UNIVERSIDADE DE BRASÍLIA
Departamento de Ciência da Computação
Instituto de Ciências Exatas
Disciplina: Métodos de Programação
Aluno: Paulo da Cunha Passos Matrícula: 10/0118577

TRABALHO 1

Descrição: A presente parte do trabalho apresenta a descrição das funções da biblioteca Grafo, com seus respectivos parâmetros e retornos e apresenta os testes realizados por meio do framework CUnit para validar a biblioteca, conforme solicitação dos itens 1 e 2 da descrição do trabalho.

Quanto ao questionamento do item 3 do trabalho:
3) Responda em que casos a função não retornar um resultado válido.
Na biblioteca buscou-se estruturar as funções para que sempre retornem valores válidos, mesmo que seja indicando que a função não realizou o que se esperava ou que tenha recebido parâmetros com valores inválidos.
3.1) Existem funções que podem corromper a estrutura de dados? Como?
Na biblioteca, as funções que recebem ponteiros como parâmetro podem corromper a estrutura de dados, caso sejam passados as funções sem serem inicializados.
3.2) O que pode ser feito para evitar este problema?
Necessariamente tem que se inicializar os ponteiros antes de passa-los como parâmetros para as funções.

FUNÇÕES E SEUS TESTES

Função: Grafo *criaGrafo(char *nome)

Descrição da função:
Cria uma estrutura Grafo e lhe atribui um nome.

Parâmetro *nome - Nome do grafo
Returno g – Ponteiro para estrutura de dados do tipo Grafo

Função Teste: void testaCriaGrafo(void)

- Descrição:
Testa se a função criaGrafo está funcionando corretamente.
Chama a função criaGrafo e atribui seu valor de retorno a um ponteiro Grafo *grafoT, e por meio dele chama as assertivas de teste.

- Assertivas:
1. CU_ASSERT_PTR_NOT_NULL(grafoT):
   - Teste: Se o retorno da função criaGrafo não é um ponteiro nulo.
   - Entrada: Ponteiro grafoT
   - Saída: 1 ou 0

- Critério para passar no teste: retornar 1

- A sua função efetivamente passou no teste: Sim

2. CU_ASSERT_EQUAL(grafoT->ordem, 0):
   - Teste: Se foi atribuído 0 para a ordem inicial do Grafo.
   - Entrada: Ponteiro grafoT->ordem e 0

   - Saída: 1 ou 0

   - Critério para passar no teste: retornar 1

   - A sua função efetivamente passou no teste: Sim
3. CU_ASSERT_EQUAL(grafoT->tamanho, 0):
   - Teste: Se foi atribuído 0 para o tamanho inicial do Grafo.

   - Entrada: Ponteiro grafoT->tamanho e 0

   - Saída: 1 ou 0

   - Critério para passar no teste: retornar 1

   - A sua função efetivamente passou no teste: Sim

4. CU_ASSERT_PTR_NULL(grafoT->nolnicio):
   - Teste: Se foi atribuído NULL para o ponteiro nolnicio do Grafo.

   - Entrada: Ponteiro grafoT->nolnicio

   - Saída: 1 ou 0

   - Critério para passar no teste: retornar 1

   - A sua função efetivamente passou no teste: Sim

5. CU_ASSERT_STRING_EQUAL(grafoT->nome, "GrafoTeste"):
   - Teste: Se foi atribuído a string "GrafoTeste" para o nome do Grafo.

   - Entrada: Ponteiro grafoT->nome e string "GrafoTeste"

   - Saída: 1 ou 0

   - Critério para passar no teste: retornar 1

   - A sua função efetivamente passou no teste: Sim

Função: int constroiGrafo(Grafo *g, char *arquivo)

Descrição da função:

Constrói um grafo a partir de um arquivo contido no diretório com os dados dos Nos e Arestas

Parâmetro *g - Ponteiro para a estrutura do grafo.
*arquivo - String com o nome do arquivo de onde se buscara os dados dos Nos e Arestas.

Returno 0 - Caso construa o Grafo com sucesso.
1 - Caso não construa o Grafo com sucesso.

Função Teste: void testaConstroiGrafo(void)

- Descrição:
Testa se um grafo foi construído com sucesso a partir de um arquivo de testo

- Assertivas:

1. CU_ASSERT_FALSE(constroiGrafo(grafoT, ""))
CU_ASSERT_FALSE(constroiGrafo(grafoT, "t2.txt"));
- Teste: Se o ao passar um nome de arquivo inválido ou que não exista no diretório a função retorna 0.
- Entrada: Ponteiro *grafoT e String “” ou “t2.txt”
- Saída: 1 ou 0
- Critério para passar no teste: retornar 1
- A sua função efetivamente passou no teste: Sim
2. CU_ASSERT_TRUE(constroiGrafo(grafoT, "t1.txt")):
- Teste: Se o ao passar um nome de arquivo válido e que exista a função retorna 1.
- Entrada: Ponteiro *grafoT e String “t1.txt”
- Saída: 1 ou 0
- Critério para passar no teste: retornar 1
- A sua função efetivamente passou no teste: Sim
3. CU_ASSERT_EQUAL(grafoT->ordem, 5);
- Teste: Se a ordem do grafo criado foi incrementada para 5.
- Entrada: grafoT->ordem e 5
- Saída: 1 ou 0
- Critério para passar no teste: retornar 1
- A sua função efetivamente passou no teste: Sim
4. CU_ASSERT_EQUAL(grafoT->tamanho, 12):
- Teste: Se o tamanho do grafo criado foi incrementado para 12.
- Entrada: grafoT->tamanho e 12
- Saída: 1 ou 0
- Critério para passar no teste: retornar 1
- A sua função efetivamente passou no teste: Sim
5. CU_ASSERT_PTR_NOT_NULL(grafoT->noInicio):
- Teste: Se o nó de inicio da lista de nós do grafo não é NULL.

