Banco de Testes - Métodos de Programação

Lukas Ferreira Machado

Setembro 08, 2016

Data Realizada:	Setembro 08, 2016
Integrantes:	Lukas Ferreira Machado - 12/0127377
Instrutor:	Professor Jan Correa

1 Introdução

Este documento apresenta a descrição dos testes executados no software em C criado como template para a geração de uma estrutura de dados do tipo Grafo, capaz de representar e manipular um grafo direcionado.

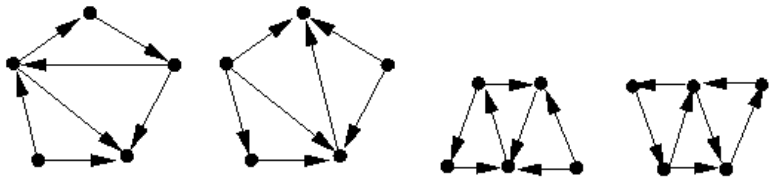

Figure 1: Exemplos de grafos direcionados cíclicos e acíclico

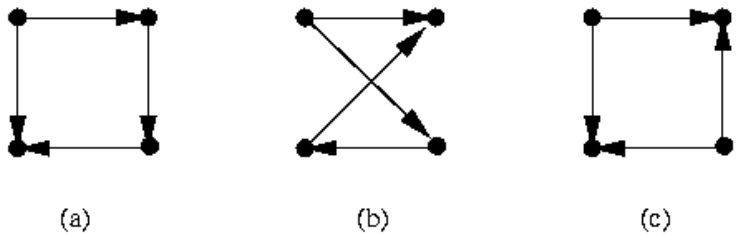

Figure 2: Exemplos de grafos direcionados acíclicos

2 Descrição da Estrutura de Dados Graph

Para a implementação deste projeto, foi criada uma estrutura do tipo graph dada por:

```
typedef struct graph *Graph;

struct graph{
    struct vertex * vertices;
    char * nome;
    float ** arestas;
    int total_vertices;

};

struct vertex{
    int id;
    float valor;
};
```

Um objeto do tipo Graph é um ponteiro para uma estrutura de dados composta por um vetor de vértices (que é uma estrutura do tipo vertex), um vetor para o nome do grafo. Além disso, a estrutura Graph contém um inteiro simbolizando o total de vértices do grafo e uma matriz do tipo float que representa os pesos das arestas do grafo. Quando não existe uma aresta entre dois vértices, o peso atribuído para tal aresta é 0.

Já a estrutura de dados vertex contém um inteiro que identifica cada vértice e um valor atribuído a ele. Quando um vértice é inicializado, o valor atribuído a ele é 0.

3 Testes Executados

3.1 Função cria_grafo:

Função testada: Graph cria_grafo (char * nome);

Descrição: Esta função cria uma estrutura de dados do tipo Graph.

Parâmetros: Ponteiro para um array do tipo char contendo o nome desejado para o grafo.

Casos de erro: A função pode retornar erro apenas se em algum momento no stack de execuç ao funções como **malloc()** ou **realloc()** acusarem acesso indevido de alguma partição de memória ou realocação dinâmica de um tipo abstrato

não alocado por **malloc()**, respectivamente. Para se evitar tal erro, verifica-se sempre, no caso do **malloc()**, se a função ao tentar alocar memória não retornou *NULL* e para o **realloc()**, em alguns casos específicos, pode ocorrer de o mesmo alocar uma outra partição de memória e que na execução do programa pode acabar afetando a referência esperada, onde tal erro é evitado usando uma variável intermediária que recebe o retorno do **realloc()** do tipo abstrato inicial e o mesmo recebe a referência para essa estrutura intermediária.

3.1.1 Teste 1

a. Nome do Teste: Teste 1

b. O que vai ser testado: o objetivo deste teste é verificar se dado um nome para o grafo, ele é corretamente inicializado

c. Entrada: nome do grafo. Neste caso, utilizou-se o nome "Meu Grafo"

d. Saída Esperada: uma estrutura do tipo Graph cujo nome é "Meu Grafo", arestas e vértices são nulos e o total de vértices é zero.

e. Critério para passar no teste: estrutura retornada contém as seguintes caracter'isticas: nome = "Meu Grafo", vértices = NULL, arestas = NULL, total_vertices = 0;

f. Resultado do teste: a função passou no teste, o que pode ser comprovado pela impressão em tela dos dados do grafo recém criado.

g. Script de execução do teste:

```
Graph g = cria_grafo("Meu Grafo");
print_graph(g);
```

3.2 Função retorna_nome_grafo:

Função testada: char * retorna_nome_grafo(Graph);

Descrição: Esta função devolve ao usuário o nome dado ao grafo passado para a função.

Parâmetros: Estrutura de dados do tipo grafo.

Casos de erro: Pode ocorrer um erro se o usuário tentar retornar o nome de uma estrutura do tipo grafo que não foi devidamente inicializada. Para se evitar esse problema, retorna-se uma string com conteúdo '*null*' que mostra ao usuàrio que ele tentou retornar o nome de uma estrutura vazia.

