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()

```c
#include <stdio.h>
#include <stdbool.h>
#include "matriz.h"

int main(){
    Matriz *M, *TS, *TI, *D, *T;
    int elemento, linha, coluna, escolha, inicio, fim;

    printf("-<X Menu X>-\n");
    printf("1 - Criar Matriz\n");
    printf("2 - Zerar Matriz\n");
    printf("3 - Preencher Aleatoriamente\n");
    printf("4 - Inserir Elemento\n");
    printf("5 - Consultar Elemento\n");
    printf("6 - É Matriz quadrada?\n");
    printf("7 - Criar Triangulo Sup\n");
    printf("8 - Criar Triangulo Inf\n");
    printf("9 - Criar Diagonal\n");
    printf("10 - Criar Transposta\n");
    printf("11 - É Simetrica?\n");
    printf("12 - Imprimir Matriz\n");
    printf("13 - Sair\n");

    while(true){
        printf("Escolha: ");
        scanf("%d", &escolha);

        if(escolha == 1){
            printf("-> Insira a quantidade de linha: ");
            scanf("%d",&linha);
            printf("-> Insira a quantidade de coluna: ");
            scanf("%d",&coluna);
            M = criaMatriz(linha, coluna);
            printf("-> Criado com sucesso!\n");
        }else if(escolha == 2){
            zeraMatriz(M);
            printf("-> Matriz Zerada com sucesso!\n");
        }else if(escolha == 3){
            printf("-> Insira um valor inicial: ");
            scanf("%d",&inicio);
            printf("-> Insira um valor final: ");
            scanf("%d",&fim);
            preencheAleatorio(M, inicio, fim);
            printf("-> Matriz preenchida aleatoriamente!\n");
        }else if(escolha == 4){
            printf("-> Elemento a ser inserido: ");
            scanf("%d",&elemento);
```

```
            printf("-> Insira a linha: ");
            scanf("%d",&linha);
            printf("-> Insira a coluna: ");
            scanf("%d",&coluna);
            insereElem(M, elemento, linha, coluna);
            printf("-> Inserido com sucesso!\n");
        }else if(escolha == 5){
            printf("-> Insira a linha: ");
            scanf("%d",&linha);
            printf("-> Insira a coluna: ");
            scanf("%d",&coluna);
            consultaElem(M, &elemento, linha, coluna);
            printf("-> Elemento consultado: %d\n", elemento);
        }else if(escolha == 6){
            if(e_matrizQuadrada(M)){
                printf("-> Matriz Quadrada!\n");
            }else{
                printf("-> Matriz não é Quadrada!\n");
            }
        }else if(escolha == 7){
            TS = criaTriangularSup(M);
            imprime(TS);
        }else if(escolha == 8){
            TI = criaTriangularInf(M);
            imprime(TI);
        }else if(escolha == 9){
            D = criaDiagonal(M);
            imprime(D);
        }else if(escolha == 10){
            T = criaTransposta(M);
            imprime(T);
        }else if(escolha == 11){
            if(e_Simetrica(M)){
                printf("-> Matriz Simétrica!\n");
            }else{
                printf("-> Matriz não Simétrica!\n");
            }
        }else if(escolha == 12){
            imprime(M);
        }else if(escolha == 13){
            destroiMatriz(M);
            printf("-> Finalizando!\n");
            break;
        }else{
            printf("-> Número inválido!\n");
        }
    }
    return 0;
}
```

