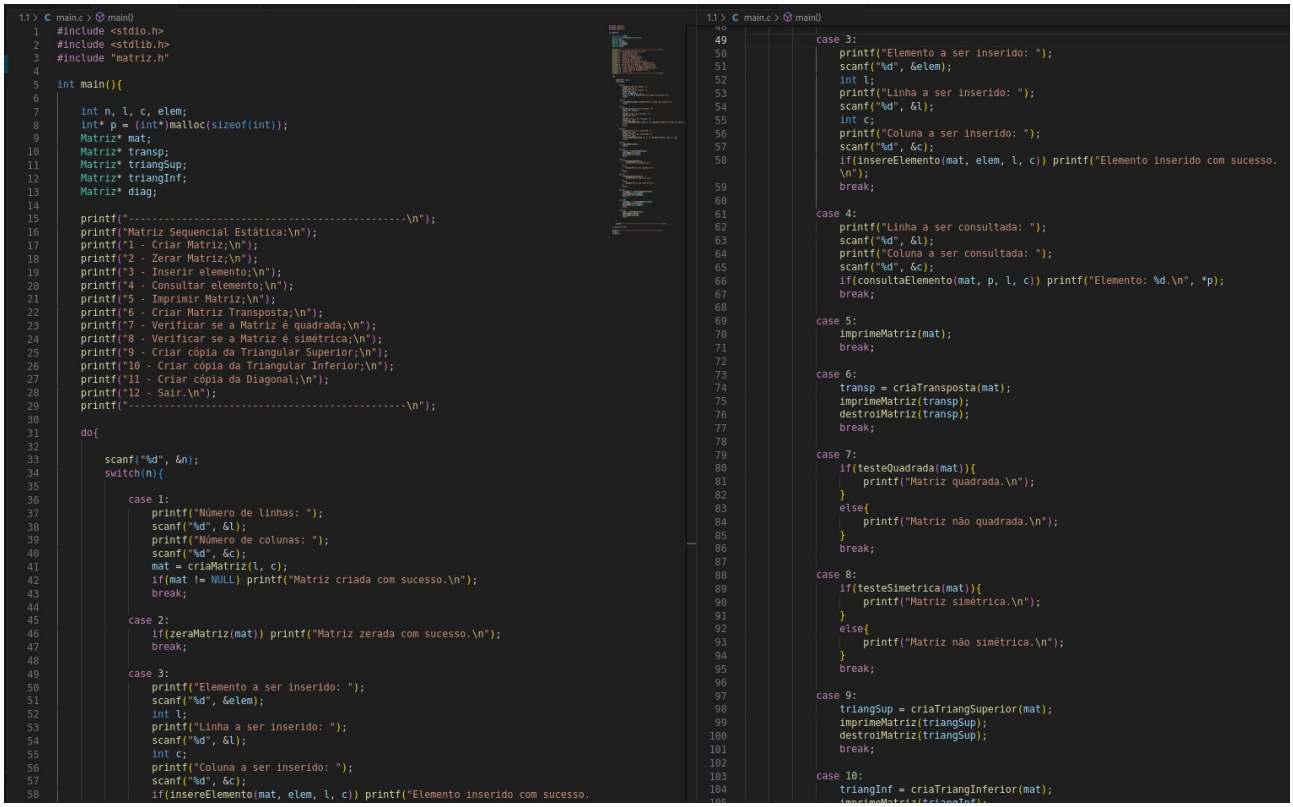
