

# ROTEIRO 7 – LUCAS EDUARDO LEITE COSTA

## 1.1

```
11> C main.c @ main0
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include "matriz.h"
4
5 int main(){
6
7     int n, l, c, elem;
8     int* p = (int*)malloc(sizeof(int));
9     Matriz* mat;
10    Matriz* transp;
11    Matriz* triangSup;
12    Matriz* triangInf;
13    Matriz* diag;
14
15    printf("-----\n");
16    printf("Matriz Sequencial Estática:\n");
17    printf("1 - Criar Matriz;\n");
18    printf("2 - Zerar Matriz;\n");
19    printf("3 - Inserir elemento;\n");
20    printf("4 - Consultar elemento;\n");
21    printf("5 - Imprimir Matriz;\n");
22    printf("6 - Criar Matriz Transposta;\n");
23    printf("7 - Verificar se a Matriz é quadrada;\n");
24    printf("8 - Verificar se a Matriz é simétrica;\n");
25    printf("9 - Criar cópia da Triangular Superior;\n");
26    printf("10 - Criar cópia da Triangular Inferior;\n");
27    printf("11 - Criar cópia da Diagonal;\n");
28    printf("12 - Sair;\n");
29    printf("-----\n");
30
31    do{
32
33        scanf("%d", &n);
34        switch(n){
35
36            case 1:
37                printf("Número de linhas: ");
38                scanf("%d", &l);
39                printf("Número de colunas: ");
40                scanf("%d", &c);
41                mat = criaMatriz(l, c);
42                if(mat != NULL) printf("Matriz criada com sucesso.\n");
43                break;
44
45            case 2:
46                if(zeraMatriz(mat)) printf("Matriz zerada com sucesso.\n");
47                break;
48
49            case 3:
50                printf("Elemento a ser inserido: ");
51                scanf("%d", &elem);
52                int l;
53                printf("Linha a ser inserido: ");
54                scanf("%d", &l);
55                int c;
56                printf("Coluna a ser inserido: ");
57                scanf("%d", &c);
58                if(inserirElemento(mat, elem, l, c)) printf("Elemento inserido com sucesso.\n");
59                break;
60
61            case 4:
62                printf("Linha a ser consultada: ");
63                scanf("%d", &l);
64                printf("Coluna a ser consultada: ");
65                scanf("%d", &c);
66                if(consultarElemento(mat, p, l, c)) printf("Elemento: %d.\n", *p);
67                break;
68
69            case 5:
70                imprimeMatriz(mat);
71                break;
72
73            case 6:
74                transp = criaTransposta(mat);
75                imprimeMatriz(transp);
76                destroiMatriz(transp);
77                break;
78
79            case 7:
80                if(testeQuadrada(mat)){
81                    printf("Matriz quadrada.\n");
82                }
83                else{
84                    printf("Matriz não quadrada.\n");
85                }
86                break;
87
88            case 8:
89                if(testeSimetrica(mat)){
90                    printf("Matriz simétrica.\n");
91                }
92                else{
93                    printf("Matriz não simétrica.\n");
94                }
95                break;
96
97            case 9:
98                triangSup = criaTriangSuperior(mat);
99                imprimeMatriz(triangSup);
100                destroiMatriz(triangSup);
101                break;
102
103            case 10:
104                triangInf = criaTriangInferior(mat);
105                imprimeMatriz(triangInf);
106                destroiMatriz(triangInf);
107                break;
108
109            case 11:
110                diag = criaDiagonal(mat);
111                imprimeMatriz(diag);
112                destroiMatriz(diag);
113                break;
114
115        }
116
117        printf("-----\n");
118
119    } while(n != 12);
120
121    printf("-----\n");
122    free(p);
123    return 0;
124
125 }
```

```
11> C main.c @ main0
49 case 3:
50     printf("Elemento a ser inserido: ");
51     scanf("%d", &elem);
52     int l;
53     printf("Linha a ser inserido: ");
54     scanf("%d", &l);
55     int c;
56     printf("Coluna a ser inserido: ");
57     scanf("%d", &c);
58     if(inserirElemento(mat, elem, l, c)) printf("Elemento inserido com sucesso.\n");
59     break;
60
61 case 4:
62     printf("Linha a ser consultada: ");
63     scanf("%d", &l);
64     printf("Coluna a ser consultada: ");
65     scanf("%d", &c);
66     if(consultarElemento(mat, p, l, c)) printf("Elemento: %d.\n", *p);
67     break;
68
69 case 5:
70     imprimeMatriz(mat);
71     break;
72
73 case 6:
74     transp = criaTransposta(mat);
75     imprimeMatriz(transp);
76     destroiMatriz(transp);
77     break;
78
79 case 7:
80     if(testeQuadrada(mat)){
81         printf("Matriz quadrada.\n");
82     }
83     else{
84         printf("Matriz não quadrada.\n");
85     }
86     break;
87
88 case 8:
89     if(testeSimetrica(mat)){
90         printf("Matriz simétrica.\n");
91     }
92     else{
93         printf("Matriz não simétrica.\n");
94     }
95     break;
96
97 case 9:
98     triangSup = criaTriangSuperior(mat);
99     imprimeMatriz(triangSup);
100     destroiMatriz(triangSup);
101     break;
102
103 case 10:
104     triangInf = criaTriangInferior(mat);
105     imprimeMatriz(triangInf);
106     destroiMatriz(triangInf);
107     break;
108
109 case 11:
110     diag = criaDiagonal(mat);
111     imprimeMatriz(diag);
112     destroiMatriz(diag);
113     break;
114
115 }
```

```
11> C main.c @ main0
82     }
83     else{
84         printf("Matriz não quadrada.\n");
85     }
86     break;
87
88 case 8:
89     if(testeSimetrica(mat)){
90         printf("Matriz simétrica.\n");
91     }
92     else{
93         printf("Matriz não simétrica.\n");
94     }
95     break;
96
97 case 9:
98     triangSup = criaTriangSuperior(mat);
99     imprimeMatriz(triangSup);
100     destroiMatriz(triangSup);
101     break;
102
103 case 10:
104     triangInf = criaTriangInferior(mat);
105     imprimeMatriz(triangInf);
106     destroiMatriz(triangInf);
107     break;
108
109 case 11:
110     diag = criaDiagonal(mat);
111     imprimeMatriz(diag);
112     destroiMatriz(diag);
113     break;
114
115 }
116
117 printf("-----\n");
118
119 } while(n != 12);
120
121 printf("-----\n");
122 free(p);
123 return 0;
124
125 }
```

```
11> C matriz.h ...
1 #ifndef MATRIZ_H
2 #define MATRIZ_H
3
4 #include <stdio.h>
5 #include <stdlib.h>
6 #define MAX 100
7
8 typedef struct{
9
10     int dados[MAX][MAX];
11     int lin, col;
12 } Matriz;
13
14 int zeraMatriz(Matriz* mat);
15
16 Matriz* criaMatriz(int l, int c){
17
18     if(l < 0 || l > MAX || c < 0 || c > MAX){
19         printf("Valores de linha e/ou coluna inválido(s).\n");
20         return NULL;
21     }
22
23     Matriz* mat = (Matriz*)malloc(sizeof(Matriz));
24
25     if(mat != NULL){
26         mat->lin = l;
27         mat->col = c;
28         zeraMatriz(mat);
29     }
30
31     return mat;
32
33 }
34
35 int destroiMatriz(Matriz* mat){
36
37     if(mat == NULL){
38         printf("Matriz não encontrada.\n");
39         return 0;
40     }
41
42     free(mat);
43     return 1;
44
45 }
46
47 int zeraMatriz(Matriz* mat){
48
49     if(mat == NULL){
50         printf("Matriz não encontrada.\n");
51         return 0;
52     }
53
54     for(int i = 0; i < mat->lin; i++){
55         for(int j = 0; j < mat->col; j++){
56             mat->dados[i][j] = 0;
57         }
58     }
59 }
```

```

11> C matrizh > ...
46 }
47
48 int zeraMatriz(Matriz* mat){
49
50     if(mat == NULL){
51         printf("Matriz não encontrada.\n");
52         return 0;
53     }
54
55     for(int i = 0; i < mat->lin; i++){
56         for(int j = 0; j < mat->col; j++){
57             mat->dados[i][j] = 0;
58         }
59     }
60
61     return 1;
62 }
63
64 int insereElemento(Matriz* mat, int elem, int l, int c){
65
66     if(mat == NULL){
67         printf("Matriz não encontrada.\n");
68         return 0;
69     }
70
71     if(l < 0 || l > MAX || c < 0 || c > MAX){
72         printf("Valores de linha e/ou coluna inválido(s).\n");
73         return 0;
74     }
75
76     mat->dados[l][c] = elem;
77     return 1;
78 }
79
80 }
81
82 int consultaElemento(Matriz* mat, int* p, int l, int c){
83
84     if(mat == NULL){
85         printf("Matriz não encontrada.\n");
86         return 0;
87     }
88
89     if(l < 0 || l > MAX || c < 0 || c > MAX){
90         printf("Valores de linha e/ou coluna inválido(s).\n");
91         return 0;
92     }
93
94     *p = mat->dados[l][c];
95     return 1;
96 }
97
98 }
99 void imprimeMatriz(Matriz* mat){
100
101     if(mat == NULL){
102         printf("Matriz não encontrada.\n");
103         return;
104     }
105
106     printf("Elementos:\n");
107     for(int i = 0; i < mat->lin; i++){
108         printf("\t");
109         for(int j = 0; j < mat->col; j++){
110             printf("%d\t", mat->dados[i][j]);
111         }
112         printf("\n");
113     }
114 }
115
116 Matriz* criaTransposta(Matriz* mat){
117
118     if(mat == NULL){
119         printf("Matriz não encontrada.\n");
120         return NULL;
121     }
122
123     Matriz* transp = criaMatriz(mat->col, mat->lin);
124
125     for(int i = 0; i < transp->lin; i++){
126         for(int j = 0; j < transp->col; j++){
127             transp->dados[i][j] = mat->dados[j][i];
128         }
129     }
130
131     return transp;
132 }
133
134 }
135
136 int testeQuadrada(Matriz* mat){
137
138     if(mat == NULL){
139         printf("Matriz não encontrada.\n");
140         return 0;
141     }
142
143     return (mat->lin == mat->col);
144 }
145
146 }
147
148 int testeSimetrica(Matriz* mat){
149
150     if(mat == NULL){
151         printf("Matriz não encontrada.\n");
152         return 0;
153     }
154
155     if(!testeQuadrada(mat)){
156         printf("Matriz não quadrada.\n");
157         return 0;
158     }
159
160     for(int i = 0; i < mat->lin; i++){
161         for(int j = 0; j < mat->col; j++){
162             if(mat->dados[i][j] != mat->dados[j][i]){
163                 return 0;
164             }
165         }
166     }
167
168     return 1;
169 }
170
171 Matriz* criaTriangSuperior(Matriz* mat){
172
173     if(mat == NULL){
174         printf("Matriz não encontrada.\n");
175         return 0;
176     }
177
178     if(!testeQuadrada(mat)){
179         printf("Matriz não quadrada.\n");
180         return 0;
181     }
182
183     Matriz* triangSup = criaMatriz(mat->lin, mat->col);
184
185     for(int i = 0; i < mat->lin; i++){
186         for(int j = 0; j < mat->col; j++){
187             if(i <= j){
188                 triangSup->dados[i][j] = mat->dados[i][j];
189             }
190         }
191     }
192
193     return triangSup;
194 }
195
196 }
197
198 Matriz* criaTriangInferior(Matriz* mat){
199
200     if(mat == NULL){
201         printf("Matriz não encontrada.\n");
202         return 0;
203     }
204
205     if(!testeQuadrada(mat)){
206         printf("Matriz não quadrada.\n");
207         return 0;
208     }
209
210     Matriz* triangInf = criaMatriz(mat->lin, mat->col);
211
212     for(int i = 0; i < mat->lin; i++){
213         for(int j = 0; j < mat->col; j++){
214             if(i >= j){
215                 triangInf->dados[i][j] = mat->dados[i][j];
216             }
217         }
218     }
219
220     return triangInf;
221 }
222
223 Matriz* criaDiagonal(Matriz* mat){
224
225     if(mat == NULL){
226         printf("Matriz não encontrada.\n");
227         return 0;
228     }
229
230     if(!testeQuadrada(mat)){
231         printf("Matriz não quadrada.\n");
232         return 0;
233     }
234
235     Matriz* diag = criaMatriz(mat->lin, mat->col);
236
237     for(int i = 0; i < diag->lin; i++){
238         diag->dados[i][i] = mat->dados[i][i];
239     }
240
241     return diag;
242 }
243
244 }
245 #endif