3.2.1 Teste 2

a. Nome do Teste: Teste 2

b. O que vai ser testado: o objetivo deste teste é verificar se dada uma estrutura de dados do tipo Graph, a função retorna corretamente o nome atribuído para o grafo.

c. Entrada: estrutura de dados do tipo Graph. Neste caso, a estrutura analisada pela função é o grafo g criado no teste anterior.

d. Saída Esperada: vetor do tipo char contendo a string "Meu Grafo".

e. Critério para passar no teste: função retorna a string "Meu Grafo".

f. Resultado do teste: a função passou no teste, o que pode ser comprovado pela impressão em tela do nome do grafo.

g. Script de execução do teste:

```
puts(retorna_nome_grafo(g));
```

3.2.2 Teste 3

a. Nome do Teste: Teste 3

b. O que vai ser testado: o objetivo deste teste é verificar o que acontece quando tenta-se verificar o nome de um grafo que não foi inicializado.

c. Entrada: estrutura de dados do tipo Graph. Neste caso, a estrutura analisada pela função é um grafo m não inicializado.

d. Saída Esperada: null.

e. Critério para passar no teste: receber a string "null".

f. Resultado do teste: a função passou no teste, o que pode ser comprovado pela impressão em tela do nome do grafo m.

g. Script de execução do teste:

```
puts(retorna_nome_grafo(m));
```

3.3 Função adiciona_vertice:

Função testada: void adiciona_vertice(Graph, int x);

Descrição: Esta função adiciona um vértice numa estrutura de dados do tipo Graph.

Parâmetros: Estrutura do tipo Graph e um identificador (x) a ser atribuído ao vértice a ser criado.

Casos de Erro: Pode ocorre erro se o usário tentar inserir um vértive já inserido no grafo ou, durante a execução da função, tentarmos inseir um vértice que não foi devidamente inicializado e criado. Utilizando-se o mesmo raciocínio dos testes anteriores, para corrigir tais erros e verficado através de uma função de busca se o vertice a ser inserido já não existe ou se o grafo a ser inserido o vertice foi devidamente inicializado. Em ambos os casos, se ocorrer algum erro, a função apenas dá um *return* e, com isso, tais erros não tem o perigo de comprometer a estrutura de dados.

3.3.1 Teste 4

a. Nome do Teste: Teste 4

b. O que vai ser testado: o objetivo deste teste é verificar se dado um identificador para um vértice, este é criado na estrutura de dados Graph dada.

c. Entrada: estrutura de dados do tipo Graph e um identificador do tipo inteiro.

d. Saída Esperada: uma estrutura do tipo Graph (g) contendo um vértice cujo identificador é 1.

e. Critério para passar no teste: estrutura retornada contém o vértice 1 com valor 0.

f. Resultado do teste: a função passou no teste, o que pode ser comprovado pela impressão em tela dos dados do grafo recém criado.

g. Script de execução do teste:

```
adiciona_vertice(g, 1);
print_graph(g);
```

3.3.2 Teste 5

a. Nome do Teste: Teste 5

b. O que vai ser testado: o objetivo deste teste é verificar o comportamento do software quando se tenta inserir um vértice com identificador existente na estrutura Graph.

c. Entrada: estrutura de dados do tipo Graph e o identificador do tipo inteiro igual a 1.

d. Saída Esperada: uma estrutura do tipo Graph (g) contendo apenas um vértice cujo identificador é 1 (aquele que foi inserido no teste 4).

e. Critério para passar no teste: estrutura retornada contém um vértice 1 com valor 0.

f. Resultado do teste: a função passou no teste, o que pode ser comprovado pela impressão em tela dos dados do grafo recém criado.

g. Script de execução do teste:

```
adiciona_vertice(g, 1);
print_graph(g);
```

3.3.3 Teste 6

a. Nome do Teste: Teste 6

b. O que vai ser testado: o objetivo deste teste é verificar o comportamento do software quando se tenta inserir um vértice numa estrutura do tipo Graph não inicializada.

c. Entrada: estrutura de dados do tipo Graph não inicializada e o identificador do tipo inteiro igual a 1.

d. Saída Esperada: nada deve acontecer. Ou seja, não há como inserir um vértice num grafo não inicializado.

e. Critério para passar no teste: não ocorre erro.

f. Resultado do teste: a função passou no teste, o que pode ser comprovado pela tentativa da impressão em tela dos dados do grafo não inicializado.

g. Script de execução do teste:

```
adiciona_vertice(m, 1);
print_graph(m);
```

Ao final deste teste foram inseridos na estrutura Graph g, os vértices com identificadores 2, 3, 4 e 5.

3.4 Função remove_vertice:

Função testada: void remove_vertice(Graph, int x);

Descrição: Esta função remove um vértice com identificador x de uma estrutura de dados do tipo Graph.

Parâmetros: Estrutura do tipo Graph e o identificador (x) do vértice a ser removido do grafo.

Casos de Erro: Pode ocorrer um erro se tentarmos remover um vertice que não exite ou de um grafo que não foi devidamente inicializado. Para se evitar tais erros, verifica-se primeiramente se o vertice exite no grafo, senão a função sai de sua execucão. Mesmo comportamente se é passado um grafo não inicializado para a função.

3.4.1 Teste 7

a. Nome do Teste: Teste 7

b. O que vai ser testado: o objetivo deste teste é verificar se dado um identificador para um vértice, este é removido da estrutura de dados Graph dada.

c. Entrada: estrutura de dados do tipo Graph e um identificador do tipo inteiro. Neste caso de teste a entrada foi o grafo g e o identificador 1.

d. Saída Esperada: uma estrutura do tipo Graph (g) que não contém um vértice com identificador 1.

e. Critério para passar no teste: estrutura retornada não contém o vértice 1 em sua lista de vértices.

f. Resultado do teste: a função passou no teste, o que pode ser comprovado pela impressão em tela dos dados do grafo recém criado.

g. Script de execução do teste:

```
remove_vertice(g, 1);
print_graph(g);
```

3.4.2 Teste 8

a. Nome do Teste: Teste 8

b. O que vai ser testado: o objetivo deste teste é verificar o comportamento do software quando se tenta remover um vértice cujo identificador não existe na lista de vértices da estrutura de dados Graph dada.

c. Entrada: estrutura de dados do tipo Graph e um identificador do tipo inteiro. Neste caso de teste a entrada foi o grafo g e o identificador 1.