- Entrada: grafoT->noInicio

- Saída: 1 ou 0

- Critério para passar no teste: retornar 1

- A sua função efetivamente passou no teste: Sim

Função: char *retornaNomeGrafo(Grafo *g)

Descrição da função:
Retorna o nome de um grafo.

Parâmetro *g - Ponteiro para a estrutura grafo a qual deve retornar o nome.

Returno nome - Nome do grafo g.
NULL - Retorno se o Grafo g não existe.

Função Teste: void testaRetornaNomeGrafo(void)

- Descrição:
Testa se a função *retornaNomeGrafo retorna o nome do grafo g.

- Assertivas:
1. CU_ASSERT_STRING_EQUAL(retornaNomeGrafo(grafoT1), "GrafoTeste"):
- Teste: Se passado um ponteiro que aponta para a estrutura de um grafo a função retorna o nome do grafo.

- Entrada: Ponteiro grafoT1

- Saída: 1 ou 0

- Critério para passar no teste: retornar 1

- A sua função efetivamente passou no teste: Sim

2. CU_ASSERT_PTR_NULL(retornaNomeGrafo(grafoT2)):
- Teste: Se passado um ponteiro NULL a função retorna NULL.

- Entrada: Ponteiro grafoT2

- Saída: 1 ou 0

- Critério para passar no teste: retornar 1

- A sua função efetivamente passou no teste: Sim

Função: int destroiGrafo(Grafo *g)

Descrição da função:
Destrói um grafo liberando o espaço de memória.

Parâmetro *g - Ponteiro para a estrutura grafo o qual deve ser destruído.

Returno 0 - Se não destruiu o Grafo.
1 - Se destruiu o Grafo.

Função Teste: void testaDestroiGrafo(void)

- Descrição:
Testa se a função destroiGrafo destrói um grafo.

- Assertivas:
1. CU_ASSERT_EQUAL(zeraGrafo(grafoT1), 1);:
- Teste: Se passado um ponteiro que aponta para a estrutura de um grafo a função destrói o grafo e retorna 1.
- Entrada: Ponteiro grafoT1 e 1
- Saída: 1 ou 0
- Critério para passar no teste: retornar 1
- A sua função efetivamente passou no teste: Sim
2. CU_ASSERT_EQUAL(zeraGrafo(grafoT2), 0):
- Teste: Se passado um ponteiro NULL a função retorna 0.
- Entrada: Ponteiro grafoT2 e 0
- Saída: 1 ou 0
- Critério para passar no teste: retornar 1
- A sua função efetivamente passou no teste: Sim

Função: int zeraGrafo(Grafo *g)

Descrição da função:
Limpa um grafo deixando-o sem nenhum No ou Aresta.

Parâmetro *g - Ponteiro para a estrutura grafo o qual deve ser esvaziado.

Returno 0 - Se não zerou o Grafo.
1 - Se zerou o Grafo.

Função Teste: void testaZeraGrafo(void)

- Descrição:
Testa se a função ZeraGrafo deixa um grafo vazio.

- Assertivas:

1. CU_ASSERT_EQUAL(zeraGrafo(grafoT1), 1):
   - Teste: Se passado um ponteiro que aponta para a estrutura de um grafo a função esvazio o grafo e retorna 1.

   - Entrada: Ponteiro grafoT1 e 1

   - Saída: 1 ou 0

   - Critério para passar no teste: retornar 1

   - A sua função efetivamente passou no teste: Sim

2. CU_ASSERT_EQUAL(zeraGrafo(grafoT2), 0):
   - Teste: Se passado um ponteiro NULL a função retorna 0.

   - Entrada: Ponteiro grafoT2 e 0

   - Saída: 1 ou 0

   - Critério para passar no teste: retornar 1

   - A sua função efetivamente passou no teste: Sim

Função: No *retornaNo(Grafo *g, int nold)

Descrição da função:
Percorre o grafo G em procura de um nó que tenha id == nold e retorna um ponteiro para esse nó.

Parâmetro *g - Ponteiro para a estrutura grafo no qual deve-se procurar o nó.
nold - Identificador do nó procurado.

Returno no1 - Ponteiro para o nó procurado.
NULL - Retorno caso o ponteiro não seja encontrado.

Função Teste: void testaRetornaNo(void)

- Descrição:
Testa se a função RetornaNo retorna o nó procurado.