matriz.c

atividade1.1 > C matriz.c > ...

```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include "matriz.h"

void zeraMatriz(Matriz* mat){
    int i, j;
    for(i=0; i<mat->lin; i++)
        for(j=0; j<mat->col; j++)
            mat->dados[i][j] = 0;
}

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

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

int preencheAleatorio(Matriz* mat, int ini, int fim){
    if(mat == NULL) return 0;
    srand(time(NULL));
    int i, j;
    for(i=0; i<mat->lin; i++)
        for(j=0; j<mat->col; j++)
            mat->dados[i][j] = ini + rand() % (fim-ini + 1);
    return 1;
}

int insereElem(Matriz* mat, int elem, int l, int c){
    if(mat == NULL) return 0;
    if(l < 0 || c < 0 || l >= mat->lin || c >= mat->col){
        printf("Valores invalidos, elem nao inserido!\n");
        return 0;
```

```c
    }
    mat->dados[l][c] = elem;
    return 1;
}

int consultaElem(Matriz* mat, int *p, int l, int c){
    if(mat == NULL) return 0;
    if(l < 0 || c < 0 || l >= mat->lin || c >= mat->col){
        printf("Valores invalidos, elem nao existe!\n");
        return 0;
    }
    *p = mat->dados[l][c];
    return 1;
}

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

int e_matrizQuadrada(Matriz *mat){
    if(mat == NULL) return 0;
    return (mat->lin == mat->col);
}

Matriz* criaTriangularSup(Matriz* mat){
    if(mat == NULL) return NULL;
    if(!e_matrizQuadrada(mat)){
        printf("Matriz nao Quadrada!\n");
        return NULL;
    }
    int i, j;
    Matriz* ts = criaMatriz(mat->lin, mat->col);
    for(i=0; i<mat->lin; i++)
        for(j=0; j<mat->col; j++)
            if(i <= j)
                ts->dados[i][j] = mat->dados[i][j];
    return ts;
```

```c
}

Matriz* criaTriangularInf(Matriz* mat){
    if(mat == NULL) return NULL;
    if(!e_matrizQuadrada(mat)){
        printf("Matriz nao Quadrada!\n");
        return NULL;
    }
    int i, j;
    Matriz* ti = criaMatriz(mat->lin, mat->col);
    for(i=0; i<mat->lin; i++)
        for(j=0; j<mat->col; j++)
            if(i >= j)
                ti->dados[i][j] = mat->dados[i][j];
    return ti;
}

Matriz* criaDiagonal(Matriz* mat){
    if(mat == NULL) return NULL;
    if(!e_matrizQuadrada(mat)){
        printf("Matriz nao Quadrada!\n");
        return NULL;
    }
    int i, j;
    Matriz* d = criaMatriz(mat->lin, mat->col);
    for(i=0; i<mat->lin; i++)
        d->dados[i][i] = mat->dados[i][i];
    return d;
}

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

Matriz* criaTransposta(Matriz* mat){
    if(mat == NULL) return NULL;
    Matriz* t = criaMatriz(mat->col, mat->lin);
    int i, j;
    for(i=0; i<mat->lin; i++)
        for(j=0; j<mat->col; j++)
            t->dados[j][i] = mat->dados[i][j];
    return t;
}

```

matriz.h

atividade1.1 > C matriz.h > imprime(Matriz *)

```c
#ifndef MATRIZ_H
#define MATRIZ_H

#define MAX 100

typedef struct{
    int dados[MAX][MAX];
    int lin, col;
}Matriz;

void zeraMatriz(Matriz* mat);

Matriz* criaMatriz(int l, int c);

void destroiMatriz(Matriz* mat);

int preencheAleatorio(Matriz* mat, int ini, int fim);

int insereElem(Matriz* mat, int elem, int l, int c);

int consultaElem(Matriz* mat, int *p, int l, int c);

void imprime(Matriz* mat);

int e_matrizQuadrada(Matriz *mat);

Matriz* criaTriangularSup(Matriz* mat);

Matriz* criaTriangularInf(Matriz* mat);

Matriz* criaDiagonal(Matriz* mat);

int e_Simetrica(Matriz* mat);

Matriz* criaTransposta(Matriz* mat);

#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 Simetrica?
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()

```c
#include <stdio.h>
#include <stdbool.h>
#include "matrizDin.h"

int main(){
    Matriz *M, *TS, *TI, *D, *T;
    int elemento, linha, coluna, escolha, inicio, fim;

    printf("-<X Menu X>-\n");
    printf("1 - Criar Matriz\n");
    printf("2 - Zerar Matriz\n");
    printf("3 - Preencher Aleatoriamente\n");
    printf("4 - Inserir Elemento\n");
    printf("5 - Consultar Elemento\n");
    printf("6 - É Matriz quadrada?\n");
    printf("7 - Criar Triangulo Sup\n");
    printf("8 - Criar Triangulo Inf\n");
    printf("9 - Criar Diagonal\n");
    printf("10 - Criar Transposta\n");
    printf("11 - É Simetrica?\n");
    printf("12 - Imprimir Matriz\n");
    printf("13 - Sair\n");

    while(true){
        printf("Escolha: ");
        scanf("%d", &escolha);

        if(escolha == 1){
            printf("-> Insira a quantidade de linha: ");
            scanf("%d",&linha);
            printf("-> Insira a quantidade de coluna: ");
            scanf("%d",&coluna);
            M = criaMatriz(linha, coluna);
            printf("-> Criado com sucesso!\n");
        }else if(escolha == 2){
            zeraMatriz(M);
            printf("-> Matriz Zerada com sucesso!\n");
        }else if(escolha == 3){
            printf("-> Insira um valor inicial: ");
            scanf("%d",&inicio);
            printf("-> Insira um valor final: ");
            scanf("%d",&fim);
            preencheAleatorio(M, inicio, fim);
            printf("-> Matriz preenchida aleatoriamente!\n");
        }else if(escolha == 4){
            printf("-> Elemento a ser inserido: ");
            scanf("%d",&elemento);
```