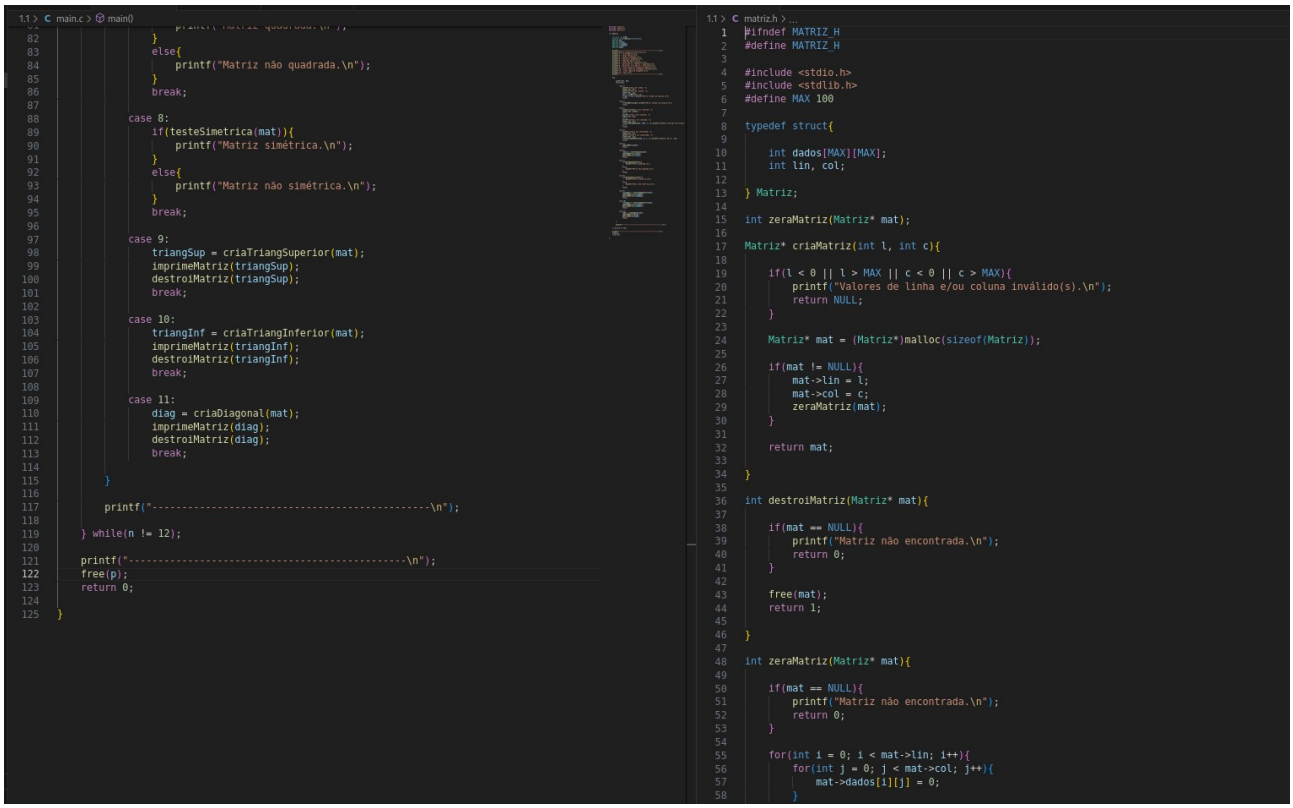
ROTEIRO 7 – LUCAS EDUARDO LEITE COSTA