- Assertivas:
1. CU_ASSERT_PTR_NOT_NULL(retornaNo(grafoT1, 1));
   CU_ASSERT_EQUAL(retornaNo(grafoT1, 1)->id, 1);
   CU_ASSERT_EQUAL(retornaNo(grafoT1, 1)->dado, 'a'):

   - Teste: Se retorna o nó se o id passado for referente ao primeiro nó da lista de nós do grafo.

   - Entrada: Ponteiro grafoT1 e 1

   - Saída: 1 ou 0

   - Critério para passar no teste: retornar 1

   - A sua função efetivamente passou no teste: Sim

2. CU_ASSERT_PTR_NOT_NULL(retornaNo(grafoT1, 4));
CU_ASSERT_EQUAL(retornaNo(grafoT1, 4)->id, 4);
CU_ASSERT_EQUAL(retornaNo(grafoT1, 4)->dado, 'd');
- Teste: Se retorna o nó se o id passado for referente a um nó no meio da lista de nós do grafo.
- Entrada: Ponteiro grafoT1 e 4
- Saída: 1 ou 0
- Critério para passar no teste: retornar 1
- A sua função efetivamente passou no teste: Sim

3. CU_ASSERT_PTR_NOT_NULL(retornaNo(grafoT1, 5));
CU_ASSERT_EQUAL(retornaNo(grafoT1, 5)->id, 5);
CU_ASSERT_EQUAL(retornaNo(grafoT1, 5)->dado, 'f'):
- Teste: Se retorna o nó se o id passado for referente ao ultimo nó da lista de nós do grafo.
- Entrada: Ponteiro grafoT1 e 5
- Saída: 1 ou 0
- Critério para passar no teste: retornar 1
- A sua função efetivamente passou no teste: Sim

4. CU_ASSERT_PTR_NULL(retornaNo(grafoT1, 0)):
- Teste: Se retorna NULL ao se passar para a função o id de um nó que não está na lista de nós do grafo.
- Entrada: Ponteiro grafoT1 e 0
- Saída: 1 ou 0
- Critério para passar no teste: retornar 1
- A sua função efetivamente passou no teste: Sim

5. CU_ASSERT_PTR_NULL(retornaNo(grafoT2, 0));
- Teste: Se retorna NULL ao se passar para a função um ponteiro que aponta para um grafo que não existe.
- Entrada: Ponteiro grafoT2 e 0
- Saída: 1 ou 0
- Critério para passar no teste: retornar 1
- A sua função efetivamente passou no teste: Sim

Função: int adicionaVertice(Grafo *g, char dado)

Descrição da função:
Adiciona um no/vértice ao Grafo G.

Parâmetro *g - Ponteiro para a estrutura grafo que receberá o nó.
dado - Valor do No que será adicionado ao grafo G

Returno 0 - Se não adicionou o vértice.
1 - Se adicionou o vértice.

Função Teste: void testaAdicionaVertice(void)

- Descrição:
Testa se a função adicionaVertice adiciona um nó corretamente no grafo indicado.

- Assertivas:
1. CU_ASSERT_EQUAL(adicionaVertice(grafoT2, 'a'), 0)
- Teste: Se ao tentar adicionar um nó em um grafo não existente a função retorna 0.
- Entrada: Ponteiro grafoT2 e 'a'
- Saída: 1 ou 0
- Critério para passar no teste: retornar 1
- A sua função efetivamente passou no teste: Sim

2. CU_ASSERT_EQUAL(adicionaVertice(grafoT1, 'a'), 1);
CU_ASSERT_EQUAL(grafoT1->contId, 1);
CU_ASSERT_EQUAL(grafoT1->ordem, 1);
CU_ASSERT_EQUAL(grafoT1->tamanho, 0);
CU_ASSERT_PTR_NOT_NULL(grafoT1->nolnicio);
- Teste: Se ao tentar adicionar um nó de valor 'a' a um grafo vazio são incrementados os valores de contId e ordem para 1, se nolnicio recebe o endereço para o nó adicionado e se tamanho permanece igual a 0.
- Entrada: Ponteiro grafoT1 e 'a'
- Saída: 1 ou 0
- Critério para passar no teste: retornar 1
- A sua função efetivamente passou no teste: Sim

3. CU_ASSERT_EQUAL(adicionaVertice(grafoT2, 't'), 1);
CU_ASSERT_EQUAL(grafoT2->ordem, 6);
CU_ASSERT_EQUAL(grafoT2->contId, 6);
CU_ASSERT_EQUAL(grafoT2->tamanho, 12);
CU_ASSERT_PTR_NOT_NULL(grafoT2->nolnicio);
- Teste: Se ao tentar adicionar um nó de valor 't' a um grafo de ordem = 5, contId = 6 e tamanho = 12 serão incrementados os valores de contId e ordem para 6, se nolnicio recebe o endereço para o nó adicionado e se tamanho permanece igual a 12.
- Entrada: Ponteiro grafoT2 e 't'
- Saída: 1 ou 0
- Critério para passar no teste: retornar 1

- A sua função efetivamente passou no teste: Sim

Função: int removeVertice(Grafo *g, int nold)

Descrição da função:

Remove um no/vertice do Grafo G.

Parâmetro *g – Ponteiro para o grafo do qual será removido o no.
nold1 - Identificador do No que será removido.