```

```

11> C matrizh > ...
146
147 int testeSimetrica(Matriz* mat){
148
149     if(mat == NULL){
150         printf("Matriz não encontrada.\n");
151         return 0;
152     }
153
154     if(!testeQuadrada(mat)){
155         printf("Matriz não quadrada.\n");
156         return 0;
157     }
158
159     for(int i = 0; i < mat->lin; i++){
160         for(int j = i+1; j < mat->col; j++){
161             if(mat->dados[i][j] != mat->dados[j][i]){
162                 return 0;
163             }
164         }
165     }
166
167     return 1;
168 }
169
170 }
171
172 Matriz* criaTriangSuperior(Matriz* mat){
173
174     if(mat == NULL){
175         printf("Matriz não encontrada.\n");
176         return 0;
177     }
178
179     if(!testeQuadrada(mat)){
180         printf("Matriz não quadrada.\n");
181         return 0;
182     }
183
184     Matriz* triangSup = criaMatriz(mat->lin, mat->col);
185
186     for(int i = 0; i < mat->lin; i++){
187         for(int j = 0; j < mat->col; j++){
188             if(i <= j){
189                 triangSup->dados[i][j] = mat->dados[i][j];
190             }
191         }
192     }
193
194     return triangSup;
195 }
196
197 }
198
199 Matriz* criaTriangInferior(Matriz* mat){
200
201     if(mat == NULL){
202         printf("Matriz não encontrada.\n");
203         return 0;
204     }
205
206     if(!testeQuadrada(mat)){
207         printf("Matriz não quadrada.\n");
208         return 0;
209     }
210
211     Matriz* triangInf = criaMatriz(mat->lin, mat->col);
212
213     for(int i = 0; i < mat->lin; i++){
214         for(int j = 0; j < mat->col; j++){
215             if(i >= j){
216                 triangInf->dados[i][j] = mat->dados[i][j];
217             }
218         }
219     }
220
221     return triangInf;
222 }
223
224 Matriz* criaDiagonal(Matriz* mat){
225
226     if(mat == NULL){
227         printf("Matriz não encontrada.\n");
228         return 0;
229     }
230
231     if(!testeQuadrada(mat)){
232         printf("Matriz não quadrada.\n");
233         return 0;
234     }
235
236     Matriz* diag = criaMatriz(mat->lin, mat->col);
237
238     for(int i = 0; i < diag->lin; i++){
239         diag->dados[i][i] = mat->dados[i][i];
240     }
241
242     return diag;
243 }
244
245 #endif