```
            printf("-> Insira a linha: ");
            scanf("%d",&linha);
            printf("-> Insira a coluna: ");
            scanf("%d",&coluna);
            insereElem(M, elemento, linha, coluna);
            printf("-> Inserido com sucesso!\n");
        }else if(escolha == 5){
            printf("-> Insira a linha: ");
            scanf("%d",&linha);
            printf("-> Insira a coluna: ");
            scanf("%d",&coluna);
            consultaElem(M, &elemento, linha, coluna);
            printf("-> Elemento consultado: %d\n", elemento);
        }else if(escolha == 6){
            if(e_matrizQuadrada(M)){
                printf("-> Matriz Quadrada!\n");
            }else{
                printf("-> Matriz não é Quadrada!\n");
            }
        }else if(escolha == 7){
            TS = criaTriangularSup(M);
            imprime(TS);
        }else if(escolha == 8){
            TI = criaTriangularInf(M);
            imprime(TI);
        }else if(escolha == 9){
            D = criaDiagonal(M);
            imprime(D);
        }else if(escolha == 10){
            T = criaTransposta(M);
            imprime(T);
        }else if(escolha == 11){
            if(e_Simetrica(M)){
                printf("-> Matriz Simétrica!\n");
            }else{
                printf("-> Matriz não Simétrica!\n");
            }
        }else if(escolha == 12){
            imprime(M);
        }else if(escolha == 13){
            destroiMatriz(M);
            printf("-> Finalizando!\n");
            break;
        }else{
            printf("-> Número inválido!\n");
        }
    }
    return 0;
}
```

matrizDin.c

atividade1.2 > C matrizDin.c > ...

```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include "matrizDin.h"

void zeraMatriz(Matriz* mat){
    int i, j;
    for(i=0; i<mat->lin; i++)
        for(j=0; j<mat->col; j++)
            mat->dados[i][j] = 0;
}

Matriz* criaMatriz(int l, int c){
    Matriz* mat;
    mat = (Matriz*) malloc (sizeof(Matriz));
    if(mat != NULL){
        if(l <= 0 || c <= 0){
            printf("Valores invalidos, matriz nao criada!\n");
            return NULL;
        }
        int i;
        mat->lin = l;
        mat->col = c;
        mat->dados = (int**) malloc (l*sizeof(int*));
        for(i=0; i<l; i++)
            mat->dados[i] = (int*) malloc (c*sizeof(int));
        zeraMatriz(mat);
    }
    return mat;
}

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

int preencheAleatorio(Matriz* mat, int ini, int fim){
    if(mat == NULL) return 0;
    srand(time(NULL));
    int i, j;
    for(i=0; i<mat->lin; i++)
        for(j=0; j<mat->col; j++)
            mat->dados[i][j] = ini + rand() % (fim-ini + 1);
    return 1;
}

int insereElem(Matriz* mat, int elem, int l, int c){
    if(mat == NULL) return 0;
    if(l < 0 || c < 0 || l > mat->lin || c > mat->col){
        printf("Valores invalidos, elem nao inserido!\n");
        return 0;
    }
    mat->dados[l][c] = elem;
    return 1;
}

int consultaElem(Matriz* mat, int *p, int l, int c){
    if(mat == NULL) return 0;
    if(l < 0 || c < 0 || l > mat->lin || c > mat->col){
        printf("Valores invalidos, elem nao existe!\n");
        return 0;
    }
    *p = mat->dados[l][c];
    return 1;
}

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

int e_matrizQuadrada(Matriz *mat){
    if(mat == NULL) return 0;
    return (mat->lin == mat->col);
}

Matriz* criaTriangularSup(Matriz* mat){
    if(mat == NULL) return NULL;
    if(!e_matrizQuadrada(mat)){
        printf("Matriz nao Quadrada!\n");
```

```
        return NULL;
    }
    int i, j;
    Matriz* ts = criaMatriz(mat->lin, mat->col);
    for(i=0; i<mat->lin; i++)
        for(j=0; j<mat->col; j++)
            if(i <= j)
                ts->dados[i][j] = mat->dados[i][j];
    return ts;
}

Matriz* criaTriangularInf(Matriz* mat){
    if(mat == NULL) return NULL;
    if(!e_matrizQuadrada(mat)){
        printf("Matriz nao Quadrada!\n");
        return NULL;
    }
    int i, j;
    Matriz* ti = criaMatriz(mat->lin, mat->col);
    for(i=0; i<mat->lin; i++)
        for(j=0; j<mat->col; j++)
            if(i >= j)
                ti->dados[i][j] = mat->dados[i][j];
    return ti;
}

Matriz* criaDiagonal(Matriz* mat){
    if(mat == NULL) return NULL;
    if(!e_matrizQuadrada(mat)){
        printf("Matriz nao Quadrada!\n");
        return NULL;
    }
    int i, j;
    Matriz* d = criaMatriz(mat->lin, mat->col);
    for(i=0; i<mat->lin; i++)
        d->dados[i][i] = mat->dados[i][i];
    return d;
}

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

Matriz* criaTransposta(Matriz* mat){
    if(mat == NULL) return NULL;
    Matriz* t = criaMatriz(mat->col, mat->lin);
    int i, j;
    for(i=0; i<mat->lin; i++)
        for(j=0; j<mat->col; j++)
            t->dados[j][i] = mat->dados[i][j];
    return t;
}
```

matrizDin.h

atividade1.2 > matrizDin.h > e_matrizQuadrada(Matriz *)

```c
#ifndef MATRIZDIN_H
#define MATRIZDIN_H

typedef struct{
    int **dados;
    int lin, col;
}Matriz;

void zeraMatriz(Matriz* mat);

