matriz_sequencial_estática.h

```c
#ifndef MSE_H
#define MSE_H

#include <stdio.h>
#include <stdlib.h>
#define MAX 100
typedef struct{
    int dados[MAX][MAX];
    int linha, coluna;
}Matriz;

void zeraMatriz(Matriz* matriz){
    int i, j;
    if (matriz != NULL){
        for (i=0; i<matriz->linha;i++)
            for (j=0; j<matriz->coluna;j++)
                matriz->dados[i][j] = 0;
    }
}

Matriz* criaMatriz(int linha, int coluna){
    Matriz* matriz;
    matriz = (Matriz*) malloc(sizeof(Matriz));
    if (matriz != NULL){
        if (linha <= 0 || coluna <=0 || linha > MAX || coluna > MAX){
            printf ("Valores invalidos, matriz nao criada!\n");
            return NULL;
        }
        matriz->linha = linha;
        matriz->coluna = coluna;
        zeraMatriz(matriz);
    }
    return matriz;
}

void destroiMatriz(Matriz* matriz){
    if (matriz != NULL)
        free(matriz);
}

int insereElemento(Matriz* matriz, int elemento, int linha, int coluna){
    if (matriz == NULL)
        return 0;
    if (linha < 0 || coluna < 0 || linha >= matriz->linha || coluna >= matriz->coluna){
        printf ("Valores invalidos, elemento nao foi inserido.\n");
        return 0;
    }
    matriz->dados[linha][coluna] = elemento;
    return 1;
}

int consultaElemento(Matriz* matriz, int *elemento, int linha, int coluna){
    if (matriz == NULL)
        return 0;
    if (linha < 0 || coluna < 0 || linha >= matriz->linha || coluna >= matriz->coluna){
        printf ("Valores invalidos, elemento nao existe na matriz!\n");
        return 0;
```

```c
    }
    *elemento = matriz->dados[linha][coluna];
    return 1;
}

void imprime (Matriz* matriz){
    if (matriz == NULL)
        return;
    int i, j;
    printf ("Matriz %d x %d: \n", matriz->linha, matriz->coluna);
    for (i=0; i<matriz->linha; i++){
        for (j=0; j<matriz->coluna; j++)
            printf ("%d ", matriz->dados[i][j]);
        printf ("\n");
    }
    printf ("\n");
}

Matriz* criaTransposta (Matriz* matriz){
    if (matriz == NULL)
        return NULL;
    Matriz* transposta = criaMatriz(matriz->coluna, matriz->linha);
    int i, j;
    for (i=0; i< matriz->linha; i++){
        for (j=0; j<matriz->coluna; j++)
            transposta->dados[j][i] = matriz->dados[i][j];
    }
    return transposta;
}

int e_matrizQuadrada(Matriz *matriz){
    if (matriz == NULL)
        return 0;
    return (matriz->linha == matriz->coluna);
}

int e_Simetrica(Matriz* matriz){
    if (matriz == NULL)
        return 0;
    if (!e_matrizQuadrada(matriz)){
        printf ("Matriz nao quadrada!\n");
        return 0;
    }
    int i, j;
    for (i=0; i<matriz->linha; i++){
        for (j=0; j<matriz->coluna; j++)
            if (matriz->dados[i][j] != matriz->dados[j][i])
                return 0;
    }
    return 1;
}

Matriz* criaTriangularSuperior(Matriz* matriz){
    if (matriz == NULL)
        return 0;
    if (!e_matrizQuadrada(matriz)){
        printf ("Matriz nao quadrada!\n");
        return NULL;
    }
    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

```c
#ifndef MSD_H
#define MSD_H
#include <stdio.h>
#include <stdlib.h>

typedef struct{
    int **dados;
    int linhas, colunas;
}Matriz;

void zeraMatriz(Matriz* matriz){
    int i, j;
    if (matriz != NULL){
        for (i=0; i<matriz->linhas;i++)
            for (j=0; j<matriz->colunas;j++)
                matriz->dados[i][j] = 0;
    }
}