```

```

[lucascosta@fedora roteiro 7]$ cd 1.1
[lucascosta@fedora 1.1]$ gcc main.c -o main
[lucascosta@fedora 1.1]$ ./main
-----
Matriz Sequencial Estática:
1 - Criar Matriz;
2 - Zerar Matriz;
3 - Inserir elemento;
4 - Consultar elemento;
5 - Imprimir Matriz;
6 - Criar Matriz Transposta;
7 - Verificar se a Matriz é quadrada;
8 - Verificar se a Matriz é simétrica;
9 - Criar cópia da Triangular Superior;
10 - Criar cópia da Triangular Inferior;
11 - Criar cópia da Diagonal;
12 - Sair.
-----
1
Número de linhas: 3
Número de colunas: 3
Matriz criada com sucesso.
-----
3
Elemento a ser inserido: 15
Linha a ser inserido: 0
Coluna a ser inserido: 1
Elemento inserido com sucesso.
-----
3
Elemento a ser inserido: 65
Linha a ser inserido: 2
Coluna a ser inserido: 2
Elemento inserido com sucesso.
-----
5
Elementos:
[ 0 15 0 ]
[ 0 0 0 ]
[ 0 0 65 ]
-----
6
Elementos:
[ 0 0 0 ]
[ 15 0 0 ]
[ 0 0 65 ]
-----
4
Linha a ser consultada: 2
Coluna a ser consultada: 2
Elemento: 65.
-----
7
Matriz quadrada.
-----
8
Matriz não simétrica.
-----
9
Elementos:
[ 0 15 0 ]
[ 0 0 0 ]
[ 0 0 65 ]
-----
10
Elementos:
[ 0 0 0 ]
[ 0 0 0 ]
[ 0 0 65 ]
-----
11
Elementos:
[ 0 0 0 ]
[ 0 0 0 ]
[ 0 0 65 ]
-----
2
Matriz zerada com sucesso.
-----
5
Elementos:
[ 0 0 0 ]
[ 0 0 0 ]
[ 0 0 0 ]
-----
12
-----
[lucascosta@fedora 1.1]$

```

```

-----
9
Elementos:
[ 0 15 0 ]
[ 0 0 0 ]
[ 0 0 65 ]
-----
10
Elementos:
[ 0 0 0 ]
[ 0 0 0 ]
[ 0 0 65 ]
-----
11
Elementos:
[ 0 0 0 ]
[ 0 0 0 ]
[ 0 0 65 ]
-----
2
Matriz zerada com sucesso.
-----
5
Elementos:
[ 0 0 0 ]
[ 0 0 0 ]
[ 0 0 0 ]
-----
12
-----
[lucascosta@fedora 1.1]$

```

## 1.2

```

12> C main.c @ main0
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include "matriz.h"
4
5 int main()
6 {
7     int n, l, c, elem;
8     int* p = (int*)malloc(sizeof(int));
9     Matriz* mat;
10    Matriz* transp;
11    Matriz* triangSup;
12    Matriz* triangInf;
13    Matriz* diag;
14
15    printf("-----\n");
16    printf("Matriz Sequencial Estática:\n");
17    printf("1 - Criar Matriz;\n");
18    printf("2 - Zerar Matriz;\n");
19    printf("3 - Inserir elemento;\n");
20    printf("4 - Consultar elemento;\n");
21    printf("5 - Imprimir Matriz;\n");
22    printf("6 - Criar Matriz Transposta;\n");
23    printf("7 - Verificar se a Matriz é quadrada;\n");
24    printf("8 - Verificar se a Matriz é simétrica;\n");
25    printf("9 - Criar cópia da Triangular Superior;\n");
26    printf("10 - Criar cópia da Triangular Inferior;\n");
27    printf("11 - Criar cópia da Diagonal;\n");
28    printf("12 - Sair;\n");
29    printf("-----\n");
30
31    do{
32
33        scanf("%d", &n);
34        switch(n){
35
36            case 1:
37                printf("Número de linhas: ");
38                scanf("%d", &l);
39                printf("Número de colunas: ");
40                scanf("%d", &c);
41                mat = criaMatriz(l, c);
42                if(mat != NULL) printf("Matriz criada com sucesso.\n");
43                break;
44
45            case 2:
46                if(zeraMatriz(mat)) printf("Matriz zerada com sucesso.\n");
47                break;
48
49            case 3:
50                printf("Elemento a ser inserido: ");
51                scanf("%d", &elem);
52                int l;
53                printf("Linha a ser inserido: ");
54                scanf("%d", &l);
55                int c;
56                printf("Coluna a ser inserido: ");
57                scanf("%d", &c);
58                if(inserirElemento(mat, elem, l, c)) printf("Elemento inserido com sucesso.\n");
59                break;
60
61            case 4:
62                printf("Linha a ser consultada: ");
63                scanf("%d", &l);
64                printf("Coluna a ser consultada: ");
65                scanf("%d", &c);
66                if(consultaElemento(mat, p, l, c)) printf("Elemento: %d.\n", *p);
67                break;
68
69            case 5:
70                imprimeMatriz(mat);
71                break;
72
73            case 6:
74                transp = criaTransposta(mat);
75                imprimeMatriz(transp);
76                destroiMatriz(transp);
77                break;
78
79            case 7:
80                if(testeQuadrada(mat)){
81                    printf("Matriz quadrada.\n");
82                }
83                else{
84                    printf("Matriz não quadrada.\n");
85                }
86                break;
87
88            case 8:
89                if(testeSimetrica(mat)){
90                    printf("Matriz simétrica.\n");
91                }
92                else{
93                    printf("Matriz não simétrica.\n");
94                }
95                break;
96
97            case 9:
98                triangSup = criaTriangSuperior(mat);
99                imprimeMatriz(triangSup);
100                destroiMatriz(triangSup);
101                break;
102
103            case 10:
104                triangInf = criaTriangInferior(mat);
105                imprimeMatriz(triangInf);
106                destroiMatriz(triangInf);
107                break;
108
109            case 11:
110                diag = criaDiagonal(mat);
111                imprimeMatriz(diag);
112                destroiMatriz(diag);
113                break;
114
115            }
116
117        printf("-----\n");
118    } while(n != 12);
119
120    printf("-----\n");
121    free(p);
122    return 0;
123 }