Matriz* criaMatriz(int l, int c);

void destroiMatriz(Matriz* mat);

int preencheAleatorio(Matriz* mat, int ini, int fim);

int insereElem(Matriz* mat, int elem, int l, int c);

int consultaElem(Matriz* mat, int *p, int l, int c);

void imprime(Matriz* mat);

int e_matrizQuadrada(Matriz *mat);

Matriz* criaTriangularSup(Matriz* mat);

Matriz* criaTriangularInf(Matriz* mat);

Matriz* criaDiagonal(Matriz* mat);

int e_Simetrica(Matriz* mat);

Matriz* criaTransposta(Matriz* mat);

#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
-<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 Simetrica?
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

atividade2.1 > C main.c > main()

```c
#include <stdio.h>
#include <stdbool.h>
#include "mFaixa.h"

int main(){
    MFaixa *M;
    int elemento, linha, coluna, valor, escolha, inicio, fim;

    printf("-<X Menu X>-\n");
    printf("1 - Criar Matriz\n");
    printf("2 - Zerar Matriz\n");
    printf("3 - Preencher Aleatoriamente\n");
    printf("4 - Inserir Elemento\n");
    printf("5 - Consultar Elemento\n");
    printf("6 - Imprimir Matriz de Faixa(Vetores)\n");
    printf("7 - Imprimir Matriz de Faixa \n");
    printf("8 - Sair\n");

    while(true){
        printf("Escolha: ");
        scanf("%d", &escolha);

        if(escolha == 1){
            printf("-> Insira o valor para a matriz: ");
            scanf("%d",&valor);
            M = criaMatriz(valor);
            printf("-> Criado com sucesso!\n");
        }else if(escolha == 2){
            zeraMatriz(M);
            printf("-> Matriz Zerada com sucesso!\n");
        }else if(escolha == 3){
            printf("-> Insira um valor inicial: ");
            scanf("%d",&inicio);
            printf("-> Insira um valor final: ");
            scanf("%d",&fim);
            preencheAleatorio(M, inicio, fim);
            printf("-> Matriz preenchida aleatoriamente!\n");
        }else if(escolha == 4){
            printf("-> Elemento a ser inserido: ");
            scanf("%d",&elemento);
            printf("-> Insira a linha: ");
            scanf("%d",&linha);
            printf("-> Insira a coluna: ");
            scanf("%d",&coluna);
            insereElem(M, elemento, linha, coluna);
            printf("-> Inserido com sucesso!\n");
        }else if(escolha == 5){
            printf("-> Insira a linha: ");
            scanf("%d",&linha);
            printf("-> Insira a coluna: ");
            scanf("%d",&coluna);
            printf("-> Elemento consultado: %d\n", consultaElem(M, linha, coluna));
        }else if(escolha == 6){
            imprimeFaixaVetores(M);
        }else if(escolha == 7){
            imprimeFaixa(M);
        }else if(escolha == 8){
            destroiMatriz(M);
            printf("-> Finalizando!\n");
            break;
        }else{
            printf("-> Número inválido!\n");
        }
    }
    return 0;
}
```

mFaixa.c

atividade2.1 > C mFaixa.c > ...

```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include "mFaixa.h"

void zeraMatriz(MFaixa* mf){
    int i;
    for(i=0; i<mf->tam; i++){
        mf->diagonal[i] = 0;
        if(i < mf->tam -1){
            mf->superior[i] = 0;
            mf->inferior[i] = 0;
        }
    }
}

MFaixa* criaMatriz(int t){
    MFaixa *mf;
    mf = (MFaixa*) malloc (sizeof(MFaixa));
    if(mf != NULL){
        if(t <= 1){
            printf("Dimensao deve ser > 1, matriz nao criada!");
            return NULL;
        }
        mf->tam = t;
        mf->diagonal = (int*) malloc (t*sizeof(int));
        mf->superior = (int*) malloc ((t-1)*sizeof(int));
        mf->inferior = (int*) malloc ((t-1)*sizeof(int));
        if(mf->diagonal == NULL || mf->superior == NULL || mf->inferior == NULL)
          return NULL;
        zeraMatriz(mf);
    }
    return mf;
}

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

int preencheAleatorio(MFaixa* mf, int ini, int fim){
    if(mf == NULL) return 0;
    srand(time(NULL));
```

```c
    int i;
    for(i=0; i<mf->tam; i++){
        mf->diagonal[i] = ini + rand() % (fim-ini + 1);
        if(i < mf->tam -1){
            mf->superior[i] = ini + rand() % (fim-ini + 1);
            mf->inferior[i] = ini + rand() % (fim-ini + 1);
        }
    }
    return 1;
}

int insereElem(MFaixa* mf, int elem, int i, int j){
  if(mf == NULL) return 0;
  if(i < 0 || j < 0 || i >= mf->tam || j >= mf->tam){
    printf("Valores invalidos, elem nao inserido!\n");
    return 0;
  }
  if(i == j) mf->diagonal[i] = elem;
  else if(i + 1 == j) mf->superior[i] = elem;
  else if(i == j + 1) mf->inferior[j] = elem;
  else{
    printf("Indices fora da faixa, elem nao inserido!\n");
    return 0;
  }
  return 1;
}

int consultaElem(MFaixa* mf, int i, int j){
  if(mf == NULL) return 0;
  if(i < 0 || j < 0 || i >= mf->tam || j >= mf->tam){
    printf("Valores invalidos, elem inexistente!\n");
    return 0;
  }
  if(i == j) return mf->diagonal[i];
  else if(i + 1 == j) return mf->superior[i];
  else if(i == j + 1) return mf->inferior[j];
  else return 0;
}

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

void imprimeFaixa(MFaixa* mf){
  if(mf == NULL) return;
  int i, j;
  imprimeFaixaVetores(mf);
  printf("Matriz Original:\n");
  for(i=0; i<mf->tam; i++){
    for(j=0; j<mf->tam; j++)
      printf("%d\t", consultaElem(mf, i, j));
    printf("\n");
  }
}
```