d. Saída Esperada: uma estrutura do tipo Graph (g) que não contém um vértice com identificador 1.

e. Critério para passar no teste: estrutura retornada não contém o v'ertice 1 em sua lista de vértices e não ocorrem erros de execução do software.

f. Resultado do teste: a função passou no teste, o que pode ser comprovado pela impressão em tela dos dados do grafo recém criado.

g. Script de execução do teste:

```
remove_vertice(g, 1);
print_graph(g);
```

3.4.3 Teste 9

a. Nome do Teste: Teste 9

b. O que vai ser testado: o objetivo deste teste é verificar o comportamento do software quando se tenta remover um vértice de uma estrutura de dados Graph não inicializada.

c. Entrada: estrutura de dados do tipo Graph e um identificador do tipo inteiro. Neste caso de teste a entrada foi o grafo m (não inicializado) e o identificador 6.

d. Saída Esperada: nada deve acontecer. Ou seja, não há como remover um vértice num grafo não inicializado.

e. Critério para passar no teste: não ocorre erro.

f. Resultado do teste: a função passou no teste, o que pode ser comprovado pela impressão em tela dos dados do grafo recém criado.

g. Script de execução do teste:

```
remove_vertice(m, 6);
print_graph(m);
```

3.5 Função adiciona_aresta:

Função testada: void adiciona_aresta(Graph, int x, int y);

Descrição: Esta função adiciona uma aresta que flui do vértice de identificador x para o vértice de identificador y em uma estrutura de dados do tipo Graph.

Parâmetros: Estrutura do tipo Graph e os identificador (x) e (y) dos vértices extremos da aresta a ser adicionada no grafo.

Casos de erro: Pode ocorrer um erro se tentarmos adcionar uma aresta com vertice(s) removido(s), que não existe(m) no grafo, uma aresta que já existe ou em uma estrutura tipo grafo não inicializada. Para esses casos, verifica-se se ambos os vertices existem no grafo com funções de busca, onde a função para sua execução e nas mesmas já é verificado se o grafo foi iniciliazado ou não, onde, se não foi, a função para sua execução.

3.5.1 Teste 10

a. Nome do Teste: Teste 10

b. O que vai ser testado: o objetivo deste teste é verificar o comportamento da aplicação quando se tenta inserir uma aresta em que uma de suas

extremidades se refere a um vértice excluído da estrutura de dados Graph dada.

c. Entrada: estrutura de dados do tipo Graph e os identificadores x e y dos dois vértices extremos da aresta a ser inserida na estrutura de grafos dada. A aresta a ser inserida sai de x e chega em y.

d. Saída Esperada: uma estrutura do tipo Graph (g) que não contém uma aresta fluindo do vértice com identificador x e chega no vértice com identificador. Não podem ocorrer erros de execução.

e. Critério para passar no teste: estrutura retornada não contém a aresta x para y e não ocorrem erros de execuccão.

f. Resultado do teste: a função passou no teste, não ocorreram erros de execução e a aresta xy não está presente na estrutura grafo passada para a função.

g. Script de execução do teste:

```
adiciona_aresta(g, 1, 2);
print_graph(g);
```

3.5.2 Teste 11

a. Nome do Teste: Teste 11

b. O que vai ser testado: o objetivo deste teste é verificar o comportamento da aplicação quando se tenta inserir uma aresta entre dois vértices existentes na estrutura dados Graph dada.

c. Entrada: estrutura de dados do tipo Graph e os identificadores x e y dos dois vértices extremos da aresta a ser inserida na estrutura de grafos dada. A aresta a ser inserida sai de x e chega em y.

d. Saída Esperada: uma estrutura do tipo Graph (g) que contém uma aresta fluindo do vértice com identificador x e chega no vértice com identificador y. Não podem ocorrer erros de execução.

e. Critério para passar no teste: estrutura retornada contém a aresta x para y e não ocorrem erros de execuccão.

f. Resultado do teste: a função passou no teste, não ocorreram erros de execução e a aresta xy é inserida na estrutura grafo passada para a função.

g. Script de execução do teste:

```
adiciona_aresta(g, 2, 4);
print_graph(g);
```

3.5.3 Teste 12

a. Nome do Teste: Teste 12

b. O que vai ser testado: o objetivo deste teste é verificar o comportamento da aplicação quando se tenta inserir uma aresta já existente na estrutura dados Graph dada.

c. Entrada: estrutura de dados do tipo Graph e os identificadores x e y dos dois vértices extremos da aresta a ser inserida na estrutura de grafos dada. A aresta a ser inserida sai de x e chega em y.

d. Saída Esperada: uma estrutura do tipo Graph (g) que contém a aresta que já estava inserida entre os vértices com identificador x e y. Não podem ocorrer erros de execução.

e. Critério para passar no teste: estrutura retornada contém a aresta x para y e não ocorrem erros de execuccão.

f. Resultado do teste: a função passou no teste, não ocorreram erros de execução e a aresta xy anteriormente inserida está presente na estrutura grafo passada para a função.

g. Script de execução do teste:

```
adiciona_aresta(g, 2, 4);
print_graph(g);
```

3.5.4 Teste 13

a. Nome do Teste: Teste 13

b. O que vai ser testado: o objetivo deste teste é verificar o comportamento da aplicação quando se tenta inserir uma aresta entre dois vértices que não estão presentes na estrutura dados Graph dada.

c. Entrada: estrutura de dados do tipo Graph e os identificadores x e y dos dois vértices extremos da aresta a ser inserida na estrutura de grafos dada. A aresta a ser inserida sai de x e chega em y.

d. Saída Esperada: não há mudanças na estrutura do tipo Graph (g). Não podem ocorrer erros de execução.

e. Critério para passar no teste: estrutura retornada não contém a aresta x para y e não ocorrem erros de execuccão.