```

```

12> C matriz.h @ main0
1 #ifndef MATRIZ_H
2 #define MATRIZ_H
3
4 #include <stdio.h>
5 #include <stdlib.h>
6
7 typedef struct{
8     int** dados;
9     int lin, col;
10 } Matriz;
11
12 int zeraMatriz(Matriz* mat);
13
14 Matriz* criaMatriz(int l, int c){
15
16     if(l < 0 || c < 0){
17         printf("Valores de linha e/ou coluna inválidos.\n");
18         return NULL;
19     }
20
21     Matriz* mat = (Matriz*)malloc(sizeof(Matriz));
22
23     if(mat != NULL){
24
25         mat->lin = l;
26         mat->col = c;
27
28         mat->dados = (int**)malloc(sizeof(int*));
29         for(int i = 0; i < mat->lin; i++){
30             mat->dados[i] = (int*)malloc(sizeof(int));
31         }
32
33         zeraMatriz(mat);
34
35     }
36
37     return mat;
38 }
39
40 int destroiMatriz(Matriz* mat){
41
42     if(mat == NULL){
43         printf("Matriz não encontrada.\n");
44         return 0;
45     }
46
47     for(int i = 0; i < mat->lin; i++){
48         free(mat->dados[i]);
49     }
50
51     free(mat->dados);
52     free(mat);
53
54     return 1;
55 }

```

```

12> C matriz.h > ...
59 }
60
61 int zeraMatriz(Matriz* mat){
62
63     if(mat == NULL){
64         printf("Matriz não encontrada.\n");
65         return 0;
66     }
67
68     for(int i = 0; i < mat->lin; i++){
69         for(int j = 0; j < mat->col; j++){
70             mat->dados[i][j] = 0;
71         }
72     }
73
74     return 1;
75 }
76
77
78 int insereElemento(Matriz* mat, int elem, int l, int c){
79
80     if(mat == NULL){
81         printf("Matriz não encontrada.\n");
82         return 0;
83     }
84
85     if(l < 0 || c < 0){
86         printf("Valores de linha e/ou coluna inválido(s).\n");
87         return 0;
88     }
89
90     mat->dados[l][c] = elem;
91     return 1;
92 }
93
94
95 int consultaElemento(Matriz* mat, int* p, int l, int c){
96
97     if(mat == NULL){
98         printf("Matriz não encontrada.\n");
99         return 0;
100     }
101
102     if(l < 0 || c < 0){
103         printf("Valores de linha e/ou coluna inválido(s).\n");
104         return 0;
105     }
106
107     *p = mat->dados[l][c];
108     return 1;
109 }
110
111
112 void imprimeMatriz(Matriz* mat){
113
114     if(mat == NULL){
115         printf("Matriz não encontrada.\n");
116         return;
117     }
118
119     printf("Elementos:\n");
120     for(int i = 0; i < mat->lin; i++){
121         printf("\t\t");
122         for(int j = 0; j < mat->col; j++){
123             printf("%d\t", mat->dados[i][j]);
124         }
125         printf("\n");
126     }
127 }
128
129
130 Matriz* criaTransposta(Matriz* mat){
131
132     if(mat == NULL){
133         printf("Matriz não encontrada.\n");
134         return NULL;
135     }
136
137     Matriz* transp = criaMatriz(mat->col, mat->lin);
138
139     for(int i = 0; i < transp->lin; i++){
140         for(int j = 0; j < transp->col; j++){
141             transp->dados[i][j] = mat->dados[j][i];
142         }
143     }
144
145     return transp;
146 }
147
148
149 int testeQuadrada(Matriz* mat){
150
151     if(mat == NULL){
152         printf("Matriz não encontrada.\n");
153         return 0;
154     }
155
156     return (mat->lin == mat->col);
157 }
158
159
160 int testeSimetrica(Matriz* mat){
161
162     if(mat == NULL){
163         printf("Matriz não encontrada.\n");
164         return 0;
165     }
166
167     if(!testeQuadrada(mat)){
168         printf("Matriz não quadrada.\n");
169         return 0;
170     }
171 }

```

```

12> C matriz.h > ...
159
160 int testeSimetrica(Matriz* mat){
161
162     if(mat == NULL){
163         printf("Matriz não encontrada.\n");
164         return 0;
165     }
166
167     if(!testeQuadrada(mat)){
168         printf("Matriz não quadrada.\n");
169         return 0;
170     }
171
172     for(int i = 0; i < mat->lin; i++){
173         for(int j = i+1; j < mat->col; j++){
174             if(mat->dados[i][j] != mat->dados[j][i]){
175                 return 0;
176             }
177         }
178     }
179
180     return 1;
181 }
182
183
184 Matriz* criaTriangSuperior(Matriz* mat){
185
186     if(mat == NULL){
187         printf("Matriz não encontrada.\n");
188         return 0;
189     }
190
191     if(!testeQuadrada(mat)){
192         printf("Matriz não quadrada.\n");
193         return 0;
194     }
195
196     Matriz* triangSup = criaMatriz(mat->lin, mat->col);
197
198     for(int i = 0; i < mat->lin; i++){
199         for(int j = 0; j < mat->col; j++){
200             if(i <= j){
201                 triangSup->dados[i][j] = mat->dados[i][j];
202             }
203         }
204     }
205
206     return triangSup;
207 }
208
209
210 Matriz* criaTriangInferior(Matriz* mat){
211
212     if(mat == NULL){
213         printf("Matriz não encontrada.\n");
214         return 0;
215     }
216
217     if(!testeQuadrada(mat)){
218         printf("Matriz não quadrada.\n");
219         return 0;
220     }
221
222     Matriz* triangInf = criaMatriz(mat->lin, mat->col);
223
224     for(int i = 0; i < mat->lin; i++){
225         for(int j = 0; j < mat->col; j++){
226             if(i >= j){
227                 triangInf->dados[i][j] = mat->dados[i][j];
228             }
229         }
230     }
231
232     return triangInf;
233 }
234
235
236 Matriz* criaDiagonal(Matriz* mat){
237
238     if(mat == NULL){
239         printf("Matriz não encontrada.\n");
240         return 0;
241     }
242
243     if(!testeQuadrada(mat)){
244         printf("Matriz não quadrada.\n");
245         return 0;
246     }
247
248     Matriz* diag = criaMatriz(mat->lin, mat->col);
249
250     for(int i = 0; i < diag->lin; i++){
251         diag->dados[i][i] = mat->dados[i][i];
252     }
253
254     return diag;
255 }
256
257
258 #endif