1.1



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

int main(){

    int n, l, c, elem;
    int* p = (int*)malloc(sizeof(int));
    Matriz* mat;
    Matriz* transp;
    Matriz* triangSup;
    Matriz* triangInf;
    Matriz* diag;

    printf("-----------------------------------------------\n");
    printf("Matriz Sequencial Estática:\n");
    printf("1 - Criar Matriz;\n");
    printf("2 - Zerar Matriz;\n");
    printf("3 - Inserir elemento;\n");
    printf("4 - Consultar elemento;\n");
    printf("5 - Imprimir Matriz;\n");
    printf("6 - Criar Matriz Transposta;\n");
    printf("7 - Verificar se a Matriz é quadrada;\n");
    printf("8 - Verificar se a Matriz é simétrica;\n");
    printf("9 - Criar cópia da Triangular Superior;\n");
    printf("10 - Criar cópia da Triangular Inferior;\n");
    printf("11 - Criar cópia da Diagonal;\n");
    printf("12 - Sair.\n");
    printf("-----------------------------------------------\n");

    do{

        scanf("%d", &n);
        switch(n){

            case 1:
                printf("Número de linhas: ");
                scanf("%d", &l);
                printf("Número de colunas: ");
                scanf("%d", &c);
                mat = criaMatriz(l, c);
                if(mat != NULL) printf("Matriz criada com sucesso.\n");
                break;

            case 2:
                if(zeraMatriz(mat)) printf("Matriz zerada com sucesso.\n");
                break;

            case 3:
                printf("Elemento a ser inserido: ");
                scanf("%d", &elem);
                int l;
                printf("Linha a ser inserido: ");
                scanf("%d", &l);
                int c;
                printf("Coluna a ser inserido: ");
                scanf("%d", &c);
                if(insereElemento(mat, elem, l, c)) printf("Elemento inserido com sucesso.
```

```c
            case 3:
                printf("Elemento a ser inserido: ");
                scanf("%d", &elem);
                int l;
                printf("Linha a ser inserido: ");
                scanf("%d", &l);
                int c;
                printf("Coluna a ser inserido: ");
                scanf("%d", &c);
                if(insereElemento(mat, elem, l, c)) printf("Elemento inserido com sucesso.
                \n");
                break;

            case 4:
                printf("Linha a ser consultada: ");
                scanf("%d", &l);
                printf("Coluna a ser consultada: ");
                scanf("%d", &c);
                if(consultaElemento(mat, p, l, c)) printf("Elemento: %d.\n", *p);
                break;

            case 5:
                imprimeMatriz(mat);
                break;

            case 6:
                transp = criaTransposta(mat);
                imprimeMatriz(transp);
                destroiMatriz(transp);
                break;

            case 7:
                if(testeQuadrada(mat)){
                    printf("Matriz quadrada.\n");
                }
                else{
                    printf("Matriz não quadrada.\n");
                }
                break;

            case 8:
                if(testeSimetrica(mat)){
                    printf("Matriz simétrica.\n");
                }
                else{
                    printf("Matriz não simétrica.\n");
                }
                break;

            case 9:
                triangSup = criaTriangSuperior(mat);
                imprimeMatriz(triangSup);
                destroiMatriz(triangSup);
                break;

            case 10:
                triangInf = criaTriangInferior(mat);
```



```c
                }
                else{
                    printf("Matriz não quadrada.\n");
                }
                break;

            case 8:
                if(testeSimetrica(mat)){
                    printf("Matriz simétrica.\n");
                }
                else{
                    printf("Matriz não simétrica.\n");
                }
                break;

            case 9:
                triangSup = criaTriangSuperior(mat);
                imprimeMatriz(triangSup);
                destroiMatriz(triangSup);
                break;

            case 10:
                triangInf = criaTriangInferior(mat);
                imprimeMatriz(triangInf);
                destroiMatriz(triangInf);
                break;

            case 11:
                diag = criaDiagonal(mat);
                imprimeMatriz(diag);
                destroiMatriz(diag);
                break;

        }

        printf("-----------------------------------------------\n");

    } while(n != 12);

    printf("-----------------------------------------------\n");
    free(p);
    return 0;
}
```

```c
#ifndef MATRIZ_H
#define MATRIZ_H

#include <stdio.h>
#include <stdlib.h>
#define MAX 100

typedef struct{

    int dados[MAX][MAX];
    int lin, col;

} Matriz;

int zeraMatriz(Matriz* mat);

Matriz* criaMatriz(int l, int c){

    if(l < 0 || l > MAX || c < 0 || c > MAX){
        printf("Valores de linha e/ou coluna inválido(s).\n");
        return NULL;
    }

    Matriz* mat = (Matriz*)malloc(sizeof(Matriz));

    if(mat != NULL){
        mat->lin = l;
        mat->col = c;
        zeraMatriz(mat);
    }

    return mat;

}

int destroiMatriz(Matriz* mat){

    if(mat == NULL){
        printf("Matriz não encontrada.\n");
        return 0;
    }

    free(mat);
    return 1;

}

int zeraMatriz(Matriz* mat){

    if(mat == NULL){
        printf("Matriz não encontrada.\n");
        return 0;
    }

    for(int i = 0; i < mat->lin; i++){
        for(int j = 0; j < mat->col; j++){
            mat->dados[i][j] = 0;
        }
```

1.1 > C matriz.h > ...

