

# Operações com Matrizes

Lucas do Vale Santos

16 de Setembro de 2013

## 1. Introdução

Este trabalho consiste na elaboração de um programa que realize algumas operações com matrizes, através do conceito intuitivo de grafos. Por meio de algumas manipulações básicas, espera-se que o programa forneça uma rápida compreensão sobre essas operações, sendo aplicado de maneira prática e flexível o conceito de grafos.

## 2. Implementação do Programa

O programa desenvolvido foi escrito utilizando a linguagem de programação **Pascal** e sua biblioteca **Crt**, que é de fundamental importância para o funcionamento dos comandos **writeln** e **readln** (entrada e saída de dados), além do **clrsrc** que serve para apagar os dados impressos na tela que antecedam este mesmo comando.

Dentre os vários recursos que a linguagem proporciona, além dos citados anteriormente, tem-se o uso de estruturas de iteração indefinida (**do ... while**), estruturas condicionais (**if ... then ... else, case ... of**) e procedimentos(**procedure**). Outro recurso usado foi a criação de **types**, e foi feito também, o uso da passagem de parâmetros por referência.

Neste programa, foram utilizadas estruturas de dados homogêneas (**array**), cuja aplicação foi de essencial importância para as operações principais.

Não foi encontrado nenhum obstáculo para a implementação do programa. Algumas restrições foram aplicadas, como será mostrado a seguir.

## 3. Casos de Uso

Aqui serão apresentados os casos de usos referentes a entrada, saída e o tratamentos de alguns dados.

O programa disponibiliza na tela de execução, um **menu** fixo, contendo opções e suas respectivas finalidades, como pode ser visto na Figura 1.

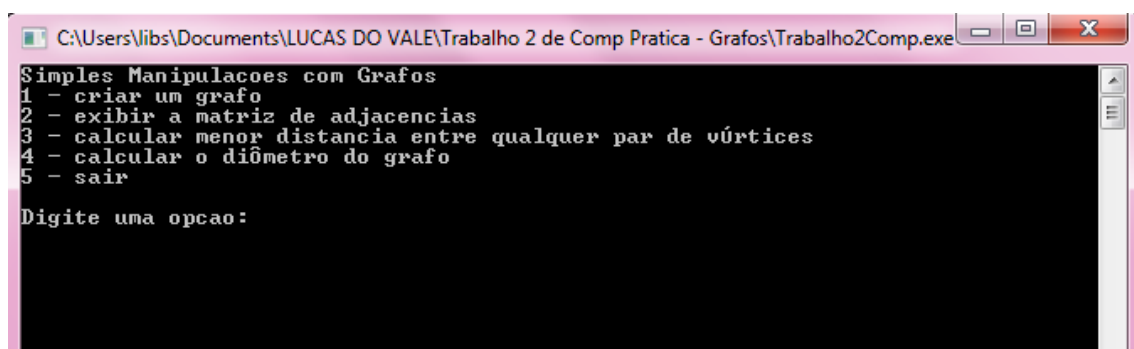


Figura 1: menu do programa

Em todo primeiro acesso ao programa, não é permitido que as opções 2,3 e 4 sejam utilizadas, pois não há grafos criados. Para tanto, foram criadas condições para essas opções, a fim de induzir o usuário a utilizar a opção 1 (criar grafo) ou no mínimo, utilizar a opção 5 para sair do programa sem nenhum grafo criado. Observe a Figura 2.