```

```
[lucascosta@fedora 1.2]$ ./main
-----
Matriz Sequencial Estática:
1 - Criar Matriz;
2 - Zerar Matriz;
3 - Inserir elemento;
4 - Consultar elemento;
5 - Imprimir Matriz;
6 - Criar Matriz Transposta;
7 - Verificar se a Matriz é quadrada;
8 - Verificar se a Matriz é simétrica;
9 - Criar cópia da Triangular Superior;
10 - Criar cópia da Triangular Inferior;
11 - Criar cópia da Diagonal;
12 - Sair.
-----
1
Número de linhas: 3
Número de colunas: 3
Matriz criada com sucesso.
-----
3
Elemento a ser inserido: 5
Linha a ser inserido: 2
Coluna a ser inserido: 2
Elemento inserido com sucesso.
-----
3
Elemento a ser inserido: 1
Linha a ser inserido: 0
Coluna a ser inserido: 0
Elemento inserido com sucesso.
-----
3
Elemento a ser inserido: 5
Linha a ser inserido: 2
Coluna a ser inserido: 1
Elemento inserido com sucesso.
-----
5
Elementos:
[ 1 0 0 ]
[ 0 0 0 ]
[ 0 5 5 ]
-----
6
Elementos:
[ 1 0 0 ]
[ 0 0 5 ]
[ 0 0 5 ]
-----
4
Linha a ser consultada: 2
Coluna a ser consultada: 1
Elemento: 5.
-----
6
Elementos:
[ 1 0 0 ]
[ 0 0 5 ]
[ 0 0 5 ]
-----
```

```
7
-----
Matriz quadrada.
-----
8
Matriz não simétrica.
-----
9
Elementos:
[ 1 0 0 ]
[ 0 0 0 ]
[ 0 0 5 ]
-----
11
Elementos:
[ 1 0 0 ]
[ 0 0 0 ]
[ 0 0 5 ]
-----
10
Elementos:
[ 1 0 0 ]
[ 0 0 0 ]
[ 0 5 5 ]
-----
2
Matriz zerada com sucesso.
-----
5
Elementos:
[ 0 0 0 ]
[ 0 0 0 ]
[ 0 0 0 ]
-----
12
-----
[lucascosta@fedora 1.2]$
```



## 2.1

```

21> C main.c > @ main0
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include "matriz.h"
4
5 int main(){
6
7     int n, tam, i, j, elem;
8     int* p = (int*)malloc(sizeof(int));
9     MFaixa* mat;
10
11     printf("-----\n");
12     printf("Matriz de Faixa (Tridiagonal):\n");
13     printf("1 - Criar Matriz;\n");
14     printf("2 - Zerar Matriz;\n");
15     printf("3 - Inserir elemento;\n");
16     printf("4 - Consultar elemento;\n");
17     printf("5 - Imprimir Matriz;\n");
18     printf("6 - Destruir Matriz;\n");
19     printf("7 - Sair;\n");
20     printf("-----\n");
21     do{
22         scanf("%d", &n);
23
24         switch(n){
25
26             case 1:
27                 printf("Tamanho da Matriz: ");
28                 scanf("%d", &tam);
29                 mat = criaMatriz(tam);
30                 if(mat != NULL) printf("Matriz criada com sucesso.\n");
31                 break;
32
33             case 2:
34                 if(zeraMatriz(mat)) printf("Matriz zerada com sucesso.\n");
35                 break;
36
37             case 3:
38                 printf("Elemento a ser inserido: ");
39                 scanf("%d", &elem);
40                 printf("Linha a ser inserido: ");
41                 scanf("%d", &i);
42                 printf("Coluna a ser inserido: ");
43                 scanf("%d", &j);
44                 if(inserirElemento(mat, elem, i, j)) printf("Elemento inserido com sucesso.\n");
45                 break;
46
47             case 4:
48                 printf("Linha a ser consultada: ");
49                 scanf("%d", &i);
50                 printf("Coluna a ser consultada: ");
51                 scanf("%d", &j);
52                 printf("Elemento: %d\n", consultaElemento(mat, i, j));
53                 break;
54
55             case 5:
56                 imprimirMatriz(mat);
57
58             case 6:
59                 if(destroiMatriz(mat)) printf("Matriz destruida com sucesso.\n");
60                 break;
61
62             case 7:
63                 break;
64
65             default:
66                 printf("-----\n");
67         }
68     } while(n != 7);
69
70     printf("-----\n");
71     free(p);
72     return 0;
73
74 }

```

```

21> C matriz.h > ...
1 #ifndef MATRIZ_H
2 #define MATRIZ_H
3
4 #include <stdio.h>
5 #include <stdlib.h>
6
7 typedef struct{
8     int* diagonal;
9     int* superior;
10    int* inferior;
11    int tam;
12
13 } MFaixa;
14
15 int zeraMatriz(MFaixa* mat);
16
17 MFaixa* criaMatriz(int tam){
18
19     MFaixa* mat = (MFaixa*)malloc(sizeof(MFaixa));
20
21     if(mat != NULL){
22         if(tam <= 1){
23             printf("ERRO: Tamanho da matriz inválido.\n");
24             return NULL;
25         }
26
27         mat->tam = tam;
28         mat->diagonal = (int*)malloc(tam * sizeof(int));
29         mat->superior = (int*)malloc((tam-1) * sizeof(int));
30         mat->inferior = (int*)malloc((tam-1) * sizeof(int));
31
32         if(mat->diagonal == NULL || mat->superior == NULL || mat->inferior == NULL){
33             return NULL;
34         }
35         zeraMatriz(mat);
36     }
37
38     return mat;
39 }
40
41 int destroiMatriz(MFaixa* mat){
42
43     if(mat == NULL){
44         printf("ERRO: Matriz não encontrada.\n");
45         return 0;
46     }
47
48     free(mat->diagonal);
49     free(mat->superior);
50     free(mat->inferior);
51     free(mat);
52
53     return 1;
54 }
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100

```

```

21> C:matriz.h> ...
80 }
81
82 int insereElemento(MFaixa* mat, int elem, int i, int j){
83
84     if(mat == NULL){
85         printf("ERRO: Matriz não encontrada.\n");
86         return 0;
87     }
88
89     if(i == j){
90         mat->diagonal[i] = elem;
91     }
92
93     else if(i+1 == j){
94         mat->superior[i] = elem;
95     }
96
97     else if(i == j+1){
98         mat->inferior[j] = elem;
99     }
100
101     else{
102         printf("ERRO: Posição do elemento fora dos limites da matriz.\n");
103         return 0;
104     }
105
106     return 1;
107
108 }
109
110 int consultaElemento(MFaixa* mat, int i, int j){
111
112     if(mat == NULL){
113         printf("ERRO: Matriz não encontrada.\n");
114         return 0;
115     }
116
117     if(i == j){
118         return mat->diagonal[i];
119     }
120
121     else if(i+1 == j){
122         return mat->superior[i];
123     }
124
125     else if(i == j+1){
126         return mat->inferior[j];
127     }
128
129     else{
130         printf("ERRO: Posição do elemento fora dos limites da matriz.\n");
131         return 0;
132     }
133
134 }
135
136 void imprimeMatriz(MFaixa* mat){
137
138     printf("ERRO: Posição do elemento fora dos limites da matriz.\n");
139     return 0;
140 }
141
142 void imprimeMatriz(MFaixa* mat){
143
144     if(mat == NULL){
145         printf("ERRO: Matriz não encontrada.\n");
146     }
147
148     for(int i = 0; i < mat->tam; i++){
149         for(int j = 0; j < mat->tam; j++){
150
151             if(i == j){
152                 printf("%d\t", mat->diagonal[i]);
153             }
154
155             else if(i+1 == j){
156                 printf("%d\t", mat->superior[i]);
157             }
158
159             else if(i == j+1){
160                 printf("%d\t", mat->inferior[j]);
161             }
162
163             else{
164                 printf("0\t");
165             }
166
167             printf("\n");
168         }
169     }
170 }
171 #endif

