

## Lista 7 de Laboratório de Programação II

Nome : Messias Feres Curi Melo

Matrícula : 2022003764

### Problema 1

#### 1.1

main.c

```
atividade1.1 > C main.c > main()
1  #include <stdio.h>
2  #include <stdbool.h>
3  #include "matriz.h"
4
5  int main()
6  {
7      Matriz *M, *TS, *TI, *D, *T;
8      int elemento, linha, coluna, escolha, inicio, fim;
9
10     printf("-<X Menu X>-\n");
11     printf("1 - Criar Matriz\n");
12     printf("2 - Zerar Matriz\n");
13     printf("3 - Preencher Aleatoriamente\n");
14     printf("4 - Inserir Elemento\n");
15     printf("5 - Consultar Elemento\n");
16     printf("6 - É Matriz quadrada?\n");
17     printf("7 - Criar Triangulo Sup\n");
18     printf("8 - Criar Triangulo Inf\n");
19     printf("9 - Criar Diagonal\n");
20     printf("10 - Criar Transposta\n");
21     printf("11 - É Simetrica?\n");
22     printf("12 - Imprimir Matriz\n");
23     printf("13 - Sair\n");
24
25     while(true){
26         printf("Escolha: ");
27         scanf("%d", &escolha);
28
29         if(escolha == 1){
30             printf("-> Insira a quantidade de linha: ");
31             scanf("%d",&linha);
32             printf("-> Insira a quantidade de coluna: ");
33             scanf("%d",&coluna);
34             M = criaMatriz(linha, coluna);
35             printf("-> Criado com sucesso!\n");
36         }else if(escolha == 2){
37             zeraMatriz(M);
38             printf("-> Matriz Zerada com sucesso!\n");
39         }else if(escolha == 3){
40             printf("-> Insira um valor inicial: ");
41             scanf("%d",&inicio);
42             printf("-> Insira um valor final: ");
43             scanf("%d",&fim);
44             preencheAleatorio(M, inicio, fim);
45             printf("-> Matriz preenchida aleatoriamente!\n");
46         }else if(escolha == 4){
47             printf("-> Elemento a ser inserido: ");
48             scanf("%d",&elemento);
```

```

48     printf("-> Insira a linha: ");
49     scanf("%d",&linha);
50     printf("-> Insira a coluna: ");
51     scanf("%d",&coluna);
52     insereElem(M, elemento, linha, coluna);
53     printf("-> Inserido com sucesso!\n");
54 }else if(escolha == 5){
55     printf("-> Insira a linha: ");
56     scanf("%d",&linha);
57     printf("-> Insira a coluna: ");
58     scanf("%d",&coluna);
59     consultaElem(M, &elemento, linha, coluna);
60     printf("-> Elemento consultado: %d\n", elemento);
61 }else if(escolha == 6){
62     if(e_matrizQuadrada(M)){
63         printf("-> Matriz Quadrada!\n");
64     }else{
65         printf("-> Matriz não é Quadrada!\n");
66     }
67 }else if(escolha == 7){
68     TS = criaTriangularSup(M);
69     imprime(TS);
70 }else if(escolha == 8){
71     TI = criaTriangularInf(M);
72     imprime(TI);
73 }else if(escolha == 9){
74     D = criaDiagonal(M);
75     imprime(D);
76 }else if(escolha == 10){
77     T = criaTransposta(M);
78     imprime(T);
79 }else if(escolha == 11){
80     if(e_Simetrica(M)){
81         printf("-> Matriz Simétrica!\n");
82     }else{
83         printf("-> Matriz não Simétrica!\n");
84     }
85 }else if(escolha == 12){
86     imprime(M);
87 }else if(escolha == 13){
88     destroiMatriz(M);
89     printf("-> Finalizando!\n");
90     break;
91 }else{
92     printf("-> Número inválido!\n");
93 }
94 }
95 return 0;
96 }

```

## matriz.c

```
atividade1.1 > C matriz.c > ...
1  √ #include <stdio.h>
2  √ #include <stdlib.h>
3  √ #include <time.h>
4  √ #include "matriz.h"
5
6  √ void zeraMatriz(Matriz* mat){
7      int i, j;
8      for(i=0; i<mat->lin; i++)
9          for(j=0; j<mat->col; j++)
10         mat->dados[i][j] = 0;
11 }
12
13 √ Matriz* criaMatriz(int l, int c){
14     Matriz* mat;
15     mat = (Matriz*) malloc (sizeof(Matriz));
16     if(mat != NULL){
17         if(l <= 0 || c <= 0 || l > MAX || c > MAX){
18             printf("Valores invalidos, matriz nao criada!\n");
19             return NULL;
20         }
21         mat->lin = l;
22         mat->col = c;
23         zeraMatriz(mat);
24     }
25     return mat;
26 }
27
28 √ void destroiMatriz(Matriz* mat){
29     if(mat != NULL)
30         free(mat);
31 }
32
33 √ int preencheAleatorio(Matriz* mat, int ini, int fim){
34     if(mat == NULL) return 0;
35     srand(time(NULL));
36     int i, j;
37     for(i=0; i<mat->lin; i++)
38         for(j=0; j<mat->col; j++)
39             mat->dados[i][j] = ini + rand() % (fim-ini + 1);
40     return 1;
41 }
42
43 √ int insereElem(Matriz* mat, int elem, int l, int c){
44     if(mat == NULL) return 0;
45     if(l < 0 || c < 0 || l >= mat->lin || c >= mat->col){
46         printf("Valores invalidos, elem nao inserido!\n");
47         return 0;
48     }
```

```

48     }
49     mat->dados[l][c] = elem;
50     return 1;
51 }
52
53 ∨ int consultaElem(Matriz* mat, int *p, int l, int c){
54     if(mat == NULL) return 0;
55 ∨     if(l < 0 || c < 0 || l >= mat->lin || c >= mat->col){
56         printf("Valores invalidos, elem nao existe!\n");
57         return 0;
58     }
59     *p = mat->dados[l][c];
60     return 1;
61 }
62
63 ∨ void imprime(Matriz* mat){
64     if(mat == NULL) return;
65     int i, j;
66     printf("Matriz %d x %d:\n", mat->lin, mat->col);
67 ∨     for(i=0; i<mat->lin; i++){
68         for(j=0; j<mat->col; j++){
69             printf("\t%d", mat->dados[i][j]);
70             printf("\n");
71         }
72         printf("\n");
73     }
74
75 ∨ int e_matrizQuadrada(Matriz *mat){
76     if(mat == NULL) return 0;
77     return (mat->lin == mat->col);
78 }
79
80 ∨ Matriz* criaTriangularSup(Matriz* mat){
81     if(mat == NULL) return NULL;
82 ∨     if(!e_matrizQuadrada(mat)){
83         printf("Matriz nao Quadrada!\n");
84         return NULL;
85     }
86     int i, j;
87     Matriz* ts = criaMatriz(mat->lin, mat->col);
88     for(i=0; i<mat->lin; i++){
89         for(j=0; j<mat->col; j++){
90             if(i <= j)
91                 ts->dados[i][j] = mat->dados[i][j];
92     }
    return ts;