```c
}

int zeraMatriz(Matriz* mat){

    if(mat == NULL){
        printf("Matriz não encontrada.\n");
        return 0;
    }

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

    return 1;

}

int insereElemento(Matriz* mat, int elem, int l, int c){

    if(mat == NULL){
        printf("Matriz não encontrada.\n");
        return 0;
    }

    if(l < 0 || l > MAX || c < 0 || c > MAX){
        printf("Valores de linha e/ou coluna inválido(s).\n");
        return 0;
    }

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

}

int consultaElemento(Matriz* mat, int* p, int l, int c){

    if(mat == NULL){
        printf("Matriz não encontrada.\n");
        return 0;
    }

    if(l < 0 || l > MAX || c < 0 || c > MAX){
        printf("Valores de linha e/ou coluna inválido(s).\n");
        return 0;
    }

    *p = mat->dados[l][c];
    return 1;

}

void imprimeMatriz(Matriz* mat){

    if(mat == NULL){
        printf("Matriz não encontrada.\n");
        return;
```

1.1 > C matriz.h > ...

```c
void imprimeMatriz(Matriz* mat){

    if(mat == NULL){
        printf("Matriz não encontrada.\n");
        return;
    }

    printf("Elementos:\n");
    for(int i = 0; i < mat->lin; i++){
        printf("[\t");
        for(int j = 0; j < mat->col; j++){
            printf("%d\t", mat->dados[i][j]);
        }
        printf("]\n");
    }

}

Matriz* criaTransposta(Matriz* mat){

    if(mat == NULL){
        printf("Matriz não encontrada.\n");
        return NULL;
    }

    Matriz* transp = criaMatriz(mat->col, mat->lin);

    for(int i = 0; i < transp->lin; i++){
        for(int j = 0; j < transp->col; j++){
            transp->dados[i][j] = mat->dados[j][i];
        }
    }

    return transp;

}

int testeQuadrada(Matriz* mat){

    if(mat == NULL){
        printf("Matriz não encontrada.\n");
        return 0;
    }

    return (mat->lin == mat->col);

}

int testeSimetrica(Matriz* mat){

    if(mat == NULL){
        printf("Matriz não encontrada.\n");
        return 0;
    }

    if(!testeQuadrada(mat)){
        printf("Matriz não quadrada.\n");
```

1.1 > C matriz.h > criaMatriz(int, int)

```c

int testeSimetrica(Matriz* mat){

    if(mat == NULL){
        printf("Matriz não encontrada.\n");
        return 0;
    }

    if(!testeQuadrada(mat)){
        printf("Matriz não quadrada.\n");
        return 0;
    }

    for(int i = 0; i < mat->lin; i++){
        for(int j = i+1; j < mat->col; j++){
            if(mat->dados[i][j] != mat->dados[j][i]){
                return 0;
            }
        }
    }

    return 1;

}

Matriz* criaTriangSuperior(Matriz* mat){

    if(mat == NULL){
        printf("Matriz não encontrada.\n");
        return 0;
    }

    if(!testeQuadrada(mat)){
        printf("Matriz não quadrada.\n");
        return 0;
    }

    Matriz* triangSup = criaMatriz(mat->lin, mat->col);

    for(int i = 0; i < mat->lin; i++){
        for(int j = 0; j < mat->col; j++){
            if(i <= j){
                triangSup->dados[i][j] = mat->dados[i][j];
            }
        }
    }

    return triangSup;

}

Matriz* criaTriangInferior(Matriz* mat){

    if(mat == NULL){
        printf("Matriz não encontrada.\n");
        return 0;
    }
```

1.1 > C matriz.h > ...

