

matriz_sequencial_estática.h

```
1  #ifndef MSE_H
2  #define MSE_H
3
4  #include <stdio.h>
5  #include <stdlib.h>
6  #define MAX 100
7  typedef struct{
8      int dados[MAX][MAX];
9      int linha, coluna;
10 }Matriz;
11
12 void zeraMatriz(Matriz* matriz){
13     int i, j;
14     if (matriz != NULL){
15         for (i=0; i<matriz->linha;i++)
16             for (j=0; j<matriz->coluna;j++)
17                 matriz->dados[i][j] = 0;
18     }
19 }
20
21 Matriz* criaMatriz(int linha, int coluna){
22     Matriz* matriz;
23     matriz = (Matriz*) malloc(sizeof(Matriz));
24     if (matriz != NULL){
25         if (linha <= 0 || coluna <=0 || linha > MAX || coluna > MAX){
26             printf ("Valores invalidos, matriz nao criada!\n");
27             return NULL;
28         }
29         matriz->linha = linha;
30         matriz->coluna = coluna;
31         zeraMatriz(matriz);
32     }
33     return matriz;
34 }
35
36 void destroiMatriz(Matriz* matriz){
37     if (matriz != NULL)
38         free(matriz);
39 }
40
41 int insereElemento(Matriz* matriz, int elemento, int linha, int coluna){
42     if (matriz == NULL)
43         return 0;
44     if (linha < 0 || coluna < 0 || linha >= matriz->linha || coluna >= matriz->coluna){
45         printf ("Valores invalidos, elemento nao foi inserido.\n");
46         return 0;
47     }
48     matriz->dados[linha][coluna] = elemento;
49     return 1;
50 }
51
52 int consultaElemento(Matriz* matriz, int *elemento, int linha, int coluna){
53     if (matriz == NULL)
54         return 0;
55     if (linha < 0 || coluna < 0 || linha >= matriz->linha || coluna >= matriz->coluna){
56         printf ("Valores invalidos, elemento nao existe na matriz!\n");
57         return 0;
```

```
58     }
59     *elemento = matriz->dados[linha][coluna];
60     return 1;
61 }
62
63 void imprime (Matriz* matriz){
64     if (matriz == NULL)
65         return;
66     int i, j;
67     printf ("Matriz %d x %d: \n", matriz->linha, matriz->coluna);
68     for (i=0; i<matriz->linha; i++){
69         for (j=0; j<matriz->coluna; j++){
70             printf ("%d ", matriz->dados[i][j]);
71             printf ("\n");
72         }
73         printf ("\n");
74     }
75
76     Matriz* criaTransposta (Matriz* matriz){
77         if (matriz == NULL)
78             return NULL;
79         Matriz* transposta = criaMatriz(matriz->coluna, matriz->linha);
80         int i, j;
81         for (i=0; i< matriz->linha; i++){
82             for (j=0; j<matriz->coluna; j++){
83                 transposta->dados[j][i] = matriz->dados[i][j];
84             }
85             return transposta;
86         }
87
88         int e_matrizQuadrada(Matriz *matriz){
89             if (matriz == NULL)
90                 return 0;
91             return (matriz->linha == matriz->coluna);
92         }
93
94         int e_Simetrica(Matriz* matriz){
95             if (matriz == NULL)
96                 return 0;
97             if (!e_matrizQuadrada(matriz)){
98                 printf ("Matriz nao quadrada!\n");
99                 return 0;
100             }
101             int i, j;
102             for (i=0; i<matriz->linha; i++){
103                 for (j=0; j<matriz->coluna; j++){
104                     if (matriz->dados[i][j] != matriz->dados[j][i])
105                         return 0;
106                 }
107             }
108             return 1;
109         }
110
111         Matriz* criaTriangularSuperior(Matriz* matriz){
112             if (matriz == NULL)
113                 return 0;
114             if (!e_matrizQuadrada(matriz)){
115                 printf ("Matriz nao quadrada!\n");
116                 return NULL;
117             }
118             int i, j;
```

```
118     Matriz* triangularSuperior = criaMatriz(matriz->linha, matriz->coluna);
119     for (i=0; i<matriz->linha; i++){
120         for (j=0; j<matriz->coluna; j++){
121             if (i <= j)
122                 triangularSuperior->dados[i][j] = matriz->dados[i][j];
123         }
124     return triangularSuperior;
125 }
126
127 Matriz* criaTriangularInferior(Matriz* matriz){
128     if (matriz == NULL)
129         return 0;
130     if (!e_matrizQuadrada(matriz)){
131         printf ("Matriz nao quadrada!\n");
132         return NULL;
133     }
134     int i, j;
135     Matriz* triangularInferior = criaMatriz(matriz->linha, matriz->coluna);
136     for (i=0; i<matriz->linha; i++){
137         for (j=0; j<matriz->coluna; j++){
138             if (i >= j)
139                 triangularInferior->dados[i][j] = matriz->dados[i][j];
140         }
141     return triangularInferior;
142 }
143
144 Matriz* criaDiagonal(Matriz* matriz){
145     if (matriz == NULL)
146         return 0;
147     if (!e_matrizQuadrada(matriz)){
148         printf ("Matriz nao quadrada!\n");
149         return NULL;
150     }
151     int i, j;
152     Matriz* diagonal = criaMatriz(matriz->linha, matriz->coluna);
153     for (i=0; i < matriz->linha; i++)
154         diagonal->dados[i][i] = matriz->dados[i][i];
155     return diagonal;
156 }
157
158 #endif //MSE_H
```