mFaixa.h

atividade2.1 > C mFaixa.h > ...

```c
#ifndef MFAIXA_H
#define MFAIXA_H

typedef struct{
    int *diagonal;
    int *superior;
    int *inferior;
    int tam;
}MFaixa;

void zeraMatriz(MFaixa* mf);

MFaixa* criaMatriz(int t);

void destroiMatriz(MFaixa* mf);

int preencheAleatorio(MFaixa* mf, int ini, int fim);

int insereElem(MFaixa* mf, int elem, int i, int j);

int consultaElem(MFaixa* mf, int i, int j);

void imprimeFaixaVetores(MFaixa* mf);

void imprimeFaixa(MFaixa* mf);

#endif
```

Console

```
messiasfcm@MessiasFCM:/mnt/c/Users/Messi/OneDrive/Área de Trabalho/lista7/atividade2.1$ gcc main.c mFaixa.c mFaixa.h -o tp3
messiasfcm@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!
messiasfcm@MessiasFCM:/mnt/c/Users/Messi/OneDrive/Área de Trabalho/lista7/atividade2.1$
```

2.2

main.c

atividade2.2 > C main.c > ...

```c
#include <stdio.h>
#include <stdbool.h>
#include "matrizDin.h"
#include "mEsparsaCSR.h"

int main(){
    Matriz *M, *TS, *TI, *D, *T;
    MEsparsaCSR *MS;
    int elemento, linha, coluna, escolha, inicio, fim, qnn;

    printf("-<X Menu Matriz Din X>-\n");
    printf("1 - Criar Matriz Din\n");
    printf("2 - Zerar Matriz Din\n");
    printf("3 - Preencher Aleatoriamente\n");
    printf("4 - Inserir Elemento\n");
    printf("5 - Consultar Elemento\n");
    printf("6 - É Matriz quadrada?\n");
    printf("7 - Criar Triangulo Sup\n");
    printf("8 - Criar Triangulo Inf\n");
    printf("9 - Criar Diagonal\n");
    printf("10 - Criar Transposta\n");
    printf("11 - É Simetrica?\n");
    printf("12 - Imprimir Matriz Din\n\n");
    // ------------------------------------------------------------------------
    printf("-<X Menu Matriz Esparsa X>-\n");
    printf("13 - Criar Matriz Esparsa\n");
    printf("14 - Inserir Elemento\n");
    printf("15 - Remover Elemento\n");
    printf("16 - Transformar em CSR\n");
    printf("17 - Consultar Elemento\n");
    printf("18 - Imprimir Matriz Esparsa(Vetores)\n");
    printf("19 - Imprimir Matriz Esparsa\n");
    printf("20 - Finalizar\n");

    while(true){
        printf("Escolha: ");
        scanf("%d", &escolha);

        if(escolha == 1){
            printf("-> Insira a quantidade de linha: ");
            scanf("%d",&linha);
            printf("-> Insira a quantidade de coluna: ");
            scanf("%d",&coluna);
            M = criaMatriz(linha, coluna);
            printf("-> Criado com sucesso!\n");
        }else if(escolha == 2){
            zeraMatriz(M);
```

```
            printf("-> Matriz Zerada com sucesso!\n");
        }else if(escolha == 3){
            printf("-> Insira um valor inicial: ");
            scanf("%d",&inicio);
            printf("-> Insira um valor final: ");
            scanf("%d",&fim);
            preencheAleatorio(M, inicio, fim);
            printf("-> Matriz preenchida aleatoriamente!\n");
        }else if(escolha == 4){
            printf("-> Elemento a ser inserido: ");
            scanf("%d",&elemento);
            printf("-> Insira a linha: ");
            scanf("%d",&linha);
            printf("-> Insira a coluna: ");
            scanf("%d",&coluna);
            insereElem(M, elemento, linha, coluna);
            printf("-> Inserido com sucesso!\n");
        }else if(escolha == 5){
            printf("-> Insira a linha: ");
            scanf("%d",&linha);
            printf("-> Insira a coluna: ");
            scanf("%d",&coluna);
            consultaElem(M, &elemento, linha, coluna);
            printf("-> Elemento consultado: %d\n", elemento);
        }else if(escolha == 6){
            if(e_matrizQuadrada(M)){
                printf("-> Matriz Quadrada!\n");
            }else{
                printf("-> Matriz não é Quadrada!\n");
            }
        }else if(escolha == 7){
            TS = criaTriangularSup(M);
            imprime(TS);
        }else if(escolha == 8){
            TI = criaTriangularInf(M);
            imprime(TI);
        }else if(escolha == 9){
            D = criaDiagonal(M);
            imprime(D);
        }else if(escolha == 10){
            T = criaTransposta(M);
            imprime(T);
        }else if(escolha == 11){
            if(e_Simetrica(M)){
                printf("-> Matriz Simétrica!\n");