```

```

[Lucascosta@fedora 2.1]$ gcc main.c -o main
[Lucascosta@fedora 2.1]$ ./main
-----
Matriz de Faixa (Tridiagonal):
1 - Criar Matriz;
2 - Zerar Matriz;
3 - Inserir elemento;
4 - Consultar elemento;
5 - Imprimir Matriz;
6 - Destruir Matriz;
7 - Sair.
-----
1
Tamanho da Matriz: 3
Matriz criada com sucesso.
-----
3
Elemento a ser inserido: 5
Linha a ser inserido: 1
Coluna a ser inserido: 2
Elemento inserido com sucesso.
-----
3
Elemento a ser inserido: 15
Linha a ser inserido:
0
Coluna a ser inserido: 1
Elemento inserido com sucesso.
-----
4
Linha a ser consultada: 0
Coluna a ser consultada: 1
Elemento: 15
-----
5
0      15      0
0      0       5
0      0       0
-----
2
Matriz zerada com sucesso.
-----
5
0      0      0
0      0      0
0      0      0
-----
6
Matriz destruída com sucesso.
-----
7
-----
[Lucascosta@fedora 2.1]$

```



## 2.2

```

22 > C main.c > ...
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include "matrizdin.h"
4 #include "matrizcsr.h"
5
6 int main(){
7
8     int n, l, c, elem, QNN;
9     Matriz* mat;
10
11     MESparsiCSR* matE;
12
13     printf("-----\n");
14     printf("Matriz Esparsa CSR:\n");
15     printf("1 - Criar Matriz;\n");
16     printf("2 - Criar Matriz Dinamica;\n");
17     printf("3 - Inserir elemento;\n");
18     printf("4 - Remover elemento;\n");
19     printf("5 - Consultar elemento;\n");
20     printf("6 - Imprimir Matriz;\n");
21     printf("7 - Transformar Matriz Dinamica em CSR;\n");
22     printf("8 - Sair;\n");
23     printf("-----\n");
24
25     do{
26
27         scanf("%d", &n);
28
29         switch(n){
30
31             case 1:
32                 printf("Número de elementos nao nulos: ");
33                 scanf("%d", &QNN);
34                 printf("Número de linhas: ");
35                 scanf("%d", &l);
36                 printf("Número de colunas: ");
37                 scanf("%d", &c);
38                 matE = criaMatrizEsparsa(l, c, QNN);
39                 if(matE != NULL) printf("Matriz criada com sucesso.\n");
40                 break;
41
42             case 2:
43                 printf("Número de linhas: ");
44                 scanf("%d", &l);
45                 printf("Número de colunas: ");
46                 scanf("%d", &c);
47                 mat = criaMatriz(l, c);
48                 if(mat != NULL) printf("Matriz criada com sucesso.\n");
49                 preencheAleatorio(mat, 0, 9);
50                 break;
51
52             case 3:
53                 printf("Elemento a ser inserido: ");
54                 scanf("%d", &elem);
55                 printf("Linha a ser inserido: ");
56                 scanf("%d", &l);
57                 printf("Coluna a ser inserido: ");
58                 scanf("%d", &c);
59
60                 case 3:
61                     printf("Elemento a ser inserido: ");
62                     scanf("%d", &elem);
63                     printf("Linha a ser inserido: ");
64                     scanf("%d", &l);
65                     printf("Coluna a ser inserido: ");
66                     scanf("%d", &c);
67                     if(InserirElemEsparsa(matE, elem, l, c)) printf("Elemento inserido com sucesso.\n");
68                     break;
69
70             case 4:
71                 printf("Linha a ser removido: ");
72                 scanf("%d", &l);
73                 printf("Coluna a ser removido: ");
74                 scanf("%d", &c);
75                 if(removeElemEsparsa(matE, l, c)) printf("Elemento removido com sucesso.\n");
76                 break;
77
78             case 5:
79                 printf("Linha a ser consultada: ");
80                 scanf("%d", &l);
81                 printf("Coluna a ser consultada: ");
82                 scanf("%d", &c);
83                 printf("Elemento: %d.\n", consultaElemEsparsa(matE, l, c));
84                 break;
85
86             case 6:
87                 imprimeEsparsa(matE);
88                 break;
89
90             case 7:
91                 matE = transformarEmCSR(mat);
92                 imprimeEsparsa(matE);
93                 break;
94
95             default:
96                 printf("-----\n");
97         }
98     } while(n != 8);
99
100     destroiMatriz(mat);
101     destroiMatrizEsparsa(matE);
102
103     printf("-----\n");
104
105     return 0;
106 }

