArq)

4.1.1 Descrição Detalhada

Define funções usadas pelo usuario.

4.1.2 Definições dos tipos

4.1.2.1 typedef struct Celula_priv Celula_priv_t

Definição na linha 19 do arquivo grafo.h.

4.1.2.2 typedef struct grafo_priv grafo_priv_t

Definição na linha 17 do arquivo grafos.h.

4.1.2.3 typedef enum resp resposta

4.1.3 Enumerações

4.1.3.1 enum resp

Valores de enumerações

FALSE_T

TRUE_T

Definição na linha 9 do arquivo grafos.h.

4.1.4 Funções

4.1.4.1 Celula_priv_t* achar_celula (const grafo_priv_t * meu_grafos, int id_externo)

Achar célula.

4.1.5 Descrição

Encontra a célula e os dados dados pelo usuário ao vértice de identificador id

Parâmetros

<i>meu_grafos</i>	- Deve ser passado um grafos inicializado
<i>id</i>	- Deve ser passado um inteiro não negativo válido, ou seja, que represente um vértice

Retorna

Retorna um ponteiro para uma célula. Caso não se ache um vértice com id, retorna-se NULL.

4.1.6 Assertiva de entrada

O grafos já deve ter sido inicializado por [criar_grafos\(\)](#), se não for o programa pode ser interrompido. $0 \leq id$ Há um vértice representado por id

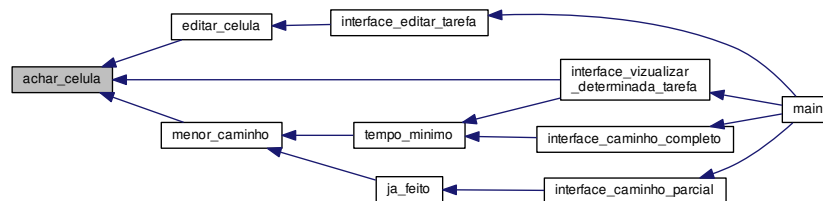
Assertivas de saída Se existir um vértice id, a resposta será um ponteiro != NULL, que aponta para onde está armazenado a célula. Se não existir, retorna-se NULL Assertivas de saída O retorno um ponteiro para uma célula. O conteúdo do grafos não será modificado.

Definição na linha 117 do arquivo grafos.cpp.

Referências lista_vert_codigo::dado, Celula_priv::id_externo, lista_vert_codigo::next e grafos_priv::tabela.

Referenciado por editar_celula(), interface_vizualizar_determinada_tarefa() e menor_caminho().

Este é o diagrama das funções que utilizam esta função:



4.1.6.1 `int achar_id (const grafo_priv_t * meu_grafo, int id_externo)`

Achar id.

4.1.7 Descrição

Todo vértice armazenado no grafo possui um nome, e um identificador (`id_externo`). Essa função acha esse identificador

Parâmetros

<i>meu_grafo</i>	- Deve ser passado um grafo inicializado
<i>id_externo</i>	- um inteiro que representa o identificador da tarefa.

Retorna

Retorna um inteiro `id`. Caso não se ache o vértice `id`, esse inteiro será -1, caso contrário será o valor do identificador da tarefa.

4.1.8 Assertiva de entrada

O grafo já deve ter sido inicializado por [criar_grafo\(\)](#), se não for o programa pode ser interrompido.

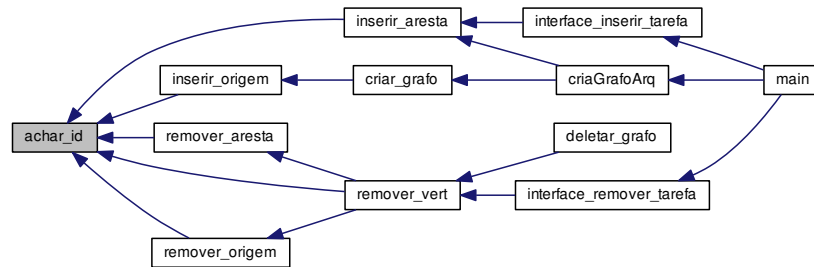
Já foi checado que `existe_vert(meu_grafo, nome) == TRUE` Assertivas de saída Se existir um vértice `nome`, $0 \leq id \leq \max(int)$, sendo que $\max(int)$ é o maior valor que pode ser representado num inteiro Se não existir `id = -1` Assertivas de saída O retorno será TRUE ou FALSE O conteúdo do grafo não será modificado.

Definição na linha 107 do arquivo `grafo.cpp`.

Referências `lista_vert_codigo::dado`, `lista_vert_codigo::id`, `Celula_priv::id_externo`, `lista_vert_codigo::next` e `grafo_priv::tabela`.

Referenciado por `inserir_aresta()`, `inserir_origem()`, `remover_aresta()`, `remover_origem()` e `remover_vert()`.

Este é o diagrama das funções que utilizam esta função:



4.1.8.1 Celula_priv_t* cria_celula (int id_externo, int executada, int duracao, int ini_min, int pre_req, int * reqs, const char * nome)

Criar nova célula.

4.1.9 Descrição

Cria uma nova célula com os os dados inseridos

Parâmetros

<i>meu_grafo</i>	- Deve ser passado um ponteiro para um grafo inicializado
<i>id_externo</i>	- Deve ser um inteiro que é o ID da tarefa
<i>executada</i>	- Deve ser um inteiro que indica se a tarefa ja foi ou nao executada.
<i>duracao</i>	- Deve ser um inteiro que representa a duracao da tarefa
<i>ini_min</i>	- Deve ser um inteiro que representa o tempo minimo para iniciar a tarefa.
<i>pre_req</i>	- Deve ser um inteiro que representa a quantidade de pre requisitos que a tarefa possui.
<i>reqs</i>	- Deve ser um ponteiro pra inteiro contendo todos os IDs dos pre requisitos da tarefa.
<i>nome</i>	- Deve ser passado uma string de até 100 caracteres, mais o caracter zero terminal que será o nome da célula.

Retorna

Retorna uma célula com os dados que foram inseridos na chamada

4.1.10 Assertiva de entrada

Os numeros devem ser inteiros maiores que 0. A string de nome deve ser de ate 100 caracteres. Assertivas de saída O retorno será uma célula contendo os dados que foram passados na entrada

Definição na linha 43 do arquivo grafo.cpp.

Referências Celula_priv::duracao, Celula_priv::executada, Celula_priv::id_externo, Celula_priv::ini_min, Celula_priv::nome, Celula_priv::pre_req e Celula_priv::reqs.

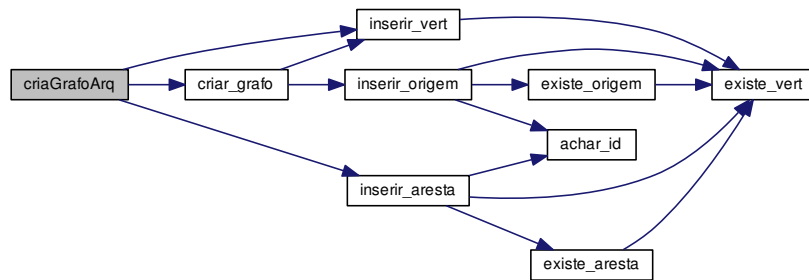
4.1.10.1 grafo_priv_t* criaGrafoArq (char * nomeArq)

Definição na linha 779 do arquivo grafo.cpp.

Referências criar_grafo(), Celula_priv::duracao, Celula_priv::executada, Celula_priv::id_externo, Celula_priv::ini_min, inserir_aresta(), inserir_vert(), Celula_priv::nome, Celula_priv::pre_req e Celula_priv::reqs.

Referenciado por main().

Este é o diagrama das funções utilizadas por esta função:



Este é o diagrama das funções que utilizam esta função:



4.1.10.2 grafo_priv_t*: criar_grafo (void)

Cria grafo.

4.1.11 Descrição

Aloca a memória necessária e inicializa um grafo

4.1.12 Parâmetros

Não ha parâmetros, a alocação e inicializam não dependem de nenhum parâmetro do usuário

Retorna

Se retorna um ponteiro para a grafo criado

4.1.13 Assertiva de saída

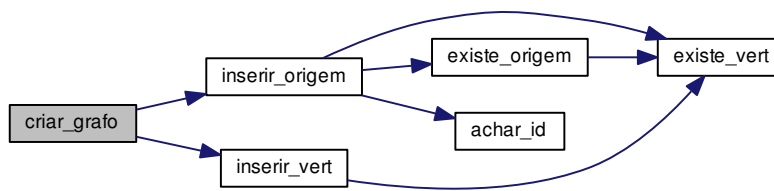
O grafo gerado é consistente e não possui nenhum vértice, origem ou aresta.

Definição na linha 12 do arquivo grafo.cpp.

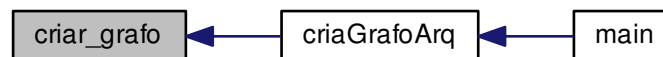
Referências Celula_priv::executada, Celula_priv::id_externo, Celula_priv::ini_min, inserir_origem(), inserir_vert(), Celula_priv::nome, grafo_priv::origem, Celula_priv::pre_req, grafo_priv::tabela e grafo_priv::vert.

Referenciado por `criaGrafoArq()`.

Este é o diagrama das funções utilizadas por esta função:



Este é o diagrama das funções que utilizam esta função:



4.1.13.1 `grafo_priv_t* deletar_grafo (grafo_priv_t * meu_grafo)`

Deleta grafo.

4.1.14 Descrição

Desaloca toda a memória utilizada pelo grafo

Parâmetros

<code>meu_grafo</code>	- Deve ser passado um ponteiro para um grafo inicializado
------------------------	---

Retorna

Retorna um ponteiro para o grafo, que será NULL. O valor de retorno é muito importante, uma vez que se ele não for utilizado o grafo do usuário apontará para um endereço não alocado e qualquer tentativa de utilizá-lo poderá gerar erros no sistema. Caso o grafo passado não tenha sido inicializado, o programa poderá parar a execução

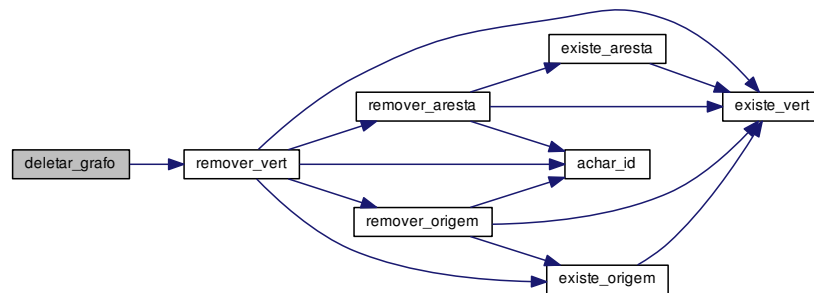
4.1.15 Assertiva de saída

O grafo já deve ter sido inicializado por `criar_grafo()` Assertiva de saída Se retornará um ponteiro para o mesmo grafo passado, após a deleção que será NULL.

Definição na linha 32 do arquivo `grafo.cpp`.

Referências `lista_vert_codigo::dado`, `Celula_priv::id_externo`, `remover_vert()` e `grafo_priv::tabela`.

Este é o diagrama das funções utilizadas por esta função:



4.1.15.1 void editar_celula (grafo_priv_t * meu_grafo, int ID)

Definição na linha 676 do arquivo grafo.cpp.

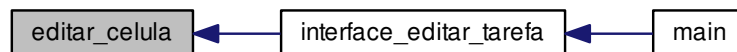
Referências achar_celula(), Celula_priv::duracao, Celula_priv::executada, Celula_priv::id_externo, Celula_priv::ini_min, Celula_priv::nome, Celula_priv::pre_req e Celula_priv::reqs.

Referenciado por interface_editar_tarefa().

Este é o diagrama das funções utilizadas por esta função:



Este é o diagrama das funções que utilizam esta função:



4.1.15.2 resposta eh_conexo (const grafo_priv_t * meu_grafo)

É conexo?

4.1.16 Descrição

Verifica se partindo da origem podem-se alcançar todos os vértices do grafo. SE for possível retorna TRUE, caso contrário FALSE.

Parâmetros

<i>meu_grafo</i>	- Deve ser passado um grafo inicializado
------------------	--

Retorna

Se for possível alcançar todos os vértices partindo das origens retorna TRUE, caso contrário FALSE. Um grafo sem vértices é conexo.

4.1.17 Assertiva de entrada

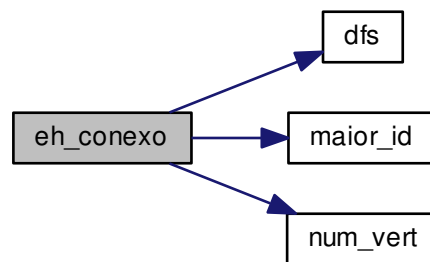
O grafo já deve ter sido inicializado por [criar_grafo\(\)](#), se não for o programa pode ser interrompido.

Assertivas de saída O grafo não será modificado.

Definição na linha 593 do arquivo grafo.cpp.

Referências `lista_origem::destino`, `dfs()`, `FALSE_T`, `lista_vert::id`, `maior_id()`, `lista_origem::next`, `num_vert()`, `grafo` ← `_priv::origem` e `TRUE_T`.