matriz_sequencial_dinamica.h

```
1  #ifndef MSD_H
2  #define MSD_H
3  #include <stdio.h>
4  #include <stdlib.h>
5
6  typedef struct{
7      int **dados;
8      int linhas, colunas;
9  }Matriz;
10
11 void zeraMatriz(Matriz* matriz){
12     int i, j;
13     if (matriz != NULL){
14         for (i=0; i<matriz->linhas;i++)
15             for (j=0; j<matriz->colunas;j++)
16                 matriz->dados[i][j] = 0;
17     }
18 }
19
20 Matriz* criaMatriz(int linhas, int colunas){
21     Matriz* matriz;
22     matriz = (Matriz*) malloc (sizeof(Matriz));
23     if (matriz != NULL){
24         if (linhas <= 0 || colunas <= 0){
25             printf ("Valores invalidos!");
26             return NULL;
27         }
28         int i;
29         matriz->linhas = linhas;
30         matriz->colunas = colunas;
31         matriz->dados = (int**) malloc (linhas * sizeof(int*));
32         for (i=0; i < linhas; i++)
33             matriz->dados[i] = (int*) malloc (colunas * sizeof(int));
34         zeraMatriz(matriz);
35     }
36 }
37
38 void destroiMatriz(Matriz* matriz){
39     if (matriz != NULL){
40         int i;
41         for (i=0; i < matriz->linhas; i++)
42             free(matriz->dados[i]);
43         free(matriz->dados);
44         free(matriz);
45     }
46 }
47
48 int insereElemento(Matriz* matriz, int elemento, int linha, int coluna){
49     if (matriz == NULL)
50         return 0;
51     if (linha < 0 || coluna < 0 || linha >= matriz->linhas || coluna >= matriz->colunas){
52         printf ("Valores invalidos, elemento nao foi inserido.\n");
53         return 0;
54     }
55     matriz->dados[linha][coluna] = elemento;
56     return 1;
57 }
```

```
58
59 int consultaElemento(Matriz* matriz, int *elemento, int linha, int coluna){
60     if (matriz == NULL)
61         return 0;
62     if (linha < 0 || coluna < 0 || linha >= matriz->linhas || coluna >= matriz->colunas){
63         printf ("Valores invalidos, elemento nao existe na matriz!\n");
64         return 0;
65     }
66     *elemento = matriz->dados[linha][coluna];
67     return 1;
68 }
69
70 void imprime (Matriz* matriz){
71     if (matriz == NULL)
72         return;
73     int i, j;
74     printf ("Matriz %d x %d: \n", matriz->linhas, matriz->colunas);
75     for (i=0; i<matriz->linhas; i++){
76         for (j=0; j<matriz->colunas; j++){
77             printf ("%d ", matriz->dados[i][j]);
78             printf ("\n");
79         }
80         printf ("\n");
81     }
82
83 Matriz* criaTransposta (Matriz* matriz){
84     if (matriz == NULL)
85         return NULL;
86     Matriz* transposta = criaMatriz(matriz->colunas, matriz->linhas);
87     int i, j;
88     for (i=0; i< matriz->linhas; i++){
89         for (j=0; j<matriz->colunas; j++){
90             transposta->dados[j][i] = matriz->dados[i][j];
91         }
92     }
93     return transposta;
94 }
95
96 int e_matrizQuadrada(Matriz *matriz){
97     if (matriz == NULL)
98         return 0;
99     return (matriz->linhas == matriz->colunas);
100 }
101
102 int e_Simetrica(Matriz* matriz){
103     if (matriz == NULL)
104         return 0;
105     if (!e_matrizQuadrada(matriz)){
106         printf ("Matriz nao quadrada!\n");
107         return 0;
108     }
109     int i, j;
110     for (i=0; i<matriz->linhas; i++){
111         for (j=0; j<matriz->colunas; j++){
112             if (matriz->dados[i][j] != matriz->dados[j][i])
113                 return 0;
114         }
115     }
116     return 1;
117 }
118
119 Matriz* criaTriangularSuperior(Matriz* matriz){
```

```
118     if (matriz == NULL)
119         return 0;
120     if (!e_matrizQuadrada(matriz)){
121         printf ("Matriz nao quadrada!\n");
122         return NULL;
123     }
124     int i, j;
125     Matriz* triangularSuperior = criaMatriz(matriz->linhas, matriz->colunas);
126     for (i=0; i<matriz->linhas; i++){
127         for (j=0; j<matriz->colunas; j++){
128             if (i <= j)
129                 triangularSuperior->dados[i][j] = matriz->dados[i][j];
130         }
131     }
132     return triangularSuperior;
133 }
134
135 Matriz* criaTriangularInferior(Matriz* matriz){
136     if (matriz == NULL)
137         return 0;
138     if (!e_matrizQuadrada(matriz)){
139         printf ("Matriz nao quadrada!\n");
140         return NULL;
141     }
142     int i, j;
143     Matriz* triangularInferior = criaMatriz(matriz->linhas, matriz->colunas);
144     for (i=0; i<matriz->linhas; i++){
145         for (j=0; j<matriz->colunas; j++){
146             if (i >= j)
147                 triangularInferior->dados[i][j] = matriz->dados[i][j];
148         }
149     }
150     return triangularInferior;
151 }
152
153 Matriz* criaDiagonal(Matriz* matriz){
154     if (matriz == NULL)
155         return 0;
156     if (!e_matrizQuadrada(matriz)){
157         printf ("Matriz nao quadrada!\n");
158         return NULL;
159     }
160     int i, j;
161     Matriz* diagonal = criaMatriz(matriz->linhas, matriz->colunas);
162     for (i=0; i < matriz->linhas; i++){
163         diagonal->dados[i][i] = matriz->dados[i][i];
164     }
165     return diagonal;
166 }
167 #endif //MSD_H
```