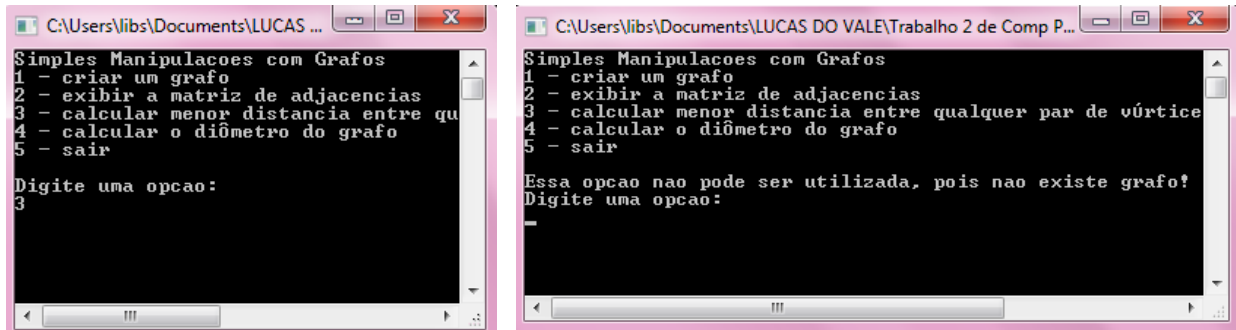


Figura 2: o usuário é advertido de utilizar as opções 2,3 e 4 quando não há grafo.

Ao utilizar a opção 1, o usuário solicita a criação de um grafo. Neste caso, o programa exige que o usuário entre com o número de vértices que o grafo deva possuir, sendo que este limite deve estar entre 2 e 10 vértices. Posteriormente, o usuário deve rotular todos os vértices e informar as arestas existentes. As arestas são construídas a partir da relação entre dois vértices, os quais o usuário informa através de seus respectivos rótulos. A criação do grafo é finalizada quando o usuário digita 0 para o rótulo. Na Figura 3 é possível observar tais etapas.

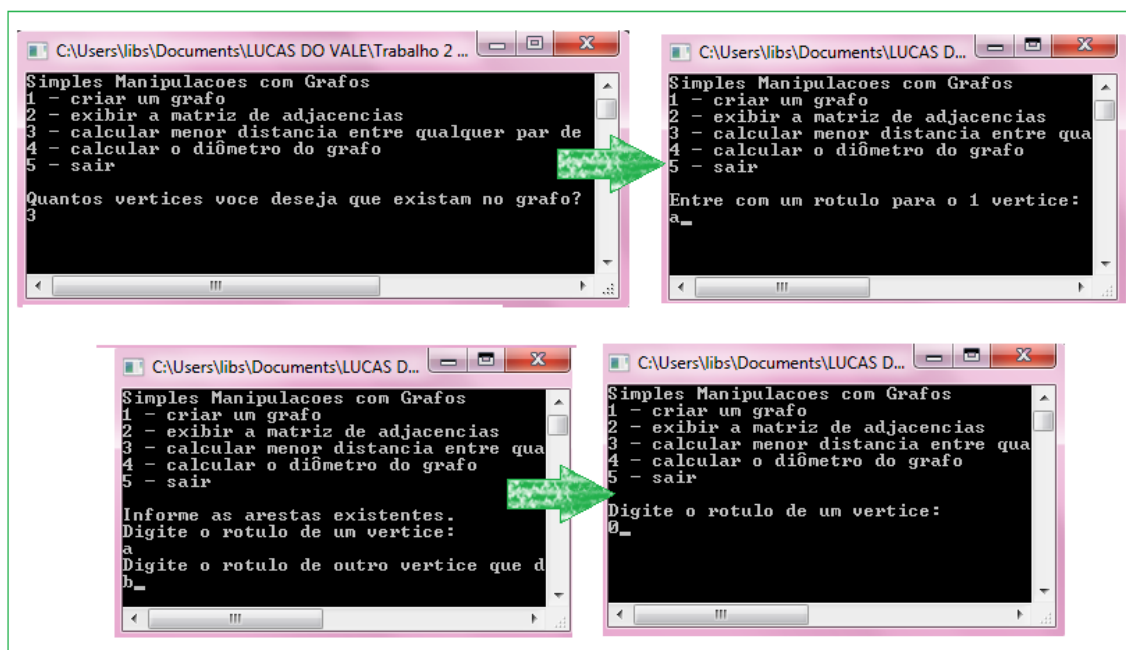


Figura 3: após digitar a opção 1, o programa faz algumas exigências para criar o grafo.

Uma observação importante a ser feita é que toda vez em que o usuário digita a opção 1, o programa “zera” as posições da matriz de adjacências, para que ele sempre possa criar um novo grafo. Isto pode ser verificado no código que se encontra no final da seção 3 de Casos de uso.