Este é o diagrama das funções utilizadas por esta função:

**4.1.17.1 resposta existe_aresta (const grafo_priv_t * meu_grafo, int id_externo1, int id_externo2)**

Existe aresta?

4.1.18 Descrição

Verifica a existência de um vértice dado

Parâmetros

<i>meu_grafo</i>	- Deve ser passado um grafo inicializado
<i>id_externo1</i>	
<i>id_externo2</i>	- Devem ser passadas dois IDs de tarefa, que são números inteiros.

Retorna

Retorna uma resposta, que será TRUE caso exista a aresta, e FALSE caso não exista.

4.1.19 Assertiva de entrada

O grafo já deve ter sido inicializado por `criar_grafo()`, se não for o programa pode ser interrompido.

Assertivas de saída O retorno será TRUE ou FALSE O conteúdo do grafo não será modificado.

Definição na linha 87 do arquivo `grafos.cpp`.

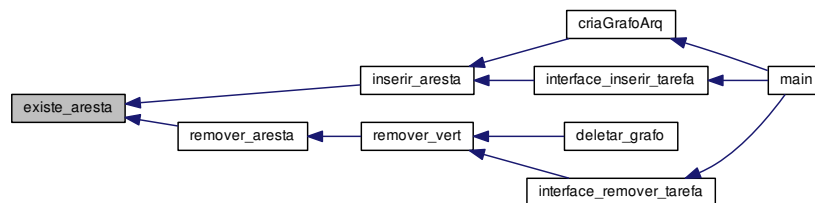
Referências `lista_aresta::destino`, `existe_vert()`, `FALSE_T`, `lista_vert::id_externo`, `lista_aresta::next`, `lista_vert::next`, `lista_vert::sucessores`, `TRUE_T` e `grafo_priv::vert`.

Referenciado por `inserir_aresta()` e `remover_aresta()`.

Este é o diagrama das funções utilizadas por esta função:



Este é o diagrama das funções que utilizam esta função:



4.1.19.1 resposta `existe_vert (const grafo_priv_t * meu_grafo, int id_externo)`

Existe vértice nome?

4.1.20 Descrição

Verifica a existência de um vértice dado

Parâmetros

<i>meu_grafo</i>	- Deve ser passado um ponteiro para um grafo inicializado
<i>nome</i>	- Deve ser passado o número de identificação da tarefa.

Retorna

Retorna uma resposta, que será TRUE caso exista o vértice, e FALSE caso não exista.

4.1.21 Assertiva de entrada

O grafo já deve ter sido inicializado por `criar_grafo()`, se não for o programa pode ser interrompido.

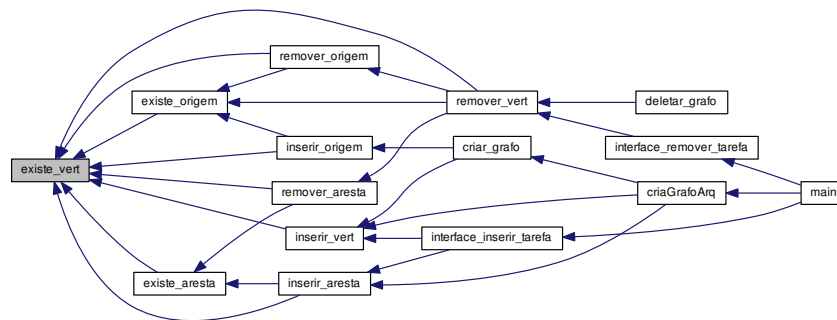
Assertivas de saída O retorno será TRUE ou FALSE O conteúdo do grafo não será modificado

Definição na linha 64 do arquivo grafo.cpp.

Referências `lista_vert_codigo::dado`, `FALSE_T`, `Celula_priv::id_externo`, `lista_vert_codigo::next`, `grafo_priv::tabela` e `TRUE_T`.

Referenciado por `existe_aresta()`, `existe_origem()`, `inserir_aresta()`, `inserir_origem()`, `inserir_vert()`, `remover_aresta()`, `remover_origem()` e `remover_vert()`.

Este é o diagrama das funções que utilizam esta função:



4.1.21.1 void Grava_Arq (grafo_priv_t * meu_grafo, char * NomeArq)

Definição na linha 649 do arquivo grafo.cpp.

Referências `lista_vert_codigo::dado`, `Celula_priv::duracao`, `Celula_priv::executada`, `Celula_priv::id_externo`, `Celula_priv::ini_min`, `lista_vert_codigo::next`, `Celula_priv::nome`, `Celula_priv::pre_req`, `Celula_priv::reqs` e `grafo_priv::tabela`.

Referenciado por `main()`.

Este é o diagrama das funções que utilizam esta função:



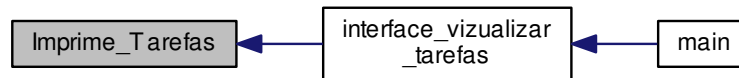
4.1.21.2 void Imprime_Tarefas (const grafo_priv_t * meu_grafo, int linha, int coluna)

Definição na linha 628 do arquivo grafo.cpp.

Referências `lista_vert_codigo::dado`, `Celula_priv::duracao`, `Celula_priv::executada`, `Celula_priv::id_externo`, `Celula_priv::ini_min`, `lista_vert_codigo::next`, `Celula_priv::nome`, `Celula_priv::pre_req`, `Celula_priv::reqs` e `grafo_priv::tabela`.

Referenciado por `interface_vizualizar_tarefas()`.

Este é o diagrama das funções que utilizam esta função:



4.1.21.3 void inserir_aresta (grafo_priv_t * meu_grafo, int id_externo1, Celula_priv_t * celula2, int peso)

Inserir aresta.

4.1.22 Descrição

Caso `id_externo1` e `celula2` sejam vértices, `peso > 0` e ainda não haja uma aresta entre eles, insere-se uma.

Se não existir algum dos vértices ou já exista uma aresta ou se `peso <= 0`, e caso a opção de DEBUG seja ativada, (macro DEBUG é igual à 1), será enviada uma mensagem à saída padrão de erro. Independente de DEBUG, a aresta não será adicionada.

Assim se for desejado adicionar uma aresta que não se sabe se já existem os vértices ou não, use `inserir_vert`, para cada um e depois `inserir_aresta`.

Parâmetros

<i>meu_grafo</i>	- Deve ser passado um grafo inicializado
<i>id_externo1</i>	- Deve ser passado o id da tarefa de pre requisito
<i>celula2</i>	- Deve ser passado um ponteiro para a célula destino
<i>peso</i>	- Deve ser um número real maior que zero

Retorna

Retorna o grafo modificado por referência, ou seja, o grafo passado será modificado sem a necessidade de receber um valor de retorno.

4.1.23 Assertiva de entrada

O grafo já deve ter sido inicializado por `criar_grafo()`, se não for o programa pode ser interrompido. `peso > 0`

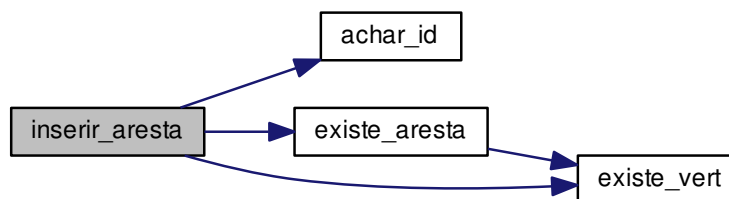
Assertivas de saída O grafo retornado foi adicionado de uma aresta, não se assegura sua conexividade. O conteúdo das células passadas e do número inteiro não serão modificados

Definição na linha 250 do arquivo `grafo.cpp`.

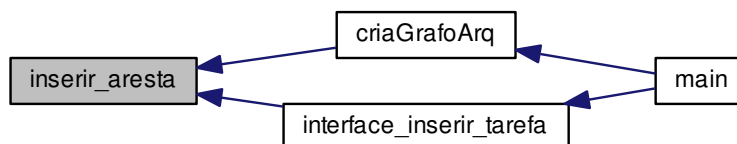
Referências `achar_id()`, `lista_aresta::destino`, `existe_aresta()`, `existe_vert()`, `FALSE_T`, `Celula_priv::id_externo`, `lista_aresta::next`, `lista_vert::next`, `lista_aresta::peso`, `lista_vert::sucessores`, `TRUE_T` e `grafo_priv::vert`.

Referenciado por `criaGrafoArq()` e `interface_inserir_tarefa()`.

Este é o diagrama das funções utilizadas por esta função:



Este é o diagrama das funções que utilizam esta função:



4.1.23.1 void inserir_vert (grafo_priv_t * meu_grafo, Celula_priv_t * celula)

Inserir vértice.

4.1.24 Descrição

Caso não existir um vértice nome, insere-se um.

Se já existir o vertice, e caso a opção de DEBUG seja ativada, (macro DEBUG é igual à 1), será enviada uma mensagem à saída padrão de erro. Independente de DEBUG, o vértice não será adicionado.

Para isso guarda-se uma cópia da célula dada pelo usuário

Parâmetros

<i>meu_grafo</i>	- Deve ser passado um grafo inicializado
<i>celula</i>	- Deve ser passado uma célula com todas as informações que nela são necessárias existir.

Retorna

Retorna a grafo modificado por referência, ou seja, o grafo passado será modificado sem a necessidade de receber um valor de retorno.

4.1.25 Assertiva de entrada

O grafo já deve ter sido inicializado por [criar_grafo\(\)](#), se não for o programa pode ser interrompido.

Assertivas de saída O grafo retornado por referência foi adicionado de um vértice, não se assegura sua conexividade. O conteúdo da célula passada não será modificado, porém será copiado

Definição na linha 128 do arquivo grafos.cpp.

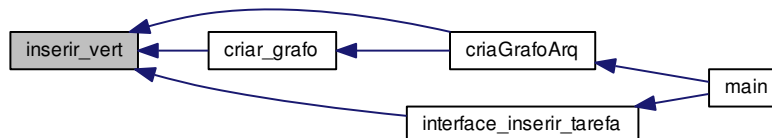
Referências lista_vert::antecessores, existe_vert(), FALSE_T, lista_vert_codigo::id, lista_vert::id, Celula_priv::id↔_externo, lista_vert::id_externo, Celula_priv::ini_min, lista_vert_codigo::next, lista_vert::next, Celula_priv::nome, Celula_priv::pre_req, lista_vert::sucessores, grafo_priv::tabela e grafo_priv::vert.

Referenciado por criaGrafoArq(), criar_grafo() e interface_inserir_tarefa().

Este é o diagrama das funções utilizadas por esta função:



Este é o diagrama das funções que utilizam esta função:



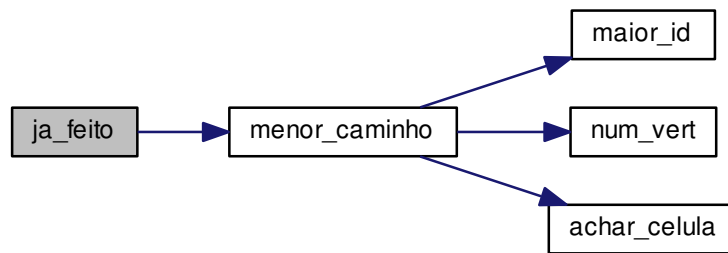
4.1.25.1 void ja_feito (const grafo_priv_t * meu_grafo, int d)

Definição na linha 747 do arquivo grafos.cpp.

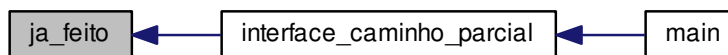
Referências lista_vert_codigo::id, menor_caminho(), lista_vert_codigo::next e grafo_priv::tabela.

Referenciado por interface_caminho_parcial().

Este é o diagrama das funções utilizadas por esta função:



Este é o diagrama das funções que utilizam esta função:



4.1.25.2 void Ler_Tarefas (grafo_priv_t * meu_grafo, Celula_priv_t * celula, const char * NomeArq)

4.1.25.3 int maior_id (const grafo_priv_t * meu_grafo)

Maior id.

4.1.26 Descrição

Retorna o valor do maior identificador (id) usado.

Se não foi utilizado nenhum id, ou seja, se não há nenhum vértice retorna -1

Parâmetros

<i>meu_grafo</i>	- Deve ser passado um grafo inicializado
------------------	--

Retorna

Será retornado um inteiro.

4.1.27 Assertiva de entrada

O grafo já deve ter sido inicializado por [criar_grafo\(\)](#), se não for o programa pode ser interrompido.

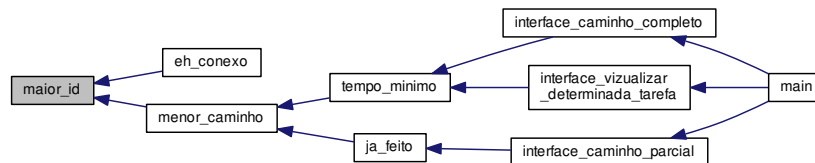
Assertivas de saída O grafo não será modificado. Se não há nenhum vértice (`num_vert(meu_grafo) == 0`), retorna -1 Caso contrário, o valor de retorno ≥ 0

Definição na linha 490 do arquivo grafos.cpp.