```

```

93 }
94
95 Matriz* criaTriangularInf(Matriz* mat){
96     if(mat == NULL) return NULL;
97     if(!e_matrizQuadrada(mat)){
98         printf("Matriz nao Quadrada!\n");
99         return NULL;
100     }
101     int i, j;
102     Matriz* ti = criaMatriz(mat->lin, mat->col);
103     for(i=0; i<mat->lin; i++)
104         for(j=0; j<mat->col; j++)
105             if(i >= j)
106                 ti->dados[i][j] = mat->dados[i][j];
107     return ti;
108 }
109
110 Matriz* criaDiagonal(Matriz* mat){
111     if(mat == NULL) return NULL;
112     if(!e_matrizQuadrada(mat)){
113         printf("Matriz nao Quadrada!\n");
114         return NULL;
115     }
116     int i, j;
117     Matriz* d = criaMatriz(mat->lin, mat->col);
118     for(i=0; i<mat->lin; i++)
119         d->dados[i][i] = mat->dados[i][i];
120     return d;
121 }
122
123 int e_Simetrica(Matriz* mat){
124     if(mat == NULL) return 0;
125     if(!e_matrizQuadrada(mat)){
126         printf("Matriz nao Quadrada!\n");
127         return 0;
128     }
129     int i, j;
130     for(i=0; i<mat->lin; i++)
131         for(j=i+1; j<mat->col; j++)
132             if(mat->dados[i][j] != mat->dados[j][i])
133                 return 0;
134     return 1;
135 }
136
137 Matriz* criaTransposta(Matriz* mat){
138     if(mat == NULL) return NULL;
139     Matriz* t = criaMatriz(mat->col, mat->lin);
140     int i, j;
141     for(i=0; i<mat->lin; i++)
142         for(j=0; j<mat->col; j++)
143             t->dados[j][i] = mat->dados[i][j];
144     return t;
145 }
146

```

## matriz.h

```
atividade1.1 > C matriz.h > ⓘ imprime(Matriz *)
1  #ifndef MATRIZ_H
2  #define MATRIZ_H
3
4  #define MAX 100
5
6  typedef struct{
7      int dados[MAX][MAX];
8      int lin, col;
9  }Matriz;
10
11 void zeraMatriz(Matriz* mat);
12
13 Matriz* criaMatriz(int l, int c);
14
15 void destroiMatriz(Matriz* mat);
16
17 int preencheAleatorio(Matriz* mat, int ini, int fim);
18
19 int insereElem(Matriz* mat, int elem, int l, int c);
20
21 int consultaElem(Matriz* mat, int *p, int l, int c);
22
23 void imprime(Matriz* mat);
24
25 int e_matrizQuadrada(Matriz *mat);
26
27 Matriz* criaTriangularSup(Matriz* mat);
28
29 Matriz* criaTriangularInf(Matriz* mat);
30
31 Matriz* criaDiagonal(Matriz* mat);
32
33 int e_Simetrica(Matriz* mat);
34
35 Matriz* criaTransposta(Matriz* mat);
36
37 #endif
```

## Console

```
messiasfcm@MessiasFCM:/mnt/c/Users/Messi/OneDrive/Área de Trabalho/lista7/atividade1.1$ gcc main.c matriz.c matriz.h -o tp1
messiasfcm@MessiasFCM:/mnt/c/Users/Messi/OneDrive/Área de Trabalho/lista7/atividade1.1$ ./tp1
-<X Menu X>-
1 - Criar Matriz
2 - Zerar Matriz
3 - Preencher Aleatoriamente
4 - Inserir Elemento
5 - Consultar Elemento
6 - É Matriz quadrada?
7 - Criar Triangulo Sup
8 - Criar Triangulo Inf
9 - Criar Diagonal
10 - Criar Transposta
11 - É Simétrica?
12 - Imprimir Matriz
13 - Sair
Escolha: 1
-> Insira a quantidade de linha: 10
-> Insira a quantidade de coluna: 10
-> Criado com sucesso!
Escolha: 3
-> Insira um valor inicial: 10
-> Insira um valor final: 50
-> Matriz preenchida aleatoriamente!
Escolha: 6
-> Matriz Quadrada!
Escolha: 7
Matriz 10 x 10:
 34    18    36    47    36    37    46    45    47    28
 0    46    47    17    45    19    21    12    39    43
 0     0    35    26    10    42    21    31    37    10
 0     0     0    44    20    48    30    15    42    29
 0     0     0     0    42    21    43    12    23    33
 0     0     0     0     0    31    26    10    13    14
 0     0     0     0     0     0    21    39    44    12
 0     0     0     0     0     0     0    50    34    19
 0     0     0     0     0     0     0     0    23    40
 0     0     0     0     0     0     0     0     0    38

Escolha: 8
Matriz 10 x 10:
 34     0     0     0     0     0     0     0     0     0
 41    46     0     0     0     0     0     0     0     0
 49    47    35     0     0     0     0     0     0     0
 18    22    20    44     0     0     0     0     0     0
 36    34    24    32    42     0     0     0     0     0
```

...

```
 38    27    12    15    41    19    43    11    48    38

Escolha: 9
Matriz 10 x 10:
 34     0     0     0     0     0     0     0     0     0
 0    46     0     0     0     0     0     0     0     0
 0     0    35     0     0     0     0     0     0     0
 0     0     0    44     0     0     0     0     0     0
 0     0     0     0    42     0     0     0     0     0
 0     0     0     0     0    31     0     0     0     0
 0     0     0     0     0     0    21     0     0     0
 0     0     0     0     0     0     0    50     0     0
 0     0     0     0     0     0     0     0    23     0
 0     0     0     0     0     0     0     0     0    38

Escolha: 10
Matriz 10 x 10:
 34    41    49    18    36    47    12    17    32    38
 18    46    47    22    34    21    21    31    32    27
 36    47    35    20    24    31    26    39    32    12
 47    17    26    44    32    33    22    33    13    15
 36    45    10    20    42    40    16    14    16    41
 37    19    42    48    21    31    38    30    21    19
 46    21    21    30    43    26    21    46    36    43
 45    12    31    15    12    10    39    50    34    11
 47    39    37    42    23    13    44    34    23    48
 28    43    10    29    33    14    12    19    40    38

Escolha: 11
-> Matriz não Simétrica!
Escolha: 12
Matriz 10 x 10:
 34    18    36    47    36    37    46    45    47    28
 41    46    47    17    45    19    21    12    39    43
 49    47    35    26    10    42    21    31    37    10
 18    22    20    44    20    48    30    15    42    29
 36    34    24    32    42    21    43    12    23    33
 47    21    31    33    40    31    26    10    13    14
 12    21    26    22    16    38    21    39    44    12
 17    31    39    33    14    30    46    50    34    19
 32    32    32    13    16    21    36    34    23    40
 38    27    12    15    41    19    43    11    48    38

Escolha: 13
-> Finalizando!
messiasfcm@MessiasFCM:/mnt/c/Users/Messi/OneDrive/Área de Trabalho/lista7/atividade1.1$
```