exercicio1.c

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include "matriz_sequencial_estatica.h"
4  // #include "matriz_sequencial_dinamica.h"
5  //Esse arquivo funciona tanto para a Matriz Sequencial Estática, quanto para a Matriz
   Sequencial Dinâmica, basta escolher qual biblioteca importar.
6
7  int main() {
8      Matriz *matriz = NULL;
9      int option, linhas, colunas, elemento, valor;
10     do{
11         printf("1. Criar Matriz\n");
12         printf("2. Destruir Matriz\n");
13         printf("3. Inserir Elemento\n");
14         printf("4. Consultar Elemento\n");
15         printf("5. Imprimir Matriz\n");
16         printf("6. Criar Transposta da Matriz\n");
17         printf("7. Verificar se a matriz e quadrada\n");
18         printf("8. Verificar se a matriz e simetrica\n");
19         printf("9. Criar Matriz Triangular Superior\n");
20         printf("10. Criar Matriz Triangular Inferior\n");
21         printf("11. Criar Matriz Diagonal\n");
22         printf("12. Sair\n");
23         printf("Escolha uma opcao: ");
24         scanf("%d", &option);
25         switch(option){
26             case 1:
27                 printf("Informe o numero de linhas: ");
28                 scanf("%d", &linhas);
29                 printf("Informe o numero de colunas: ");
30                 scanf("%d", &colunas);
31                 matriz = criaMatriz(linhas, colunas);
32                 if(matriz == NULL)
33                     printf("Falha na criacao da matriz!\n");
34                 else
35                     printf("Matriz criada!\n");
36                 break;
37             case 2:
38                 destroiMatriz(matriz);
39                 matriz = NULL;
40                 printf("Matriz destruida!\n");
41                 break;
42             case 3:
43                 if(matriz == NULL){
44                     printf("Matriz nao foi criada!\n");
45                 }else{
46                     printf("Informe o elemento: ");
47                     scanf("%d", &elemento);
48                     printf("Informe a linha e coluna (linha coluna): ");
49                     scanf("%d %d", &linhas, &colunas);
50                     if(insereElemento(matriz, elemento, linhas, colunas))
51                         printf("Elemento inserido!\n");
52                     else
53                         printf("Nao foi possivel inserir o elemento!\n");
54                 }
55                 break;
56             case 4:
```

```
57     if(matriz == NULL){
58         printf("Erro: Matriz nao foi criada!\n");
59     }else{
60         printf("Informe a linha e coluna (linha coluna): ");
61         scanf("%d %d", &linhas, &colunas);
62         if(consultaElemento(matriz, &valor, linhas, colunas))
63             printf("Elemento consultado: %d\n", valor);
64         else
65             printf("Erro: Falha na consulta do elemento!\n");
66     }
67     break;
68 case 5:
69     if(matriz == NULL){
70         printf("Matriz nao foi criada!\n");
71     }else{
72         imprime(matriz);
73     }
74     break;
75 case 6:
76     if(matriz == NULL){
77         printf("Matriz nao foi criada!\n");
78     }else{
79         Matriz *transposta = criaTransposta(matriz);
80         if(transposta == NULL){
81             printf("Nao foi possivel criar a matriz transposta!\n");
82         }else{
83             printf("Matriz transposta:\n");
84             imprime(transposta);
85             destroiMatriz(transposta);
86         }
87     }
88     break;
89 case 7:
90     if(matriz == NULL){
91         printf("Matriz nao foi criada!\n");
92     }else{
93         if(e_matrizQuadrada(matriz))
94             printf("A matriz e' quadrada!\n");
95         else
96             printf("A matriz nao e' quadrada!\n");
97     }
98     break;
99 case 8:
100    if(matriz == NULL){
101        printf("Matriz nao foi criada!\n");
102    }else{
103        if(e_Simetrica(matriz))
104            printf("A matriz e' simetrica!\n");
105        else
106            printf("A matriz nao e' simetrica!\n");
107    }
108    break;
109 case 9:
110    if(matriz == NULL){
111        printf("Matriz nao foi criada!\n");
112    }else{
113        Matriz *triangularSuperior = criaTriangularSuperior(matriz);
114        if(triangularSuperior == NULL) {
115            printf("Nao foi possivel criar a matriz triangular superior!\n");
116        }else{
```