Matriz* criaMatriz(int linhas, int colunas){
    Matriz* matriz;
    matriz = (Matriz*) malloc (sizeof(Matriz));
    if (matriz != NULL){
        if (linhas <= 0 || colunas <= 0){
            printf ("Valores invalidos!");
            return NULL;
        }
        int i;
        matriz->linhas = linhas;
        matriz->colunas = colunas;
        matriz->dados = (int**) malloc (linhas * sizeof(int*));
        for (i=0; i < linhas; i++)
            matriz->dados[i] = (int*) malloc (colunas * sizeof(int));
        zeraMatriz(matriz);
    }
}

void destroiMatriz(Matriz* matriz){
    if (matriz != NULL){
        int i;
        for (i=0; i < matriz->linhas; i++)
            free(matriz->dados[i]);
        free(matriz->dados);
        free(matriz);
    }
}

int insereElemento(Matriz* matriz, int elemento, int linha, int coluna){
    if (matriz == NULL)
        return 0;
    if (linha < 0 || coluna < 0 || linha >= matriz->linhas || coluna >= matriz->colunas){
        printf ("Valores invalidos, elemento nao foi inserido.\n");
        return 0;
    }
    matriz->dados[linha][coluna] = elemento;
    return 1;
}
```

```c

int consultaElemento(Matriz* matriz, int *elemento, int linha, int coluna){
    if (matriz == NULL)
        return 0;
    if (linha < 0 || coluna < 0 || linha >= matriz->linhas || coluna >= matriz->colunas){
        printf ("Valores invalidos, elemento nao existe na matriz!\n");
        return 0;
    }
    *elemento = matriz->dados[linha][coluna];
    return 1;
}

void imprime (Matriz* matriz){
    if (matriz == NULL)
        return;
    int i, j;
    printf ("Matriz %d x %d: \n", matriz->linhas, matriz->colunas);
    for (i=0; i<matriz->linhas; i++){
        for (j=0; j<matriz->colunas; j++)
            printf ("%d ", matriz->dados[i][j]);
        printf ("\n");
    }
    printf ("\n");
}

Matriz* criaTransposta (Matriz* matriz){
    if (matriz == NULL)
        return NULL;
    Matriz* transposta = criaMatriz(matriz->colunas, matriz->linhas);
    int i, j;
    for (i=0; i< matriz->linhas; i++){
        for (j=0; j<matriz->colunas; j++)
            transposta->dados[j][i] = matriz->dados[i][j];
    }
    return transposta;
}

int e_matrizQuadrada(Matriz *matriz){
    if (matriz == NULL)
        return 0;
    return (matriz->linhas == matriz->colunas);
}

int e_Simetrica(Matriz* matriz){
    if (matriz == NULL)
        return 0;
    if (!e_matrizQuadrada(matriz)){
        printf ("Matriz nao quadrada!\n");
        return 0;
    }
    int i, j;
    for (i=0; i<matriz->linhas; i++){
        for (j=0; j<matriz->colunas; j++)
            if (matriz->dados[i][j] != matriz->dados[j][i])
                return 0;
    }
    return 1;
}

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

exercicio1.c

```c
#include <stdio.h>
#include <stdlib.h>
#include "matriz_sequencial_estatica.h"
//#include "matriz_sequencial_dinamica.h"
//Esse arquivo funciona tanto para a Matriz Sequencial Estática, quanto para a Matriz Sequencial Dinâmica, basta escolher qual biblioteca importar.

int main() {
    Matriz *matriz = NULL;
    int option, linhas, colunas, elemento, valor;
    do{
        printf("1. Criar Matriz\n");
        printf("2. Destruir Matriz\n");
        printf("3. Inserir Elemento\n");
        printf("4. Consultar Elemento\n");
        printf("5. Imprimir Matriz\n");
        printf("6. Criar Transposta da Matriz\n");
        printf("7. Verificar se a matriz e quadrada\n");
        printf("8. Verificar se a matriz e simetrica\n");
        printf("9. Criar Matriz Triangular Superior\n");
        printf("10. Criar Matriz Triangular Inferior\n");
        printf("11. Criar Matriz Diagonal\n");
        printf("12. Sair\n");
        printf("Escolha uma opcao: ");
        scanf("%d", &option);
        switch(option){
            case 1:
                printf("Informe o numero de linhas: ");
                scanf("%d", &linhas);
                printf("Informe o numero de colunas: ");
                scanf("%d", &colunas);
                matriz = criaMatriz(linhas, colunas);
                if(matriz == NULL)
                    printf("Falha na criacao da matriz!\n");
                else
                    printf("Matriz criada!\n");
                break;
            case 2:
                destroiMatriz(matriz);
                matriz = NULL;
                printf("Matriz destruida!\n");
                break;
            case 3:
                if(matriz == NULL){
                    printf("Matriz nao foi criada!\n");
                }else{
                    printf("Informe o elemento: ");
                    scanf("%d", &elemento);
                    printf("Informe a linha e coluna (linha coluna): ");
                    scanf("%d %d", &linhas, &colunas);
                    if(insereElemento(matriz, elemento, linhas, colunas))
                        printf("Elemento inserido!\n");
                    else
                        printf("Nao foi possivel inserir o elemento!\n");
                }
                break;
            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{
```

```c
                        printf("Matriz triangular superior:\n");
                        imprime(triangularSuperior);
                        destroiMatriz(triangularSuperior);
                    }
                }
                break;
            case 10:
                if(matriz == NULL){
                    printf("Erro: Matriz nao foi criada!\n");
                }else{
                    Matriz *triangularInferior = criaTriangularInferior(matriz);
                    if(triangularInferior == NULL){
                        printf("Erro: Falha na criacao da matriz triangular inferior!\n");
                    }else{
                        printf("Matriz triangular inferior:\n");
                        imprime(triangularInferior);
                        destroiMatriz(triangularInferior);
                    }
                }
                break;
            case 11:
                if(matriz == NULL){
                    printf("Matriz nao foi criada!\n");
                }else{
                    Matriz *diagonal = criaDiagonal(matriz);
                    if(diagonal == NULL){
                        printf("Nao foi possivel criar a matriz diagonal!\n");
                    }else{
                        printf("Matriz diagonal:\n");
                        imprime(diagonal);
                        destroiMatriz(diagonal);
                    }
                }
                break;
            case 12:
                printf("Saindo.\n");
                break;
            default:
                printf("Opcao invalida!\n");
        }
    }while(option != 12);
    return 0;
}
```

```
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: 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: 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: 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: 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: 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: 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: 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: 11
Matriz diagonal:
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: 12
Saindo.
```