## 1.2

### main.c

```
atividade1.2 > C main.c > main()
1  #include <stdio.h>
2  #include <stdbool.h>
3  #include "matrizDin.h"
4
5  int main()
6  {
7      Matriz *M, *TS, *TI, *D, *T;
8      int elemento, linha, coluna, escolha, inicio, fim;
9
10     printf("-<X Menu X>-\\n");
11     printf("1 - Criar Matriz\\n");
12     printf("2 - Zerar Matriz\\n");
13     printf("3 - Preencher Aleatoriamente\\n");
14     printf("4 - Inserir Elemento\\n");
15     printf("5 - Consultar Elemento\\n");
16     printf("6 - É Matriz quadrada?\\n");
17     printf("7 - Criar Triangulo Sup\\n");
18     printf("8 - Criar Triangulo Inf\\n");
19     printf("9 - Criar Diagonal\\n");
20     printf("10 - Criar Transposta\\n");
21     printf("11 - É Simetrica?\\n");
22     printf("12 - Imprimir Matriz\\n");
23     printf("13 - Sair\\n");
24
25     while(true){
26         printf("Escolha: ");
27         scanf("%d", &escolha);
28
29         if(escolha == 1){
30             printf("-> Insira a quantidade de linha: ");
31             scanf("%d", &linha);
32             printf("-> Insira a quantidade de coluna: ");
33             scanf("%d", &coluna);
34             M = criaMatriz(linha, coluna);
35             printf("-> Criado com sucesso!\\n");
36         }else if(escolha == 2){
37             zeraMatriz(M);
38             printf("-> Matriz Zerada com sucesso!\\n");
39         }else if(escolha == 3){
40             printf("-> Insira um valor inicial: ");
41             scanf("%d", &inicio);
42             printf("-> Insira um valor final: ");
43             scanf("%d", &fim);
44             preencheAleatorio(M, inicio, fim);
45             printf("-> Matriz preenchida aleatoriamente!\\n");
46         }else if(escolha == 4){
47             printf("-> Elemento a ser inserido: ");
48             scanf("%d", &elemento);
```



```

48     printf("-> Insira a linha: ");
49     scanf("%d",&linha);
50     printf("-> Insira a coluna: ");
51     scanf("%d",&coluna);
52     insereElem(M, elemento, linha, coluna);
53     printf("-> Inserido com sucesso!\n");
54 }else if(escolha == 5){
55     printf("-> Insira a linha: ");
56     scanf("%d",&linha);
57     printf("-> Insira a coluna: ");
58     scanf("%d",&coluna);
59     consultaElem(M, &elemento, linha, coluna);
60     printf("-> Elemento consultado: %d\n", elemento);
61 }else if(escolha == 6){
62     if(e_matrizQuadrada(M)){
63         printf("-> Matriz Quadrada!\n");
64     }else{
65         printf("-> Matriz não é Quadrada!\n");
66     }
67 }else if(escolha == 7){
68     TS = criaTriangularSup(M);
69     imprime(TS);
70 }else if(escolha == 8){
71     TI = criaTriangularInf(M);
72     imprime(TI);
73 }else if(escolha == 9){
74     D = criaDiagonal(M);
75     imprime(D);
76 }else if(escolha == 10){
77     T = criaTransposta(M);
78     imprime(T);
79 }else if(escolha == 11){
80     if(e_Simetrica(M)){
81         printf("-> Matriz Simétrica!\n");
82     }else{
83         printf("-> Matriz não Simétrica!\n");
84     }
85 }else if(escolha == 12){
86     imprime(M);
87 }else if(escolha == 13){
88     destroiMatriz(M);
89     printf("-> Finalizando!\n");
90     break;
91 }else{
92     printf("-> Número inválido!\n");
93 }
94 }
95 return 0;
96 }

```

## matrizDin.c

```
atividade1.2 > C matrizDin.c > ...
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <time.h>
4  #include "matrizDin.h"
5
6  void zeraMatriz(Matriz* mat){
7      int i, j;
8      for(i=0; i<mat->lin; i++)
9          for(j=0; j<mat->col; j++)
10             mat->dados[i][j] = 0;
11 }
12
13 Matriz* criaMatriz(int l, int c){
14     Matriz* mat;
15     mat = (Matriz*) malloc (sizeof(Matriz));
16     if(mat != NULL){
17         if(l <= 0 || c <= 0){
18             printf("Valores invalidos, matriz nao criada!\n");
19             return NULL;
20         }
21         int i;
22         mat->lin = l;
23         mat->col = c;
24         mat->dados = (int**) malloc (l*sizeof(int*));
25         for(i=0; i<l; i++)
26             mat->dados[i] = (int*) malloc (c*sizeof(int));
27         zeraMatriz(mat);
28     }
29     return mat;
30 }
31
32 void destroiMatriz(Matriz* mat){
33     if(mat != NULL){
34         int i;
35         for(i=0; i<mat->lin; i++)
36             free(mat->dados[i]);
37         free(mat->dados);
38         free(mat);
39     }
40 }
41
42 int preencheAleatorio(Matriz* mat, int ini, int fim){
43     if(mat == NULL) return 0;
44     srand(time(NULL));
45     int i, j;
46     for(i=0; i<mat->lin; i++)
47         for(j=0; j<mat->col; j++)
48             mat->dados[i][j] = ini + rand() % (fim-ini + 1);
49     return 1;
50 }
51
52 int insereElem(Matriz* mat, int elem, int l, int c){
53     if(mat == NULL) return 0;
54     if(l < 0 || c < 0 || l > mat->lin || c > mat->col){
55         printf("Valores invalidos, elem nao inserido!\n");
56         return 0;
57     }
58     mat->dados[l][c] = elem;
59     return 1;
60 }
61
62 int consultaElem(Matriz* mat, int *p, int l, int c){
63     if(mat == NULL) return 0;
64     if(l < 0 || c < 0 || l > mat->lin || c > mat->col){
65         printf("Valores invalidos, elem nao existe!\n");
66         return 0;
67     }
68     *p = mat->dados[l][c];
69     return 1;
70 }
71
72 void imprime(Matriz* mat){
73     if(mat == NULL) return;
74     int i, j;
75     printf("Matriz %d x %d:\n", mat->lin, mat->col);
76     for(i=0; i<mat->lin; i++){
77         for(j=0; j<mat->col; j++)
78             printf("%d ", mat->dados[i][j]);
79         printf("\n");
80     }
81     printf("\n");
82 }
83
84 int e_matrizQuadrada(Matriz *mat){
85     if(mat == NULL) return 0;
86     return (mat->lin == mat->col);
87 }
88
89 Matriz* criaTriangularSup(Matriz* mat){
90     if(mat == NULL) return NULL;
91     if(!e_matrizQuadrada(mat)){
92         printf("Matriz nao Quadrada!\n");
```

```

93     return NULL;
94 }
95 int i, j;
96 Matriz* ts = criaMatriz(mat->lin, mat->col);
97 for(i=0; i<mat->lin; i++)
98     for(j=0; j<mat->col; j++)
99         if(i <= j)
100             ts->dados[i][j] = mat->dados[i][j];
101 return ts;
102 }
103
104 Matriz* criaTriangularInf(Matriz* mat){
105     if(mat == NULL) return NULL;
106     if(!e_matrizQuadrada(mat)){
107         printf("Matriz nao Quadrada!\n");
108         return NULL;
109     }
110     int i, j;
111     Matriz* ti = criaMatriz(mat->lin, mat->col);
112     for(i=0; i<mat->lin; i++)
113         for(j=0; j<mat->col; j++)
114             if(i >= j)
115                 ti->dados[i][j] = mat->dados[i][j];
116     return ti;
117 }
118
119 Matriz* criaDiagonal(Matriz* mat){
120     if(mat == NULL) return NULL;
121     if(!e_matrizQuadrada(mat)){
122         printf("Matriz nao Quadrada!\n");
123         return NULL;
124     }
125     int i, j;
126     Matriz* d = criaMatriz(mat->lin, mat->col);
127     for(i=0; i<mat->lin; i++)
128         d->dados[i][i] = mat->dados[i][i];
129     return d;
130 }
131
132 int e_Simetrica(Matriz* mat){
133     if(mat == NULL) return 0;
134     if(!e_matrizQuadrada(mat)){
135         printf("Matriz nao Quadrada!\n");
136         return 0;
137     }
138     int i, j;
139     for(i=0; i<mat->lin; i++)
140         for(j=i+1; j<mat->col; j++)
141             if(mat->dados[i][j] != mat->dados[j][i])
142                 return 0;
143     return 1;
144 }
145
146 Matriz* criaTransposta(Matriz* mat){
147     if(mat == NULL) return NULL;
148     Matriz* t = criaMatriz(mat->col, mat->lin);
149     int i, j;
150     for(i=0; i<mat->lin; i++)
151         for(j=0; j<mat->col; j++)
152             t->dados[j][i] = mat->dados[i][j];
153     return t;
154 }