Referências lista_vert_codigo::id, lista_vert_codigo::next e grafos_priv::tabela.

Referenciado por eh_conexo() e menor_caminho().

Este é o diagrama das funções que utilizam esta função:



4.1.27.1 int menor_caminho (const grafos_priv_t * meu_grafos, int ** dist)

Menor caminho.

4.1.28 Descrição

Retorna o tamanho de menor caminho de inicio até fim.

Caso não haja um caminho entre os dois, ou um deles não for vértice retorna -1.

Parâmetros

<i>meu_grafos</i>	- Deve ser passado um grafos inicializado
<i>inicio</i>	
<i>fim</i>	- Devem ser passadas duas string de até 100 caracteres, sem contar o caracter zero terminal.

Retorna

Será retornado um número real.

4.1.29 Assertiva de entrada

O grafos já deve ter sido inicializado por [criar_grafos\(\)](#), se não for o programa pode ser interrompido.

4.1.30 de saída

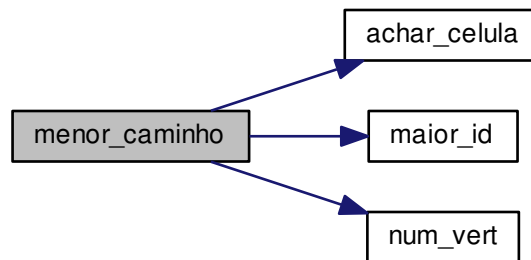
O grafos não será modificado. Se não houver caminho retorna -1. Caso contrário, o valor de retorno é maior ou igual à 0, e é o menor caminho de inicio até fim O menor caminho de um vértice à ele mesmo é 0.

Definição na linha 524 do arquivo grafos.cpp.

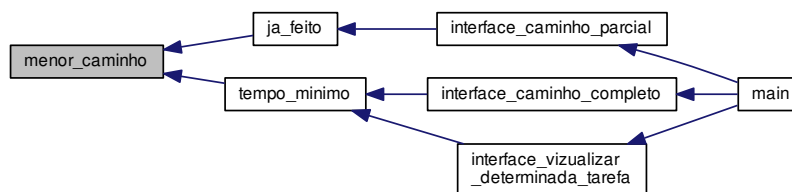
Referências achar_celula(), lista_aresta::destino, Celula_priv::executada, lista_vert::id, lista_vert::id_externo, Celula_priv::ini_min, maior_id(), lista_aresta::next, lista_vert::next, num_vert(), lista_aresta::peso, lista_vert::sucessores e grafos_priv::vert.

Referenciado por ja_feito() e tempo_minimo().

Este é o diagrama das funções utilizadas por esta função:



Este é o diagrama das funções que utilizam esta função:



4.1.30.1 `int num_arestas (const grafo_priv_t * meu_grafo)`

Número de arestas.

4.1.31 Descrição

Retorna o número de arestas do grafo

Parâmetros

<code>meu_grafo</code>	- Deve ser passado um grafo inicializado
------------------------	--

Retorna

Será retornado um inteiro ≥ 0 .

4.1.32 Assertiva de entrada

O grafo já deve ter sido inicializado por `criar_grafo()`, se não for o programa pode ser interrompido.

Assertivas de saída O grafo não será modificado. O valor de retorno é maior ou igual 0

Definição na linha 511 do arquivo `grafo.cpp`.

Referências `lista_aresta::next`, `lista_vert::next`, `lista_vert::sucessores` e `grafo_priv::vert`.

4.1.32.1 int num_vert (const grafo_priv_t * meu_grafo)

Número de vértices.

4.1.33 Descrição

Retorna o número de vértices do grafo

Parâmetros

<i>meu_grafo</i>	- Deve ser passado um grafo inicializado
------------------	--

Retorna

Será retornado um inteiro ≥ 0 .

4.1.34 Assertiva de entrada

O grafo já deve ter sido inicializado por [criar_grafo\(\)](#), se não for o programa pode ser interrompido.

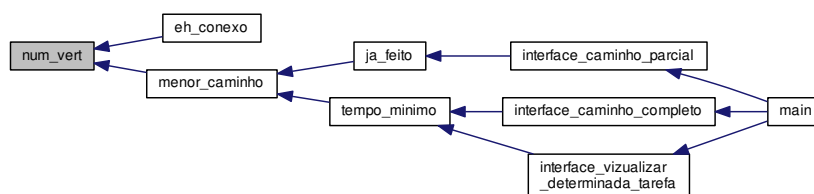
Assertivas de saída O grafo não será modificado. O valor de retorno é maior ou igual 0

Definição na linha 501 do arquivo grafos.cpp.

Referências lista_vert::next e grafo_priv::vert.

Referenciado por eh_conexo() e menor_caminho().

Este é o diagrama das funções que utilizam esta função:



4.1.34.1 void remover_aresta (grafo_priv_t * meu_grafo, int id_externo1, int id_externo2)

Remover aresta.

4.1.35 Descrição

Caso não existirem vértices id_externo1 e id_externo2 ou não existir uma aresta de id_externo1 para id_externo2, nada será feito, não será gerado nenhum warning.

Se existir a aresta, ela será removida. Os vértices permanecerão no grafo.

Parâmetros

<i>meu_grafo</i>	- Deve ser passado um grafo inicializado
<i>id_externo1</i>	
<i>id_externo2</i>	- Deve ser passado os identificadores

Retorna

Retorna a grafo modificado por referência, ou seja, o grafo passado será modificado sem a necessidade de receber um valor de retorno.

4.1.36 Assertiva de entrada

O grafo já deve ter sido inicializado por [criar_grafo\(\)](#), se não for o programa pode ser interrompido.

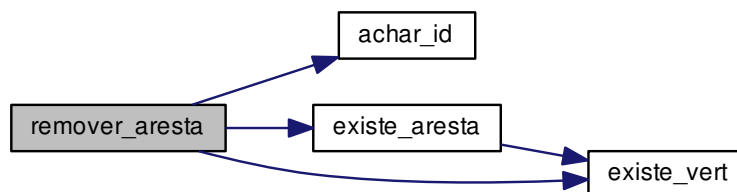
Assertivas de saída O grafo retornado por referência foi removido de uma aresta, não se assegura sua conexividade.

Definição na linha 440 do arquivo grafo.cpp.

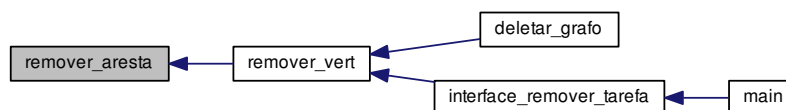
Referências [achar_id\(\)](#), [lista_vert::antecessores](#), [existe_aresta\(\)](#), [existe_vert\(\)](#), [lista_vert::id](#), [lista_aresta::next](#), [lista_vert::next](#), [lista_vert::sucessores](#), [TRUE_T](#) e [grafo_priv::vert](#).

Referenciado por [remover_vert\(\)](#).

Este é o diagrama das funções utilizadas por esta função:



Este é o diagrama das funções que utilizam esta função:



4.1.36.1 void remover_vert (grafo_priv_t * meu_grafo, int id_externo)

Remover vértice.

4.1.37 Descrição

Caso não existir um vértice `id_externo`, nada será feito, não será gerado nenhum warning.

Se existir o vertice, ele será removido. Se houverem arestas incidentes nele, saindo ou entrando, elas serão removidas, incluindo dos outros vértices.

Parâmetros

<i>meu_grafo</i>	- Deve ser passado um grafo inicializado
<i>id_externo</i>	- Deve ser passado um inteiro que é o identificador de uma tarefa

Retorna

Retorna o grafo modificado por referência, ou seja, o grafo passado será modificado sem a necessidade de receber um valor de retorno.

4.1.38 Assertiva de entrada

O grafo já deve ter sido inicializado por [criar_grafo\(\)](#), se não for o programa pode ser interrompido.

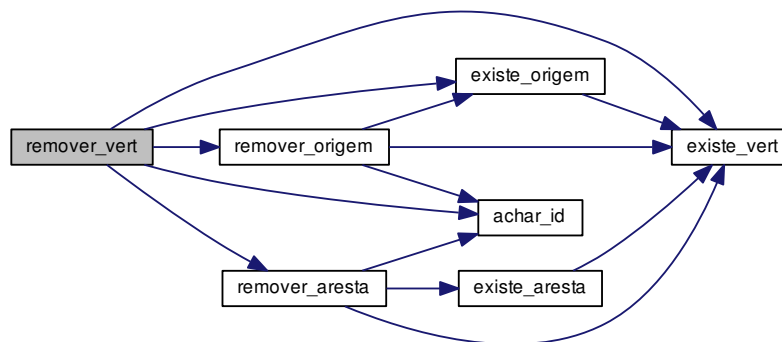
Assertivas de saída O grafo retornado por referência foi removido de um vértice, não se assegura sua conexividade.

Definição na linha 340 do arquivo grafos.cpp.

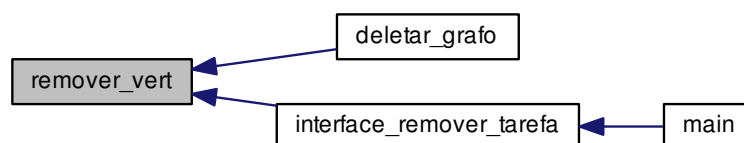
Referências `achar_id()`, `lista_aresta::destino`, `existe_origem()`, `existe_vert()`, `lista_vert_codigo::id`, `lista_vert::id`, `lista_vert::id_externo`, `lista_vert_codigo::next`, `lista_vert::next`, `remover_aresta()`, `remover_origem()`, `grafo_privado::tabela`, `TRUE_T` e `grafo_privado::vert`.

Referenciado por `deletar_grafo()` e `interface_remover_tarefa()`.

Este é o diagrama das funções utilizadas por esta função:



Este é o diagrama das funções que utilizam esta função:



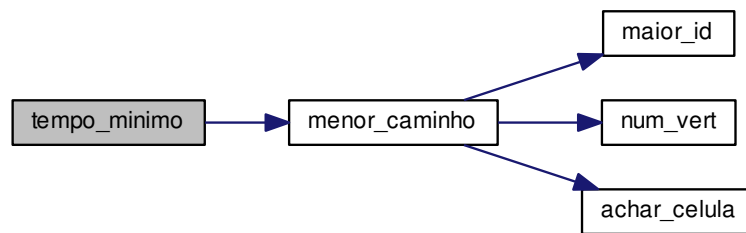
4.1.38.1 `int tempo_minimo (const grafo_priv_t * meu_grafo, int id_fim)`

Definição na linha 732 do arquivo `grafo.cpp`.

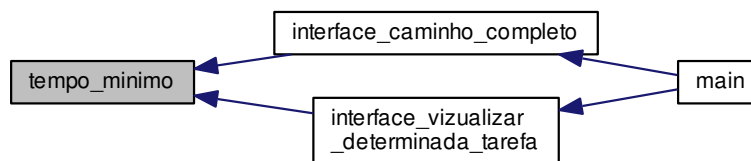
Referências `menor_caminho()`.

Referenciado por `interface_caminho_completo()` e `interface_vizualizar_determinada_tarefa()`.

Este é o diagrama das funções utilizadas por esta função:



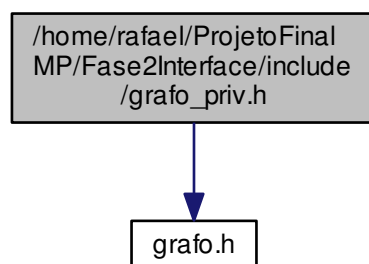
Este é o diagrama das funções que utilizam esta função:



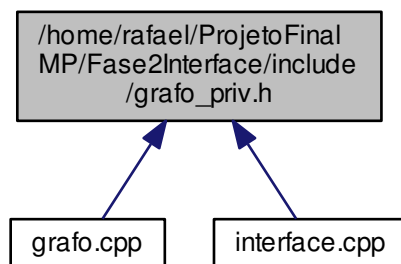
4.2 Referência do Arquivo `/home/rafael/ProjetoFinalIMP/Fase2Interface/include/grafo_priv.h`

```
#include "grafo.h"
```

Gráfico de dependência de inclusões para grafo_priv.h:



Este grafo mostra quais arquivos estão direta ou indiretamente relacionados com este arquivo:



Estruturas de Dados

- struct [Celula_priv](#)
- struct [lista_vert_codigo](#)
- struct [lista_aresta](#)
- struct [lista_vert](#)
- struct [lista_origem](#)
- struct [grafo_priv](#)

Definições de Tipos

- typedef struct [Celula_priv](#) [Celula_priv_t](#)
- typedef struct [lista_vert_codigo](#) [lista_vert_codigo_t](#)
- typedef struct [lista_aresta](#) [lista_aresta_t](#)
- typedef struct [lista_vert](#) [lista_vert_t](#)
- typedef struct [lista_origem](#) [lista_origem_t](#)
- typedef struct [grafo_priv](#) [grafo_priv_t](#)

Funções

- `resposta existe_origem` (`const grafo_priv_t *meu_grafo`, `int id_externo`)
- `void inserir_origem` (`grafo_priv_t *meu_grafo`, `Celula_priv_t *celula`)
- `void remover_origem` (`grafo_priv_t *meu_grafo`, `int id_externo`)

4.2.1 Definições dos tipos

4.2.1.1 `typedef struct Celula_priv Celula_priv_t`

4.2.1.2 `typedef struct grafo_priv grafo_priv_t`

4.2.1.3 `typedef struct lista_aresta lista_aresta_t`

4.2.1.4 `typedef struct lista_origem lista_origem_t`

4.2.1.5 `typedef struct lista_vert_codigo lista_vert_codigo_t`

4.2.1.6 `typedef struct lista_vert lista_vert_t`

4.2.2 Funções

4.2.2.1 `resposta existe_origem (const grafo_priv_t * meu_grafo, int id_externo)`

Definição na linha 74 do arquivo `grafo.cpp`.