```
117         printf("Matriz triangular superior:\n");
118         imprime(triangularSuperior);
119         destroiMatriz(triangularSuperior);
120     }
121 }
122 break;
123 case 10:
124     if(matriz == NULL){
125         printf("Erro: Matriz nao foi criada!\n");
126     }else{
127         Matriz *triangularInferior = criaTriangularInferior(matriz);
128         if(triangularInferior == NULL){
129             printf("Erro: Falha na criacao da matriz triangular inferior!\n");
130         }else{
131             printf("Matriz triangular inferior:\n");
132             imprime(triangularInferior);
133             destroiMatriz(triangularInferior);
134         }
135     }
136     break;
137 case 11:
138     if(matriz == NULL){
139         printf("Matriz nao foi criada!\n");
140     }else{
141         Matriz *diagonal = criaDiagonal(matriz);
142         if(diagonal == NULL){
143             printf("Nao foi possivel criar a matriz diagonal!\n");
144         }else{
145             printf("Matriz diagonal:\n");
146             imprime(diagonal);
147             destroiMatriz(diagonal);
148         }
149     }
150     break;
151 case 12:
152     printf("Saindo.\n");
153     break;
154 default:
155     printf("Opcao invalida!\n");
156 }
157 }while(option != 12);
158 return 0;
159 }
```

PS C:\Users\USER\OneDrive\Área de Trabalho\LabProg2\Lista 7\output> & .\'exercicio1.exe'

1. Criar Matriz
2. Destruir Matriz
3. Inserir Elemento
4. Consultar Elemento
5. Imprimir Matriz
6. Criar Transposta da Matriz
7. Verificar se a matriz e quadrada
8. Verificar se a matriz e simetrica
9. Criar Matriz Triangular Superior
10. Criar Matriz Triangular Inferior
11. Criar Matriz Diagonal
12. Sair

Escolha uma opcao: 1
Informe o numero de linhas: 3
Informe o numero de colunas: 3
Matriz criada!

1. Criar Matriz
2. Destruir Matriz
3. Inserir Elemento
4. Consultar Elemento
5. Imprimir Matriz
6. Criar Transposta da Matriz
7. Verificar se a matriz e quadrada
8. Verificar se a matriz e simetrica
9. Criar Matriz Triangular Superior
10. Criar Matriz Triangular Inferior
11. Criar Matriz Diagonal
12. Sair

Escolha uma opcao: 6
Matriz transposta:
Matriz 3 x 3:
0 0 0
0 10 0
0 0 0

1. Criar Matriz
2. Destruir Matriz
3. Inserir Elemento
4. Consultar Elemento
5. Imprimir Matriz
6. Criar Transposta da Matriz
7. Verificar se a matriz e quadrada
8. Verificar se a matriz e simetrica
9. Criar Matriz Triangular Superior
10. Criar Matriz Triangular Inferior
11. Criar Matriz Diagonal
12. Sair

Escolha uma opcao: 2
Matriz destruida!

1. Criar Matriz
2. Destruir Matriz
3. Inserir Elemento
4. Consultar Elemento
5. Imprimir Matriz
6. Criar Transposta da Matriz
7. Verificar se a matriz e quadrada
8. Verificar se a matriz e simetrica
9. Criar Matriz Triangular Superior
10. Criar Matriz Triangular Inferior
11. Criar Matriz Diagonal
12. Sair

Escolha uma opcao: 7
A matriz e' quadrada!

1. Criar Matriz
2. Destruir Matriz
3. Inserir Elemento
4. Consultar Elemento
5. Imprimir Matriz
6. Criar Transposta da Matriz
7. Verificar se a matriz e quadrada
8. Verificar se a matriz e simetrica
9. Criar Matriz Triangular Superior
10. Criar Matriz Triangular Inferior
11. Criar Matriz Diagonal
12. Sair

Escolha uma opcao: 12
Saindo.



1. Criar Matriz
2. Destruir Matriz
3. Inserir Elemento
4. Consultar Elemento
5. Imprimir Matriz
6. Criar Transposta da Matriz
7. Verificar se a matriz e quadrada
8. Verificar se a matriz e simetrica
9. Criar Matriz Triangular Superior
10. Criar Matriz Triangular Inferior
11. Criar Matriz Diagonal
12. Sair

Escolha uma opcao: 3
Informe o elemento: 10
Informe a linha e coluna (linha coluna): 1
1
Elemento inserido!

1. Criar Matriz
2. Destruir Matriz
3. Inserir Elemento
4. Consultar Elemento
5. Imprimir Matriz
6. Criar Transposta da Matriz
7. Verificar se a matriz e quadrada
8. Verificar se a matriz e simetrica
9. Criar Matriz Triangular Superior
10. Criar Matriz Triangular Inferior
11. Criar Matriz Diagonal
12. Sair

Escolha uma opcao: 8
A matriz e' simetrica!

1. Criar Matriz
2. Destruir Matriz
3. Inserir Elemento
4. Consultar Elemento
5. Imprimir Matriz
6. Criar Transposta da Matriz
7. Verificar se a matriz e quadrada
8. Verificar se a matriz e simetrica
9. Criar Matriz Triangular Superior
10. Criar Matriz Triangular Inferior
11. Criar Matriz Diagonal
12. Sair

Escolha uma opcao: 5
Matriz 3 x 3:
0 0 0
0 10 0
0 0 0

1. Criar Matriz
2. Destruir Matriz
3. Inserir Elemento
4. Consultar Elemento
5. Imprimir Matriz
6. Criar Transposta da Matriz
7. Verificar se a matriz e quadrada
8. Verificar se a matriz e simetrica
9. Criar Matriz Triangular Superior
10. Criar Matriz Triangular Inferior
11. Criar Matriz Diagonal
12. Sair

Escolha uma opcao: 9
Matriz triangular superior:
Matriz 3 x 3:
0 0 0
0 10 0
0 0 0

1. Criar Matriz
2. Destruir Matriz
3. Inserir Elemento
4. Consultar Elemento
5. Imprimir Matriz
6. Criar Transposta da Matriz
7. Verificar se a matriz e quadrada
8. Verificar se a matriz e simetrica
9. Criar Matriz Triangular Superior
10. Criar Matriz Triangular Inferior
11. Criar Matriz Diagonal
12. Sair

Escolha uma opcao: 10
Matriz triangular inferior:
Matriz 3 x 3:
0 0 0
0 10 0
0 0 0

1. Criar Matriz
2. Destruir Matriz
3. Inserir Elemento
4. Consultar Elemento
5. Imprimir Matriz
6. Criar Transposta da Matriz
7. Verificar se a matriz e quadrada
8. Verificar se a matriz e simetrica
9. Criar Matriz Triangular Superior
10. Criar Matriz Triangular Inferior
11. Criar Matriz Diagonal
12. Sair

Escolha uma opcao: 4
Informe a linha e coluna (linha coluna): 1
1
Elemento consultado: 10

1. Criar Matriz
2. Destruir Matriz
3. Inserir Elemento
4. Consultar Elemento
5. Imprimir Matriz
6. Criar Transposta da Matriz
7. Verificar se a matriz e quadrada
8. Verificar se a matriz e simetrica
9. Criar Matriz Triangular Superior
10. Criar Matriz Triangular Inferior
11. Criar Matriz Diagonal
12. Sair

Escolha uma opcao: 11
Matriz diagonal:
Matriz 3 x 3:
0 0 0
0 10 0
0 0 0

matriz_de_faixa.h