```

## matrizDin.h

```
atividade1.2 > C matrizDin.h > e_matrizQuadrada(Matriz *)
1  #ifndef MATRIZDIN_H
2  #define MATRIZDIN_H
3
4  typedef struct{
5      int **dados;
6      int lin, col;
7  }Matriz;
8
9  void zeraMatriz(Matriz* mat);
10
11 Matriz* criaMatriz(int l, int c);
12
13 void destroiMatriz(Matriz* mat);
14
15 int preencheAleatorio(Matriz* mat, int ini, int fim);
16
17 int insereElem(Matriz* mat, int elem, int l, int c);
18
19 int consultaElem(Matriz* mat, int *p, int l, int c);
20
21 void imprime(Matriz* mat);
22
23 int e_matrizQuadrada(Matriz *mat);
24
25 Matriz* criaTriangularSup(Matriz* mat);
26
27 Matriz* criaTriangularInf(Matriz* mat);
28
29 Matriz* criaDiagonal(Matriz* mat);
30
31 int e_Simetrica(Matriz* mat);
32
33 Matriz* criaTransposta(Matriz* mat);
34
35 #endif
```

## Console

```
messiasfcm@MessiasFCM:/mnt/c/Users/Messi/OneDrive/Área de Trabalho/lista7/atividade1.2$ gcc main.c matrizDin.c matrizDin.h -o tp2
messiasfcm@MessiasFCM:/mnt/c/Users/Messi/OneDrive/Área de Trabalho/lista7/atividade1.2$ ./tp2
-cX Menu X-
1 - Criar Matriz
2 - Zerar Matriz
3 - Preencher Aleatoriamente
4 - Inserir Elemento
5 - Consultar Elemento
6 - É Matriz quadrada?
7 - Criar Triangulo Sup
8 - Criar Triangulo Inf
9 - Criar Diagonal
10 - Criar Transposta
11 - É Simétrica?
12 - Imprimir Matriz
13 - Sair
Escolha: 1
-> Insira a quantidade de linha: 5
-> Insira a quantidade de coluna: 5
-> Criado com sucesso!
Escolha: 2
-> Matriz Zerada com sucesso!
Escolha: 12
Matriz 5 x 5:
0 0 0 0 0
0 0 0 0 0
0 0 0 0 0
0 0 0 0 0
0 0 0 0 0

Escolha: 3
-> Insira um valor inicial: 10
-> Insira um valor final: 99
-> Matriz preenchida aleatoriamente!
Escolha: 7
Matriz 5 x 5:
94 49 43 97 21
0 32 86 97 41
0 0 55 43 29
0 0 0 26 75
0 0 0 0 21

Escolha: 8
Matriz 5 x 5:
94 0 0 0 0
34 32 0 0 0
36 24 55 0 0
31 22 81 26 0
32 11 79 55 21

Escolha: 9
Matriz 5 x 5:
94 0 0 0 0
0 32 0 0 0
0 0 55 0 0
0 0 0 26 0
0 0 0 0 21

Escolha: 10
Matriz 5 x 5:
94 34 36 31 32
49 32 24 22 11
43 86 55 81 79
97 97 43 26 55
21 41 29 75 21

Escolha: 11
-> Matriz não Simétrica!
Escolha: 13
-> Finalizando!
messiasfcm@MessiasFCM:/mnt/c/Users/Messi/OneDrive/Área de Trabalho/lista7/atividade1.2$
```

## Problema 2

### 2.1

main.c

```
atividade21 > C main.c > main()
1  #include <stdio.h>
2  #include <stdbool.h>
3  #include "mFaixa.h"
4
5  int main(){
6      Mfaixa *M;
7      int elemento, linha, coluna, valor, escolha, inicio, fim;
8
9      printf("-> Menu X->\n");
10     printf("1 - Criar Matriz\n");
11     printf("2 - Zerar Matriz\n");
12     printf("3 - Preencher Aleatoriamente\n");
13     printf("4 - Inserir Elemento\n");
14     printf("5 - Consultar Elemento\n");
15     printf("6 - Imprimir Matriz de Faixa(Vetores)\n");
16     printf("7 - Imprimir Matriz de Faixa\n");
17     printf("8 - Sair\n");
18
19     while(true){
20         printf("Escolha: ");
21         scanf("%d", &escolha);
22
23         if(escolha == 1){
24             printf("-> Insira o valor para a matriz: ");
25             scanf("%d",&valor);
26             M = criaMatriz(valor);
27             printf("-> Criado com sucesso!\n");
28         }else if(escolha == 2){
29             zeraMatriz(M);
30             printf("-> Matriz Zerada com sucesso!\n");
31         }else if(escolha == 3){
32             printf("-> Insira um valor inicial: ");
33             scanf("%d",&inicio);
34             printf("-> Insira um valor final: ");
35             scanf("%d",&fim);
36             preencheAleatorio(M, inicio, fim);
37             printf("-> Matriz preenchida aleatoriamente!\n");
38         }else if(escolha == 4){
39             printf("-> Elemento a ser inserido: ");
40             scanf("%d",&elemento);
41             printf("-> Insira a linha: ");
42             scanf("%d",&linha);
43             printf("-> Insira a coluna: ");
44             scanf("%d",&coluna);
45             insereElem(M, elemento, linha, coluna);
46             printf("-> Inserido com sucesso!\n");
47         }else if(escolha == 5){
48             printf("-> Insira a linha: ");
49             scanf("%d",&linha);
50             printf("-> Insira a coluna: ");
51             scanf("%d",&coluna);
52             printf("-> Elemento consultado: %d\n", consultaElem(M, linha, coluna));
53         }else if(escolha == 6){
54             imprimeFaixaVetores(M);
55         }else if(escolha == 7){
56             imprimeFaixa(M);
57         }else if(escolha == 8){
58             destroiMatriz(M);
59             printf("-> Finalizando!\n");
60             break;
61         }else{
62             printf("-> Número inválido!\n");
63         }
64     }
65     return 0;
66 }
```