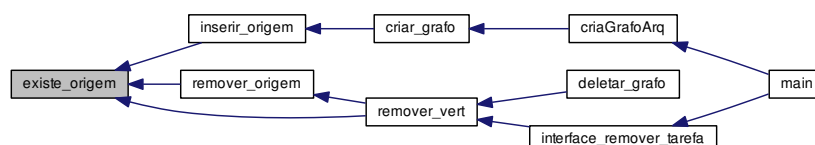
Referências `lista_origem::destino`, `existe_vert()`, `FALSE_T`, `lista_vert::id_externo`, `lista_origem::next`, `grafo_priv_t::origem` e `TRUE_T`.

Referenciado por `inserir_origem()`, `remover_origem()` e `remover_vert()`.

Este é o diagrama das funções utilizadas por esta função:



Este é o diagrama das funções que utilizam esta função:



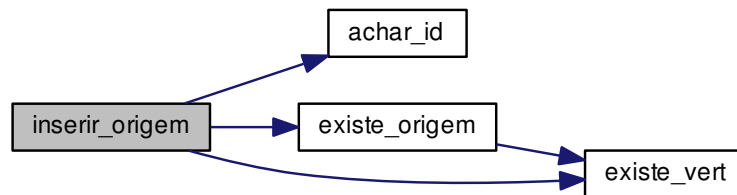
4.2.2.2 void inserir_origem (grafo_priv_t * meu_grafo, Celula_priv_t * celula)

Definição na linha 202 do arquivo grafo.cpp.

Referências achar_id(), lista_origem::destino, existe_origem(), existe_vert(), FALSE_T, lista_vert::id, Celula_priv::id_externo, lista_vert::next, lista_origem::next, Celula_priv::nome, grafo_priv::origem, TRUE_T e grafo_priv::vert.

Referenciado por criar_grafo().

Este é o diagrama das funções utilizadas por esta função:



Este é o diagrama das funções que utilizam esta função:



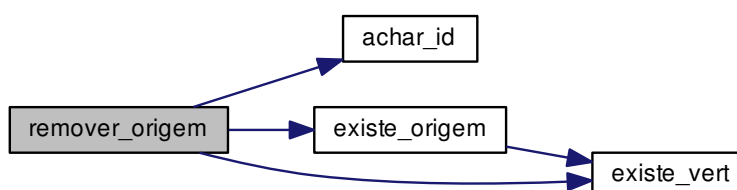
4.2.2.3 void remover_origem (grafo_priv_t * meu_grafo, int id_externo)

Definição na linha 413 do arquivo grafo.cpp.

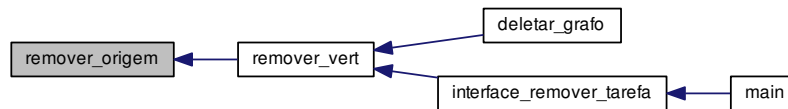
Referências achar_id(), lista_origem::destino, existe_origem(), existe_vert(), lista_vert::id, lista_origem::next, grafo_priv::origem e TRUE_T.

Referenciado por remover_vert().

Este é o diagrama das funções utilizadas por esta função:



Este é o diagrama das funções que utilizam esta função:



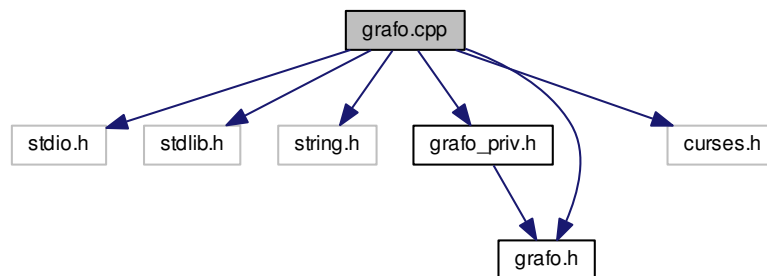
4.3 Referência do Arquivo grafo.cpp

```

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "grafo_priv.h"
#include "grafo.h"
#include <curses.h>

```

Gráfico de dependência de inclusões para grafo.cpp:



Definições e Macros

- #define **DEBUG**

Funções

- **grafo_priv_t * criar_grafo** (void)
Cria grafo.
- **grafo_priv_t * deletar_grafo** (grafo_priv_t *meu_grafo)
Deleta grafo.
- **Celula_priv_t * cria_celula** (int id_externo, int executada, int duracao, int ini_min, int pre_req, int *reqs, const char *nome)
Criar nova célula.
- **resposta existe_vert** (const **grafo_priv_t** *meu_grafo, int id_externo)
Existe vértice nome?
- **resposta existe_origem** (const **grafo_priv_t** *meu_grafo, int id_externo)

- `resposta existe_aresta` (const `grafo_priv_t` *meu_grafo, int id_externo1, int id_externo2)
Existe aresta?
- int `achar_id` (const `grafo_priv_t` *meu_grafo, int id_externo)
Achar id.
- `Celula_priv_t` * `achar_celula` (const `grafo_priv_t` *meu_grafo, int id_externo)
Achar célula.
- void `inserir_vert` (`grafo_priv_t` *meu_grafo, `Celula_priv_t` *celula)
Inserir vértice.
- void `inserir_origem` (`grafo_priv_t` *meu_grafo, `Celula_priv_t` *celula)
- void `inserir_aresta` (`grafo_priv_t` *meu_grafo, int id_externo1, `Celula_priv_t` *celula2, int peso)
Inserir aresta.
- void `remover_vert` (`grafo_priv_t` *meu_grafo, int id_externo)
Remover vértice.
- void `remover_origem` (`grafo_priv_t` *meu_grafo, int id_externo)
- void `remover_aresta` (`grafo_priv_t` *meu_grafo, int id_externo1, int id_externo2)
Remover aresta.
- int `maior_id` (const `grafo_priv_t` *meu_grafo)
Maior id.
- int `num_vert` (const `grafo_priv_t` *meu_grafo)
Número de vértices.
- int `num_arestas` (const `grafo_priv_t` *meu_grafo)
Número de arestas.
- int `menor_caminho` (const `grafo_priv_t` *meu_grafo, int **dist)
Menor caminho.
- int `dfs` (const `grafo_priv_t` *meu_grafo, `lista_vert_t` *atual, int *marc)
- `resposta eh_conexo` (const `grafo_priv_t` *meu_grafo)
É conexo?
- void `Imprime_Tarefas` (const `grafo_priv_t` *meu_grafo, int linha, int coluna)
- void `Grava_Arq` (`grafo_priv_t` *meu_grafo, char *NomeArq)
- void `editar_celula` (`grafo_priv_t` *meu_grafo, int ID)
- int `tempo_minimo` (const `grafo_priv_t` *meu_grafo, int id_fim)
- void `ja_feito` (const `grafo_priv_t` *meu_grafo, int d)
- `grafo_priv_t` * `criaGrafoArq` (char *nomeArq)

4.3.1 Definições e macros

4.3.1.1 #define DEBUG

Definição na linha 10 do arquivo grafo.cpp.

4.3.2 Funções

4.3.2.1 `Celula_priv_t`* `achar_celula` (const `grafo_priv_t` * *meu_grafo*, int *id_externo*)

Achar célula.

4.3.3 Descrição

Encontra a célula e os dados dados pelo usuário ao vértice de identificador id

Parâmetros

<i>meu_grafo</i>	- Deve ser passado um grafo inicializado
<i>id</i>	- Deve ser passado um inteiro não negativo válido, ou seja, que represente um vértice

Retorna

Retorna um ponteiro para uma célula. Caso não se ache um vértice com id, retorna-se NULL.

4.3.4 Assertiva de entrada

O grafo já deve ter sido inicializado por [criar_grafo\(\)](#), se não for o programa pode ser interrompido. $0 \leq id$ Há um vértice representado por id

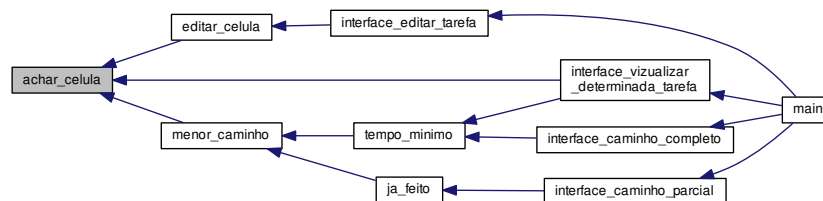
Assertivas de saída Se existir um vértice id, a resposta será um ponteiro != NULL, que aponta para onde está armazenado a célula. Se não existir, retorna-se NULL Assertivas de saída O retorno um ponteiro para uma célula. O conteúdo do grafo não será modificado.

Definição na linha 117 do arquivo grafo.cpp.

Referências lista_vert_codigo::dado, Celula_priv::id_externo, lista_vert_codigo::next e grafo_priv::tabela.

Referenciado por editar_celula(), interface_vizualizar_determinada_tarefa() e menor_caminho().

Este é o diagrama das funções que utilizam esta função:

**4.3.4.1 int achar_id (const grafo_priv_t * meu_grafo, int id_externo)**

Achar id.

4.3.5 Descrição

Todo vértice armazenado no grafo possui um nome, e um identificador (id_externo). Essa função acha esse identificador

Parâmetros

<i>meu_grafo</i>	- Deve ser passado um grafo inicializado
<i>id_externo</i>	- um inteiro que representa o identificador da tarefa.

Retorna

Retorna um inteiro id. Caso não se ache o vértice id, esse inteiro será -1, caso contrário será o valor do identificador da tarefa.

4.3.6 Assertiva de entrada

O grafo já deve ter sido inicializado por `criar_grafo()`, se não for o programa pode ser interrompido.

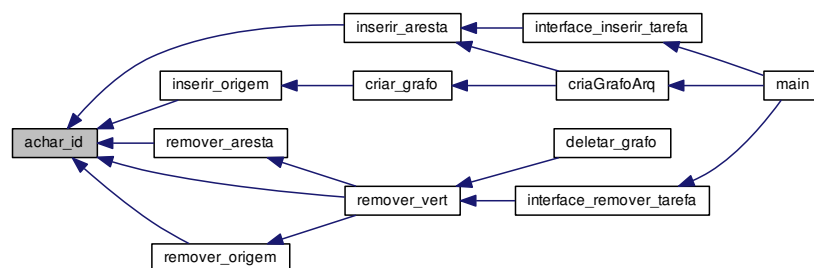
Já foi checado que existe `vert(meu_grafo, nome) == TRUE` Assertivas de saída Se existir um vértice nome, $0 \leq id \leq \max(int)$, sendo que $\max(int)$ é o maior valor que pode ser representado num inteiro Se não existir `id = -1` Assertivas de saída O retorno será TRUE ou FALSE O conteúdo do grafo não será modificado.

Definição na linha 107 do arquivo grafo.cpp.

Referências `lista_vert_codigo::dados`, `lista_vert_codigo::id`, `Celula_priv::id_externo`, `lista_vert_codigo::next` e `grafo->_priv::tabela`.

Referenciado por `inserir_aresta()`, `inserir_origem()`, `remover_aresta()`, `remover_origem()` e `remover_vert()`.

Este é o diagrama das funções que utilizam esta função:



4.3.6.1 `Celula_priv_t* cria_celula (int id_externo, int executada, int duracao, int ini_min, int pre_req, int * reqs, const char * nome)`

Criar nova célula.

4.3.7 Descrição

Cria uma nova célula com os os dados inseridos

Parâmetros

<i>meu_grafo</i>	- Deve ser passado um ponteiro para um grafo inicializado
<i>id_externo</i>	- Deve ser um inteiro que é o ID da tarefa
<i>executada</i>	- Deve ser um inteiro que indica se a tarefa ja foi ou nao executada.
<i>duracao</i>	- Deve ser um inteiro que representa a duracao da tarefa
<i>ini_min</i>	- Deve ser um inteiro que representa o tempo minimo para iniciar a tarefa.
<i>pre_req</i>	- Deve ser um inteiro que representa a quantidade de pre requisitos que a tarega possui.
<i>reqs</i>	- Deve ser um ponteiro pra inteiro contendo todos os IDs dos pre requisitos da tarefa.
<i>nome</i>	- Deve ser passado uma string de até 100 caracteres, mais o caracter zero terminal que será o nome da célula.