A opção 2 permite que o usuário visualize a matriz de adjacências, responsável por informar as arestas existentes entre pares de vértices. Os enlaces são identificados pelo número 1 nas posições da matriz. A Figura 4 exemplifica o que é exibido na tela.

```
C:\Users\luis\Documents\LUCAS DO VALE\Trabalho 2 de Comp Pratica - Grafos\Trabalho 2 de Comp Pratica - Grafos
Simples Manipulacoes com Grafos
1 - criar um grafo
2 - exibir a matriz de adjacencias
3 - calcular menor distancia entre qualquer par de vórtices
4 - calcular o diâmetro do grafo
5 - sair

Matriz de Adjacencias:

  a b c
a 0 1 1
b 1 0 1
c 1 1 0
```

Figura 4: matriz de adjacências sendo exibida na tela.

O processo utilizado pelo programa é extremamente simples, pois consiste em percorrer toda matriz, e imprimir na tela cada valor contido em cada posição. Isso pode ser verificado no código do programa que se encontra no final da seção 3 de Casos de uso.

A opção 3 informa ao usuário a distância mínima entre um vértice e outro no grafo, onde a distância é dada pelas arestas (saltos) que conectam esses vértices. Para tanto, o usuário deve informar dois rótulos correspondentes a dois vértices, e verificar essa distância mínima entre eles. Essa informação pode ser obtida através de uma matriz de possibilidade de saltos, onde cada campo contém o número de possibilidades de “n” saltos entre todos os pares de vértices. Essa matriz pode ser calculada elevando-se a matriz de adjacências ao número “n”.

A menor distância vai ser dada pelo “n”, quando o campo correspondente a esses dois vértices assumir o primeiro valor diferente de 0. A seguir, o programa imprime o resultado na tela, como mostra a Figura 5.

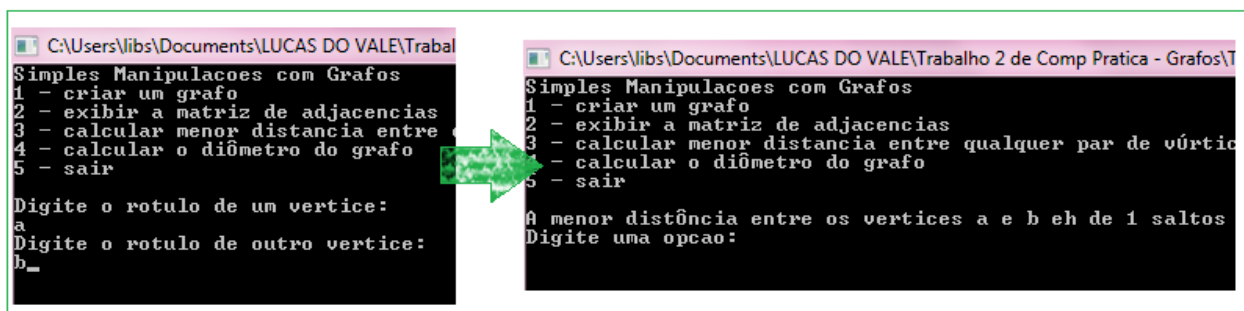
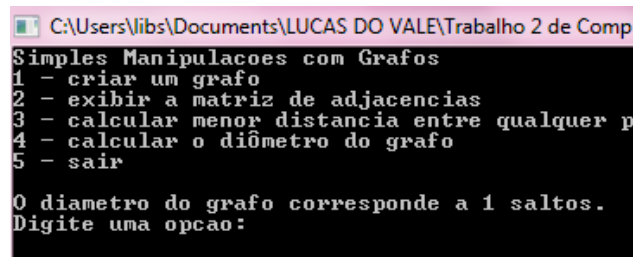


Figura 5: processo de verificação da menor distância entre um par de vértices.

No final da Seção 3 de Casos de Uso, podemos conferir os detalhes minuciosos essenciais para a construção do raciocínio explicado anteriormente, como por exemplo, a multiplicação feita entre matrizes.