Returno 0 - Se não removeu o vértice por não existir o Grafo ou o No.
1 - Se removeu o vértice.

Função Teste: void testaRemoveVertice(void){

- Descrição:

Testa se a função removeVertice remove um nó corretamente no grafo indicado.

- Assertivas:

1. CU_ASSERT_EQUAL(removeVertice(grafoT2, 1), 0):
   - Teste: Se ao tentar remover um nó em um grafo não existente a função retorna 0.
   - Entrada: Ponteiro grafoT2 e 1
   - Saída: 1 ou 0
   - Critério para passar no teste: retornar 1
   - A sua função efetivamente passou no teste: Sim
2. CU_ASSERT_EQUAL(removeVertice(grafoT1, 1), 0);
   CU_ASSERT_EQUAL(grafoT1->ordem, 0);
   CU_ASSERT_EQUAL(grafoT1->contId, 0);
   CU_ASSERT_EQUAL(grafoT1->tamanho, 0);
   CU_ASSERT_PTR_NULL(grafoT1->noInicio);
   - Teste: Se ao tentar remover um nó em um grafo vazio a função retorna 0 e a ordem, contId e tamanho permanecem com valor 0 e noInicio com valor NULL.
   - Entrada: Ponteiro grafoT1 e 1
   - Saída: 1 ou 0
   - Critério para passar no teste: retornar 1
   - A sua função efetivamente passou no teste: Sim
3. CU_ASSERT_EQUAL(removeVertice(grafoT1, 8), 0);
   CU_ASSERT_EQUAL(grafoT1->ordem, 5);
   CU_ASSERT_EQUAL(grafoT1->contId, 5);
   CU_ASSERT_EQUAL(grafoT1->tamanho, 12);
   CU_ASSERT_PTR_NOT_NULL(grafoT1->noInicio);

- Teste: Se ao tentar remover um nó que não existe em um grafo a função retorna 0 e a ordem, contId, tamanho permanecem e nolnicio permanecem inalterados.

- Entrada: Ponteiro grafoT1 e 1

- Saída: 1 ou 0

- Critério para passar no teste: retornar 1

- A sua função efetivamente passou no teste: Sim

4. CU_ASSERT_EQUAL(removeVertice(grafoT1, 1), 1);
CU_ASSERT_EQUAL(grafoT1->ordem, 4);
CU_ASSERT_EQUAL(grafoT1->contId, 5);
CU_ASSERT_EQUAL(grafoT1->tamanho, 6);
CU_ASSERT_PTR_NOT_NULL(grafoT1->nolnicio);
CU_ASSERT_EQUAL(retornaNo(grafoT1, 2)->numVizinhos, 3);
CU_ASSERT_EQUAL(retornaNo(grafoT1, 5)->numVizinhos, 0);
CU_ASSERT_EQUAL(retornaNo(grafoT1, 4)->numVizinhos, 2);

- Teste: Se ao tentar remover um nó que existe em um grafo a função retorna 1 e a ordem é decrementada, o contId permanece igual, o tamanho é decrementado da quantidade de arestas que tem origem ou destino o nó removido.

- Entrada: Ponteiro grafoT1 e 1

- Saída: 1 ou 0

- Critério para passar no teste: retornar 1

- A sua função efetivamente passou no teste: Sim

Função: Aresta *retornaAresta(No *x, No *y)

Descrição da função:

Retorna uma aresta que tenha como origem o No x e destino o No y.

Parâmetro *x - Ponteiro para o No de origem.
*y - Ponteiro para o No de destino.

Retorno NULL - Se não existir aresta entre x e y.
NULL - Se algum dos Nós não existe.
pare - Ponteiro para a aresta de x para y.

Função Teste: void testaRetornaAresta(void)

- Descrição:

Testa se a função retornaAresta retorna corretamente uma aresta de x para y.

- Assertivas:

1. x = retornaNo(grafoT1, 8);
y = retornaNo(grafoT1, 10);
CU_ASSERT_PTR_NULL(retornaAresta(x, y));

```
CU_ASSERT_EQUAL(grafoT1->ordem, 5);
CU_ASSERT_EQUAL(grafoT1->contId, 5);
CU_ASSERT_EQUAL(grafoT1->tamanho, 12);
    CU_ASSERT_PTR_NOT_NULL(grafoT1->nolnicio);
```

- Teste: Se ao tentar encontrar uma aresta, tal que um dos Nos ou ambos não pertença no Grafo, esta retorna NULL, bem como não altera os valores da ordem, tamanho, contId e nolnicio.

- Entrada: Nó x e Nó y

- Saída: 1 ou 0

- Critério para passar no teste: retornar 1

- A sua função efetivamente passou no teste: Sim

2.
```
x = retornaNo(grafoT1, 1);
y = retornaNo(grafoT1, 5);
CU_ASSERT_PTR_NULL(retornaAresta(x, y));
CU_ASSERT_EQUAL(grafoT1->ordem, 5);
CU_ASSERT_EQUAL(grafoT1->contId, 5);
CU_ASSERT_EQUAL(grafoT1->tamanho, 12);
    CU_ASSERT_PTR_NOT_NULL(grafoT1->nolnicio);
```

- Teste: Se ao tentar encontrar uma aresta, tal que esta aresta não exista no Grafo, a função retorna NULL, bem como não altera os valores da ordem, tamanho, contId e nolnicio.