## mFaixa.c

```
atividade21 > C mFaixa.c > ...
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <time.h>
4  #include "mFaixa.h"
5
6  void zeraMatriz(MFaixa* mf){
7      int i;
8      for(i=0; i<mf->tam; i++){
9          mf->diagonal[i] = 0;
10         if(i < mf->tam -1){
11             mf->superior[i] = 0;
12             mf->inferior[i] = 0;
13         }
14     }
15 }
16
17 MFaixa* criaMatriz(int t){
18     MFaixa *mf;
19     mf = (MFaixa*) malloc (sizeof(MFaixa));
20     if(mf != NULL){
21         if(t <= 1){
22             printf("Dimensao deve ser > 1, matriz nao criada!");
23             return NULL;
24         }
25         mf->tam = t;
26         mf->diagonal = (int*) malloc (t*sizeof(int));
27         mf->superior = (int*) malloc ((t-1)*sizeof(int));
28         mf->inferior = (int*) malloc ((t-1)*sizeof(int));
29         if(mf->diagonal == NULL || mf->superior == NULL || mf->inferior == NULL)
30             return NULL;
31         zeraMatriz(mf);
32     }
33     return mf;
34 }
35
36 void destroiMatriz(MFaixa* mf){
37     if(mf != NULL){
38         free(mf->diagonal);
39         free(mf->superior);
40         free(mf->inferior);
41         free(mf);
42     }
43 }
44
45 int preencheAleatorio(MFaixa* mf, int ini, int fim){
46     if(mf == NULL) return 0;
47     srand(time(NULL));
48 }
```

```

48     int i;
49     for(i=0; i<mf->tam; i++){
50         mf->diagonal[i] = ini + rand() % (fim-ini + 1);
51         if(i < mf->tam -1){
52             mf->superior[i] = ini + rand() % (fim-ini + 1);
53             mf->inferior[i] = ini + rand() % (fim-ini + 1);
54         }
55     }
56     return 1;
57 }
58
59 int insereElem(MFaixa* mf, int elem, int i, int j){
60     if(mf == NULL) return 0;
61     if(i < 0 || j < 0 || i >= mf->tam || j >= mf->tam){
62         printf("Valores invalidos, elem nao inserido!\n");
63         return 0;
64     }
65     if(i == j) mf->diagonal[i] = elem;
66     else if(i + 1 == j) mf->superior[i] = elem;
67     else if(i == j + 1) mf->inferior[j] = elem;
68     else{
69         printf("Indices fora da faixa, elem nao inserido!\n");
70         return 0;
71     }
72     return 1;
73 }
74
75 int consultaElem(MFaixa* mf, int i, int j){
76     if(mf == NULL) return 0;
77     if(i < 0 || j < 0 || i >= mf->tam || j >= mf->tam){
78         printf("Valores invalidos, elem inexistente!\n");
79         return 0;
80     }
81     if(i == j) return mf->diagonal[i];
82     else if(i + 1 == j) return mf->superior[i];
83     else if(i == j + 1) return mf->inferior[j];
84     else return 0;
85 }
86
87 void imprimeFaixaVetores(MFaixa* mf){
88     if(mf == NULL) return;
89     int i;
90     printf("Matriz Faixa, Tam: %d x %d:\n", mf->tam, mf->tam);
91     printf("Diagonal = [");
92     for(i=0; i<mf->tam; i++)

```

```

93         printf("%d ", mf->diagonal[i]);
94     printf("]\n");
95     printf("Superior = [");
96     for(i=0; i<mf->tam-1; i++){
97         printf("%d ", mf->superior[i]);
98     }
99     printf("]\n");
100     printf("Inferior = [");
101     for(i=0; i<mf->tam-1; i++){
102         printf("%d ", mf->inferior[i]);
103     }
104     printf("]\n");
105 }
106
107 void imprimeFaixa(MFaixa* mf){
108     if(mf == NULL) return;
109     int i, j;
110     imprimeFaixaVetores(mf);
111     printf("Matriz Original:\n");
112     for(i=0; i<mf->tam; i++){
113         for(j=0; j<mf->tam; j++){
114             printf("%d\t", consultaElem(mf, i, j));
115         }
116         printf("\n");
117     }

```



## mFaixa.h

```
atividade2.1 > C mFaixa.h > ...
1  ▾ #ifndef MFAIXA_H
2  #define MFAIXA_H
3
4  ▾ typedef struct{
5      int *diagonal;
6      int *superior;
7      int *inferior;
8      int tam;
9  }MFaixa;
10
11 void zeraMatriz(MFaixa* mf);
12
13 MFaixa* criaMatriz(int t);
14
15 void destroiMatriz(MFaixa* mf);
16
17 int preencheAleatorio(MFaixa* mf, int ini, int fim);
18
19 int insereElem(MFaixa* mf, int elem, int i, int j);
20
21 int consultaElem(MFaixa* mf, int i, int j);
22
23 void imprimeFaixaVetores(MFaixa* mf);
24
25 void imprimeFaixa(MFaixa* mf);
26
27 #endif
```

## Console

```
messiasfcn@MessiasFCM:/mnt/c/Users/Messi/OneDrive/Área de Trabalho/lista7/atividade2.1$ gcc main.c mFaixa.c mFaixa.h -o tp3
messiasfcn@MessiasFCM:/mnt/c/Users/Messi/OneDrive/Área de Trabalho/lista7/atividade2.1$ ./tp3
-<X Menu X>-
1 - Criar Matriz
2 - Zerar Matriz
3 - Preencher Aleatoriamente
4 - Inserir Elemento
5 - Consultar Elemento
6 - Imprimir Matriz de Faixa(Vetores)
7 - Imprimir Matriz de Faixa
8 - Sair
Escolha: 1
-> Insira o valor para a matriz: 10
-> Criado com sucesso!
Escolha: 3
-> Insira um valor inicial: 20
-> Insira um valor final: 100
-> Matriz preenchida aleatoriamente!
Escolha: 6
Matriz Faixa, Tam: 10 x 10:
Diagonal = [63 95 23 37 75 25 85 97 43 61 ]
Superior = [49 95 91 29 53 36 70 47 50 ]
Inferior = [31 51 37 44 93 26 71 88 78 ]

Escolha: 7
Matriz Faixa, Tam: 10 x 10:
Diagonal = [63 95 23 37 75 25 85 97 43 61 ]
Superior = [49 95 91 29 53 36 70 47 50 ]
Inferior = [31 51 37 44 93 26 71 88 78 ]

Matriz Original:
63   49   0   0   0   0   0   0   0   0
31   95   95   0   0   0   0   0   0   0
0    51   23   91   0   0   0   0   0   0
0    0    37   37   29   0   0   0   0   0
0    0    0    44   75   53   0   0   0   0
0    0    0    0    93   25   36   0   0   0
0    0    0    0    0    26   85   70   0   0
0    0    0    0    0    0    71   97   47   0
0    0    0    0    0    0    0    88   43   50
0    0    0    0    0    0    0    0    78   61

Escolha: 8
-> Finalizando!
messiasfcn@MessiasFCM:/mnt/c/Users/Messi/OneDrive/Área de Trabalho/lista7/atividade2.1$
```