            }else{
                printf("-> Matriz não Simétrica!\n");
            }
        }else if(escolha == 12){
            imprime(M);
        // ------------------------------------------------------------------------
        }else if(escolha == 13){
            printf("-> Insira a quantidade de linha: ");
            scanf("%d",&linha);
            printf("-> Insira a quantidade de coluna: ");
            scanf("%d",&coluna);
            printf("-> Insira a quantidade de não nulos: ");
            scanf("%d",&qnn);
            MS = criaMatrizEsparsa(linha, coluna, qnn);
        }else if(escolha == 14){
            printf("-> Insira o elemento: ");
            scanf("%d",&elemento);
            printf("-> Insira a linha: ");
            scanf("%d",&linha);
            printf("-> Insira a coluna: ");
            scanf("%d",&coluna);
            insereElemEsparsa(MS, elemento, linha, coluna);
        }else if(escolha == 15){
            printf("-> Insira a linha: ");
            scanf("%d",&linha);
            printf("-> Insira a coluna: ");
            scanf("%d",&coluna);
            removeElemEsparsa(MS, linha, coluna);
        }else if(escolha == 16){
            MS = transformarEmCSR(M);
            printf("-> Matriz tranformada com sucesso");
        }else if(escolha == 17){
            printf("-> Insira a linha: ");
            scanf("%d",&linha);
            printf("-> Insira a coluna: ");
            scanf("%d",&coluna);
            printf("-> Elemento consultado: %d\n", consultaElemEsparsa(MS, linha, coluna));
        }else if(escolha == 18){
            imprimeEsparsaVetores(MS);
        }else if(escolha == 19){
            imprimeEsparsa(MS);
        }else if(escolha == 20){
            destroiMatriz(M);
            destroiMatrizEsparsa(MS);
            printf("-> Finalizando!\n");
```

```
            break;
        }else{
            printf("-> Número inválido!\n");
        }
    }
    return 0;
}
```

matrizDin.h

atividade2.2 > C matrizDin.h > ...

```
/*---------------- File: MatrizDin.h  -----------------+
|Matriz Dinamica sequencial (malloc)                   |
|                                                      |
|                                                      |
| Implementado por Guilherme C. Pena em 29/09/2023     |
+------------------------------------------------------+ */

#ifndef MATRIZDIN_H
#define MATRIZDIN_H

#include <stdio.h>
#include <stdlib.h>
#include <time.h>

typedef struct{
    int **dados;
    int lin, col;
}Matriz;

void zeraMatriz(Matriz* mat){
    int i, j;
    for(i=0; i<mat->lin; i++)
        for(j=0; j<mat->col; j++)
            mat->dados[i][j] = 0;
}

Matriz* criaMatriz(int l, int c){
    Matriz* mat;
    mat = (Matriz*) malloc (sizeof(Matriz));
    if(mat != NULL){
        if(l <= 0 || c <= 0){
            printf("Valores invalidos, matriz nao criada!\n");
            return NULL;
        }
        int i;
        mat->lin = l;
        mat->col = c;
        mat->dados = (int**) malloc (l*sizeof(int*));
        for(i=0; i<l; i++)
            mat->dados[i] = (int*) malloc (c*sizeof(int));
        zeraMatriz(mat);
    }
    return mat;
}

void destroiMatriz(Matriz* mat){
    if(mat != NULL){
```

```c
        int i;
        for(i=0; i<mat->lin; i++)
            free(mat->dados[i]);
        free(mat->dados);
        free(mat);
    }
}

int preencheAleatorio(Matriz* mat, int ini, int fim){
    if(mat == NULL) return 0;
    srand(time(NULL));
    int i, j;
    for(i=0; i<mat->lin; i++)
        for(j=0; j<mat->col; j++)
            mat->dados[i][j] = ini + rand() % (fim-ini + 1);
    return 1;
}

int insereElem(Matriz* mat, int elem, int l, int c){
    if(mat == NULL) return 0;
    if(l < 0 || c < 0 || l >= mat->lin || c >= mat->col){
        printf("Valores invalidos, elem nao inserido!\n");
        return 0;
    }
    mat->dados[l][c] = elem;
    return 1;
}

int consultaElem(Matriz* mat, int *p, int l, int c){
    if(mat == NULL) return 0;
    if(l < 0 || c < 0 || l >= mat->lin || c >= mat->col){
        printf("Valores invalidos, elem nao existe!\n");
        return 0;
    }
    *p = mat->dados[l][c];
    return 1;
}

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