- Entrada: Nó x e Nó y

- Saída: 1 ou 0

- Critério para passar no teste: retornar 1

- A sua função efetivamente passou no teste: Sim

3.
```
x = retornaNo(grafoT1, 2);
y = retornaNo(grafoT1, 1);
CU_ASSERT_PTR_NOT_NULL(retornaAresta(x, y));
CU_ASSERT_EQUAL(grafoT1->ordem, 5);
CU_ASSERT_EQUAL(grafoT1->contId, 5);
CU_ASSERT_EQUAL(grafoT1->tamanho, 12);
  CU_ASSERT_PTR_NOT_NULL(grafoT1->nolnicio);
```

- Teste: Se ao tentar encontrar uma aresta, tal que esta aresta exista no Grafo, a função retorne a aresta procurada, bem como não altera os valores da ordem, tamanho, contId e nolnicio.

- Entrada: Nó x e Nó y

- Saída: 1 ou 0

- Critério para passar no teste: retornar 1

- A sua função efetivamente passou no teste: Sim

Função: int adicionaAresta(Grafo *g, int nold1, int nold2, int valorAresta)
Descrição da função:

Adiciona uma aresta que tenha como origem o nold1 e destino o nold2.

Parâmetro *g - Ponteiro para o grafo g.
*x - Ponteiro que aponta para o No de origem da Aresta.
*y - Ponteiro que aponta para o No de destino da Aresta.
valorAresta - Valor/peso da aresta que liga os dois nos.

Retorno 0 - Se não foi possível adicionar a Aresta.
1 - Se a Aresta foi adicionada com sucesso.

Função Teste: void testaAdicionaAresta(void)
- Descrição:

Testa se a função adicionaAresta adiciona uma aresta corretamente.

- Assertivas:

1. CU_ASSERT_EQUAL(adicionaAresta(grafoT1, 8, 10, 10), 0);
   CU_ASSERT_EQUAL(grafoT1->ordem, 5);
   CU_ASSERT_EQUAL(grafoT1->contId, 5);
   CU_ASSERT_EQUAL(grafoT1->tamanho, 12);
   CU_ASSERT_PTR_NOT_NULL(grafoT1->noInicio);
   - Teste: Se ao tentar adicionar uma aresta entre Nós, tal que um dos Nós ou ambos não pertencem ao Grafo, a função não adiciona a aresta e não altera os valores da ordem, tamanho, contId e noInicio.
   - Entrada: Ponteiro grafoT1, Id do Nó x, Id do Nó y, Valor da Aresta
   - Saída: 1 ou 0
   - Critério para passar no teste: retornar 1
   - A sua função efetivamente passou no teste: Sim

2. CU_ASSERT_EQUAL(adicionaAresta(grafoT1, 3, 1, 10), 0);
   CU_ASSERT_EQUAL(grafoT1->ordem, 5);
   CU_ASSERT_EQUAL(grafoT1->contId, 5);
   CU_ASSERT_EQUAL(grafoT1->tamanho, 12);
   CU_ASSERT_PTR_NOT_NULL(grafoT1->noInicio);
   - Teste: Se ao tentar adicionar uma aresta entre os Nós, tal que os Nós pertençam ao Grafo e já exista uma aresta entre os Nós, a função não adiciona a aresta e não altera os valores da ordem, tamanho, contId e noInicio.
   - Entrada: Ponteiro grafoT1, Id do Nó x, Id do Nó y, Valor da Aresta
   - Saída: 1 ou 0
   - Critério para passar no teste: retornar 1
   - A sua função efetivamente passou no teste: Sim
3. CU_ASSERT_EQUAL(adicionaAresta(grafoT1, 4, 5, 10), 1);

CU_ASSERT_EQUAL(grafoT1->ordem, 5);
CU_ASSERT_EQUAL(grafoT1->contId, 5);
CU_ASSERT_EQUAL(grafoT1->tamanho, 13);
CU_ASSERT_PTR_NOT_NULL(grafoT1->noInicio);

- Teste: Se ao tentar adicionar uma aresta entre os Nós, tal que os Nós pertençam ao Grafo mas não exista aresta entre eles, a função adiciona a aresta e não altera os valores da ordem, contId e noInicio, mas incremente o tamanho do grafo.
- Entrada: Ponteiro grafoT1, Id do Nó x, Id do Nó y, Valor da Aresta
- Saída: 1 ou 0
- Critério para passar no teste: retornar 1
- A sua função efetivamente passou no teste: Sim

Função: int removeAresta(Grafo *g, int noId1, int noId2)

Descrição da função:

Remove uma aresta que liga No noId1 de origem a seu adjacente No noId2.
Essa funcao parte do presuposto que existe somente uma aresta de noId1 para noId2.

Parâmetro *g - Ponteiro para o grafo g.
noId1 - Identificador do No de origem.
noId2 - Identificador do No de destino da Aresta.

Retorno 0 - Se não foi possível remover a Aresta.
1 - Se a Aresta foi removida com sucesso.