Ao digitar a opção 4, o usuário verifica qual é o diâmetro do grafo, que corresponde ao menor número de saltos que permitem conectar qualquer par de vértices. Para isso, o processo é basicamente o mesmo do executado pela opção 3, com a diferença de que o programa deve encontrar o menor valor obtido na opção anterior. Esta pequena diferença

poderá ser verificada no final da seção 3 de Casos de Uso. A seguir, a Figura 6 mostra o conteúdo que é impresso na tela, após o usuário solicitar a opção 4.

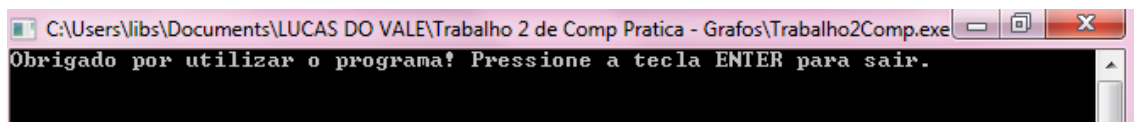


```
C:\Users\lucs\Documents\LUCAS DO VALE\Trabalho 2 de Comp
Simples Manipulacoes com Grafos
1 - criar um grafo
2 - exibir a matriz de adjacencias
3 - calcular menor distancia entre qualquer p
4 - calcular o diâmetro do grafo
5 - sair

0 diametro do grafo corresponde a 1 saltos.
Digite uma opcao:
```

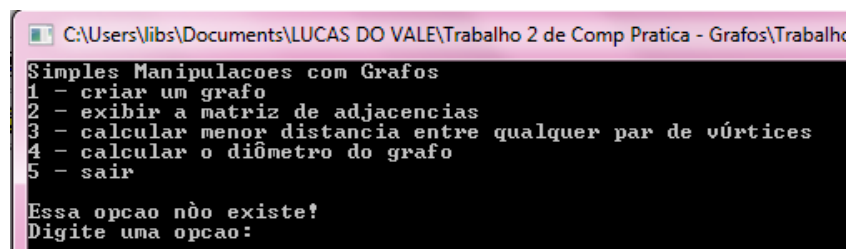
Figura 6: diâmetro do grafo

Como já citado anteriormente, o programa disponibiliza a opção 5. Após digitar essa opção, o usuário sai do programa, e é impressa na tela uma mensagem de finalização, como está demonstrado na figura 7. Por fim, caso o usuário digite uma opção diferente das anteriores, o programa exibe uma mensagem, alertando sobre a inexistência da opção, como está esclarecido na Figura 8.



```
C:\Users\lucs\Documents\LUCAS DO VALE\Trabalho 2 de Comp Pratica - Grafos\Trabalho2Comp.exe
Obrigado por utilizar o programa! Pressione a tecla ENTER para sair.
```

Figura 7: alerta de saída do programa.



```
C:\Users\lucs\Documents\LUCAS DO VALE\Trabalho 2 de Comp Pratica - Grafos\Trabalho2Comp.exe
Simples Manipulacoes com Grafos
1 - criar um grafo
2 - exibir a matriz de adjacencias
3 - calcular menor distancia entre qualquer par de vórtices
4 - calcular o diâmetro do grafo
5 - sair

Essa opcao não existe!
Digite uma opcao:
```

Figura 8: quando uma opção inválida é digitada.

## Código do Programa

⇒ Programa Principal

```
298 var
299     mat: matriz;
300     rot: vetor;
301     op,ord,existeGrafo: integer;
302
303 begin
304
305     existeGrafo := 0;
306
307     menu;
308
309     writeln('Digite uma opcao: ');
310     readln(op);
311
312     while (op <> 5) do
313     begin
314         case op of
315             1: begin
316                 clrscr;
317                 menu;
318                 criaGrafo(mat,rot,ord); //cria grafo, ou edita o já existente
319                 existeGrafo := 1;
320             end;
321             2: begin
322                 clrscr;
323                 menu;
324                 if(existeGrafo <> 0) then
325                 begin
326                     exibeMatAdj(mat,ord,rot); //imprime a matriz de adjacências na tela
327                 end
328                 else
329                 begin
330                     writeln('Essa opcao nao pode ser utilizada, pois nao existe grafo!');
331                 end;
332             end;
333             3: begin
334                 clrscr;
335                 menu;
336                 if(existeGrafo <> 0) then
337                 begin
338                     calculaDistMin(mat,rot,ord); //calcula a menor distancia entre um par de vértices
339                 end
340                 else
341                 begin
342                     writeln('Essa opcao nao pode ser utilizada, pois nao existe grafo!');
343                 end;
344             end;
345             4: begin
346                 clrscr;
347                 menu;
348                 if(existeGrafo <> 0) then
349                 begin
350                     calculaDmtGrafo(mat,ord); //calcula o diâmetro do grafo
351                 end
352                 else
353                 begin
354                     writeln('Essa opcao nao pode ser utilizada, pois nao existe grafo!');
355                 end;
356             end;
357             else
358             begin
359                 if (op <> 5) then
360                 begin
361                     clrscr;
362                     menu;
363                     writeln('Essa opcao não existe!');
364                 end;
365             end;
366         end;
367         writeln('Digite uma opcao: ');
368         readln(op);
369     end;
370
371     clrscr;
372     writeln('Obrigado por utilizar o programa! Pressione a tecla ENTER para sair.');
```