```c
        printf("\n");
    }
    printf("\n");
}

//Matrizes Quadradas e propriedades

int e_matrizQuadrada(Matriz *mat){
    if(mat == NULL) return 0;
    return (mat->lin == mat->col);
}

Matriz* criaTriangularSup(Matriz* mat){
    if(mat == NULL) return NULL;
    if(!e_matrizQuadrada(mat)){
        printf("Matriz nao Quadrada!\n");
        return NULL;
    }
    int i, j;
    Matriz* ts = criaMatriz(mat->lin, mat->col);
    for(i=0; i<mat->lin; i++)
        for(j=0; j<mat->col; j++)
            if(i <= j)
                ts->dados[i][j] = mat->dados[i][j];
    return ts;
}

Matriz* criaTriangularInf(Matriz* mat){
    if(mat == NULL) return NULL;
    if(!e_matrizQuadrada(mat)){
        printf("Matriz nao Quadrada!\n");
        return NULL;
    }
    int i, j;
    Matriz* ti = criaMatriz(mat->lin, mat->col);
    for(i=0; i<mat->lin; i++)
        for(j=0; j<mat->col; j++)
            if(i >= j)
                ti->dados[i][j] = mat->dados[i][j];
    return ti;
}

Matriz* criaDiagonal(Matriz* mat){
    if(mat == NULL) return NULL;
    if(!e_matrizQuadrada(mat)){
        printf("Matriz nao Quadrada!\n");
        return NULL;
    }
    int i, j;
    Matriz* d = criaMatriz(mat->lin, mat->col);
    for(i=0; i<mat->lin; i++)
        d->dados[i][i] = mat->dados[i][i];
    return d;
}

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

Matriz* criaTransposta(Matriz* mat){
    if(mat == NULL) return NULL;
    Matriz* t = criaMatriz(mat->col, mat->lin);
    int i, j;
    for(i=0; i<mat->lin; i++)
        for(j=0; j<mat->col; j++)
            t->dados[j][i] = mat->dados[i][j];
    return t;
}

#endif
```

mEsparsaCSR.h

atividade2.2 > C mEsparsaCSR.h > ...

```c
/*---------------- File: MEsparsaCSR.h  ----------------+
|Matriz Esparsa (malloc)                                |
|Compressed Sparse Row - CSR                            |
|                                                       |
| Implementado por Guilherme C. Pena em 09/10/2023      |
+-------------------------------------------------------+ */

#ifndef MESPARSACSR_H
#define MESPARSACSR_H

#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include "MatrizDin.h"

/*
Matriz Esparsa (Compressed Sparce Row - CSR)
3 vetores:
A -> tamanho QNN (Quantidade de nao nulos)
IA -> tamanho N+1 (QNN por linha)
    IA[0] = 0;
    IA[i] = IA[i-1] + QNN da linha (i-1)
JA -> tamanho QNN (indices de coluna de cada elemento em A)

Input : 10  20  0  0  0  0
        0  30  0  4  0  0
        0   0 50 60 70  0
        0   0  0  0  0 80

Output :  A = [10 20 30 4 50 60 70 80],
          IA = [0 2 4 7 8]
          JA = [0  1 1 3 2 3 4 5]
*/

typedef struct{
    int *A; //Valores
    int *IA;
    int *JA;
    int lin, col, QNN, QI;
}MEsparsaCSR;
//QNN - Quantidade de Nao Nulos
//QI - Quantidade de Inseridos

MEsparsaCSR* criaMatrizEsparsa(int l, int c, int qnn){
    MEsparsaCSR *ms;
    ms = (MEsparsaCSR*) malloc (sizeof(MEsparsaCSR));
    if(ms != NULL){
```

```c
        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;
}

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;
}

void imprimeEsparsaVetores(MEsparsaCSR* ms){
  if(ms == NULL) return;
  int i, j;
  printf("Matriz Esparsa, Tam: %d x %d:\n", ms->lin, ms->col);
  printf("%d elementos nao nulos.\n", ms->QNN);
  printf("A = [");
  for(i=0; i<ms->QNN; i++)
    printf("%d ", ms->A[i]);
  printf("]\n");
  printf("IA = [");
  for(i=0; i<ms->lin+1; i++)
    printf("%d ", ms->IA[i]);
  printf("]\n");
```