Função Teste: void testaRemoveAresta(void)
- Descrição:

Testa se a função removeAresta remove uma aresta do grafo corretamente.

- Assertivas:

1. CU_ASSERT_EQUAL(removeAresta(grafoT1, 8, 10), 0);
CU_ASSERT_EQUAL(grafoT1->ordem, 5);
CU_ASSERT_EQUAL(grafoT1->contId, 5);
CU_ASSERT_EQUAL(grafoT1->tamanho, 12);
CU_ASSERT_PTR_NOT_NULL(grafoT1->noInicio);
- Teste: Se ao tentar remover uma aresta entre Nós, tal que um dos Nós ou ambos não pertencem ao Grafo, a função não altera os valores da ordem, tamanho, contId e noInicio.
- Entrada: Ponteiro grafoT1, Id do Nó x e Id do Nó y.
- Saída: 1 ou 0
- Critério para passar no teste: retornar 1
- A sua função efetivamente passou no teste: Sim
2. CU_ASSERT_EQUAL(removeAresta(grafoT1, 4, 3), 1);
CU_ASSERT_EQUAL(grafoT1->ordem, 5);

CU_ASSERT_EQUAL(grafoT1->contId, 5);
CU_ASSERT_EQUAL(grafoT1->tamanho, 11);
CU_ASSERT_PTR_NOT_NULL(grafoT1->nolnicio);

- Teste: Se ao tentar remover uma aresta entre os Nós, tal que os Nós pertençam ao Grafo e exista uma aresta entre eles, a função remove a aresta e não altera os valores da ordem, contId e nolnicio, mas decremente o tamanho do grafo.
- Entrada: Ponteiro grafoT1, Id do Nó x e Id do Nó y.
- Saída: 1 ou 0
- Critério para passar no teste: retornar 1
- A sua função efetivamente passou no teste: Sim

3. CU_ASSERT_EQUAL(removeAresta(grafoT1, 5, 4), 0);

CU_ASSERT_EQUAL(grafoT1->ordem, 5);
CU_ASSERT_EQUAL(grafoT1->contId, 5);
CU_ASSERT_EQUAL(grafoT1->tamanho, 10);
CU_ASSERT_PTR_NOT_NULL(grafoT1->nolnicio);

- Teste: Se ao tentar remover uma aresta entre os Nós, tal que os Nós pertençam ao Grafo e não exista uma aresta entre eles, a função não altera os valores da ordem, contId e nolnicio e tamanho do grafo.
- Entrada: Ponteiro grafoT1, Id do Nó x e Id do Nó y.
- Saída: 1 ou 0
- Critério para passar no teste: retornar 1
- A sua função efetivamente passou no teste: Sim

Função: char retornaValorVertice(Grafo *g, int nold)

Descrição da função:

Retorna o valor associado a um dado vértice do grafo.

Parâmetro *g - Ponteiro para o grafo g.
nold1 - Identificador do No de origem.

Retorno dado - Valor associado ao no de nold1.
0 - O grafo ou o nó não existam.

Função Teste: void testaRetornaValorVertice(void)

- Descrição:

Testa se a função retornaValorVertice retorna o valor correto associado a um Nó do grafo corretamente.

- Assertivas:

1. CU_ASSERT_EQUAL(retornaValorVertice(grafoT1, 1), 0);

- Teste: Se ao passar um grafo que não exista como argumento a função retorna 0.
- Entrada: Ponteiro grafoT1 e 1.

- Saída: 1 ou 0

- Critério para passar no teste: retornar 1

- A sua função efetivamente passou no teste: Sim

2. CU_ASSERT_EQUAL(retornaValorVertice(grafoT1, 10), 0);
   - Teste: Se ao passar o identificador de um nó que não exista no grafo a função retorna 0.
   - Entrada: Ponteiro grafoT1 e 10.
   - Saída: 1 ou 0
   - Critério para passar no teste: retornar 1
   - A sua função efetivamente passou no teste: Sim

3. CU_ASSERT_EQUAL(retornaValorVertice(grafoT1, 2), 'b');
   - Teste: Se ao passar o identificador de um nó que exista no grafo a função retorna o valor do nó.
   - Entrada: Ponteiro grafoT1 e 2.
   - Saída: 1 ou 0
   - Critério para passar no teste: retornar 1
   - A sua função efetivamente passou no teste: Sim

Função: int mudaValorVertice(Grafo *g, int nold, char valor)

Descrição da função:

Muda o valor da variável dado de um vértice que possua identificador nold.

Parâmetro *g - Ponteiro para o grafo g.
nold - Identificador do nó.
Valor – Novo valor do nó.

Retorno 1 - Se o valor foi alterado com sucesso.
0 - Se não houve a alteração do valor.

Função Teste: void testaMudaValorVertice(void)
- Descrição:

Testa se a função mudaValorVertice altera o valor de um vértice com sucesso.