## 2.2

### main.c

```
atividade22 > C main.c > ...
1  #include <stdio.h>
2  #include <stdbool.h>
3  #include "matrizDin.h"
4  #include "mEsparsaCSR.h"
5
6  int main(){
7      Matriz *M, *TS, *TI, *D, *T;
8      MEsparsaCSR *MS;
9      int elemento, linha, coluna, escolha, inicio, fim, qnn;
10
11     printf("-<X Menu Matriz Din X>-\\n");
12     printf("1 - Criar Matriz Din\\n");
13     printf("2 - Zerar Matriz Din\\n");
14     printf("3 - Preencher Aleatoriamente\\n");
15     printf("4 - Inserir Elemento\\n");
16     printf("5 - Consultar Elemento\\n");
17     printf("6 - É Matriz quadrada?\\n");
18     printf("7 - Criar Triangulo Sup\\n");
19     printf("8 - Criar Triangulo Inf\\n");
20     printf("9 - Criar Diagonal\\n");
21     printf("10 - Criar Transposta\\n");
22     printf("11 - É Simetrica?\\n");
23     printf("12 - Imprimir Matriz Din\\n\\n");
24     // -----
25     printf("-<X Menu Matriz Esparsa X>-\\n");
26     printf("13 - Criar Matriz Esparsa\\n");
27     printf("14 - Inserir Elemento\\n");
28     printf("15 - Remover Elemento\\n");
29     printf("16 - Transformar em CSR\\n");
30     printf("17 - Consultar Elemento\\n");
31     printf("18 - Imprimir Matriz Esparsa(Vetores)\\n");
32     printf("19 - Imprimir Matriz Esparsa\\n");
33     printf("20 - Finalizar\\n");
34
35     while(true){
36         printf("Escolha: ");
37         scanf("%d", &escolha);
38
39         if(escolha == 1){
40             printf("-> Insira a quantidade de linha: ");
41             scanf("%d",&linha);
42             printf("-> Insira a quantidade de coluna: ");
43             scanf("%d",&coluna);
44             M = criaMatriz(linha, coluna);
45             printf("-> Criado com sucesso!\\n");
46         }else if(escolha == 2){
47             zeraMatriz(M);
```

```

48     printf("-> Matriz Zerada com sucesso!\n");
49 }else if(escolha == 3){
50     printf("-> Insira um valor inicial: ");
51     scanf("%d",&inicio);
52     printf("-> Insira um valor final: ");
53     scanf("%d",&fim);
54     preencheAleatorio(M, inicio, fim);
55     printf("-> Matriz preenchida aleatoriamente!\n");
56 }else if(escolha == 4){
57     printf("-> Elemento a ser inserido: ");
58     scanf("%d",&elemento);
59     printf("-> Insira a linha: ");
60     scanf("%d",&linha);
61     printf("-> Insira a coluna: ");
62     scanf("%d",&coluna);
63     insereElem(M, elemento, linha, coluna);
64     printf("-> Inserido com sucesso!\n");
65 }else if(escolha == 5){
66     printf("-> Insira a linha: ");
67     scanf("%d",&linha);
68     printf("-> Insira a coluna: ");
69     scanf("%d",&coluna);
70     consultaElem(M, &elemento, linha, coluna);
71     printf("-> Elemento consultado: %d\n", elemento);
72 }else if(escolha == 6){
73     if(e_matrizQuadrada(M)){
74         printf("-> Matriz Quadrada!\n");
75     }else{
76         printf("-> Matriz não é Quadrada!\n");
77     }
78 }else if(escolha == 7){
79     TS = criaTriangularSup(M);
80     imprime(TS);
81 }else if(escolha == 8){
82     TI = criaTriangularInf(M);
83     imprime(TI);
84 }else if(escolha == 9){
85     D = criaDiagonal(M);
86     imprime(D);
87 }else if(escolha == 10){
88     T = criaTransposta(M);
89     imprime(T);
90 }else if(escolha == 11){
91     if(e_Simetrica(M)){
92         printf("-> Matriz Simétrica!\n");

```

```

93     }else{
94         printf("-> Matriz não Simétrica!\n");
95     }
96 }else if(escolha == 12){
97     imprime(M);
98     // -----
99 }else if(escolha == 13){
100     printf("-> Insira a quantidade de linha: ");
101     scanf("%d",&linha);
102     printf("-> Insira a quantidade de coluna: ");
103     scanf("%d",&coluna);
104     printf("-> Insira a quantidade de não nulos: ");
105     scanf("%d",&qnn);
106     MS = criaMatrizEsparsa(linha, coluna, qnn);
107 }else if(escolha == 14){
108     printf("-> Insira o elemento: ");
109     scanf("%d",&elemento);
110     printf("-> Insira a linha: ");
111     scanf("%d",&linha);
112     printf("-> Insira a coluna: ");
113     scanf("%d",&coluna);
114     insereElemEsparsa(MS, elemento, linha, coluna);
115 }else if(escolha == 15){
116     printf("-> Insira a linha: ");
117     scanf("%d",&linha);
118     printf("-> Insira a coluna: ");
119     scanf("%d",&coluna);
120     removeElemEsparsa(MS, linha, coluna);
121 }else if(escolha == 16){
122     MS = transformarEmCSR(M);
123     printf("-> Matriz transformada com sucesso");
124 }else if(escolha == 17){
125     printf("-> Insira a linha: ");
126     scanf("%d",&linha);
127     printf("-> Insira a coluna: ");
128     scanf("%d",&coluna);
129     printf("-> Elemento consultado: %d\n", consultaElemEsparsa(MS, linha, coluna));
130 }else if(escolha == 18){
131     imprimeEsparsaVetores(MS);
132 }else if(escolha == 19){
133     imprimeEsparsa(MS);
134 }else if(escolha == 20){
135     destroiMatriz(M);
136     destroiMatrizEsparsa(MS);
137     printf("-> Finalizando!\n");

```

```

138         break;
139     }else{
140         printf("-> Número inválido!\n");
141     }
142 }
143 return 0;
144 }

```

## matrizDin.h

```

atividade22 > C matrizDin.h > ...
1  /*----- File: MatrizDin.h -----+
2  |Matriz Dinamica sequencial (malloc) |
3  |                                   |
4  |                                   |
5  | Implementado por Guilherme C. Pena em 29/09/2023 |
6  +-----+ */
7
8  #ifndef MATRIZDIN_H
9  #define MATRIZDIN_H
10
11  #include <stdio.h>
12  #include <stdlib.h>
13  #include <time.h>
14
15  typedef struct{
16      int **dados;
17      int lin, col;
18  }Matriz;
19
20  void zeraMatriz(Matriz* mat){
21      int i, j;
22      for(i=0; i<mat->lin; i++)
23          for(j=0; j<mat->col; j++)
24              mat->dados[i][j] = 0;
25  }
26
27  Matriz* criaMatriz(int l, int c){
28      Matriz* mat;
29      mat = (Matriz*) malloc (sizeof(Matriz));
30      if(mat != NULL){
31          if(l <= 0 || c <= 0){
32              printf("Valores invalidos, matriz nao criada!\n");
33              return NULL;
34          }
35          int i;
36          mat->lin = l;
37          mat->col = c;
38          mat->dados = (int**) malloc (l*sizeof(int*));
39          for(i=0; i<l; i++)
40              mat->dados[i] = (int*) malloc (c*sizeof(int));
41          zeraMatriz(mat);
42      }
43      return mat;
44  }
45
46  void destroiMatriz(Matriz* mat){
47      if(mat != NULL){

```

```

48         int i;
49         for(i=0; i<mat->lin; i++)
50             free(mat->dados[i]);
51         free(mat->dados);
52         free(mat);
53     }
54 }
55
56 int preencheAleatorio(Matriz* mat, int ini, int fim){
57     if(mat == NULL) return 0;
58     srand(time(NULL));
59     int i, j;
60     for(i=0; i<mat->lin; i++)
61         for(j=0; j<mat->col; j++)
62             mat->dados[i][j] = ini + rand() % (fim-ini + 1);
63     return 1;
64 }
65
66 int insereElem(Matriz* mat, int elem, int l, int c){
67     if(mat == NULL) return 0;
68     if(l < 0 || c < 0 || l >= mat->lin || c >= mat->col){
69         printf("Valores invalidos, elem nao inserido!\n");
70         return 0;
71     }
72     mat->dados[l][c] = elem;
73     return 1;
74 }
75
76 int consultaElem(Matriz* mat, int *p, int l, int c){
77     if(mat == NULL) return 0;
78     if(l < 0 || c < 0 || l >= mat->lin || c >= mat->col){
79         printf("Valores invalidos, elem nao existe!\n");
80         return 0;
81     }
82     *p = mat->dados[l][c];
83     return 1;
84 }
85
86 void imprime(Matriz* mat){
87     if(mat == NULL) return;
88     int i, j;
89     printf("Matriz %d x %d:\n", mat->lin, mat->col);
90     for(i=0; i<mat->lin; i++){
91         for(j=0; j<mat->col; j++)
92             printf("%d ", mat->dados[i][j]);