Retorna

Retorna uma célula com os dados que foram inseridos na chamada

4.3.8 Assertiva de entrada

Os numeros devem ser inteiros maiores que 0. A string de nome deve ser de ate 100 caracteres. Assertivas de saída O retorno será uma célula contendo os dados que foram passados na entrada

Definição na linha 43 do arquivo grafo.cpp.

Referências Celula_priv::duracao, Celula_priv::executada, Celula_priv::id_externo, Celula_priv::ini_min, Celula_priv::nome, Celula_priv::pre_req e Celula_priv::reqs.

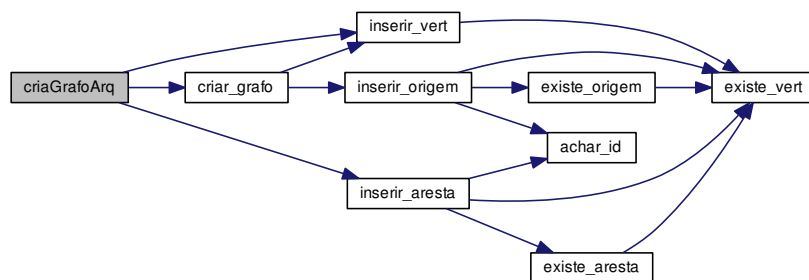
4.3.8.1 grafo_priv_t* criaGrafoArq (char * nomeArq)

Definição na linha 779 do arquivo grafo.cpp.

Referências criar_grafo(), Celula_priv::duracao, Celula_priv::executada, Celula_priv::id_externo, Celula_priv::ini_min, inserir_aresta(), inserir_vert(), Celula_priv::nome, Celula_priv::pre_req e Celula_priv::reqs.

Referenciado por main().

Este é o diagrama das funções utilizadas por esta função:



Este é o diagrama das funções que utilizam esta função:

**4.3.8.2 grafo_priv_t* criar_grafo (void)**

Cria grafo.

4.3.9 Descrição

Aloca a memória necessária e inicializa um grafo

4.3.10 Parâmetros

Não ha parâmetros, a alocação e inicializam não dependem de nenhum parâmetro do usuário

Retorna

Se retorna um ponteiro para a grafo criado

4.3.11 Assertiva de saída

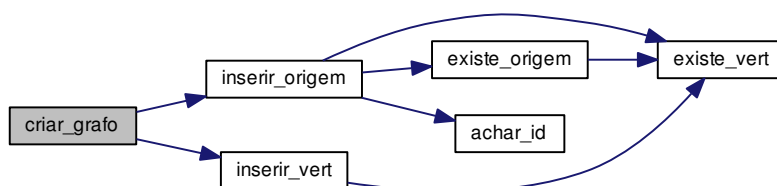
O grafo gerado é consistente e não possui nenhum vértice, origem ou aresta.

Definição na linha 12 do arquivo grafo.cpp.

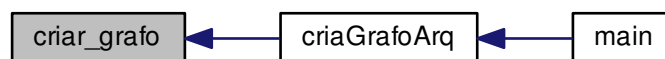
Referências Celula_priv::executada, Celula_priv::id_externo, Celula_priv::ini_min, inserir_origem(), inserir_vert(), Celula_priv::nome, grafo_priv::origem, Celula_priv::pre_req, grafo_priv::tabela e grafo_priv::vert.

Referenciado por criaGrafoArq().

Este é o diagrama das funções utilizadas por esta função:



Este é o diagrama das funções que utilizam esta função:



4.3.11.1 grafo_priv_t* deletar_grafo (grafo_priv_t * meu_grafo)

Deleta grafo.

4.3.12 Descrição

Desaloca toda a memória utilizada pelo grafo

Parâmetros

<i>meu_grafo</i>	- Deve ser passado um ponteiro para um grafo inicializado
------------------	---

Retorna

Retorna um ponteiro para o grafo, que será NULL. O valor de retorno é muito importante, uma vez que se ele não for utilizado o grafo do usuário apontará para um endereço não alocado e qualquer tentativa de utilizá-lo poderá gerar erros no sistema. Caso o grafo passado não tenha sido inicializado, o programa poderá parar a execução

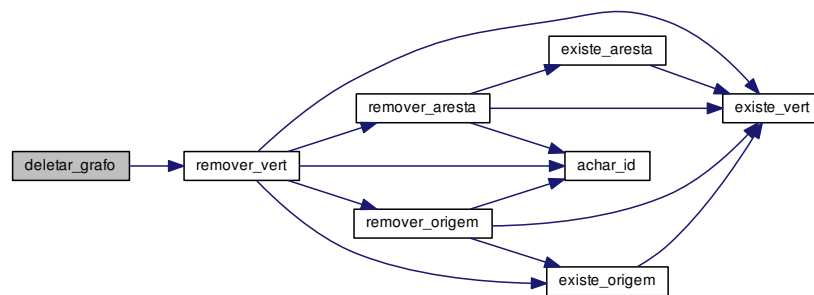
4.3.13 Assertiva de saída

O grafo já deve ter sido inicializado por [criar_grafo\(\)](#) Assertiva de saída Se retornará um ponteiro para o mesmo grafo passado, após a deleção que será NULL.

Definição na linha 32 do arquivo grafo.cpp.

Referências `lista_vert::dado`, `Celula_priv::id_externo`, `remover_vert()` e `grafo_priv::tabela`.

Este é o diagrama das funções utilizadas por esta função:

**4.3.13.1 `int dfs (const grafo_priv_t * meu_grafo, lista_vert_t * atual, int * marc)`**

Definição na linha 577 do arquivo grafo.cpp.

Referências `lista_aresta::destino`, `lista_vert::id`, `lista_aresta::next` e `lista_vert::sucessores`.

Referenciado por `eh_conexo()`.

Este é o diagrama das funções que utilizam esta função:



4.3.13.2 void editar_celula (grafo_priv_t * meu_grafo, int ID)

Definição na linha 676 do arquivo grafo.cpp.

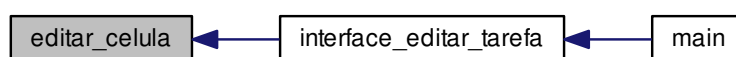
Referências achar_celula(), Celula_priv::duracao, Celula_priv::executada, Celula_priv::id_externo, Celula_priv::ini_min, Celula_priv::nome, Celula_priv::pre_req e Celula_priv::reqs.

Referenciado por interface_editar_tarefa().

Este é o diagrama das funções utilizadas por esta função:



Este é o diagrama das funções que utilizam esta função:



4.3.13.3 resposta eh_conexo (const grafo_priv_t * meu_grafo)

É conexo?

4.3.14 Descrição

Verifica se partindo da origem podem-se alcançar todos os vértices do grafo. SE for possível retorna TRUE, caso contrário FALSE.

Parâmetros

<i>meu_grafo</i>	- Deve ser passado um grafo inicializado
------------------	--

Retorna

Se for possível alcançar todos os vértices partindo das origens retorna TRUE, caso contrário FALSE. Um grafo sem vértices é conexo.

4.3.15 Assertiva de entrada

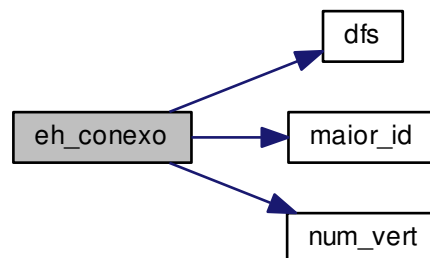
O grafo já deve ter sido inicializado por [criar_grafo\(\)](#), se não for o programa pode ser interrompido.

Assertivas de saída O grafo não será modificado.

Definição na linha 593 do arquivo grafo.cpp.

Referências `lista_origem::destino`, `dfs()`, `FALSE_T`, `lista_vert::id`, `maior_id()`, `lista_origem::next`, `num_vert()`, `grafo`, `_priv::origem` e `TRUE_T`.

Este é o diagrama das funções utilizadas por esta função:



4.3.15.1 resposta existe_aresta (const grafo_priv_t * meu_grafo, int id_externo1, int id_externo2)

Existe aresta?

4.3.16 Descrição

Verfica a existência de um vértice dado

Parâmetros

<i>meu_grafo</i>	- Deve ser passado um grafo inicializado
<i>id_externo1</i>	
<i>id_externo2</i>	- Devem ser passadas dois IDs de tarefa, que são números inteiros.

Retorna

Retorna uma resposta, que será TRUE caso exista a aresta, e FALSE caso não exista.

4.3.17 Assertiva de entrada

O grafo já deve ter sido inicializado por [criar_grafo\(\)](#), se não for o programa pode ser interrompido.

Assertivas de saída O retorno será TRUE ou FALSE O conteúdo do grafo não será modificado.

Definição na linha 87 do arquivo grafo.cpp.

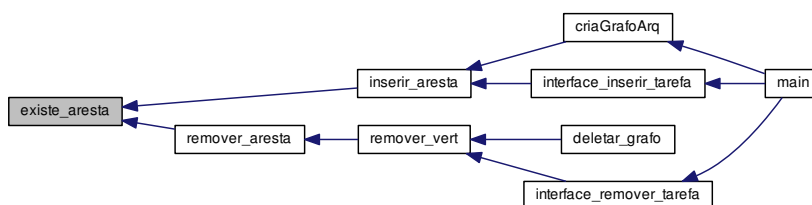
Referências `lista_aresta::destino`, `existe_vert()`, `FALSE_T`, `lista_vert::id_externo`, `lista_aresta::next`, `lista_vert::next`, `lista_vert::sucessores`, `TRUE_T` e `grafo_priv::vert`.

Referenciado por `inserir_aresta()` e `remover_aresta()`.

Este é o diagrama das funções utilizadas por esta função:



Este é o diagrama das funções que utilizam esta função:



4.3.17.1 resposta existe_origem (const grafo_priv_t * meu_grafo, int id_externo)

Definição na linha 74 do arquivo grafo.cpp.

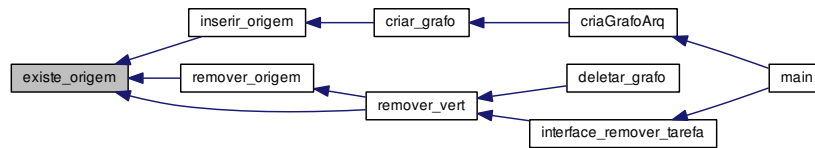
Referências lista_origem::destino, existe_vert(), FALSE_T, lista_vert::id_externo, lista_origem::next, grafo_priv->origem e TRUE_T.

Referenciado por inserir_origem(), remover_origem() e remover_vert().

Este é o diagrama das funções utilizadas por esta função:



Este é o diagrama das funções que utilizam esta função:



4.3.17.2 resposta existe_vert (const grafo_priv_t * meu_grafo, int id_externo)

Existe vértice nome?

4.3.18 Descrição

Verfica a existência de um vértice dado

Parâmetros

<i>meu_grafo</i>	- Deve ser passado um ponteiro para um grafo inicializado
<i>nome</i>	- Deve ser passado o número de identificação da tarefa.

Retorna

Retorna uma resposta, que será TRUE caso exista o vértice, e FALSE caso não exista.

4.3.19 Assertiva de entrada

O grafo já deve ter sido inicializado por [criar_grafo\(\)](#), se não for o programa pode ser interrompido.

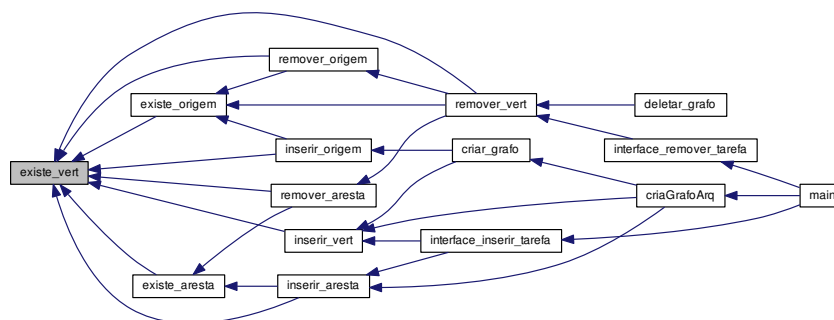
Assertivas de saída O retorno será TRUE ou FALSE O conteúdo do grafo não será modificado

Definição na linha 64 do arquivo grafo.cpp.

Referências lista_vert_codigo::dado, FALSE_T, Celula_priv::id_externo, lista_vert_codigo::next, grafo_priv::tabela e TRUE_T.

Referenciado por existe_aresta(), existe_origem(), inserir_aresta(), inserir_origem(), inserir_vert(), remover_aresta(), remover_origem() e remover_vert().

Este é o diagrama das funções que utilizam esta função:



4.3.19.1 void Grava_Arq (grafo_priv_t * meu_grafo, char * NomeArq)

Definição na linha 649 do arquivo grafo.cpp.

Referências lista_vert_codigo::dado, Celula_priv::duracao, Celula_priv::executada, Celula_priv::id_externo, Celula_priv::ini_min, lista_vert_codigo::next, Celula_priv::nome, Celula_priv::pre_req, Celula_priv::reqs e grafo_priv::tabela.

Referenciado por main().