```
1  #ifndef MF_H
2  #define MF_H
3  #include <stdio.h>
4  #include <stdlib.h>
5
6  typedef struct{
7      int *diagonal;
8      int *superior;
9      int *inferior;
10     int tamanho;
11 }MatrizFaixa;
12
13 void zeraMatriz(MatrizFaixa *matrizFaixa) {
14     if(matrizFaixa == NULL)
15         return;
16     for(int i = 0; i < matrizFaixa->tamanho; i++)
17         matrizFaixa->diagonal[i] = 0;
18     for(int i = 0; i < matrizFaixa->tamanho - 1; i++)
19         matrizFaixa->superior[i] = 0;
20     for(int i = 0; i < matrizFaixa->tamanho - 1; i++)
21         matrizFaixa->inferior[i] = 0;
22 }
23
24 MatrizFaixa* criaMatriz(int t){
25     MatrizFaixa *matrizFaixa = (MatrizFaixa*) malloc(sizeof(MatrizFaixa));
26     if(matrizFaixa != NULL){
27         if(t <= 1)
28             return NULL;
29         matrizFaixa->tamanho = t;
30         matrizFaixa->diagonal = (int*) malloc(t * sizeof(int));
31         matrizFaixa->superior = (int*) malloc((t-1) * sizeof(int));
32         matrizFaixa->inferior = (int*) malloc((t-1) * sizeof(int));
33         if(matrizFaixa->diagonal == NULL || matrizFaixa->superior == NULL || matrizFaixa->
inferior == NULL){
34             return NULL;
35         }
36         zeraMatriz(matrizFaixa);
37     }
38     return matrizFaixa;
39 }
40
41 void destroiMatriz(MatrizFaixa *matrizFaixa){
42     if(matrizFaixa != NULL){
43         free(matrizFaixa->diagonal);
44         free(matrizFaixa->superior);
45         free(matrizFaixa->inferior);
46         free(matrizFaixa);
47     }
48 }
49
50 int insereElemento(MatrizFaixa *matrizFaixa, int elemento, int i, int j){
51     if(matrizFaixa == NULL)
52         return -1;
53     if(i < 0 || j < 0 || i >= matrizFaixa->tamanho || j >= matrizFaixa->tamanho)
54         return 0;
55     if(i == j)
56         matrizFaixa->diagonal[i] = elemento;
```