```

```

22 > C matrizcsr.h > ...
1 #ifndef MATRIZCSR_H
2 #define MATRIZCSR_H
3 #include <stdio.h>
4 #include <stdlib.h>
5 #include <time.h>
6 #include "matrizdin.h"
7
8
9 typedef struct{
10     int *A; //Valores
11     int *IA;
12     int *JA;
13     int lin, col, QNN, QI;
14 }MESparsiCSR;
15 //QNN - Quantidade de Nao Nulos
16 //QI - Quantidade de Inseridos
17
18 MESparsiCSR* criaMatrizEsparsa(int l, int c, int qnn){
19     MESparsiCSR* ms;
20     ms = (MESparsiCSR*) malloc (sizeof(MEsparsiCSR));
21     if(ms != NULL){
22         if(l <= 0 || c <= 0 || qnn < 0){
23             printf("Valores invalidos, matriz nao criada!\n");
24             return NULL;
25         }
26         ms->lin = l; ms->col = c;
27         ms->QI = 0; ms->QNN = qnn;
28         ms->A = ms->JA = ms->IA = NULL;
29         if(qnn != 0){
30             ms->A = (int*) malloc (qnn*sizeof(int));
31             ms->JA = (int*) malloc (qnn*sizeof(int));
32             if(ms->A == NULL || ms->JA == NULL) return NULL;
33         }
34         ms->IA = (int*) malloc ((ms->lin+1)*sizeof(int));
35         if(ms->IA == NULL) return NULL;
36         int i; for(i=0; i<lin; i++) ms->IA[i] = 0;
37     }
38     return ms;
39 }
40
41 int* meuRealloc(int* v, int tam){
42     int* aux = (int*) malloc ((tam+1)*sizeof(int));
43     if(aux != NULL){
44         if(v != NULL){
45             int i;
46             for(i=0; i<tam; i++){
47                 aux[i] = v[i];
48             }
49             free(v);
50         }
51         return aux;
52     }
53 }
54 void imprimeEsparsaVetores(MEsparsiCSR* ms){
55     if(ms == NULL) return;
56     int i;
57     printf("Matriz Esparsa, Tam: %d x %d.\n", ms->lin, ms->col);
58     printf("%d elementos nao nulos.\n", ms->QNN);
59
60 void imprimeEsparsaVetores(MEsparsiCSR* ms){
61     if(ms == NULL) return;
62     int i;
63     printf("Matriz Esparsa, Tam: %d x %d.\n", ms->lin, ms->col);
64     printf("%d elementos nao nulos.\n", ms->QNN);
65     printf("A = [");
66     for(i=0; i<ms->QNN; i++){
67         printf("%d ", ms->A[i]);
68     }
69     printf("]\n");
70     printf("IA = [");
71     for(i=0; i<ms->lin+1; i++){
72         printf("%d ", ms->IA[i]);
73     }
74     printf("]\n");
75     printf("JA = [");
76     for(i=0; i<ms->QNN; i++){
77         printf("%d ", ms->JA[i]);
78     }
79     printf("]\n");
80 }
81
82 int inserirElemEsparsa(MEsparsiCSR* ms, int elem, int l, int j){
83     if(ms == NULL) return 0;
84     if(l < 0 || j < 0 || l > ms->lin || j > ms->col){
85         printf("Valores invalidos, elem nao inserido!\n");
86         return 0;
87     }
88     int k;
89     int index = -1;
90     int ini = ms->IA[l]; int fim = ms->IA[l+1];
91     // Encontre a posicao correta para inserir o valor
92     for(k = ini; k<fim; k++){
93         if (ms->JA[k] >= j) {
94             index = k;
95             break;
96         }
97     }
98     if (index == -1) { //NOVA INSERCAO
99         if(ms->QI == ms->QNN){//Necessita REALLOC
100             ms->A = meuRealloc(ms->A, ms->QNN);
101             ms->JA = meuRealloc(ms->JA, ms->QNN);
102             ms->QNN++;
103         }
104         //Move elementos para a nova insercao
105         for(k = ms->QNN-1; k>=fim; k--){
106             ms->A[k] = ms->A[k-1];
107             ms->JA[k] = ms->JA[k-1];
108         }
109         ms->A[fim] = elem;
110         ms->JA[fim] = j;
111         ms->QI++;
112         // Atualiza QNN acumulado
113         for (int k = i+1; k<ms->lin; k++){
114             ms->IA[k]++;
115         }
116     } else { //Atualiza um valor existente
117         ms->A[index] = elem;
118     }
119     imprimeEsparsaVetores(ms);
120     return 1;
121 }

```

```

22 > C:matrizcsr.h >...
112
113 int removeElemEsparsa(MEsparsaCSR *ms, int i, int j) {
114     if(ms == NULL) return 0;
115     if(i < 0 || j < 0 || i >= ms->lin || j >= ms->col){
116         printf("Valores invalidos, elem nao removido\n");
117         return 0;
118     }
119
120     int k;
121     int index = -1;
122     int ini = ms->IA[i]; int fim = ms->IA[i+1];
123     // Encontre a posição do valor a ser removido
124     for(k = ini; k<fim; k++){
125         if (ms->JA[k] == j) {
126             index = k;
127             break;
128         }
129     }
130
131     if (index != -1) {
132         // Move todos elementos uma posição para tras
133         for (k = index; k < ms->QNN - 1; k++) {
134             ms->A[k] = ms->A[k+1];
135             ms->JA[k] = ms->JA[k+1];
136         }
137         ms->QNN--;
138         // Atualiza QNN acumulado
139         for (int k = i+1; k<ms->lin; k++)
140             ms->IA[k]--;
141     }
142     printf("Elemento nao existente\n"); return 0;
143
144     imprimeEsparsaVetores(ms);
145     return 1;
146 }
147
148
149 MEsparsaCSR* transformarEmCSR(Matriz* mat){
150     MEsparsaCSR *ms = criaMatrizEsparsa(mat->lin, mat->col, 0);
151     if(ms != NULL){
152         if(mat == NULL){
153             printf("Matriz de entrada inexistente\n");
154             return NULL;
155         }
156
157         int i, j;
158         for(i=0; i<mat->lin; i++){
159             for(j=0; j<mat->col; j++){
160                 if(mat->dados[i][j] != 0)
161                     insereElemEsparsa(ms, mat->dados[i][j], i, j);
162             }
163         }
164     }
165     return ms;
166 }
167
168 int consultaElemEsparsa(MEsparsaCSR* ms, int i, int j){
169     if(ms == NULL) return 0;
170     if(i < 0 || j < 0 || i >= ms->lin || j >= ms->col){
171         printf("Valores invalidos, elem inexistente\n");
172         return 0;
173     }
174     int k;
175     int index = -1;
176     int ini = ms->IA[i]; int fim = ms->IA[i+1];
177     for(k = ini; k<fim; k++){
178         if (ms->JA[k] == j) {
179             index = k;
180             break;
181         }
182     }
183     if (index != -1) {
184         return ms->A[index];
185     }
186     return 0;
187 }
188
189 void imprimeEsparsa(MEsparsaCSR* ms){
190     if(ms == NULL) return;
191     int i, j;
192     imprimeEsparsaVetores(ms);
193     printf("Matriz Original:\n");
194     for(i=0; i<ms->lin; i++){
195         for(j=0; j<ms->col; j++){
196             printf("%d\t", consultaElemEsparsa(ms, i, j));
197         }
198         printf("\n");
199     }
200 }
201
202 void destroiMatrizEsparsa(MEsparsaCSR* ms){
203     if(ms != NULL){
204         free(ms->A);
205         free(ms->IA);
206         free(ms->JA);
207         free(ms);
208     }
209 }
210
211 #endif