Este é o diagrama das funções que utilizam esta função:



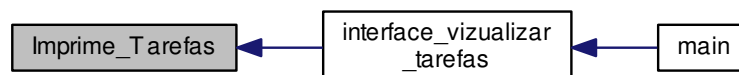
4.3.19.2 void Imprime_Tarefas (const grafo_priv_t * meu_grafo, int linha, int coluna)

Definição na linha 628 do arquivo grafo.cpp.

Referências lista_vert_codigo::dado, Celula_priv::duracao, Celula_priv::executada, Celula_priv::id_externo, Celula_priv::ini_min, lista_vert_codigo::next, Celula_priv::nome, Celula_priv::pre_req, Celula_priv::reqs e grafo_priv::tabela.

Referenciado por interface_vizualizar_tarefas().

Este é o diagrama das funções que utilizam esta função:



4.3.19.3 void inserir_aresta (grafo_priv_t * meu_grafo, int id_externo1, Celula_priv_t * celula2, int peso)

Inserir aresta.

4.3.20 Descrição

Caso id_externo1 e celula2 sejam vértices, peso > 0 e ainda não haja uma aresta entre eles, insere-se uma.

Se não existir algum dos vértices ou já exista uma aresta ou se peso <= 0, e caso a opção de DEBUG seja ativada, (macro DEBUG é igual à 1), será enviada uma mensagem à saída padrão de erro. Independente de DEBUG, a aresta não será adicionada.

Assim se for desejado adicionar uma aresta que não se sabe se já existem os vértices ou não, use inserir_vert, para cada um e depois inserir_aresta.

Parâmetros

<i>meu_grafo</i>	- Deve ser passado um grafo inicializado
<i>id_externo1</i>	- Deve ser passado o id da tarefa de pre requisito
<i>celula2</i>	- Deve ser passado um ponteiro para a célula destino
<i>peso</i>	- Deve ser um número real maior que zero

Retorna

Retorna a grafo modificado por referência, ou seja, o grafo passado será modificado sem a necessidade de receber um valor de retorno.

4.3.21 Assertiva de entrada

O grafo já deve ter sido inicializado por [criar_grafo\(\)](#), se não for o programa pode ser interrompido. $\text{peso} > 0$

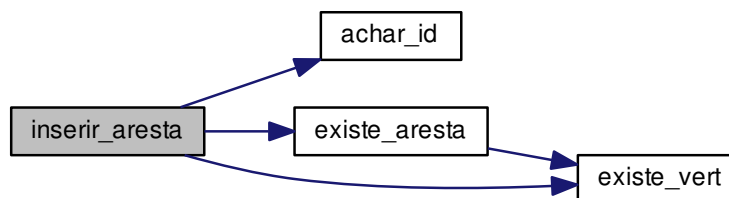
Assertivas de saída O grafo retornado foi adicionado de uma aresta, não se assegura sua conexividade. O conteúdo das células passadas e do número inteiro não serão modificados

Definição na linha 250 do arquivo grafo.cpp.

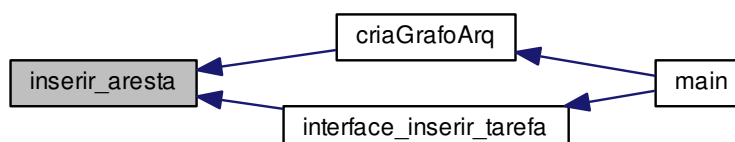
Referências [achar_id\(\)](#), [lista_aresta::destino](#), [existe_aresta\(\)](#), [existe_vert\(\)](#), [FALSE_T](#), [Celula_priv::id_externo](#), [lista_aresta::next](#), [lista_vert::next](#), [lista_aresta::peso](#), [lista_vert::sucessores](#), [TRUE_T](#) e [grafo_priv::vert](#).

Referenciado por [criaGrafoArq\(\)](#) e [interface_inserir_tarefa\(\)](#).

Este é o diagrama das funções utilizadas por esta função:



Este é o diagrama das funções que utilizam esta função:



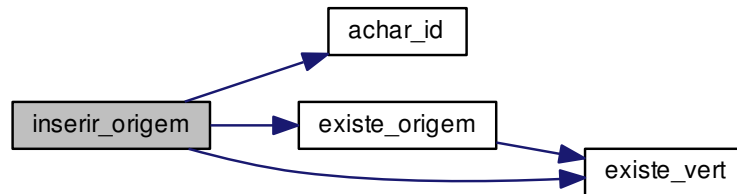
4.3.21.1 void inserir_origem (grafo_priv_t * meu_grafo, Celula_priv_t * celula)

Definição na linha 202 do arquivo grafo.cpp.

Referências achar_id(), lista_origem::destino, existe_origem(), existe_vert(), FALSE_T, lista_vert::id, Celula_priv::id_externo, lista_vert::next, lista_origem::next, Celula_priv::nome, grafo_priv::origem, TRUE_T e grafo_priv::vert.

Referenciado por criar_grafo().

Este é o diagrama das funções utilizadas por esta função:



Este é o diagrama das funções que utilizam esta função:



4.3.21.2 void inserir_vert (grafo_priv_t * meu_grafo, Celula_priv_t * celula)

Inserir vértice.

4.3.22 Descrição

Caso não existir um vértice nome, insere-se um.

Se já existir o vértice, e caso a opção de DEBUG seja ativada, (macro DEBUG é igual à 1), será enviada uma mensagem à saída padrão de erro. Independente de DEBUG, o vértice não será adicionado.

Para isso guarda-se uma cópia da célula dada pelo usuário

Parâmetros

<i>meu_grafo</i>	- Deve ser passado um grafo inicializado
<i>celula</i>	- Deve ser passado uma célula com todas as informações que nela são necessárias existir.

Retorna

Retorna o grafo modificado por referência, ou seja, o grafo passado será modificado sem a necessidade de receber um valor de retorno.

4.3.23 Assertiva de entrada

O grafo já deve ter sido inicializado por `criar_grafo()`, se não for o programa pode ser interrompido.

Assertivas de saída O grafo retornado por referência foi adicionado de um vértice, não se assegura sua conexividade. O conteúdo da célula passada não será modificado, porém será copiado

Definição na linha 128 do arquivo `grafo.cpp`.

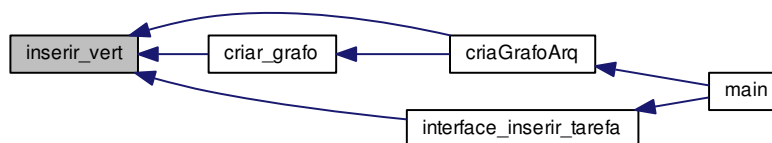
Referências `lista_vert::antecessores`, `existe_vert()`, `FALSE_T`, `lista_vert_codigo::id`, `lista_vert::id`, `Celula_priv::id`, `_externo`, `lista_vert::id_externo`, `Celula_priv::ini_min`, `lista_vert_codigo::next`, `lista_vert::next`, `Celula_priv::nome`, `Celula_priv::pre_req`, `lista_vert::sucessores`, `grafo_priv::tabela` e `grafo_priv::vert`.

Referenciado por `criaGrafoArq()`, `criar_grafo()` e `interface_inserir_tarefa()`.

Este é o diagrama das funções utilizadas por esta função:



Este é o diagrama das funções que utilizam esta função:



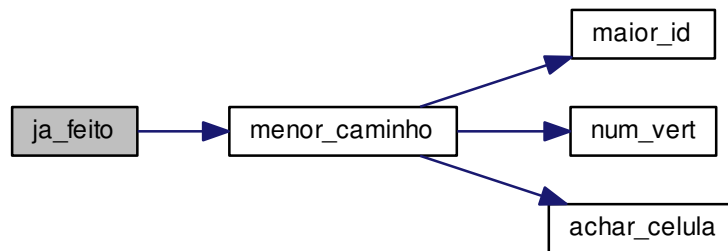
4.3.23.1 void jafeito (const grafo_priv_t * meu_grafo, int d)

Definição na linha 747 do arquivo `grafo.cpp`.

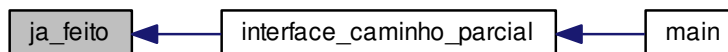
Referências `lista_vert_codigo::id`, `menor_caminho()`, `lista_vert_codigo::next` e `grafo_priv::tabela`.

Referenciado por `interface_caminho_parcial()`.

Este é o diagrama das funções utilizadas por esta função:



Este é o diagrama das funções que utilizam esta função:



4.3.23.2 int maior_id (const grafo_priv_t * meu_grafo)

Maior id.

4.3.24 Descrição

Retorna o valor do maior identificador (id) usado.

Se não foi utilizado nenhum id, ou seja, se não há nenhum vértice retorna -1

Parâmetros

<i>meu_grafo</i>	- Deve ser passado um grafo inicializado
------------------	--

Retorna

Será retornado um inteiro.

4.3.25 Assertiva de entrada

O grafo já deve ter sido inicializado por [criar_grafo\(\)](#), se não for o programa pode ser interrompido.

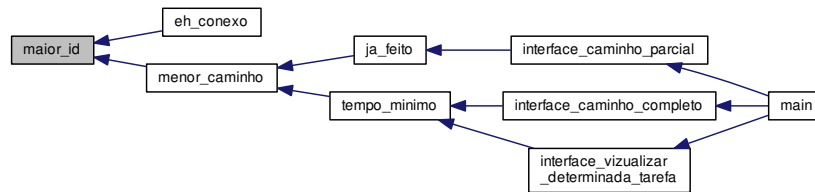
Assertivas de saída O grafo não será modificado. Se não há nenhum vértice (`num_vert(meu_grafo) == 0`), retorna -1 Caso contrário, o valor de retorno ≥ 0

Definição na linha 490 do arquivo grafo.cpp.

Referências `lista_vert_codigo::id`, `lista_vert_codigo::next` e `grafo_priv::tabela`.

Referenciado por `eh_conexo()` e `menor_caminho()`.

Este é o diagrama das funções que utilizam esta função:



4.3.25.1 `int menor_caminho (const grafo_priv_t * meu_grafo, int ** dist)`

Menor caminho.

4.3.26 Descrição

Retorna o tamanho de menor caminho de inicio até fim.

Caso não haja um caminho entre os dois, ou um deles não for vértice retorna -1.

Parâmetros

<i>meu_grafo</i>	- Deve ser passado um grafo inicializado
<i>inicio</i>	
<i>fim</i>	- Devem ser passadas duas string de até 100 caracteres, sem contar o caracter zero terminal.

Retorna

Será retornado um número real.

4.3.27 Assertiva de entrada

O grafo já deve ter sido inicializado por `criar_grafo()`, se não for o programa pode ser interrompido.

4.3.28 de saída

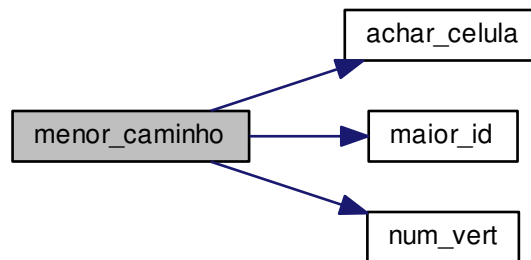
O grafo não será modificado. Se não houver caminho retorna -1. Caso contrário, o valor de retorno é maior ou igual à 0, e é o menor caminho de inicio até fim O menor caminho de um vértice à ele mesmo é 0.

Definição na linha 524 do arquivo `grafo.cpp`.

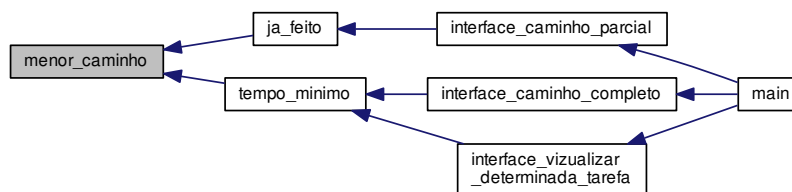
Referências `achar_celula()`, `lista_aresta::destino`, `Celula_priv::executada`, `lista_vert::id`, `lista_vert::id_externo`, `Celula_priv::ini_min`, `maior_id()`, `lista_aresta::next`, `lista_vert::next`, `num_vert()`, `lista_aresta::peso`, `lista_vert::sucessores` e `grafo_priv::vert`.

Referenciado por `ja_feito()` e `tempo_minimo()`.

Este é o diagrama das funções utilizadas por esta função:



Este é o diagrama das funções que utilizam esta função:



4.3.28.1 int num_arestas (const grafo_priv_t * meu_grafo)

Número de arestas.

4.3.29 Descrição

Retorna o número de arestas do grafo

Parâmetros

<i>meu_grafo</i>	- Deve ser passado um grafo inicializado
------------------	--

Retorna

Será retornado um inteiro ≥ 0 .

4.3.30 Assertiva de entrada

O grafo já deve ter sido inicializado por `criar_grafo()`, se não for o programa pode ser interrompido.

Assertivas de saída O grafo não será modificado. O valor de retorno é maior ou igual 0

Definição na linha 511 do arquivo grafo.cpp.

Referências `lista_aresta::next`, `lista_vert::next`, `lista_vert::sucessores` e `grafo_priv::vert`.

4.3.30.1 `int num_vert (const grafo_priv_t * meu_grafo)`

Número de vértices.