f. Resultado do teste: a função passou no teste, não ocorreram erros de execução e não existe a aresta xy na estrutura grafo passada para a função.

g. Script de execução do teste:

```
adiciona_aresta(g, 6, 7);
print_graph(g);
```

3.5.5 Teste 14

a. Nome do Teste: Teste 14

b. O que vai ser testado: o objetivo deste teste é verificar o comportamento da aplicação quando se tenta inserir uma aresta numa estrutura dados Graph não inicializada.

c. Entrada: estrutura de dados do tipo Graph não inicializada e os identificadores x e y dos dois vértices extremos da aresta a ser inserida na estrutura de grafos dada. A aresta a ser inserida sai de x e chega em y.

d. Saída Esperada: Não ocorrem erros de execução.

e. Critério para passar no teste: Nada acontece. A funccão apenas retorna sem que ocorram erros de execução.

f. Resultado do teste: a função passou no teste, não ocorreram erros de execução.

g. Script de execução do teste:

```
adiciona_aresta(m, 1, 2);
print_graph(m);
```

3.6 Função remove_aresta:

Função testada: void remove_aresta(Graph, int x, int y);

Descrição: Esta função remove uma aresta que flui do vértice de identificador x para o vértice de identificador y em uma estrutura de dados do tipo Graph.

Parâmetros: Estrutura do tipo Graph e os identificadores (x) e (y) dos vértices extremos da aresta a ser removida do grafo.

Casos de Erro: Pode ocorrer um erro se tentarmos remover uma aresta cuja uma de suas extremidades se refere a um vertice(s) excluído(s) ou de uma estrutura grafo não inicializada. Para ambos os casos, assim como no outros testes, foi verificado se os vertices realmente existem e por consequência se o grafo foi inicializado, além de ser verificado se a aresta a ser removida realmente existe. Para todos esses casos, se ocorrer algum erro, a função para sua execução.

3.6.1 Teste 15

a. Nome do Teste: Teste 15

b. O que vai ser testado: o objetivo deste teste é verificar o comportamento da aplicação quando se tenta remover uma aresta em que uma de suas extremidades se refere a um vértice excluído da estrutura de dados Graph dada.

c. Entrada: estrutura de dados do tipo Graph e os identificadores x e y dos dois vértices extremos da aresta a ser removida na estrutura de grafos dada. A aresta a ser removida sai de x e chega em y.

d. Saída Esperada: Não há alteração na estrutura do tipo Graph (g) e não ocorrem erros de execução.

e. Critério para passar no teste: estrutura retornada não sofre alterações e não ocorrem erros de execuccão.

f. Resultado do teste: a função passou no teste, não ocorreram erros de execução e a estrutura grafo dada não sofre alterações.

g. Script de execução do teste:

```
remove_aresta(g, 1, 2);
print_graph(g);
```

3.6.2 Teste 16

a. Nome do Teste: Teste 16

b. O que vai ser testado: o objetivo deste teste é verificar o comportamento da aplicação quando se tenta remover uma aresta existente na estrutura de dados Graph dada.

c. Entrada: estrutura de dados do tipo Graph e os identificadores x e y dos dois vértices extremos da aresta a ser removida na estrutura de grafos dada. A aresta a ser removida sai de x e chega em y.

d. Saída Esperada: A estrutura do tipo Graph (g) não contém a aresta xy.

e. Critério para passar no teste: estrutura retornada não contém a aresta xy e não ocorrem erros de execuccão.

f. Resultado do teste: a função passou no teste, não ocorreram erros de execução e a estrutura grafo retornada não contém a aresta xy.

g. Script de execução do teste:

```
remove_aresta(g, 4, 2);
print_graph(g);
```

3.6.3 Teste 17

a. Nome do Teste: Teste 17

b. O que vai ser testado: o objetivo deste teste é verificar o comportamento da aplicação quando se tenta remover uma aresta xy tal que os vértices x e y não existem na estrutura de dados Graph dada.

c. Entrada: estrutura de dados do tipo Graph e os identificadores x e y dos dois vértices extremos da aresta a ser removida na estrutura de grafos dada. A aresta a ser removida sai de x e chega em y.

d. Saída Esperada: a estrutura do tipo Graph (g) não sofre alterações e n]ão ocorrem erros de execução.

e. Critério para passar no teste: estrutura retornada não é alterada e não ocorrem erros de execuccão.

f. Resultado do teste: a função passou no teste, não ocorreram erros de execução e a estrutura grafo retornada não sofreu alterações.

g. Script de execução do teste:

```
remove_aresta(g, 8, 9);
print_graph(g);
```

3.6.4 Teste 18

a. Nome do Teste: Teste 18

b. O que vai ser testado: o objetivo deste teste é verificar o comportamento da aplicação quando se tenta remover uma aresta xy de uma na estrutura de dados Graph não inicializada.

c. Entrada: estrutura de dados do tipo Graph e os identificadores x e y dos dois vértices extremos da aresta a ser removida na estrutura de grafos dada. A aresta a ser removida sai de x e chega em y.

d. Saída Esperada: nada ocorre. A função apenas retorna.

e. Critério para passar no teste: não ocorrem erros de execuccão.

f. Resultado do teste: a função passou no teste, não ocorreram erros de execução.

g. Script de execução do teste:

```
remove_aresta(m, 1, 2);
print_graph(m);
```

3.7 Função muda_valor_vertice:

Função testada: void muda_valor_vertice(Graph, int x, int valor);

Descrição: Esta função altera o valor do vértice cujo identificador é x, atribuindo a ele a quantidade valor em uma estrutura de dados do tipo Graph.

Parâmetros: Estrutura do tipo Graph, o identificadores x do vértice cujo valor busca-se alterar, novo valor a ser atribuído ao vértice x.