```
57     else if(i + 1 == j)
58         matrizFaixa->superior[i] = elemento;
59     else if(i == j + 1)
60         matrizFaixa->inferior[j] = elemento;
61     else
62         return 0;
63     return 1;
64 }
65
66 int consultaElemento(MatrizFaixa *matrizFaixa, int i, int j){
67     if(matrizFaixa == NULL)
68         return 0;
69     if(i < 0 || j < 0 || i >= matrizFaixa->tamanho || j >= matrizFaixa->tamanho)
70         return 0;
71     if(i == j)
72         return matrizFaixa->diagonal[i];
73     else if(i + 1 == j)
74         return matrizFaixa->superior[i];
75     else if(i == j + 1)
76         return matrizFaixa->inferior[j];
77     else
78         return 0;
79 }
80
81 void imprimeFaixaVetores(MatrizFaixa *matrizFaixa){
82     if(matrizFaixa == NULL)
83         return;
84     int i;
85     printf("Matriz Faixa, Tam: %d x %d:\n", matrizFaixa->tamanho, matrizFaixa->tamanho);
86     printf("Diagonal = [");
87     for(i = 0; i < matrizFaixa->tamanho; i++)
88         printf("%d ", matrizFaixa->diagonal[i]);
89     printf("]\n");
90     printf("Superior = [");
91     for (i = 0; i < matrizFaixa->tamanho - 1; i++)
92         printf("%d ", matrizFaixa->superior[i]);
93     printf("]\n");
94     printf("Inferior = [");
95     for(i = 0; i < matrizFaixa->tamanho - 1; i++)
96         printf("%d ", matrizFaixa->inferior[i]);
97     printf("]\n\n");
98 }
99
100 void imprimeFaixa(MatrizFaixa *matrizFaixa){
101     if(matrizFaixa == NULL)
102         return;
103     imprimeFaixaVetores(matrizFaixa);
104     printf("Matriz original: \n");
105     for(int i = 0; i < matrizFaixa->tamanho; i++){
106         for(int j = 0; j < matrizFaixa->tamanho; j++){
107             printf("%d\t", consultaElemento(matrizFaixa, i, j));
108         }
109         printf("\n");
110     }
111     printf("\n");
112 }
113
114 #endif //MF_H
```

exercicio2-1.c

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include "matriz_de_faixa.h"
4
5  int main() {
6      MatrizFaixa *matriz = NULL;
7      int tamanho, option;
8      do{
9          printf("1. Criar matriz faixa\n");
10         printf("2. Inserir Elemento.\n");
11         printf("3. Consultar Elemento.\n");
12         printf("4. Imprimir Matriz Faixa\n");
13         printf("5. Destruir Matriz Faixa\n");
14         printf("6. Sair\n");
15         printf("Escolha uma opcao: ");
16         scanf("%d", &option);
17         switch(option){
18             case 1:
19                 if(matriz != NULL){
20                     printf("A matriz faixa ja foi criada!\n");
21                 }else{
22                     printf("Informe o tamanho da matriz faixa: ");
23                     scanf("%d", &tamanho);
24                     matriz = criaMatriz(tamanho);
25                     if(matriz == NULL){
26                         printf("Nao foi possivel criar a matriz faixa!\n");
27                     }else{
28                         printf("Matriz faixa criada com sucesso!\n");
29                     }
30                 }
31                 break;
32             case 2:
33                 if(matriz == NULL){
34                     printf("Matriz nao foi criada.\n");
35                 }else{
36                     int elemento, i, j;
37                     printf("Informe o elemento a ser inserido: ");
38                     scanf("%d", &elemento);
39                     printf("Informe a linha e a coluna: ");
40                     scanf("%d %d", &i, &j);
41                     int resultado = insereElemento(matriz, elemento, i, j);
42                     if(resultado == -1){
43                         printf("Matriz faixa nao foi criada!\n");
44                     }else if(resultado == 0){
45                         printf("Falha na insercao do elemento!\n");
46                     }else{
47                         printf("Elemento inserido.\n");
48                     }
49                 }
50                 break;
51             case 3:
52                 if(matriz == NULL){
53                     printf("Erro: Matriz faixa nao foi criada!\n");
54                 }else{
55                     int linha, coluna;
56                     printf("Informe a linha e a coluna: ");
57                     scanf("%d %d", &linha, &coluna);
```

```
58         int valor = consultaElemento(matriz, linha, coluna);
59         if(valor == 0){
60             printf("Falha ao procurar elemento!\n");
61         }else{
62             printf("Elemento consultado: %d\n", valor);
63         }
64     }
65     break;
66 case 4:
67     if(matriz == NULL){
68         printf("Matriz faixa nao foi criada!\n");
69     }else{
70         imprimeFaixa(matriz);
71     }
72     break;
73 case 5:
74     if(matriz == NULL){
75         printf("Matriz faixa nao foi criada!\n");
76     }else{
77         destroiMatriz(matriz);
78         matriz = NULL;
79         printf("Matriz faixa destruida!\n");
80     }
81     break;
82 case 6:
83     printf("Saindo.\n");
84     break;
85 default:
86     printf("Opcao invalida!\n");
87 }
88 }while(option != 6);
89 return 0;
90 }
```

PS C:\Users\USER\OneDrive\Área de Trabalho\LabProg2\Lista 7\output> & .\'exercicio2-1.exe'

1. Criar matriz faixa
2. Inserir Elemento.
3. Consultar Elemento.
4. Imprimir Matriz Faixa
5. Destruir Matriz Faixa
6. Sair

Escolha uma opcao: 1

Informe o tamanho da matriz faixa: 3

Matriz faixa criada com sucesso!

1. Criar matriz faixa
2. Inserir Elemento.
3. Consultar Elemento.
4. Imprimir Matriz Faixa
5. Destruir Matriz Faixa
6. Sair

Escolha uma opcao: 2

Informe o elemento a ser inserido: 10

Informe a linha e a coluna: 1

1

Elemento inserido.

1. Criar matriz faixa
2. Inserir Elemento.
3. Consultar Elemento.
4. Imprimir Matriz Faixa
5. Destruir Matriz Faixa
6. Sair

Escolha uma opcao: 2

Informe o elemento a ser inserido: 15

Informe a linha e a coluna: 2

2

Elemento inserido.

1. Criar matriz faixa
2. Inserir Elemento.
3. Consultar Elemento.
4. Imprimir Matriz Faixa
5. Destruir Matriz Faixa
6. Sair

Escolha uma opcao: 4

Matriz Faixa, Tam: 3 x 3:

Diagonal = [0 10 15]

Superior = [0 0]

Inferior = [0 0]

Matriz original:

0	0	0
0	10	0
0	0	15

1. Criar matriz faixa
2. Inserir Elemento.
3. Consultar Elemento.
4. Imprimir Matriz Faixa
5. Destruir Matriz Faixa
6. Sair

Escolha uma opcao: 3

Informe a linha e a coluna: 2

2

Elemento consultado: 15

1. Criar matriz faixa
2. Inserir Elemento.
3. Consultar Elemento.
4. Imprimir Matriz Faixa
5. Destruir Matriz Faixa
6. Sair

Escolha uma opcao: 5

Matriz faixa destruida!

1. Criar matriz faixa
2. Inserir Elemento.
3. Consultar Elemento.
4. Imprimir Matriz Faixa
5. Destruir Matriz Faixa
6. Sair

Escolha uma opcao: 6

Saindo.