matriz_de_faixa.h

```c
#ifndef MF_H
#define MF_H
#include <stdio.h>
#include <stdlib.h>

typedef struct{
    int *diagonal;
    int *superior;
    int *inferior;
    int tamanho;
}MatrizFaixa;

void zeraMatriz(MatrizFaixa *matrizFaixa) {
    if(matrizFaixa == NULL)
        return;
    for(int i = 0; i < matrizFaixa->tamanho; i++)
        matrizFaixa->diagonal[i] = 0;
    for(int i = 0; i < matrizFaixa->tamanho - 1; i++)
        matrizFaixa->superior[i] = 0;
    for(int i = 0; i < matrizFaixa->tamanho - 1; i++)
        matrizFaixa->inferior[i] = 0;
}

MatrizFaixa* criaMatriz(int t){
    MatrizFaixa *matrizFaixa = (MatrizFaixa*) malloc(sizeof(MatrizFaixa));
    if(matrizFaixa != NULL){
        if(t <= 1)
            return NULL;
        matrizFaixa->tamanho = t;
        matrizFaixa->diagonal = (int*) malloc(t * sizeof(int));
        matrizFaixa->superior = (int*) malloc((t-1) * sizeof(int));
        matrizFaixa->inferior = (int*) malloc((t-1) * sizeof(int));
        if(matrizFaixa->diagonal == NULL || matrizFaixa->superior == NULL || matrizFaixa->
inferior == NULL){
            return NULL;
        }
        zeraMatriz(matrizFaixa);
    }
    return matrizFaixa;
}

void destroiMatriz(MatrizFaixa *matrizFaixa){
    if(matrizFaixa != NULL){
        free(matrizFaixa->diagonal);
        free(matrizFaixa->superior);
        free(matrizFaixa->inferior);
        free(matrizFaixa);
    }
}

int insereElemento(MatrizFaixa *matrizFaixa, int elemento, int i, int j){
    if(matrizFaixa == NULL)
        return -1;
    if(i < 0 || j < 0 || i >= matrizFaixa->tamanho || j >= matrizFaixa->tamanho)
        return 0;
    if(i == j)
        matrizFaixa->diagonal[i] = elemento;
```

```
    else if(i + 1 == j)
        matrizFaixa->superior[i] = elemento;
    else if(i == j + 1)
        matrizFaixa->inferior[j] = elemento;
    else
        return 0;
    return 1;
}

int consultaElemento(MatrizFaixa *matrizFaixa, int i, int j){
    if(matrizFaixa == NULL)
        return 0;
    if(i < 0 || j < 0 || i >= matrizFaixa->tamanho || j >= matrizFaixa->tamanho)
        return 0;
    if(i == j)
        return matrizFaixa->diagonal[i];
    else if(i + 1 == j)
        return matrizFaixa->superior[i];
    else if(i == j + 1)
        return matrizFaixa->inferior[j];
    else
        return 0;
}

void imprimeFaixaVetores(MatrizFaixa *matrizFaixa){
    if(matrizFaixa == NULL)
        return;
    int i;
    printf("Matriz Faixa, Tam: %d x %d:\n", matrizFaixa->tamanho, matrizFaixa->tamanho);
    printf("Diagonal = [");
    for(i = 0; i < matrizFaixa->tamanho; i++)
        printf("%d ", matrizFaixa->diagonal[i]) ;
    printf("]\n");
    printf("Superior = [");
    for (i = 0; i < matrizFaixa->tamanho - 1; i++)
        printf("%d ", matrizFaixa->superior[i]);
    printf("]\n");
    printf("Inferior = [");
    for(i = 0; i < matrizFaixa->tamanho - 1; i++)
        printf("%d ", matrizFaixa->inferior[i]);
    printf("]\n\n");
}

void imprimeFaixa(MatrizFaixa *matrizFaixa){
    if(matrizFaixa == NULL)
        return;
    imprimeFaixaVetores(matrizFaixa);
    printf("Matriz original: \n");
    for(int i = 0; i < matrizFaixa->tamanho; i++){
        for(int j = 0; j < matrizFaixa->tamanho; j++){
            printf("%d\t", consultaElemento(matrizFaixa, i, j));
        }
        printf("\n");
    }
    printf("\n");
}

#endif  //MF_H
```