Casos de erro: Pode ocorrer erros se tentarmos mudar o valor de um vértice que não existe ou de uma estrutura grafo não inicializada. Para isso, é verificado se tanto o vertice existe quanto se o grafo foi devidamente inicializado. Caso algumas delas não for respeitada, a função termina sua execução.

3.7.1 Teste 19

a. Nome do Teste: Teste 19

b. O que vai ser testado: o objetivo deste teste é verificar o comportamento da aplicação quando se tenta alterar o valor de um vértice existente em estrutura de dados Graph dada.

c. Entrada: estrutura de dados do tipo Graph, o identificador x do vértice cujo valor se busca alterar, e o novo valor a ser atribuído ao vértice.

d. Saída Esperada: o novo valor do vértice x é valor.

e. Critério para passar no teste: o vértice 5 tem o valor 10.

f. Resultado do teste: a função passou no teste, não ocorreram erros de execução e o vértice 5 tem como valor 10.

g. Script de execução do teste:

```
muda_valor_vertice(g, 5, 10);
print_graph(g);
```

3.7.2 Teste 20

a. Nome do Teste: Teste 20

b. O que vai ser testado: o objetivo deste teste é verificar o comportamento da aplicação quando se tenta alterar o valor de um vértice inexistente em estrutura de dados Graph dada.

c. Entrada: estrutura de dados do tipo Graph, o identificador x do vértice cujo valor se busca alterar, e o novo valor a ser atribuído ao vértice.

d. Saída Esperada: a estrutura de dados Graph dada não sofre alterções e não ocorrem erros de execução.

e. Critério para passar no teste: não ocorrem erros e a estrutura Graph n ão é alterada.

f. Resultado do teste: a função passou no teste, não ocorreram erros de execução e a estrutura Graph dada não sofre alterações.

g. Script de execução do teste:

```
muda_valor_vertice(g, 1, 7);
print_graph(g);
```

3.7.3 Teste 21

a. Nome do Teste: Teste 21

b. O que vai ser testado: o objetivo deste teste é verificar o comportamento da aplicação quando se tenta alterar o valor de um vértice em uma estrutura de dados Graph não inicializada.

c. Entrada: estrutura de dados do tipo Graph não inicializada, o identificador x do vértice cujo valor se busca alterar, e o novo valor a ser atribuído ao vértice x.

d. Saída Esperada: nada ocorre.

e. Critério para passar no teste: não ocorrem erros de execução.

f. Resultado do teste: a função passou no teste, não ocorreram erros de execução.

g. Script de execução do teste:

```
muda_valor_vertice(m, 1, 3);
print_graph(g);
```

3.8 Função retorna_valor_vertice:

Função testada: float retorna_valor_vertice(Graph, int x);

Descrição: Esta função retorna ao usuário o valor do vértice cujo identificador é x em uma estrutura de dados do tipo Graph.

Parâmetros: Estrutura do tipo Graph e o identificador x do vértice cujo valor busca-se consultar.

Casos de erro: Pode ocorrer erros se tentarmos retornar o valor de um vertice excluído do grafo, um vertice inexistente no grafo ou de um grafo não devidamente inicializado. Para isso, como nos testes anteriores, verifica-se se o vertice existe no grafo e se o mesmo foi inicializado. Caso alguma das afirmativas for falsa, a função termina sua execução.

3.8.1 Teste 22

a. Nome do Teste: Teste 22

b. O que vai ser testado: comportamento da aplicação quando se tenta consultar o valor de um vértice que foi excluído de uma estrutura de dados Graph dada.

c. Entrada: estrutura de dados do tipo Graph e o identificador x do vértice cujo valor se busca consultar.

d. Saída Esperada: -1 (significando que o vértice não existe na estrutura).

e. Critério para passar no teste: o retorno é -1.

f. Resultado do teste: a função passou no teste.

g. Script de execução do teste:

```
printf("%.2f\n", retorna_valor_vertice(g, 1));
```

3.8.2 Teste 23

a. Nome do Teste: Teste 23

b. O que vai ser testado: comportamento da aplicação quando se tenta consultar o valor de um vértice presente na estrutura de dados Graph dada.

c. Entrada: estrutura de dados do tipo Graph e o identificador x do vértice cujo valor se busca consultar.

d. Saída Esperada: 10.

e. Critério para passar no teste: o retorno é 10.

f. Resultado do teste: a função passou no teste.

g. Script de execução do teste:

```
printf("%.2f\n", retorna_valor_vertice(g, 5));
```

3.8.3 Teste 24

a. Nome do Teste: Teste 24

b. O que vai ser testado: comportamento da aplicação quando se tenta consultar o valor de um vértice inexistente na estrutura de dados Graph dada.

c. Entrada: estrutura de dados do tipo Graph e o identificador x do vértice cujo valor se busca consultar.

d. Saída Esperada: -1 (significando que o vértice não existe na estrutura).

e. Critério para passar no teste: o retorno é -1.

f. Resultado do teste: a função passou no teste.

g. Script de execução do teste:

```
printf("%.2f\n", retorna_valor_vertice(g, 10));
```

3.8.4 Teste 25

a. Nome do Teste: Teste 25

b. O que vai ser testado: comportamento da aplicação quando se tenta consultar o valor de um vértice em uma estrutura de dados Graph não inicializada.

c. Entrada: estrutura de dados do tipo Graph e o identificador x do vértice cujo valor se busca consultar.

d. Saída Esperada: -1 (significando que o vértice não existe na estrutura).

e. Critério para passar no teste: o retorno é -1 e não ocorrem erros de execução.

f. Resultado do teste: a função passou no teste.

g. Script de execução do teste:

```
printf("%.2f\n", retorna_valor_vertice(m, 10));
```

3.9 Função muda_valor_aresta:

Função testada: void muda_valor_aresta(Graph, int x, int y, int valor);

Descrição: Esta função altera o valor da aresta entre os vértices cujos identificadores são x e y em uma estrutura de dados do tipo Graph.