matriz_csr.h

```
1  #ifndef CSR_H
2  #define CSR_H
3  #include <stdio.h>
4  #include <stdlib.h>
5
6  typedef struct{
7      int *A; //Valores
8      int *IA;
9      int *JA;
10     int lin, col, QNN, QI;
11 }MEsparsaCSR;
12
13 MEsparsaCSR* criaMatrizEsparsa(int l, int c, int qnn){
14     MEsparsaCSR *ms;
15     ms = (MEsparsaCSR*) malloc (sizeof(MEsparsaCSR));
16     if(ms != NULL){
17         if(l <= 0 || c <= 0 || qnn < 0){
18             printf("Valores invalidos, matriz nao criada!\n");
19             return NULL;
20         }
21         ms->lin = l; ms->col = c;
22         ms->QI = 0; ms->QNN = qnn;
23         ms->A = ms->IA = ms->JA = NULL;
24         if(qnn != 0){
25             ms->A = (int*) malloc (qnn*sizeof(int));
26             ms->JA = (int*) malloc (qnn*sizeof(int));
27             if(ms->A == NULL || ms->JA == NULL) return NULL;
28         }
29         ms->IA = (int*) malloc ((ms->lin+1)*sizeof(int));
30         if(ms->IA == NULL) return NULL;
31         int i; for(i=0; i<l+1; i++) ms->IA[i] = 0;
32     }
33     return ms;
34 }
35
36 void destroiMatrizEsparsa(MEsparsaCSR *ms){
37     if(ms != NULL){
38         free(ms->A);
39         free(ms->IA);
40         free(ms->JA);
41         free(ms);
42     }
43 }
44
45 int* meuRealloc(int* v, int tam){
46     int* aux = (int*) malloc ((tam+1)*sizeof(int));
47     if(aux != NULL){
48         if(v != NULL){
49             int i;
50             for(i=0; i<tam; i++)
51                 aux[i] = v[i];
52             free(v);
53         }
54     }
55     return aux;
56 }
57
```



```
58 int insereElemEsparsa( MEsparsaCSR *ms , int elem , int i, int j){
59     if(ms == NULL)
60         return 0;
61     if(i < 0 || j < 0 || i >= ms->lin || j >= ms->col){
62         printf("Valores invalidos, elem nao inserido!\n");
63         return 0;
64     }
65     int k; int index = -1;
66     int ini = ms->IA[i];
67     int fim = ms->IA[i + 1];
68     for(k = ini; k < fim; k++){
69         if(ms->JA[k] >= j){
70             index = k;
71             break;
72         }
73     }
74     if(index == -1){
75         if(ms->QI == ms->QNN){
76             ms->A = meuRealloc(ms->A, ms->QNN );
77             ms->JA = meuRealloc(ms->JA , ms->QNN );
78             ms->QNN++;
79         }
80         for(k = ms->QNN - 1; k >= fim; k--){
81             ms->A[k] = ms->A[k - 1];
82             ms->JA[k] = ms->JA[k - 1];
83         }
84         ms->A[fim] = elem;
85         ms->JA[fim] = j;
86         ms->QI++;
87         for(int k = i + 1; k <= ms->lin; k++){
88             ms->IA[k]++;
89         }
90     }else
91         ms->A[index] = elem;
92     return 1;
93 }
94
95 int removeElemEsparsa(MEsparsaCSR *ms , int i, int j){
96     if(ms == NULL)
97         return 0;
98     if(i < 0 || j < 0 || i >= ms->lin || j >= ms->col){
99         printf("Valores invalidos , elem nao removido!\n");
100         return 0;
101     }
102     int k;
103     int index = -1;
104     int ini = ms->IA[i];
105     int fim = ms->IA[i + 1];
106     for(k = ini; k < fim ; k++){
107         if(ms->JA[k] == j){
108             index = k;
109             break;
110         }
111     }
112     if(index != -1){
113         for(k = index; k < ms->QNN - 1; k++){
114             ms->A[k] = ms->A[k + 1];
115             ms->JA[k] = ms->JA[k + 1];
116         }
117         ms->QNN --;
118         ms->QI --;
119         for(int k = i + 1; k <= ms->lin; k++){
120             ms->IA[k] --;
```

```
118     }else{
119         printf("Elemento nao existente\n");
120         return 0;
121     }
122     return 1;
123 }
124
125 int consultaElemEsparsa(MEsparsaCSR *ms, int i, int j){
126     if(ms == NULL )
127         return 0;
128     if(i < 0 || j < 0 || i >= ms->lin || j >= ms->col){
129         printf("Valores invalidos, elem inexistente!\n");
130         return 0;
131     }
132     int k;
133     for(k = ms->IA[i]; k < ms->IA[i + 1]; k++)
134         if(ms->JA[k] == j)
135             return ms->A[k];
136     return 0;
137 }
138
139 void imprimeEsparsaVetores (MEsparsaCSR * ms){
140     if(ms == NULL)
141         return ;
142     int i;
143     printf("Matriz Esparsa , Tam: %d x %d:\n", ms->lin, ms->col);
144     printf("%d elementos nao nulos .\n", ms->QNN);
145     printf("A = [");
146     for(i = 0; i < ms->QNN; i++)
147         printf("%d ", ms->A[i]);
148     printf("]\n");
149     printf("IA = [");
150     for(i = 0; i < ms->lin + 1; i++)
151         printf("%d ", ms->IA[i]);
152     printf("]\n");
153     printf("JA = [");
154     for(i = 0; i < ms->QNN; i++)
155         printf("%d ", ms->JA[i]);
156     printf("]\n\n");
157 }
158
159 #endif //CSR_H
```