```

```

93     printf("\n");
94 }
95 printf("\n");
96 }
97
98 //Matrizes Quadradas e propriedades
99
100 int e_matrizQuadrada(Matriz *mat){
101     if(mat == NULL) return 0;
102     return (mat->lin == mat->col);
103 }
104
105 Matriz* criaTriangularSup(Matriz* mat){
106     if(mat == NULL) return NULL;
107     if(!e_matrizQuadrada(mat)){
108         printf("Matriz nao Quadrada!\n");
109         return NULL;
110     }
111     int i, j;
112     Matriz* ts = criaMatriz(mat->lin, mat->col);
113     for(i=0; i<mat->lin; i++)
114         for(j=0; j<mat->col; j++)
115             if(i <= j)
116                 ts->dados[i][j] = mat->dados[i][j];
117     return ts;
118 }
119
120 Matriz* criaTriangularInf(Matriz* mat){
121     if(mat == NULL) return NULL;
122     if(!e_matrizQuadrada(mat)){
123         printf("Matriz nao Quadrada!\n");
124         return NULL;
125     }
126     int i, j;
127     Matriz* ti = criaMatriz(mat->lin, mat->col);
128     for(i=0; i<mat->lin; i++)
129         for(j=0; j<mat->col; j++)
130             if(i >= j)
131                 ti->dados[i][j] = mat->dados[i][j];
132     return ti;
133 }
134
135 Matriz* criaDiagonal(Matriz* mat){
136     if(mat == NULL) return NULL;
137     if(!e_matrizQuadrada(mat)){
138         printf("Matriz nao Quadrada!\n");
139         return NULL;
140     }
141     int i, j;
142     Matriz* d = criaMatriz(mat->lin, mat->col);
143     for(i=0; i<mat->lin; i++)
144         d->dados[i][i] = mat->dados[i][i];
145     return d;
146 }
147
148 int e_Simetrica(Matriz* mat){
149     if(mat == NULL) return 0;
150     if(!e_matrizQuadrada(mat)){
151         printf("Matriz nao Quadrada!\n");
152         return 0;
153     }
154     int i, j;
155     for(i=0; i<mat->lin; i++)
156         for(j=i+1; j<mat->col; j++)
157             if(mat->dados[i][j] != mat->dados[j][i])
158                 return 0;
159     return 1;
160 }
161
162 Matriz* criaTransposta(Matriz* mat){
163     if(mat == NULL) return NULL;
164     Matriz* t = criaMatriz(mat->col, mat->lin);
165     int i, j;
166     for(i=0; i<mat->lin; i++)
167         for(j=0; j<mat->col; j++)
168             t->dados[j][i] = mat->dados[i][j];
169     return t;
170 }
171
172 #endif

```

## mEsparsaCSR.h

```
atividade22 > C mEsparsaCSR.h > ...
1  /*----- File: MEsparsaCSR.h -----+
2  |Matriz Esparsa (malloc)                |
3  |Compressed Sparse Row - CSR            |
4  |                                     |
5  | Implementado por Guilherme C. Pena em 09/10/2023 |
6  +-----+ */
7
8  #ifndef MESPARSACSR_H
9  #define MESPARSACSR_H
10
11 #include <stdio.h>
12 #include <stdlib.h>
13 #include <time.h>
14 #include "MatrizDin.h"
15
16 /*
17  Matriz Esparsa (Compressed Sparse Row - CSR)
18  3 vetores:
19  A -> tamanho QNN (Quantidade de nao nulos)
20  IA -> tamanho N+1 (QNN por linha)
21  |   IA[0] = 0;
22  |   IA[i] = IA[i-1] + QNN da linha (i-1)
23  JA -> tamanho QNN (indices de coluna de cada elemento em A)
24
25  Input : 10  20  0  0  0  0
26  |       | 0  30  0  4  0  0
27  |       | 0  0  50  60  70  0
28  |       | 0  0  0  0  0  80
29
30  Output : A = [10 20 30 4 50 60 70 80],
31  |       | IA = [0 2 4 7 8]
32  |       | JA = [0 1 1 3 2 3 4 5]
33  */
34
35 typedef struct{
36     int *A; //Valores
37     int *IA;
38     int *JA;
39     int lin, col, QNN, QI;
40 }MEsparsaCSR;
41 //QNN - Quantidade de Nao Nulos
42 //QI - Quantidade de Inseridos
43
44 MEsparsaCSR* criaMatrizEsparsa(int l, int c, int qnn){
45     MEsparsaCSR *ms;
46     ms = (MEsparsaCSR*) malloc (sizeof(MEsparsaCSR));
47     if(ms != NULL){
```

```

48     if(l <= 0 || c <= 0 || qnn < 0){
49         printf("Valores invalidos, matriz nao criada!\n");
50         return NULL;
51     }
52     ms->lin = l; ms->col = c;
53     ms->QI = 0; ms->QNN = qnn;
54     ms->A = ms->IA = ms->JA = NULL;
55     if(qnn != 0){
56         ms->A = (int*) malloc (qnn*sizeof(int));
57         ms->JA = (int*) malloc (qnn*sizeof(int));
58         if(ms->A == NULL || ms->JA == NULL) return NULL;
59     }
60     ms->IA = (int*) malloc ((ms->lin+1)*sizeof(int));
61     if(ms->IA == NULL) return NULL;
62     int i; for(i=0; i<lin+1; i++) ms->IA[i] = 0;
63 }
64 return ms;
65 }
66
67 int* meuRealloc(int* v, int tam){
68     int* aux = (int*) malloc ((tam+1)*sizeof(int));
69     if(aux != NULL){
70         if(v != NULL){
71             int i;
72             for(i=0; i<tam; i++)
73                 aux[i] = v[i];
74             free(v);
75         }
76     }
77     return aux;
78 }
79
80 void imprimeEsparsaVetores(MEsparsaCSR* ms){
81     if(ms == NULL) return;
82     int i, j;
83     printf("Matriz Esparsa, Tam: %d x %d:\n", ms->lin, ms->col);
84     printf("%d elementos nao nulos.\n", ms->QNN);
85     printf("A = [");
86     for(i=0; i<ms->QNN; i++)
87         printf("%d ", ms->A[i]);
88     printf("]\n");
89     printf("IA = [");
90     for(i=0; i<ms->lin+1; i++)
91         printf("%d ", ms->IA[i]);
92     printf("]\n");