exercicio2-1.c

```c
#include <stdio.h>
#include <stdlib.h>
#include "matriz_de_faixa.h"

int main() {
    MatrizFaixa *matriz = NULL;
    int tamanho, option;
    do{
        printf("1. Criar matriz faixa\n");
        printf("2. Inserir Elemento.\n");
        printf("3. Consultar Elemento.\n");
        printf("4. Imprimir Matriz Faixa\n");
        printf("5. Destruir Matriz Faixa\n");
        printf("6. Sair\n");
        printf("Escolha uma opcao: ");
        scanf("%d", &option);
        switch(option){
            case 1:
                if(matriz != NULL){
                    printf("A matriz faixa ja foi criada!\n");
                }else{
                    printf("Informe o tamanho da matriz faixa: ");
                    scanf("%d", &tamanho);
                    matriz = criaMatriz(tamanho);
                    if(matriz == NULL){
                        printf("Nao foi possivel criar a matriz faixa!\n");
                    }else{
                        printf("Matriz faixa criada com sucesso!\n");
                    }
                }
                break;
            case 2:
                if(matriz == NULL){
                    printf("Matriz nao foi criada.\n");
                }else{
                    int elemento, i, j;
                    printf("Informe o elemento a ser inserido: ");
                    scanf("%d", &elemento);
                    printf("Informe a linha e a coluna: ");
                    scanf("%d %d", &i, &j);
                    int resultado = insereElemento(matriz, elemento, i, j);
                    if(resultado == -1){
                        printf("Matriz faixa nao foi criada!\n");
                    }else if(resultado == 0){
                        printf("Falha na insercao do elemento!\n");
                    }else{
                        printf("Elemento inserido.\n");
                    }
                }
                break;
            case 3:
                if(matriz == NULL){
                    printf("Erro: Matriz faixa nao foi criada!\n");
                }else{
                    int linha, coluna;
                    printf("Informe a linha e a coluna: ");
                    scanf("%d %d", &linha, &coluna);
```

```c
                    int valor = consultaElemento(matriz, linha, coluna);
                    if(valor == 0){
                        printf("Falha ao procurar elemento!\n");
                    }else{
                        printf("Elemento consultado: %d\n", valor);
                    }
                }
                break;
            case 4:
                if(matriz == NULL){
                    printf("Matriz faixa nao foi criada!\n");
                }else{
                    imprimeFaixa(matriz);
                }
                break;
            case 5:
                if(matriz == NULL){
                    printf("Matriz faixa nao foi criada!\n");
                }else{
                    destroiMatriz(matriz);
                    matriz = NULL;
                    printf("Matriz faixa destruida!\n");
                }
                break;
            case 6:
                printf("Saindo.\n");
                break;
            default:
                printf("Opcao invalida!\n");
        }
    }while(option != 6);
    return 0;
}
```

```
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