```

```

22 > C:matriz.h >...
1 #ifndef MATRIZDIN_H
2 #define MATRIZDIN_H
3
4 #include <stdio.h>
5 #include <stdlib.h>
6 #include <time.h>
7
8 typedef struct{
9     int **dados;
10    int lin, col;
11 }Matriz;
12
13 void zeraMatriz(Matriz* mat){
14     int i, j;
15     for(i=0; i<mat->lin; i++){
16         for(j=0; j<mat->col; j++){
17             mat->dados[i][j] = 0;
18         }
19     }
20 }
21
22 Matriz* criaMatriz(int l, int c){
23     Matriz* mat;
24     mat = (Matriz*) malloc (sizeof(Matriz));
25     if(mat != NULL){
26         if(l <= 0 || c <= 0){
27             printf("Valores invalidos, matriz nao criada\n");
28             return NULL;
29         }
30         int i;
31         mat->lin = l;
32         mat->col = c;
33         mat->dados = (int**) malloc (l*sizeof(int*));
34         for(i=0; i<l; i++){
35             mat->dados[i] = (int*) malloc (c*sizeof(int));
36             zeraMatriz(mat);
37         }
38     }
39     return mat;
40 }
41
42 void destroiMatriz(Matriz* mat){
43     if(mat != NULL){
44         int i;
45         for(i=0; i<mat->lin; i++){
46             free(mat->dados[i]);
47             free(mat->dados);
48             free(mat);
49         }
50     }
51 }
52
53 int preencheAleatorio(Matriz* mat, int ini, int fim){
54     if(mat == NULL) return 0;
55     srand(time(NULL));
56     int i, j;
57     for(i=0; i<mat->lin; i++){
58         for(j=0; j<mat->col; j++){
59             mat->dados[i][j] = ini + rand() % (fim-ini + 1);
60         }
61     }
62     return 1;
63 }
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91 //Matrizes Quadradas e propriedades
92
93 int e_matrizQuadrada(Matriz *mat){
94     if(mat == NULL) return 0;
95     return (mat->lin == mat->col);
96 }
97
98 Matriz* criaTriangularSup(Matriz* mat){
99     if(mat == NULL) return NULL;
100     if(!e_matrizQuadrada(mat)){
101         printf("Matriz nao Quadrada!\n");
102         return NULL;
103     }
104     int i, j;
105     Matriz* ts = criaMatriz(mat->lin, mat->col);
106     for(i=0; i<mat->lin; i++){
107         for(j=0; j<mat->col; j++){
108             if(i <= j)
109                 ts->dados[i][j] = mat->dados[i][j];
110             else
111                 ts->dados[i][j] = 0;
112         }
113     }
114     return ts;
115 }
116
117 Matriz* criaTriangularInf(Matriz* mat){
118     if(mat == NULL) return NULL;
119     if(!e_matrizQuadrada(mat)){
120         printf("Matriz nao Quadrada!\n");
121         return NULL;
122     }
123     int i, j;
124     Matriz* ti = criaMatriz(mat->lin, mat->col);
125     for(i=0; i<mat->lin; i++){
126         for(j=0; j<mat->col; j++){
127             if(i >= j)
128                 ti->dados[i][j] = mat->dados[i][j];
129             else
130                 ti->dados[i][j] = 0;
131         }
132     }
133     return ti;
134 }

```

2.2 > C matrizdin.h > ...

```
112  
113 Matriz* criaTriangularInf(Matriz* mat){  
114     if(mat == NULL) return NULL;  
115     if(!e_matrizQuadrada(mat)){  
116         printf("Matriz nao Quadrada!\n");  
117         return NULL;  
118     }  
119     int i, j;  
120     Matriz* ti = criaMatriz(mat->lin, mat->col);  
121     for(i=0; i<mat->lin; i++)  
122         for(j=0; j<mat->col; j++)  
123             if(i >= j)  
124                 ti->dados[i][j] = mat->dados[i][j];  
125     return ti;  
126 }  
127  
128 Matriz* criaDiagonal(Matriz* mat){  
129     if(mat == NULL) return NULL;  
130     if(!e_matrizQuadrada(mat)){  
131         printf("Matriz nao Quadrada!\n");  
132         return NULL;  
133     }  
134     int i;  
135     Matriz* d = criaMatriz(mat->lin, mat->col);  
136     for(i=0; i<mat->lin; i++)  
137         d->dados[i][i] = mat->dados[i][i];  
138     return d;  
139 }  
140  
141 int e_Simetrica(Matriz* mat){  
142     if(mat == NULL) return 0;  
143     if(!e_matrizQuadrada(mat)){  
144         printf("Matriz nao Quadrada!\n");  
145         return 0;  
146     }  
147     int i, j;  
148     for(i=0; i<mat->lin; i++)  
149         for(j=i+1; j<mat->col; j++)  
150             if(mat->dados[i][j] != mat->dados[j][i])  
151                 return 0;  
152     return 1;  
153 }  
154  
155 Matriz* criaTransposta(Matriz* mat){  
156     if(mat == NULL) return NULL;  
157     Matriz* t = criaMatriz(mat->col, mat->lin);  
158     int i, j;  
159     for(i=0; i<mat->lin; i++)  
160         for(j=0; j<mat->col; j++)  
161             t->dados[j][i] = mat->dados[i][j];  
162     return t;  
163 }  
164  
165 #endif
```

2.2 > C matrizdin.h > ...

119 | 118 1, J,

PROBLEMAS SAÍDA TERMINAL PORTAS CONSOLE DE DEPURAÇÃO

[lucascosta@fedora 2.2]\$ ./main

-----  
Matriz Esparsa CSR:

- 1 - Criar Matriz;
- 2 - Criar Matriz Dinamica;
- 3 - Inserir elemento;
- 4 - Remover elemento;
- 5 - Consultar elemento;
- 6 - Imprimir Matriz;
- 7 - Transformar Matriz Dinamica em CSR;
- 8 - Sair.

-----  
1

Número de elementos nao nulos: 3

Número de linhas: 2

Número de colunas: 2

Matriz criada com sucesso.

-----  
3

Elemento a ser inserido: 12

Linha a ser inserido: 0

Coluna a ser inserido: 0

Matriz Esparsa, Tam: 2 x 2:

3 elementos nao nulos.

A = [12 0 0 ]

IA = [0 1 1 ]

JA = [0 0 0 ]

Elemento inserido com sucesso.

-----  
3

Elemento a ser inserido: 1

Linha a ser inserido: 1

Coluna a ser inserido: 0

Matriz Esparsa, Tam: 2 x 2:

3 elementos nao nulos.

A = [12 1 0 ]

IA = [0 1 2 ]

JA = [0 0 0 ]

Elemento inserido com sucesso.

-----  
2

Número de linhas: 2

Número de colunas: 2

Matriz criada com sucesso.

-----  
6

Matriz Esparsa, Tam: 2 x 2:

3 elementos nao nulos.

A = [12 1 0 ]

IA = [0 1 2 ]

JA = [0 0 0 ]

Matriz Original:

12      0

2.2 > C matrizdin.h > ...

119 | 118 1, J,

PROBLEMAS SAÍDA TERMINAL PORTAS CONSOLE DE DEPURAÇÃO

```
A = [12 1 0 ]
IA = [0 1 2 ]
JA = [0 0 0 ]
```

Matriz Original:

```
12    0
1      0
```

-----

```
4
Linha a ser removido: 1
Coluna a ser removido: 0
Matriz Esparsa, Tam: 2 x 2:
2 elementos nao nulos.
A = [12 0 ]
IA = [0 1 1 ]
JA = [0 0 ]
```

Elemento removido com sucesso.

-----

```
7
Matriz Esparsa, Tam: 2 x 2:
1 elementos nao nulos.
A = [9 ]
IA = [0 1 1 ]
JA = [0 ]
```

```
Matriz Esparsa, Tam: 2 x 2:
2 elementos nao nulos.
A = [9 5 ]
IA = [0 2 2 ]
JA = [0 1 ]
```

```
Matriz Esparsa, Tam: 2 x 2:
3 elementos nao nulos.
A = [9 5 8 ]
IA = [0 2 3 ]
JA = [0 1 0 ]
```

```
Matriz Esparsa, Tam: 2 x 2:
4 elementos nao nulos.
A = [9 5 8 5 ]
IA = [0 2 4 ]
JA = [0 1 0 1 ]
```

```
Matriz Esparsa, Tam: 2 x 2:
4 elementos nao nulos.
A = [9 5 8 5 ]
IA = [0 2 4 ]
JA = [0 1 0 1 ]
```

Matriz Original:

```
9      5
8      5
```

-----

8

-----