Parâmetros: Estrutura do tipo Graph, os identificadores x e y do vértice cujo valor da aresta busca-se alterar e o novo valor a ser atribuído para a aresta xy.

Casos de erro: Pode ocorrer um erro se tentarmos mudar o valor de uma aresta que não existe no grafo, de vertice(s) que não existem no grafo, de vertices que foram excluídos do grafo ou quando o grafo passado não foi devidamente inicializado. Para tais casos, assim como os testes anteriores, verifica-se por meio de funções auxiliares de busca se o(s) vertice(s) existe(m) e, por consequência, se o grafo passado foi devidamente inicializado e também se a aresta cujo o valor se deseja mudar realmente existe no grafo. Caso algumas dessas afirmativas não for verdadeira, a função termina sua execução.

3.9.1 Teste 26

a. Nome do Teste: Teste 26

b. O que vai ser testado: comportamento da aplicação quando se tenta alterar o valor de aresta inexistente na estrutura de dados Graph dada.

c. Entrada: estrutura de dados do tipo Graph, os identificadores x e y do vértices extremidade da aresta cujo valor se busca alterar, e o novo valor a ser atribuído à aresta.

d. Saída Esperada: nada ocorre, ou seja, a estrutura Graph não é alterada.

e. Critério para passar no teste: não ocorrem erros de execução e a estrutura Graph dada não sofre alterações.

f. Resultado do teste: a função passou no teste.

g. Script de execução do teste:

```
muda_valor_aresta(g, 2, 3, 10);
print_graph(g);
```

3.9.2 Teste 27

a. Nome do Teste: Teste 27

b. O que vai ser testado: comportamento da aplicação quando se tenta alterar o valor de aresta presente na estrutura de dados Graph dada.

c. Entrada: estrutura de dados do tipo Graph, os identificadores x e y do vértices extremidade da aresta cujo valor se busca alterar, e o novo valor a ser atribuído à aresta.

d. Saída Esperada: o novo valor da aresta 2-4 é 7.

e. Critério para passar no teste: não ocorrem erros de execução e na estrutura Graph dada a aresta 2-4 possui valor 7.

f. Resultado do teste: a função passou no teste.

g. Script de execução do teste:

```
muda_valor_aresta(g, 2, 4, 7);
print_graph(g);
```

3.9.3 Teste 28

a. Nome do Teste: Teste 28

b. O que vai ser testado: comportamento da aplicação quando se tenta alterar o valor de aresta cujas extremidades são vértices inexistentes na estrutura de dados Graph dada.

c. Entrada: estrutura de dados do tipo Graph, os identificadores x e y do vértices extremidade da aresta cujo valor se busca alterar, e o novo valor a ser atribuído à aresta.

d. Saída Esperada: não ocorrem erros de execução e não há alterações na estrutura Graph dada.

e. Critério para passar no teste: não ocorrem erros de execução e a estrutura Graph não é alterada.

f. Resultado do teste: a função passou no teste.

g. Script de execução do teste:

```
muda_valor_aresta(g, 5, 6, 12);
print_graph(g);
```

3.9.4 Teste 29

a. Nome do Teste: Teste 29

b. O que vai ser testado: comportamento da aplicação quando se tenta alterar o valor de aresta cujas extremidades são vértices excluídos da estrutura de dados Graph dada.

c. Entrada: estrutura de dados do tipo Graph, os identificadores x e y do vértices extremidade da aresta cujo valor se busca alterar, e o novo valor a ser atribuído à aresta.

d. Saída Esperada: não ocorrem erros de execução e não há alterações na estrutura Graph dada.

e. Critério para passar no teste: não ocorrem erros de execução e a estrutura Graph não é alterada.

f. Resultado do teste: a função passou no teste.

g. Script de execução do teste:

```
muda_valor_aresta(g, 1, 1, 5);
print_graph(g);
```

3.9.5 Teste 30

a. Nome do Teste: Teste 30

b. O que vai ser testado: comportamento da aplicação quando se tenta alterar o valor de uma aresta em uma estrutura de dados Graph não inicializada.

c. Entrada: estrutura de dados do tipo Graph não inicializada, os identificadores x e y do vértices extremidade da aresta cujo valor se busca alterar, e o novo valor a ser atribuído à aresta.

d. Saída Esperada: não ocorrem erros de execução.

e. Critério para passar no teste: não ocorrem erros de execução.

f. Resultado do teste: a função passou no teste.

g. Script de execução do teste:

```
muda_valor_aresta(m, 1, 1, 5);
print_graph(g);
```

3.10 Função retorna_valor_aresta:

Função testada: float retorna_valor_aresta(Graph, int x, int y);

Descrição: Esta função retorna o valor da aresta entre os vértices cujos identificadores são x e y em uma estrutura de dados do tipo Graph.

Parâmetros: estrutura do tipo Graph e os identificadores x e y do vértice cujo valor da aresta busca-se consultar.

Casos de erro: Podem ocorrer erros se tentarmos consultar uma aresta cujo algum dos vertices passados como paâmetro para a função não existirem, quando a aresta procurada não existe no grafo ou quando o grafo passado como parâmetro não foi inicializado corretamente. Caso em algum retorno das funções auxiliares de busca tivermos algum tipo de erro com tais condições a serem confirmadas, a função terminina sua execução retornando -1.