```c
#ifndef CSR_H
#define CSR_H
#include <stdio.h>
#include <stdlib.h>

typedef struct{
    int *A;  //Valores
    int *IA;
    int *JA;
    int lin, col, QNN, QI;
}MEsparsaCSR;

MEsparsaCSR* criaMatrizEsparsa(int l, int c, int qnn){
    MEsparsaCSR *ms;
    ms = (MEsparsaCSR*) malloc (sizeof(MEsparsaCSR));
    if(ms != NULL){
        if(l <= 0 || c <= 0 || qnn < 0){
            printf("Valores invalidos, matriz nao criada!\n");
            return NULL;
        }
        ms->lin = l; ms->col = c;
        ms->QI = 0; ms->QNN = qnn;
        ms->A = ms->IA = ms->JA = NULL;
        if(qnn != 0){
            ms->A = (int*) malloc (qnn*sizeof(int));
            ms->JA = (int*) malloc (qnn*sizeof(int));
            if(ms->A == NULL || ms->JA == NULL) return NULL;
        }
        ms->IA = (int*) malloc ((ms->lin+1)*sizeof(int));
        if(ms->IA == NULL) return NULL;
        int i; for(i=0; i<l+1; i++) ms->IA[i] = 0;
    }
    return ms;
}

void destroiMatrizEsparsa(MEsparsaCSR *ms){
    if(ms != NULL){
        free(ms->A);
        free(ms->IA);
        free(ms->JA);
        free(ms);
    }
}

int* meuRealloc(int* v, int tam){
    int* aux = (int*) malloc ((tam+1)*sizeof(int));
    if(aux != NULL){
        if(v != NULL){
            int i;
            for(i=0; i<tam; i++)
                aux[i] = v[i];
            free(v);
        }
    }
    return aux;
}

```

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

```c
#include <stdio.h>
#include <stdlib.h>
#include "matriz_csr.h"

int main(){
    MEsparsaCSR* esparsa = NULL;

    int option, linhas, colunas, elemento;
    do{
        printf ("1- Criar Matriz Esparsa.\n");
        printf ("2- Destruir Matriz Esparsa.\n");
        printf ("3- Inserir Elemento.\n");
        printf ("4- Remover Elemento.\n");
        printf ("5- Consultar Elemento.\n");
        printf ("6- Imprimir Matriz Esparsa.\n");
        printf ("7- Sair.\n");
        printf("Escolha uma opcao: ");
        scanf("%d", &option);
        switch (option){
            case 1:
                if (esparsa != NULL){
                    destroiMatrizEsparsa(esparsa);
                    printf ("Matriz apagada!");
                }
                printf ("Informe as dimensoes\n");
                printf ("Linhas e Colunas: ");
                scanf ("%d %d", &linhas, &colunas);
                esparsa = criaMatrizEsparsa(linhas, colunas, 0);
                break;
            case 2:
                destroiMatrizEsparsa(esparsa);
                printf ("Matriz Destruida.\n");
                break;
            case 3:
                printf ("Elemento a ser inserido: ");
                scanf ("%d", &elemento);
                printf ("Informe a linha e a coluna onde sera inserido o elemento: ");
                scanf ("%d %d", &linhas, &colunas);
                if (insereElemEsparsa(esparsa, elemento, linhas, colunas))
                    printf ("Elemento %d inserido na posicao (%d, %d)\n", elemento, linhas,
colunas);
                else
                    printf ("Nao foi possivel inserir o elemento\n");
                break;
            case 4:
                printf ("Informe a linha e a coluna do elemento a ser removido: ");
                scanf ("%d %d", &linhas, &colunas);
                if (removeElemEsparsa(esparsa, linhas, colunas))
                    printf ("Removeu o elemento %d da posicao (%d, %d)\n", elemento,
linhas, colunas);
                else
                    printf ("Nao foi possivel remover.\n");
                break;
            case 5:
                printf ("Posicao a ser consultada:\n");
                printf ("Informe a linha e a coluna: \n");
                scanf ("%d %d", &linhas, &colunas);
```

```c
                if (elemento = consultaElemEsparsa(esparsa, linhas, colunas))
                    printf ("Elemento: %d\n", elemento);
                else
                    printf ("Nao foi possivel consultar nessa posicao da matriz.\n");
                break;
            case 6:
                imprimeEsparsaVetores(esparsa);
                break;
            case 7:
                printf ("Saindo.\n");
                break;
            default:
                printf ("Opcao invalida!\n");
                break;
        }
    } while(option != 7);
    return 0;
}
```

```
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.
```