exercicio2-2.c

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include "matriz_csr.h"
4
5  int main(){
6      MEsparsaCSR* esparsa = NULL;
7
8      int option, linhas, colunas, elemento;
9      do{
10         printf ("1- Criar Matriz Esparsa.\n");
11         printf ("2- Destruir Matriz Esparsa.\n");
12         printf ("3- Inserir Elemento.\n");
13         printf ("4- Remover Elemento.\n");
14         printf ("5- Consultar Elemento.\n");
15         printf ("6- Imprimir Matriz Esparsa.\n");
16         printf ("7- Sair.\n");
17         printf("Escolha uma opcao: ");
18         scanf("%d", &option);
19         switch (option){
20             case 1:
21                 if (esparsa != NULL){
22                     destroiMatrizEsparsa(esparsa);
23                     printf ("Matriz apagada!");
24                 }
25                 printf ("Informe as dimensoes\n");
26                 printf ("Linhas e Colunas: ");
27                 scanf ("%d %d", &linhas, &colunas);
28                 esparsa = criaMatrizEsparsa(linhas, colunas, 0);
29                 break;
30             case 2:
31                 destroiMatrizEsparsa(esparsa);
32                 printf ("Matriz Destruida.\n");
33                 break;
34             case 3:
35                 printf ("Elemento a ser inserido: ");
36                 scanf ("%d", &elemento);
37                 printf ("Informe a linha e a coluna onde sera inserido o elemento: ");
38                 scanf ("%d %d", &linhas, &colunas);
39                 if (insereElemEsparsa(esparsa, elemento, linhas, colunas))
40                     printf ("Elemento %d inserido na posicao (%d, %d)\n", elemento, linhas,
colunas);
41                 else
42                     printf ("Nao foi possivel inserir o elemento\n");
43                 break;
44             case 4:
45                 printf ("Informe a linha e a coluna do elemento a ser removido: ");
46                 scanf ("%d %d", &linhas, &colunas);
47                 if (removeElemEsparsa(esparsa, linhas, colunas))
48                     printf ("Removeu o elemento %d da posicao (%d, %d)\n", elemento,
linhas, colunas);
49                 else
50                     printf ("Nao foi possivel remover.\n");
51                 break;
52             case 5:
53                 printf ("Posicao a ser consultada:\n");
54                 printf ("Informe a linha e a coluna: \n");
55                 scanf ("%d %d", &linhas, &colunas);
```

```
56         if (elemento = consultaElemEsparsa(esparsa, linhas, colunas))
57             printf ("Elemento: %d\n", elemento);
58         else
59             printf ("Nao foi possivel consultar nessa posicao da matriz.\n");
60         break;
61     case 6:
62         imprimeEsparsaVetores(esparsa);
63         break;
64     case 7:
65         printf ("Saindo.\n");
66         break;
67     default:
68         printf ("Opcao invalida!\n");
69         break;
70     }
71 } while(option != 7);
72 return 0;
73 }
```

```

PS C:\Users\USER\OneDrive\Área de Trabalho\LabProg2\Lista 7\output> & .\'exercicio2-2.exe'
1- Criar Matriz Esparsa.
2- Destruir Matriz Esparsa.
3- Inserir Elemento.
4- Remover Elemento.
5- Consultar Elemento.
6- Imprimir Matriz Esparsa.
7- Sair.
Escolha uma opcao: 1
Informe as dimensoes
Linhas e Colunas: 3
3
1- Criar Matriz Esparsa.
2- Destruir Matriz Esparsa.
3- Inserir Elemento.
4- Remover Elemento.
5- Consultar Elemento.
6- Imprimir Matriz Esparsa.
7- Sair.
Escolha uma opcao: 3
Elemento a ser inserido: 10
Informe a linha e a coluna onde sera inserido o elemento: 1
1
Elemento 10 inserido na posicao (1, 1)
1- Criar Matriz Esparsa.
2- Destruir Matriz Esparsa.
3- Inserir Elemento.
4- Remover Elemento.
5- Consultar Elemento.
6- Imprimir Matriz Esparsa.
7- Sair.
Escolha uma opcao: 3
Elemento a ser inserido: 15
Informe a linha e a coluna onde sera inserido o elemento: 2
3
Valores invalidos, elem nao inserido!
Nao foi possivel inserir o elemento
1- Criar Matriz Esparsa.
2- Destruir Matriz Esparsa.
3- Inserir Elemento.
4- Remover Elemento.
5- Consultar Elemento.
6- Imprimir Matriz Esparsa.
7- Sair.
Escolha uma opcao: 3
Elemento a ser inserido: 15
Informe a linha e a coluna onde sera inserido o elemento: 2
2
Elemento 15 inserido na posicao (2, 2)

1- Criar Matriz Esparsa.
2- Destruir Matriz Esparsa.
3- Inserir Elemento.
4- Remover Elemento.
5- Consultar Elemento.
6- Imprimir Matriz Esparsa.
7- Sair.
Escolha uma opcao: 6
Matriz Esparsa , Tam: 3 x 3:
2 elementos nao nulos .
A = [10 15 ]
IA = [0 0 1 2 ]
JA = [1 2 ]
1- Criar Matriz Esparsa.
2- Destruir Matriz Esparsa.
3- Inserir Elemento.
4- Remover Elemento.
5- Consultar Elemento.
6- Imprimir Matriz Esparsa.
7- Sair.
Escolha uma opcao: 5
Posicao a ser consultada:
Informe a linha e a coluna:
2
2
Elemento: 15
1- Criar Matriz Esparsa.
2- Destruir Matriz Esparsa.
3- Inserir Elemento.
4- Remover Elemento.
5- Consultar Elemento.
6- Imprimir Matriz Esparsa.
7- Sair.
Escolha uma opcao: 4
Informe a linha e a coluna do elemento a ser removido: 2
2
Removeu o elemento 15 da posicao (2, 2)
1- Criar Matriz Esparsa.
2- Destruir Matriz Esparsa.
3- Inserir Elemento.
4- Remover Elemento.
5- Consultar Elemento.
6- Imprimir Matriz Esparsa.
7- Sair.
Escolha uma opcao: 2
Matriz Destruida.

```