```



```

93     printf("JA = [");
94     for(i=0; i<ms->QNN; i++)
95         printf("%d ", ms->JA[i]);
96     printf("]\n\n");
97 }
98
99 int insereElemEsparsa(MEsparsaCSR *ms, int elem, int i, int j) {
100     if(ms == NULL) return 0;
101     if(i < 0 || j < 0 || i >= ms->lin || j >= ms->col){
102         printf("Valores invalidos, elem nao inserido!\n");
103         return 0;
104     }
105     int k;
106     int index = -1;
107     int ini = ms->IA[i]; int fim = ms->IA[i+1];
108     // Encontre a posição correta para inserir o valor
109     for(k = ini; k<fim; k++){
110         if (ms->JA[k] >= j) {
111             index = k;
112             break;
113         }
114     }
115     if (index == -1) { //NOVA INSERCAO
116         if(ms->QI == ms->QNN){//Necessita REALLOC
117             ms->A = meuRealloc(ms->A, ms->QNN);
118             ms->JA = meuRealloc(ms->JA, ms->QNN);
119             ms->QNN++;
120         }
121         //Move elementos para a nova insercao
122         for(k = ms->QNN-1; k>=fim; k--){
123             ms->A[k] = ms->A[k-1];
124             ms->JA[k] = ms->JA[k-1];
125         }
126         ms->A[fim] = elem;
127         ms->JA[fim] = j;
128         ms->QI++;
129         // Atualiza QNN acumulado
130         for (int k = i+1; k<=ms->lin; k++)
131             ms->IA[k]++;
132     } else { //Atualiza um valor existente
133         ms->A[index] = elem;
134     }
135     imprimeEsparsaVetores(ms);
136     return 1;
137 }

```

```

138
139 int removeElemEsparsa(MEsparsaCSR *ms, int i, int j) {
140     if(ms == NULL) return 0;
141     if(i < 0 || j < 0 || i >= ms->lin || j >= ms->col){
142         printf("Valores invalidos, elem nao removido!\n");
143         return 0;
144     }
145
146     int k;
147     int index = -1;
148     int ini = ms->IA[i]; int fim = ms->IA[i+1];
149     // Encontre a posição do valor a ser removido
150     for(k = ini; k<fim; k++){
151         if (ms->JA[k] == j) {
152             index = k;
153             break;
154         }
155     }
156
157     if (index != -1) {
158         // Move todos elementos uma posição para tras
159         for (k = index; k < ms->QNN - 1; k++) {
160             ms->A[k] = ms->A[k+1];
161             ms->JA[k] = ms->JA[k+1];
162         }
163         ms->QNN--;
164         ms->QI--;
165         // Atualiza QNN acumulado
166         for (int k = i+1; k<ms->lin; k++)
167             ms->IA[k]--;
168     }else{
169         printf("Elemento nao existente!\n"); return 0;
170     }
171     imprimeEsparsaVetores(ms);
172     return 1;
173 }
174
175 MEsparsaCSR* transformarEmCSR(Matriz* mat){
176     MEsparsaCSR *ms = criaMatrizEsparsa(mat->lin, mat->col, 0);
177     if(ms != NULL){
178         if(mat == NULL){
179             printf("Matriz de entrada inexistente!\n");
180             return NULL;
181         }
182
183         int i, j;
184         for(i=0; i<mat->lin; i++)
185             for(j=0; j<mat->col; j++)
186                 if(mat->dados[i][j] != 0)
187                     insereElemEsparsa(ms, mat->dados[i][j], i, j);
188     }
189     return ms;
190 }
191
192 int consultaElemEsparsa(MEsparsaCSR* ms, int i, int j){
193     if(ms == NULL) return 0;
194     if(i < 0 || j < 0 || i >= ms->lin || j >= ms->col){
195         printf("Valores invalidos, elem inexistente!\n");
196         return 0;
197     }
198     int k;
199     for(k = ms->IA[i]; k<ms->IA[i+1]; k++)
200         if(ms->JA[k] == j) return ms->A[k];
201     return 0;
202 }
203
204 void imprimeEsparsa(MEsparsaCSR* ms){
205     if(ms == NULL) return;
206     int i, j;
207     imprimeEsparsaVetores(ms);
208     printf("Matriz Original:\n");
209     for(i=0; i<ms->lin; i++){
210         for(j=0; j<ms->col; j++){
211             printf("%d\t", consultaElemEsparsa(ms, i, j));
212             printf("\n");
213         }
214     }
215
216 void destroiMatrizEsparsa(MEsparsaCSR* ms){
217     if(ms != NULL){
218         free(ms->A);
219         free(ms->IA);
220         free(ms->JA);
221         free(ms);
222     }
223 }
224
225 #endif

```

## Console

```
messiasfcm@MessiasFCM:/mnt/c/Users/Messi/OneDrive/Área de Trabalho/lista7/atividade2.2$ gcc main.c matrizDin.h mEsparsaCSR.h -o tp4
messiasfcm@MessiasFCM:/mnt/c/Users/Messi/OneDrive/Área de Trabalho/lista7/atividade2.2$ ./tp4

--CX Menu Matriz Din X--
1 - Criar Matriz Din
2 - Zerar Matriz Din
3 - Preencher Aleatoriamente
4 - Inserir Elemento
5 - Consultar Elemento
6 - É Matriz quadrada?
7 - Criar Triangulo Sup
8 - Criar Triangulo Inf
9 - Criar Diagonal
10 - Criar Transposta
11 - É Simetrica?
12 - Imprimir Matriz Din

--CX Menu Matriz Esparsa X--
13 - Criar Matriz Esparsa
14 - Inserir Elemento
15 - Remover Elemento
16 - Transformar em CSR
17 - Consultar Elemento
18 - Imprimir Matriz Esparsa(Vetores)
19 - Imprimir Matriz Esparsa
20 - Finalizar

Escolha: 1
-> Insira a quantidade de linha: 10
-> Insira a quantidade de coluna: 10
-> Criado com sucesso!
Escolha: 3
-> Insira um valor inicial: 10
-> Insira um valor final: 99
-> Matriz preenchida aleatoriamente!
Escolha: 12
Matriz 10 x 10:
28 78 51 40 44 89 62 10 11 94
22 90 61 62 14 69 68 27 74 10
28 87 48 57 36 84 34 65 69 64
18 88 42 21 28 38 10 80 91 64
75 13 54 88 66 20 19 86 37 83
48 55 32 49 65 20 33 89 75 54
54 45 94 86 56 22 77 19 55 68
73 82 33 79 32 51 89 41 37 27
76 38 72 98 77 37 18 10 27 45
16 33 81 11 19 37 75 48 98 30
```

```
Escolha: 13
-> Insira a quantidade de linha: 10
-> Insira a quantidade de coluna: 10
-> Insira a quantidade de não nulos: 5
Escolha: 19
Matriz Esparsa, Tam: 10 x 10:
5 elementos nao nulos.
A = [0 0 0 0 0 ]
IA = [0 0 0 0 0 0 0 0 0 0 ]
JA = [0 0 0 0 0 ]

Matriz Original:
0      0      0      0      0      0      0      0      0      0
0      0      0      0      0      0      0      0      0      0
0      0      0      0      0      0      0      0      0      0
0      0      0      0      0      0      0      0      0      0
0      0      0      0      0      0      0      0      0      0
0      0      0      0      0      0      0      0      0      0
0      0      0      0      0      0      0      0      0      0
0      0      0      0      0      0      0      0      0      0
0      0      0      0      0      0      0      0      0      0
0      0      0      0      0      0      0      0      0      0

Escolha: 16
Matriz Esparsa, Tam: 10 x 10:
1 elementos nao nulos.
A = [28 ]
IA = [0 1 1 1 1 1 1 1 1 1 ]
JA = [0 ]

Matriz Esparsa, Tam: 10 x 10:
2 elementos nao nulos.
A = [28 78 ]
IA = [0 2 2 2 2 2 2 2 2 2 ]
JA = [0 1 ]

Matriz Esparsa, Tam: 10 x 10:
3 elementos nao nulos.
A = [28 78 51 ]
IA = [0 3 3 3 3 3 3 3 3 3 ]
JA = [0 1 2 ]
```

[illegible]