3.10.1 Teste 31

a. Nome do Teste: Teste 31

b. O que vai ser testado: comportamento da aplicação quando se tenta consultar o valor de aresta presente na estrutura de dados Graph dada.

c. Entrada: estrutura de dados do tipo Graph e os identificadores x e y do vértices extremidade da aresta cujo valor se busca consultar.

d. Saída Esperada: 7.

e. Critério para passar no teste: não ocorrem erros de execução e a função retorna o valor 7.

f. Resultado do teste: a função passou no teste.

g. Script de execução do teste:

```
printf("%.2f\n", retorna_valor_aresta(g, 2, 4));
```

3.10.2 Teste 32

a. Nome do Teste: Teste 32

b. O que vai ser testado: comportamento da aplicação quando se tenta consultar o valor de aresta cujo vértice extremidade não existe na estrutura de dados Graph dada.

c. Entrada: estrutura de dados do tipo Graph e os identificadores x e y do vértices extremidade da aresta cujo valor se busca consultar.

d. Saída Esperada: -1 (indicando que o vértice não existe na estrutura Graph).

e. Critério para passar no teste: não ocorrem erros de execução e a função retorna o valor -1.

f. Resultado do teste: a função passou no teste.

g. Script de execução do teste:

```
printf("%.2f\n", retorna_valor_aresta(g, 1, 2));
```

3.10.3 Teste 33

a. Nome do Teste: Teste 33

b. O que vai ser testado: comportamento da aplicação quando se tenta consultar o valor de aresta não existente na estrutura de dados Graph dada.

c. Entrada: estrutura de dados do tipo Graph e os identificadores x e y do vértices extremidade da aresta cujo valor se busca consultar.

d. Saída Esperada: 0.00 (indicando que a aresta não existe na estrutura Graph).

e. Critério para passar no teste: não ocorrem erros de execução e a função retorna o valor 0.00.

f. Resultado do teste: a função passou no teste.

g. Script de execução do teste:

```
printf("%.2f\n", retorna_valor_aresta(g, 2, 3));
```

3.10.4 Teste 34

a. Nome do Teste: Teste 34

b. O que vai ser testado: comportamento da aplicação quando se tenta consultar o valor de uma aresta em estrutura de dados Graph não inicializada.

c. Entrada: estrutura de dados do tipo Graph não inicializada e os identificadores x e y do vértices extremidade da aresta cujo valor se busca consultar.

d. Saída Esperada: -1 (indicando que a estrutura Graph não existe).

e. Critério para passar no teste: não ocorrem erros de execução e a função retorna o valor -1.

f. Resultado do teste: a função passou no teste.

g. Script de execução do teste:

```
printf("%.2f\n", retorna_valor_aresta(m, 2, 3));
```

3.11 Função adjacente:

Função testada: bool adjacente(Graph, int x, int y);

Descrição: Esta função retorna um booleano indicando se os vértices cujos identificadores são x e y são adjacentes na estrutura de dados Graph dada.

Parâmetros: Estrutura do tipo Graph e os identificadores x e y dos vértices cuja adjacência busca-se consultar.

Casos de erro: Pode ocorrer algum erro quando verifica-se adjacência entre dois vértices removidos, entre dois vertices que não existem, adjacência entre dois vértices não adjacentes ou quando temos uma estrutura de grafo não devidamente inicializada. Para tais casos de erro, a função irá retornar falso se algum dos casos descritos acontecerem, verdadeiro do contrário.

3.11.1 Teste 35

a. Nome do Teste: Teste 35

b. O que vai ser testado: comportamento da aplicação quando se tenta verificar a adjacência entre dois vértices removidos da estrutura de dados Graph dada.

c. Entrada: estrutura de dados do tipo Graph e os identificadores x e y dos vértices cuja adjacência busca-se testar.

d. Saída Esperada: false.

e. Critério para passar no teste: não ocorrem erros de execução e a função retorna false.

f. Resultado do teste: a função passou no teste.

g. Script de execução do teste:

```
printf(adjacente(g, 1, 1) ? "true\n" : "false\n");
```

3.11.2 Teste 36

a. Nome do Teste: Teste 36

b. O que vai ser testado: comportamento da aplicação quando se tenta verificar a adjacência entre dois vértices adjacentes na estrutura de dados Graph dada.

c. Entrada: estrutura de dados do tipo Graph e os identificadores x e y dos vértices cuja adjacência busca-se testar.

d. Saída Esperada: true.

e. Critério para passar no teste: não ocorrem erros de execução e a função retorna true.

f. Resultado do teste: a função passou no teste.

g. Script de execução do teste:

```
printf(adjacente(g, 2, 4) ? "true\n" : "false\n");
```

3.11.3 Teste 37

a. Nome do Teste: Teste 37

b. O que vai ser testado: comportamento da aplicação quando se tenta verificar a adjacência entre dois vértices não adjacentes na estrutura de dados Graph dada.

c. Entrada: estrutura de dados do tipo Graph e os identificadores x e y dos vértices cuja adjacência busca-se testar.

d. Saída Esperada: false.

e. Critério para passar no teste: não ocorrem erros de execução e a função retorna false.

f. Resultado do teste: a função passou no teste.

g. Script de execução do teste:

```
printf(adjacente(g, 3, 4) ? "true\n" : "false\n");
```

3.11.4 Teste 38

a. Nome do Teste: Teste 38

b. O que vai ser testado: comportamento da aplicação quando se tenta verificar a adjacência entre dois vértices não existentes na estrutura de dados Graph dada.

c. Entrada: estrutura de dados do tipo Graph e os identificadores x e y dos vértices cuja adjacência busca-se testar.

d. Saída Esperada: false.

e. Critério para passar no teste: não ocorrem erros de execução e a função retorna false.

f. Resultado do teste: a função passou no teste.

g. Script de execução do teste:

```
printf(adjacente(g, 8, 10) ? "true\n" : "false\n");
```

3.11.5 Teste 39

a. Nome do Teste: Teste 39

b. O que vai ser testado: comportamento da aplicação quando se tenta verificar a adjacência entre dois vértices em uma estrutura de dados Graph não inicializada.

c. Entrada: estrutura de dados do tipo Graph não inicializada e os identificadores x e y dos vértices cuja adjacência busca-se testar.

d. Saída Esperada: false.

e. Critério para passar no teste: não ocorrem erros de execução e a função retorna false.

f. Resultado do teste: a função passou no teste.

g. Script de execução do teste:

```
printf(adjacente(m, 8, 10) ? "true\n" : "false\n");
```

3.12 Função vizinhos:

Função testada: struct vertex * vizinhos(Graph, int x);

Descrição: Esta função retorna uma lista com os vizinhos do vértice de identificador x numa estrutura de dados Graph dada. Os vizinhos de um vértice

x são os vértices que podem ser alcançados a partir do vértice x visitando-se apenas uma aresta.