4.3.31 Descrição

Retorna o número de vértices do grafo

Parâmetros

<code>meu_grafo</code>	- Deve ser passado um grafo inicializado
------------------------	--

Retorna

Será retornado um inteiro ≥ 0 .

4.3.32 Assertiva de entrada

O grafo já deve ter sido inicializado por [criar_grafo\(\)](#), se não for o programa pode ser interrompido.

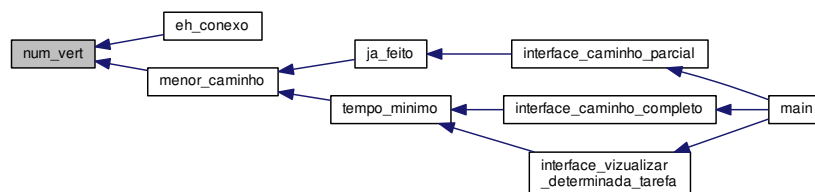
Assertivas de saída O grafo não será modificado. O valor de retorno é maior ou igual 0

Definição na linha 501 do arquivo `grafo.cpp`.

Referências `lista_vert::next` e `grafo_priv::vert`.

Referenciado por `eh_conexo()` e `menor_caminho()`.

Este é o diagrama das funções que utilizam esta função:



4.3.32.1 `void remover_aresta (grafo_priv_t * meu_grafo, int id_externo1, int id_externo2)`

Remover aresta.

4.3.33 Descrição

Caso não existirem vértices `id_externo1` e `id_externo2` ou não existir uma aresta de `id_externo1` para `id_externo2`, nada será feito, não será gerado nenhum warning.

Se existir a aresta, ela será removida. Os vértices permanecerão no grafo.

Parâmetros

<i>meu_grafo</i>	- Deve ser passado um grafo inicializado
<i>id_externo1</i>	
<i>id_externo2</i>	- Deve ser passado os identificadores

Retorna

Retorna a grafo modificado por referência, ou seja, o grafo passado será modificado sem a necessidade de receber um valor de retorno.

4.3.34 Assertiva de entrada

O grafo já deve ter sido inicializado por [criar_grafo\(\)](#), se não for o programa pode ser interrompido.

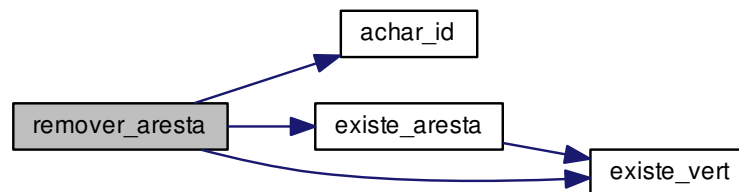
Assertivas de saída O grafo retornado por referência foi removido de uma aresta, não se assegura sua conexividade.

Definição na linha 440 do arquivo grafo.cpp.

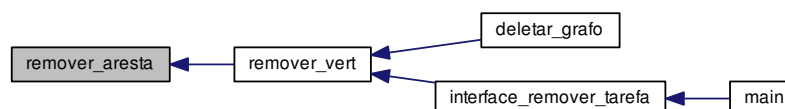
Referências [achar_id\(\)](#), [lista_vert::antecessores](#), [existe_aresta\(\)](#), [existe_vert\(\)](#), [lista_vert::id](#), [lista_aresta::next](#), [lista_vert::next](#), [lista_vert::sucessores](#), [TRUE_T](#) e [grafo_priv::vert](#).

Referenciado por [remover_vert\(\)](#).

Este é o diagrama das funções utilizadas por esta função:



Este é o diagrama das funções que utilizam esta função:

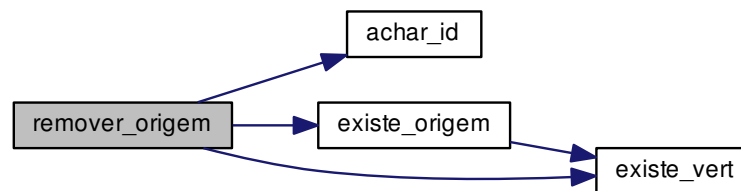
**4.3.34.1 void remover_origem (grafo_priv_t * meu_grafo, int id_externo)**

Definição na linha 413 do arquivo grafo.cpp.

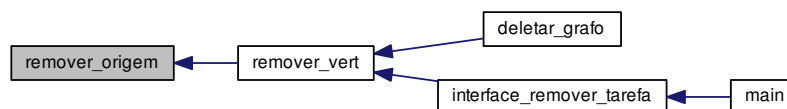
Referências [achar_id\(\)](#), [lista_origem::destino](#), [existe_origem\(\)](#), [existe_vert\(\)](#), [lista_vert::id](#), [lista_origem::next](#), [grafo_priv::origem](#) e [TRUE_T](#).

Referenciado por [remover_vert\(\)](#).

Este é o diagrama das funções utilizadas por esta função:



Este é o diagrama das funções que utilizam esta função:



4.3.34.2 void remover_vert (grafo_priv_t * meu_grafo, int id_externo)

Remover vértice.

4.3.35 Descrição

Caso não existir um vértice `id_externo`, nada será feito, não será gerado nenhum warning.

Se existir o vertice, ele será removido. Se houverem arestas incidentes nele, saindo ou entrando, elas serão removidas, incluindo dos outros vértices.

Parâmetros

<i>meu_grafo</i>	- Deve ser passado um grafo inicializado
<i>id_externo</i>	- Deve ser passado um inteiro que é o identificador de uma tarefa

Retorna

Retorna a grafo modificado por referência, ou seja, o grafo passado será modificado sem a necessidade de receber um valor de retorno.

4.3.36 Assertiva de entrada

O grafo já deve ter sido inicializado por `criar_grafo()`, se não for o programa pode ser interrompido.

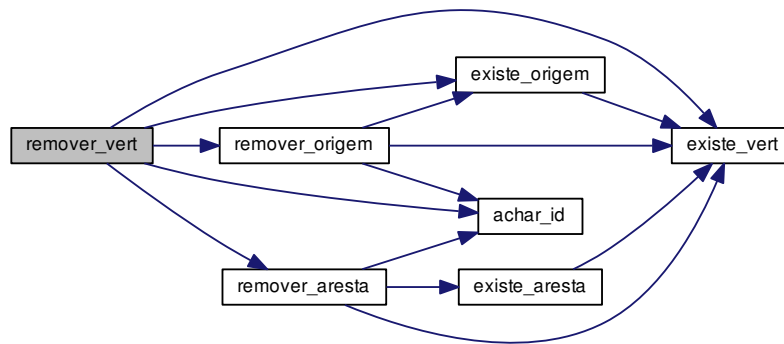
Assertivas de saída O grafo retornado por referência foi removido de um vértice, não se assegura sua conexividade.

Definição na linha 340 do arquivo `grafo.cpp`.

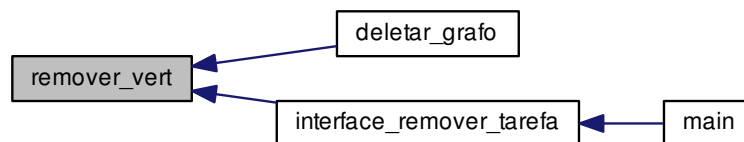
Referências `achar_id()`, `lista_aresta::destino`, `existe_origem()`, `existe_vert()`, `lista_vert_codigo::id`, `lista_vert::id`, `lista_vert::id_externo`, `lista_vert_codigo::next`, `lista_vert::next`, `remover_aresta()`, `remover_origem()`, `grafo_priv::tabela`, `TRUE_T` e `grafo_priv::vert`.

Referenciado por `deletar_grafo()` e `interface_remover_tarefa()`.

Este é o diagrama das funções utilizadas por esta função:



Este é o diagrama das funções que utilizam esta função:



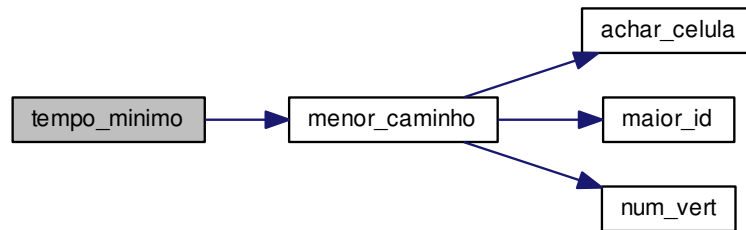
4.3.36.1 `int tempo_minimo (const grafo_priv_t * meu_grafo, int id_fim)`

Definição na linha 732 do arquivo `grafo.cpp`.

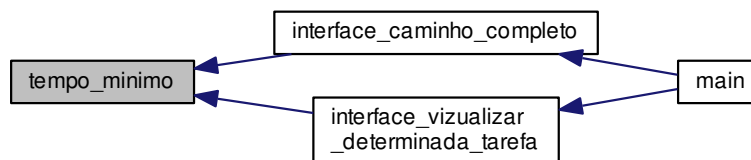
Referências `menor_caminho()`.

Referenciado por `interface_caminho_completo()` e `interface_vizualizar_determinada_tarefa()`.

Este é o diagrama das funções utilizadas por esta função:



Este é o diagrama das funções que utilizam esta função:

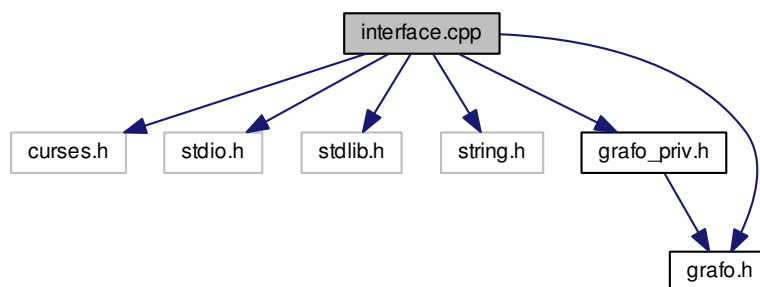


4.4 Referência do Arquivo interface.cpp

```

#include <curses.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "grafo_priv.h"
#include "grafo.h"
  
```

Gráfico de dependência de inclusões para interface.cpp:



Definições e Macros

- #define NCURSES_CONST

Funções

- void interface_remover_tarefa (grafo_priv_t *meu_grafo)
- void interface_editar_tarefa (grafo_priv_t *meu_grafo)
- void interface_inserir_tarefa (grafo_priv_t *meu_grafo)
- void interface_caminho_completo (const grafo_priv_t *meu_grafo)
- void interface_caminho_parcial (grafo_priv_t *meu_grafo)
- void interface_vizualizar_tarefas (const grafo_priv_t *meu_grafo)
- void interface_vizualizar_determinada_tarefa (const grafo_priv_t *meu_grafo)
- int main ()

4.4.1 Definições e macros

4.4.1.1 #define NCURSES_CONST

Definição na linha 1 do arquivo interface.cpp.

4.4.2 Funções

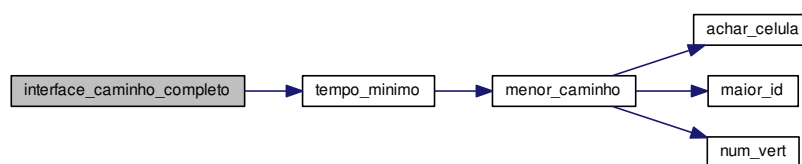
4.4.2.1 void interface_caminho_completo (const grafo_priv_t * meu_grafo)

Definição na linha 114 do arquivo interface.cpp.

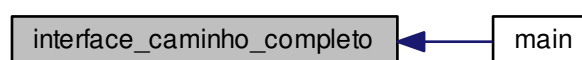
Referências lista_vert_codigo::dado, Celula_priv::id_externo, lista_vert_codigo::next, grafo_priv::tabela e tempo_minimo().

Referenciado por main().

Este é o diagrama das funções utilizadas por esta função:



Este é o diagrama das funções que utilizam esta função:



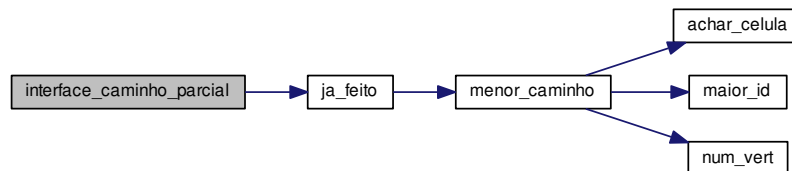
4.4.2.2 void interface_caminho_parcial (grafo_priv_t * meu_grafo)

Definição na linha 141 do arquivo interface.cpp.

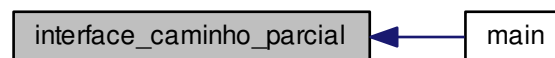
Referências ja_feito().

Referenciado por main().

Este é o diagrama das funções utilizadas por esta função:



Este é o diagrama das funções que utilizam esta função:



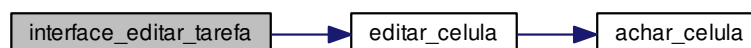
4.4.2.3 void interface_editar_tarefa (grafo_priv_t * meu_grafo)

Definição na linha 43 do arquivo interface.cpp.

Referências editar_celula().