- Assertivas:

1. grafoT1 = NULL;
   CU_ASSERT_FALSE(mudaValorVertice(grafoT1, 10, 'a'));
   - Teste: Se ao tentar de mudar o valor de um vértice, caso o grafo não exista, a função retorna 0.
   - Entrada: Ponteiro grafoT1, 10 e ‘a’.
   - Saída: 1 ou 0
   - Critério para passar no teste: retornar 1
   - A sua função efetivamente passou no teste: Sim

2. grafoT1 = criaGrafo("GrafoTeste");
   CU_ASSERT_FALSE(mudaValorVertice(grafoT1, 10, 'a'));

- Teste: Se ao tentar de mudar o valor de um vértice, caso o vertice não exista no grafo, a função retorna 0.

- Entrada: Ponteiro grafoT1, 10 e 'a'.

- Saída: 1 ou 0

- Critério para passar no teste: retornar 1

- A sua função efetivamente passou no teste: Sim

3. CU_ASSERT_TRUE(mudaValorVertice(grafoT1, 1, 'f'));
CU_ASSERT_EQUAL(retornaNo(grafoT1, 1)->dado, 'f');
CU_ASSERT_TRUE(mudaValorVertice(grafoT1, 5, 'a'));
CU_ASSERT_EQUAL(retornaNo(grafoT1, 5)->dado, 'a');
CU_ASSERT_TRUE(mudaValorVertice(grafoT1, 3, 't'));
CU_ASSERT_EQUAL(retornaNo(grafoT1, 3)->dado, 't');

- Teste: Se ao tentar de mudar o valor de um vértice, do grafo a função faz a alteração do valor e retorna 1.

- Entrada: Ponteiro grafoT1, 1 e 'f'.

- Saída: 1 ou 0

- Critério para passar no teste: retornar 1

- A sua função efetivamente passou no teste: Sim

Função: int retornaValorAresta(Grafo *g, int nold1, int nold2)

Descrição da função:

Retorna o valor associado a uma aresta que tenha origem no nó de identificador nold1 e destino o nó de identificador nold2.

Parâmetro	*g	- Ponteiro para o grafo g.
	nold1	- Identificador do nó de origem.
	nold2	- Identificador do nó de destino
Retorno	0	- Se a aresta não existe a aresta no grafo ou o grafo não exista.
	valor	- Valor/peso da aresta.

Função Teste: void testaRetornaValorAresta(void)

- Descrição:

Testa se a função retornaValorAresta retorna o valor referente a uma aresta do grafo.

- Assertivas:

1. grafoT1 = NULL;

CU_ASSERT_EQUAL(retornaValorAresta(grafoT1, 1, 2), 0);

- Teste: Se ao tentar retornar o valor de uma aresta, caso o grafo não exista, a função retorna 0.
- Entrada: Ponteiro grafoT1, 1 e 2.
- Saída: 1 ou 0
- Critério para passar no teste: retornar 1
- A sua função efetivamente passou no teste: Sim

2. grafoT1 = criaGrafo("GrafoTeste");
   CU_ASSERT_EQUAL(retornaValorAresta(grafoT1, 5, 4), 0);

- Teste: Se ao tentar retornar o valor de uma aresta, caso a aresta não exista no grafo, a função retorna 0.
- Entrada: Ponteiro grafoT1, 5 e 4.
- Saída: 1 ou 0
- Critério para passar no teste: retornar 1
- A sua função efetivamente passou no teste: Sim

3. constroiGrafo(grafoT1, "t1.txt");
   CU_ASSERT_EQUAL(retornaValorAresta(grafoT1, 3, 5), 45);
   - Teste: Se ao tentar retornar o valor de uma aresta que exista no grafo a função retorna o valor da aresta corretamente.
   - Entrada: Ponteiro grafoT1, 3 e 5.
   - Saída: 1 ou 0
   - Critério para passar no teste: retornar 1
   - A sua função efetivamente passou no teste: Sim

Função: int mudaValorAresta(Grafo *g, int nold1, int nold2, int valor)

Descrição da função:

Muda o valor da variável valoraresta de uma aresta que tenha como origem o nó de identificador nold1 e destino o nó de identificador nold2.

Parâmetro	*g	- Ponteiro para o grafo g.
	nold1	- Identificador do nó de origem.
	nold2	- Identificador do nó de destino.
	valor	– Novo valor da aresta.
Retorno	0	- Se não houve a alteração do valor.
	1	- Se o valor foi alterado com sucesso.

Função Teste: void testaMudaValorAresta(void)

- Descrição:

Testa se a função mudaValorAresta realiza a alteração do valor de uma aresta de forma

correta.

- Assertivas:
    1. grafoT1 = NULL;
    CU_ASSERT_FALSE(mudaValorAresta(grafoT1, 10, 1, 10));
    - Teste: Se ao tentar mudar o valor de uma aresta, caso o grafo não exista, a função retorna 0.
  - Entrada: Ponteiro grafoT1, 10, 1 e 10.
  - Saída: 1 ou 0
  - Critério para passar no teste: retornar 1
    - A sua função efetivamente passou no teste: Sim

    2. grafoT1 = criaGrafo("GrafoTeste");
    CU_ASSERT_FALSE(mudaValorAresta(grafoT1, 10, 1, 10));
    - Teste: Se ao tentar mudar o valor de uma aresta, caso a aresta não exista no grafo, a função retorna 0.
  - Entrada: Ponteiro grafoT1, 10, 1 e 10.
  - Saída: 1 ou 0
  - Critério para passar no teste: retornar 1
    - A sua função efetivamente passou no teste: Sim