⇒ Declaração da Constante, Criação de Tipos e Procedimento do Menu

```
1  program GRAFOS;
2
3  uses Crt;
4
5  const N = 10;      //tamanho fixo para o vetor de rótulos e a matriz de adjacências
6
7  type matriz = array[1..N,1..N] of integer;
8      vetor = array[1..N] of char;
9
10 procedure menu;
11 begin
12     writeln('Simples Manipulacoes com Grafos');
13     writeln;
14     writeln('1 - criar um grafo');
15     writeln('2 - exibir a matriz de adjacencias');
16     writeln('3 - calcular menor distancia entre qualquer par de vértices');
17     writeln('4 - calcular o diâmetro do grafo');
18     writeln('5 - sair');
19     writeln;
20     writeln('Observacao: cada rótulo deve corresponder a um caracter diferente. Use letras de preferencia.');
```

⇒ Procedimentos de Carregamento e Cópia de Matriz

```
24 procedure zeraMatriz(var mat: matriz);
25 var
26     i,j: integer;
27 begin
28     for i := 1 to 10 do
29     begin
30         for j := 1 to 10 do
31         begin
32             mat[i,j] := 0;
33         end;
34     end;
35 end;
36
37 procedure copiaMatriz(var matSal,mat: matriz; var ord: integer);
38 var
39     i,j: integer;
40 begin
41     for i := 1 to ord do
42     begin
43         for j := 1 to ord do
44         begin
45             matSal[i,j] := mat[i,j];
46         end;
47     end;
48 end;
```

⇒ Procedimento de Criação do Grafo

```

50 procedure criaGrafo(var mat: matriz; var rot: vetor; var ord: integer);
51 var
52     i,j,verdadeiro,arestas: integer;
53     r1,r2: char;
54 begin
55
56     zeraMatriz(mat);
57
58     verdadeiro := 0;
59     arestas := 0;
60
61     writeln('Quantos vertices voce deseja que existam no grafo?');
62     readln(ord);
63     clrscr;
64     menu;
65     while((ord < 2) or (ord > 10))do //permanece aqui ate o usuario entre com um valor adequado de vertices
66     begin
67         clrscr;
68         menu;
69         writeln('Este grafo deve possuir no minimo 2 vertices e no maximo 10 vertices!');
70         writeln('Digite novamente a quantidade de vertices que deseja: ');
71         readln(ord);
72     end;
73     for i := 1 to ord do //guarda em um vetor todos os rotulos dos vertices
74     begin
75         clrscr;
76         menu;
77
78         writeln('Entre com um rotulo para o ',i,' vertice: ');
79         readln(rot[i]);
80     end;
81
82     clrscr;
83     menu;
84     writeln('Informe as arestas existentes.');

```

⇒ Procedimento de Exibição da Matriz de Adjacências

```
148 procedure exhibeMatAdj(mat: matriz; var ord: integer; var rot: vetor);
149 var
150     i,j: integer;
151 begin
152     writeln('Matriz de Adjacencias:');
153     writeln;
154     write(' ');
155     for i := 1 to ord do
156     begin
157         write(' ',rot[i]); //imprime os rotulos em cima de cada coluna
158     end;
159
160     writeln;
161
162     for i := 1 to ord do
163     begin
164         for j := 1 to ord do
165         begin
166             if(j = 1)then
167             begin
168                 write(rot[i],' '); //imprime os rotulos a esquerda de cada linha
169             end;
170             write(mat[i,j],' '); //imprime a matriz
171         end;
172         writeln;
173     end;
174     writeln;
175 end;
176
```