Referenciado por main().

Este é o diagrama das funções utilizadas por esta função:



Este é o diagrama das funções que utilizam esta função:



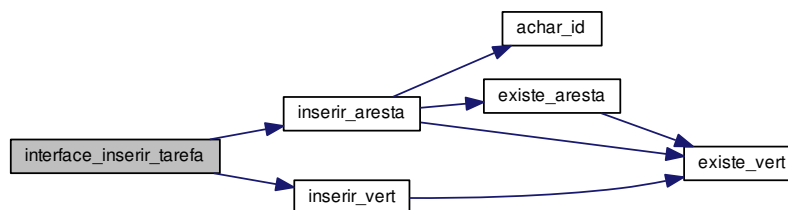
4.4.2.4 void interface_inserir_tarefa (grafo_priv_t * meu_grafo)

Definição na linha 64 do arquivo interface.cpp.

Referências Celula_priv::duracao, Celula_priv::executada, Celula_priv::id_externo, Celula_priv::ini_min, inserir_↔ aresta(), inserir_vert(), Celula_priv::nome, Celula_priv::pre_req e Celula_priv::reqs.

Referenciado por main().

Este é o diagrama das funções utilizadas por esta função:



Este é o diagrama das funções que utilizam esta função:



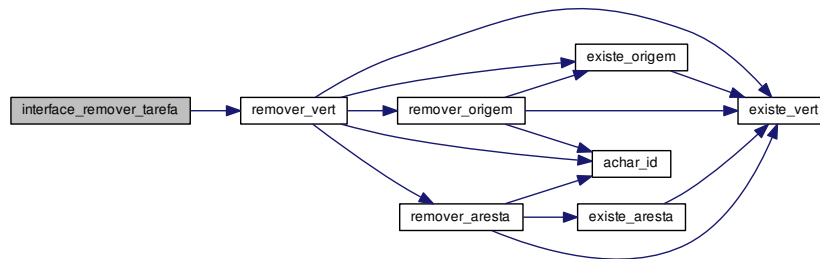
4.4.2.5 void interface_remover_tarefa (grafo_priv_t * meu_grafo)

Definição na linha 21 do arquivo interface.cpp.

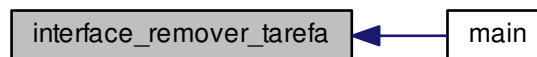
Referências remover_vert().

Referenciado por main().

Este é o diagrama das funções utilizadas por esta função:



Este é o diagrama das funções que utilizam esta função:



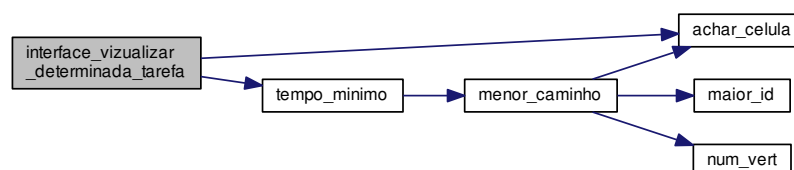
4.4.2.6 void interface_vizualizar_determinada_tarefa (const grafo_priv_t * meu_grafo)

Definição na linha 187 do arquivo interface.cpp.

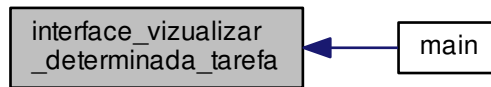
Referências `achar_celula()`, `Celula_priv::duracao`, `Celula_priv::executada`, `Celula_priv::id_externo`, `Celula_priv::ini_min`, `Celula_priv::nome`, `Celula_priv::pre_req`, `Celula_priv::reqs` e `tempo_minimo()`.

Referenciado por `main()`.

Este é o diagrama das funções utilizadas por esta função:



Este é o diagrama das funções que utilizam esta função:



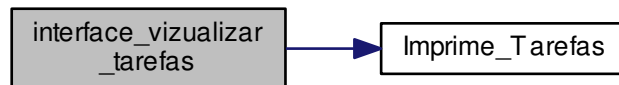
4.4.2.7 void interface_vizualizar_tarefas (const grafo_priv_t * meu_grafo)

Definição na linha 170 do arquivo interface.cpp.

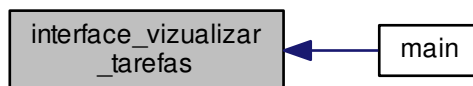
Referências `Imprime_Tarefas()`.

Referenciado por `main()`.

Este é o diagrama das funções utilizadas por esta função:



Este é o diagrama das funções que utilizam esta função:

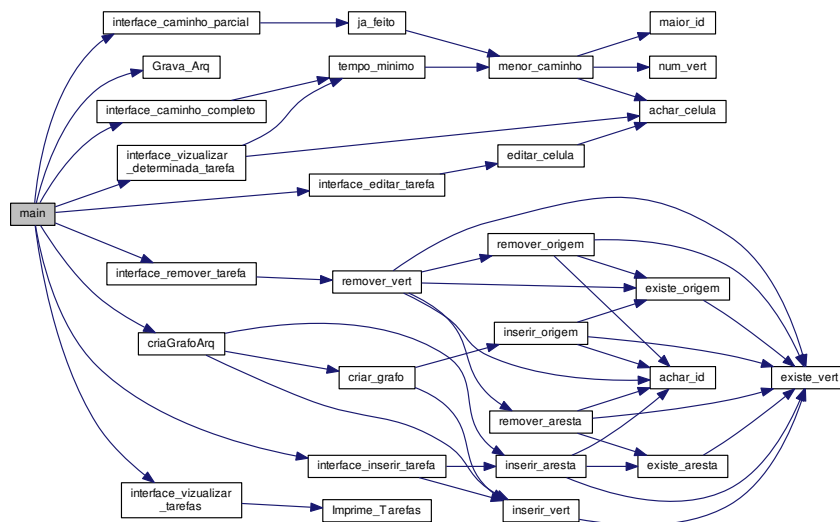


4.4.2.8 int main ()

Definição na linha 225 do arquivo interface.cpp.

Referências `criaGrafoArq()`, `Grava_Arq()`, `interface_caminho_completo()`, `interface_caminho_parcial()`, `interface_↔_editar_tarefa()`, `interface_inserir_tarefa()`, `interface_remover_tarefa()`, `interface_vizualizar_determinada_tarefa()` e `interface_vizualizar_tarefas()`.

Este é o diagrama das funções utilizadas por esta função:



Índice Remissivo

/home/rafael/ProjetoFinalIMP/Fase2Interface/include/grafo.
h, 13
/home/rafael/ProjetoFinalIMP/Fase2Interface/include/grafo.
_priv.h, 34

achar_celula
 grafo.cpp, 39
 grafo.h, 15

achar_id
 grafo.cpp, 40
 grafo.h, 16

antecessores
 lista_vert, 10

Celula_priv, 5
 duracao, 5
 executada, 5
 id_externo, 5
 ini_min, 6
 nome, 6
 pre_req, 6
 reqs, 6

Celula_priv_t
 grafo.h, 14
 grafo_priv.h, 36

cria_celula
 grafo.cpp, 41
 grafo.h, 17

criaGrafoArq
 grafo.cpp, 42
 grafo.h, 17

criar_grafo
 grafo.cpp, 42
 grafo.h, 18

DEBUG
 grafo.cpp, 39

dado
 lista_vert_codigo, 11

deletar_grafo
 grafo.cpp, 43
 grafo.h, 19

destino
 lista_aresta, 8
 lista_origem, 9

dfs
 grafo.cpp, 44

duracao
 Celula_priv, 5

 grafo.cpp, 44
 grafo.h, 20

eh_conexo
 grafo.cpp, 45
 grafo.h, 20

executada
 Celula_priv, 5

existe_aresta
 grafo.cpp, 46
 grafo.h, 22

existe_origem
 grafo.cpp, 47
 grafo_priv.h, 36

existe_vert
 grafo.cpp, 48
 grafo.h, 23

FALSE_T
 grafo.h, 15

grafo.cpp, 38
 achar_celula, 39
 achar_id, 40
 cria_celula, 41
 criaGrafoArq, 42
 criar_grafo, 42
 DEBUG, 39
 deletar_grafo, 43
 dfs, 44
 editar_celula, 44
 eh_conexo, 45
 existe_aresta, 46
 existe_origem, 47
 existe_vert, 48
 Grava_Arq, 49
 Imprime_Tarefas, 49
 inserir_aresta, 49
 inserir_origem, 50
 inserir_vert, 51
 ja_feito, 52
 maior_id, 53
 menor_caminho, 54
 num_arestas, 55
 num_vert, 55
 remover_aresta, 56
 remover_origem, 57
 remover_vert, 58
 tempo_minimo, 59

grafo.h

- achar_celula, 15
- achar_id, 16
- Celula_priv_t, 14
- cria_celula, 17
- criaGrafoArq, 17
- criar_grafo, 18
- deletar_grafo, 19
- editar_celula, 20
- eh_conexo, 20
- existe_aresta, 22
- existe_vert, 23
- FALSE_T, 15
- grafo_priv_t, 14
- Grava_Arq, 24
- Imprime_Tarefas, 24
- inserir_aresta, 25
- inserir_vert, 26
- ja_feito, 27
- Ler_Tarefas, 28
- maior_id, 28
- menor_caminho, 29
- num_arestas, 30
- num_vert, 30
- remover_aresta, 31
- remover_vert, 32
- resp, 15
- resposta, 15
- TRUE_T, 15
- tempo_minimo, 33
- grafo_priv, 6
 - origem, 7
 - tabela, 7
 - vert, 7
- grafo_priv.h
 - Celula_priv_t, 36
 - existe_origem, 36
 - grafo_priv_t, 36
 - inserir_origem, 36
 - lista_aresta_t, 36
 - lista_origem_t, 36
 - lista_vert_codigo_t, 36
 - lista_vert_t, 36
 - remover_origem, 37
- grafo_priv_t
 - grafo.h, 14
 - grafo_priv.h, 36
- Grava_Arq
 - grafo.cpp, 49
 - grafo.h, 24
- id
 - lista_vert, 10
 - lista_vert_codigo, 12
- id_externo
 - Celula_priv, 5
 - lista_vert, 10
- Imprime_Tarefas
 - grafo.cpp, 49
 - grafo.h, 24
- ini_min
 - Celula_priv, 6
- inserir_aresta
 - grafo.cpp, 49
 - grafo.h, 25
- inserir_origem
 - grafo.cpp, 50
 - grafo_priv.h, 36
- inserir_vert
 - grafo.cpp, 51
 - grafo.h, 26
- interface.cpp, 60
 - interface_caminho_completo, 61
 - interface_caminho_parcial, 61
 - interface_editar_tarefa, 62
 - interface_inserir_tarefa, 63
 - interface_remover_tarefa, 63
 - interface_vizualizar_determinada_tarefa, 64
 - interface_vizualizar_tarefas, 65
 - main, 65
 - NCURSES_CONST, 61
- interface_caminho_completo
 - interface.cpp, 61
- interface_caminho_parcial
 - interface.cpp, 61
- interface_editar_tarefa
 - interface.cpp, 62
- interface_inserir_tarefa
 - interface.cpp, 63
- interface_remover_tarefa
 - interface.cpp, 63
- interface_vizualizar_determinada_tarefa
 - interface.cpp, 64
- interface_vizualizar_tarefas
 - interface.cpp, 65
- ja_feito
 - grafo.cpp, 52
 - grafo.h, 27
- Ler_Tarefas
 - grafo.h, 28
- lista_aresta, 7
 - destino, 8
 - next, 8
 - peso, 8
- lista_aresta_t
 - grafo_priv.h, 36
- lista_origem, 9
 - destino, 9
 - next, 9
- lista_origem_t
 - grafo_priv.h, 36
- lista_vert, 9
 - antecessores, 10
 - id, 10
 - id_externo, 10
 - next, 10
 - sucessores, 11

lista_vert_codigo, 11
 dado, 11
 id, 12
 next, 12
lista_vert_codigo_t
 grafo_priv.h, 36
lista_vert_t
 grafo_priv.h, 36

main
 interface.cpp, 65
maior_id
 grafo.cpp, 53
 grafo.h, 28
menor_caminho
 grafo.cpp, 54
 grafo.h, 29

NCURSES_CONST
 interface.cpp, 61
next
 lista_aresta, 8
 lista_origem, 9
 lista_vert, 10
 lista_vert_codigo, 12
nome
 Celula_priv, 6
num_arestas
 grafo.cpp, 55
 grafo.h, 30
num_vert
 grafo.cpp, 55
 grafo.h, 30

origem
 grafo_priv, 7

peso
 lista_aresta, 8
pre_req
 Celula_priv, 6

remover_aresta
 grafo.cpp, 56
 grafo.h, 31
remover_origem
 grafo.cpp, 57
 grafo_priv.h, 37
remover_vert
 grafo.cpp, 58
 grafo.h, 32
reqs
 Celula_priv, 6
resp
 grafo.h, 15
resposta
 grafo.h, 15

sucessores
 lista_vert, 11

TRUE_T
 grafo.h, 15
tabela
 grafo_priv, 7
tempo_minimo
 grafo.cpp, 59
 grafo.h, 33

vert
 grafo_priv, 7