```c
  printf("JA = [");
  for(i=0; i<ms->QNN; i++)
    printf("%d ", ms->JA[i]);
  printf("]\n\n");
}

int insereElemEsparsa(MEsparsaCSR *ms, int elem, int i, int j) {
    if(ms == NULL) return 0;
    if(i < 0 || j < 0 || i >= ms->lin || j >= ms->col){
        printf("Valores invalidos, elem nao inserido!\n");
        return 0;
    }
    int k;
    int index = -1;
    int ini = ms->IA[i]; int fim = ms->IA[i+1];
    // Encontre a posição correta para inserir o valor
    for(k = ini; k<fim; k++)
        if (ms->JA[k] >= j) {
            index = k;
            break;
        }

    if (index == -1) { //NOVA INSERCAO
        if(ms->QI == ms->QNN){//Necessita REALLOC
            ms->A = meuRealloc(ms->A, ms->QNN);
            ms->JA = meuRealloc(ms->JA, ms->QNN);
            ms->QNN++;
        }
        //Move elementos para a nova insercao
        for(k = ms->QNN-1; k>=fim; k--){
            ms->A[k] = ms->A[k-1];
            ms->JA[k] = ms->JA[k-1];
        }
        ms->A[fim] = elem;
        ms->JA[fim] = j;
        ms->QI++;
        // Atualiza QNN acumulado
        for (int k = i+1; k<=ms->lin; k++)
            ms->IA[k]++;
    } else { //Atualiza um valor existente
        ms->A[index] = elem;
    }
    imprimeEsparsaVetores(ms);
    return 1;
}
```

```c

int removeElemEsparsa(MEsparsaCSR *ms, int i, int j) {
    if(ms == NULL) return 0;
    if(i < 0 || j < 0 || i >= ms->lin || j >= ms->col){
        printf("Valores invalidos, elem nao removido!\n");
        return 0;
    }

    int k;
    int index = -1;
    int ini = ms->IA[i]; int fim = ms->IA[i+1];
    // Encontre a posição do valor a ser removido
    for(k = ini; k<fim; k++)
        if (ms->JA[k] == j) {
            index = k;
            break;
        }

    if (index != -1) {
        // Move todos elementos uma posição para tras
        for (k = index; k < ms->QNN - 1; k++) {
            ms->A[k] = ms->A[k+1];
            ms->JA[k] = ms->JA[k+1];
        }
        ms->QNN--;
        ms->QI--;
        // Atualiza QNN acumulado
        for (int k = i+1; k<=ms->lin; k++)
            ms->IA[k]--;
    }else{
        printf("Elemento nao existente\n"); return 0;
    }
    imprimeEsparsaVetores(ms);
    return 1;
}


MEsparsaCSR* transformarEmCSR(Matriz* mat){
    MEsparsaCSR *ms = criaMatrizEsparsa(mat->lin, mat->col, 0);
    if(ms != NULL){
        if(mat == NULL){
            printf("Matriz de entrada inexistente!\n");
            return NULL;
        }

        int i, j;
        for(i=0; i<mat->lin; i++)
            for(j=0; j<mat->col; j++)
                if(mat->dados[i][j] != 0)
                    insereElemEsparsa(ms, mat->dados[i][j], i, j);
    }
    return ms;
}

int consultaElemEsparsa(MEsparsaCSR* ms, int i, int j){
  if(ms == NULL) return 0;
  if(i < 0 || j < 0 || i >= ms->lin || j >= ms->col){
    printf("Valores invalidos, elem inexistente!\n");
    return 0;
  }
  int k;
  for(k = ms->IA[i]; k<ms->IA[i+1]; k++)
    if(ms->JA[k] == j) return ms->A[k];
  return 0;
}

void imprimeEsparsa(MEsparsaCSR* ms){
  if(ms == NULL) return;
  int i, j;
  imprimeEsparsaVetores(ms);
  printf("Matriz Original:\n");
  for(i=0; i<ms->lin; i++){
    for(j=0; j<ms->col; j++)
      printf("%d\t", consultaElemEsparsa(ms, i, j));
    printf("\n");
  }
}

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

#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
-<X 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

-<X 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 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 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 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 3 ]
JA = [0 1 2 ]
```

```
JA = [0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 ]

Matriz Esparsa, Tam: 10 x 10:
99 elementos nao nulos.
A = [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 ]
IA = [0 10 20 30 40 50 60 70 80 90 99 ]
JA = [0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 ]

Matriz Esparsa, Tam: 10 x 10:
100 elementos nao nulos.
A = [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 ]
IA = [0 10 20 30 40 50 60 70 80 90 100 ]
JA = [0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 ]

-> Matriz tranformada com sucessoEscolha: 18
Matriz Esparsa, Tam: 10 x 10:
100 elementos nao nulos.
A = [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 ]
IA = [0 10 20 30 40 50 60 70 80 90 100 ]
JA = [0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 ]

Escolha: 19
Matriz Esparsa, Tam: 10 x 10:
100 elementos nao nulos.
A = [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 ]
IA = [0 10 20 30 40 50 60 70 80 90 100 ]
JA = [0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 ]

Matriz Original:
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: 20
-> Finalizando!
messiasfcm@MessiasFCM:/mnt/c/Users/Messi/OneDrive/Área de Trabalho/lista7/atividade2.2$
```