Parâmetros: estrutura do tipo Graph e o identificador x do vértice cuja lista de vizinhos busca-se consultar.

Casos de erro: Pode ocorrer erros se o vertice com o valor x não existir no grafo, quando se tenta verificar os vizinhos de um vertice excluído ou quando tenta se verificar a lista de vizinhos em um grafo não devidamente inicializado. Para tais casos, verifica-se se o vertice realmente existe e se o grafo foi devidamente inicializado. Se alguma desses erros acontecerem, a função termina sua execução, do contrário retorna a lista de vizinhos do vertice com valor x.

3.12.1 Teste 40

a. Nome do Teste: Teste 40

b. O que vai ser testado: comportamento da aplicação quando se tenta verificar a lista de vizinhos de um vértice presente da estrutura de dados Graph dada e que efetivamente possui vizinhos.

c. Entrada: estrutura de dados do tipo Graph e o identificador x do vértice cuja lista de vizinhos busca-se consultar.

d. Saída Esperada: lista de vértices contendo os vizinhos do vértice x.

e. Critério para passar no teste: não ocorrem erros de execução e a função retorna os vértices 3 e 5 como vizinhos.

f. Resultado do teste: a função passou no teste.

g. Script de execução do teste:

```
puts("Vizinhos do Vertice 4:");
vizinhos(g, 4);
```

3.12.2 Teste 41

a. Nome do Teste: Teste 41

b. O que vai ser testado: comportamento da aplicação quando se tenta verificar a lista de vizinhos de um vértice inexistente da estrutura de dados Graph dada.

c. Entrada: estrutura de dados do tipo Graph e o identificador x do vértice cuja lista de vizinhos busca-se consultar.

d. Saída Esperada: lista vazia.

e. Critério para passar no teste: não ocorrem erros de execução e a função retorna uma lista vazia.

f. Resultado do teste: a função passou no teste.

g. Script de execução do teste:

```
puts("Vizinhos do Vertice 8:");
vizinhos(g, 8);
```

3.12.3 Teste 42

a. Nome do Teste: Teste 42

b. O que vai ser testado: comportamento da aplicação quando se tenta verificar a lista de vizinhos de um vértice excluído da estrutura de dados Graph dada.

c. Entrada: estrutura de dados do tipo Graph e o identificador x do vértice cuja lista de vizinhos busca-se consultar.

d. Saída Esperada: lista vazia.

e. Critério para passar no teste: não ocorrem erros de execução e a função retorna uma lista vazia.

f. Resultado do teste: a função passou no teste.

g. Script de execução do teste:

```
puts("Vizinhos do Vertice 1:");
vizinhos(g, 1);
```

3.12.4 Teste 43

a. Nome do Teste: Teste 43

b. O que vai ser testado: comportamento da aplicação quando se tenta verificar a lista de vizinhos de um vértice em uma estrutura de dados Graph não inicializada.

c. Entrada: estrutura de dados do tipo Graph não inicializada e o identificador x do vértice cuja lista de vizinhos busca-se consultar.

d. Saída Esperada: lista nula.

e. Critério para passar no teste: não ocorrem erros de execução e a função retorna uma lista nula e imprime em tela a informação "Grafo nulo".

f. Resultado do teste: a função passou no teste.

g. Script de execução do teste:

```
puts("Vizinhos do Vertice 1:");
vizinhos(m, 8);
```

3.13 Função destroi_grafo:

Função testada: void destroi_grafo(Graph);

Descrição: Esta função destrói uma estrutura do tipo Graph, liberando o espaço alocado em memória para ela.

Parâmetros: Estrutura do tipo Graph.

Casos de erro: Pode ocorrer um erro se o grafo a ser destruído não foi sequer inicializado. Se tal erro acontecer, a função para sua execução, do contrário desaloca e reseta todos os componentes do grafo e o próprio grafo.

3.13.1 Teste 44

a. Nome do Teste: Teste 44

b. O que vai ser testado: comportamento da aplicação quando se tenta destruir uma estrutura do tipo Graph inicializada e preenchida com vértices e arestas.

c. Entrada: estrutura de dados do tipo Graph inicializada e contendo vértices e arestas.

d. Saída Esperada: nulo.

e. Critério para passar no teste: não ocorrem erros de execução e a função retorna nulo.

f. Resultado do teste: a função passou no teste.

g. Script de execução do teste:

```
destroi_grafo(g);
print_graph(g);
```

3.13.2 Teste 45

a. Nome do Teste: Teste 45

b. O que vai ser testado: comportamento da aplicação quando se tenta destruir uma estrutura do tipo Graph não inicializada.

c. Entrada: estrutura de dados do tipo Graph não inicializada.

d. Saída Esperada: nulo.

e. Critério para passar no teste: não ocorrem erros de execução e a função retorna nulo.

f. Resultado do teste: a função passou no teste.

g. Script de execução do teste:

```
destroi_grafo(m);
print_graph(m);
```