    3. constroiGrafo(grafoT1, "t1.txt");
       CU_ASSERT_TRUE(mudaValorAresta(grafoT1, 1, 2, 70));
     x = retornaNo(grafoT1, 1);
     y = retornaNo(grafoT1, 2);
   CU_ASSERT_EQUAL(retornaAresta(x, y)->valoraresta, 70);
       - Teste: Se ao tentar mudar o valor de uma aresta que exista no grafo a função altera o valor da aresta corretamente e retorna 1.
       - Entrada: Ponteiro grafoT1, 1, 2 e 70.
       - Saída: 1 ou 0
       - Critério para passar no teste: retornar 1
       - A sua função efetivamente passou no teste: Sim

Função: int *vizinhos(Grafo *g, int nold)

Descrição da função:
Retorna a lista de todos os vértices que podem ser visitados a partir do de identificador nold.

Parâmetro *g - Ponteiro para o grafo g.
nold1 - Identificador do nó.
Retorno viz - Vetor com todos os nós adjacentes a nold.

Função Teste: void testaVizinhos(void)

- Descrição:
Testa se a função vizinhos retorna uma lista contendo todos os nós do nó indicado.

- Assertivas:

1. grafoT2 = NULL;
   CU_ASSERT_PTR_NULL(vizinhos(grafoT2, 1));
   - Teste: Se ao passar um grafo que não exista como parâmetro a função retorna NULL.
   - Entrada: Ponteiro grafoT2, e 1.
   - Saída: 1 ou 0
   - Critério para passar no teste: retornar 1
   - A sua função efetivamente passou no teste: Sim

2. grafoT1 = criaGrafo("GrafoTeste");
   constroiGrafo(grafoT1, "t1.txt");
   p = vizinhos(grafoT1, 2);
   CU_ASSERT_EQUAL(p[0], 3);
   CU_ASSERT_EQUAL(p[1], 1);
   CU_ASSERT_EQUAL(p[2], 5);
   CU_ASSERT_EQUAL(p[3], 4);
   - Teste: Se retorna corretamente a listas de vizinhos do nó 2.
   - Entrada: Ponteiro grafoT1 e 2.
   - Saída: 1 ou 0
   - Critério para passar no teste: retornar 1
   - A sua função efetivamente passou no teste: Sim

3. CU_ASSERT_PTR_NULL(vizinhos(grafoT1, 6));
   - Teste: Se retorna NULL ao se passar um nó que não possua vizinhos.
   - Entrada: Ponteiro grafoT1 e 6.
   - Saída: 1 ou 0
   - Critério para passar no teste: retornar 1
   - A sua função efetivamente passou no teste: Sim

4. CU_ASSERT_PTR_NULL(vizinhos(grafoT1, 7));
   - Teste: Se retorna NULL ao se passar um nó que não exista no grafo.
   - Entrada: Ponteiro grafoT1 e 6.
   - Saída: 1 ou 0
   - Critério para passar no teste: retornar 1
   - A sua função efetivamente passou no teste: Sim

Função: int adjacente(Grafo *g, int nold1, int nold2)

Descrição da função:
Verifica se dois Nos de um grafo G sao adjacentes.

Parâmetro *g - Ponteiro para o grafo g.
nold1 - Identificador do nó 1.
nold2 - Identificador do nó 2.
Retorno 0 - Se os nós não são adjacentes ou algum deles não exista no grafo ou mesmo se o grafo não exista.
1 - Se os nós são adjacentes.

Função Teste: void testaAdjacente(void)
- Descrição:
Testa se a função adjacente funciona corretamente, retornando 1 caso os dois vértices sejam adjacentes ou 0 nos demais casos.

- Assertivas:

1. grafoT2 = NULL;
CU_ASSERT_FALSE(adjacente(grafoT2, 1, 2));
- Teste: Se ao passar um grafo que não exista como parâmetro a função retorna 0.
- Entrada: Ponteiro grafoT2, 1 e 2.
- Saída: 1 ou 0
- Critério para passar no teste: retornar 1
- A sua função efetivamente passou no teste: Sim

2. grafoT1 = criaGrafo("GrafoTeste");
constroiGrafo(grafoT1, "t1.txt");
CU_ASSERT_TRUE(adjacente(grafoT1, 1, 2));
- Teste: Se ao passar um grafo que exista e o identificador de dois nós que são adjacentes a função retorna 1.
- Entrada: Ponteiro grafoT1, 1 e 2.
- Saída: 1 ou 0
- Critério para passar no teste: retornar 1
- A sua função efetivamente passou no teste: Sim

3. CU_ASSERT_FALSE(adjacente(grafoT1, 3, 4));
- Teste: Se ao passar um grafo que exista e o identificador de dois nós que não são adjacentes ou que algum deles ou ambos não exista no grafo a função retorna 0.
- Entrada: Ponteiro grafoT1, 1 e 2.
- Saída: 1 ou 0
- Critério para passar no teste: retornar 1
- A sua função efetivamente passou no teste: Sim