⇒ Procedimento de Cálculo da Distância Mínima entre dois Vértices

```
177 procedure calculaDistMin(mat: matriz; var rot: vetor; var ord: integer);
178 var
179     i,j,x,y,z,saltos: integer;
180     r1,r2: char;
181     matAux,matSal: matriz;
182 begin
183
184     saltos := 1;
185     writeln('Digite o rotulo de um vertice: ');
186     readln(r1);
187     writeln('Digite o rotulo de outro vertice:');
188     readln(r2);
189     for i := 1 to ord do
190     begin
191
192         if(r1 = rot[i])then //corresponde com algum rotulo no vetor
193         begin
194
195             for j := 1 to ord do
196             begin
197                 if(r2 = rot[j])then //corresponde com algum rotulo no vetor
198                 begin
199
200                     copiaMatriz(matSal,mat,ord); //atualiza a matriz de possibilidades a "n" saltos, para a matriz de possibil
201
202                     while((matSal[i,j] = 0) and (matSal[j,i] = 0))do
203                     begin
204                         zeraMatriz(matAux);
205
206                         for x := 1 to ord do
207                         begin
208                             for y := 1 to ord do
209                             begin
210                                 for z := 1 to ord do
211                                 begin
212                                     matAux[x,y] := matAux[x,y] + matSal[x,z]*mat[z,y];
213                                 end;
214                             end;
215                         end;
216                     end;
217
218                     copiaMatriz(matSal,matAux,ord); //atualiza a matriz de possibilidades a "n" saltos, onde "n"="saltos
219
220                     saltos := saltos + 1;
221                 end;
222             end;
223         end;
224     end;
225
226     end;
227
228     end;
229
230     end;
231
232     end;
233     clrscr;
234     menu;
235     writeln('A menor distância entre os vertices ',r1,' e ',r2,' eh de ',saltos,' saltos');
236 end;
```



#### ⇒ Procedimento do Cálculo do Diâmetro

```
238 procedure calculaDmtGrafo(mat: matriz; var ord: integer);
239 var
240   i,j,x,y,z,saltos,diametro: integer;
241   matAux,matSal: matriz;
242 begin
243   diametro := 1;
244   for i := 1 to ord do
245   begin
246     for j := 1 to ord do
247     begin
248       saltos := 1;
249       copiaMatriz(matSal,mat,ord); //atualiza a matriz de possibilidades a "n" saltos, para a matriz de possibilidades
250       while((matSal[i,j] = 0) and (matSal[j,i] = 0) and (i <> j))do //Continua até que seja obtido um numero min
251       begin
252         zeraMatriz(matAux);
253         for x := 1 to ord do
254         begin
255           for y := 1 to ord do
256           begin
257             for z := 1 to ord do
258             begin
259               matAux[x,y] := matAux[x,y] + matSal[x,z]*mat[z,y];
260             end;
261           end;
262         end;
263       end;
264       copiaMatriz(matSal,matAux,ord); //atualiza a matriz de possibilidades a "n" saltos, onde "n"="saltos"
265       saltos := saltos + 1;
266     end;
267     if(saltos > diametro)then //verifica entre cada salto qual o menor possivel para que qualquer par
268     begin
269       diametro := saltos;
270     end;
271   end;
272 end;
273 //desconta as conexoes repetidas
274 writeln('O diametro do grafo corresponde a ',diametro,' saltos.');
```

## 4. Conclusão

O desenvolvimento do código em si não foi tão simples, visto que o processo de manipulação de matrizes, principalmente quanto à multiplicação, não é nada trivial. As restrições também devem ser consideradas nos momentos em que sejam identificadas incoerências, tomadas as entradas de dados. Visto que aplicação do conceito de grafos foi concebida com sucesso, entre outros aspectos, podemos concluir que o programa cumpriu com sua finalidade.