```c

Matriz* criaTriangInferior(Matriz* mat){

    if(mat == NULL){
        printf("Matriz não encontrada.\n");
        return 0;
    }

    if(!testeQuadrada(mat)){
        printf("Matriz não quadrada.\n");
        return 0;
    }

    Matriz* triangInf = criaMatriz(mat->lin, mat->col);

    for(int i = 0; i < mat->lin; i++){
        for(int j = 0; j < mat->col; j++){
            if(i >= j){
                triangInf->dados[i][j] = mat->dados[i][j];
            }
        }
    }

    return triangInf;

}

Matriz* criaDiagonal(Matriz* mat){

    if(mat == NULL){
        printf("Matriz não encontrada.\n");
        return 0;
    }

    if(!testeQuadrada(mat)){
        printf("Matriz não quadrada.\n");
        return 0;
    }

    Matriz* diag = criaMatriz(mat->lin, mat->col);

    for(int i = 0; i < diag->lin; i++){
        diag->dados[i][i] = mat->dados[i][i];
    }

    return diag;

}

#endif
```

```
[lucascosta@fedora roteiro_7]$ cd 1.1
[lucascosta@fedora 1.1]$ gcc main.c -o main
[lucascosta@fedora 1.1]$ ./main
-----------------------------------------------
Matriz Sequencial Estática:
1 - Criar Matriz;
2 - Zerar Matriz;
3 - Inserir elemento;
4 - Consultar elemento;
5 - Imprimir Matriz;
6 - Criar Matriz Transposta;
7 - Verificar se a Matriz é quadrada;
8 - Verificar se a Matriz é simétrica;
9 - Criar cópia da Triangular Superior;
10 - Criar cópia da Triangular Inferior;
11 - Criar cópia da Diagonal;
12 - Sair.
-----------------------------------------------
1
Número de linhas: 3
Número de colunas: 3
Matriz criada com sucesso.
-----------------------------------------------
3
Elemento a ser inserido: 15
Linha a ser inserido: 0
Coluna a ser inserido: 1
Elemento inserido com sucesso.
-----------------------------------------------
3
Elemento a ser inserido: 65
Linha a ser inserido: 2
Coluna a ser inserido: 2
Elemento inserido com sucesso.
-----------------------------------------------
5
Elementos:
[       0       15      0       ]
[       0       0       0       ]
[       0       0       65      ]
-----------------------------------------------
6
Elementos:
[       0       0       0       ]
[       15      0       0       ]
[       0       0       65      ]
-----------------------------------------------
4
Linha a ser consultada: 2
Coluna a ser consultada: 2
Elemento: 65.
-----------------------------------------------
7
Matriz quadrada.
-----------------------------------------------
8
Matriz não simétrica.
-----------------------------------------------
9
Elementos:
[       0       15      0       ]
[       0       0       0       ]
```

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

1.2

1.2 > C main.c > main()

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

int main(){

    int n, l, c, elem;
    int* p = (int*)malloc(sizeof(int));
    Matriz* mat;
    Matriz* transp;
    Matriz* triangSup;
    Matriz* triangInf;
    Matriz* diag;

    printf("------------------------------------------------\n");
    printf("Matriz Sequencial Estática:\n");
    printf("1 - Criar Matriz;\n");
    printf("2 - Zerar Matriz;\n");
    printf("3 - Inserir elemento;\n");
    printf("4 - Consultar elemento;\n");
    printf("5 - Imprimir Matriz;\n");
    printf("6 - Criar Matriz Transposta;\n");
    printf("7 - Verificar se a Matriz é quadrada;\n");
    printf("8 - Verificar se a Matriz é simétrica;\n");
    printf("9 - Criar cópia da Triangular Superior;\n");
    printf("10 - Criar cópia da Triangular Inferior;\n");
    printf("11 - Criar cópia da Diagonal;\n");
    printf("12 - Sair.\n");
    printf("------------------------------------------------\n");

    do{

        scanf("%d", &n);
        switch(n){

            case 1:
                printf("Número de linhas: ");
                scanf("%d", &l);
                printf("Número de colunas: ");
                scanf("%d", &c);
                mat = criaMatriz(l, c);
                if(mat != NULL) printf("Matriz criada com sucesso.\n");
                break;

            case 2:
                if(zeraMatriz(mat)) printf("Matriz zerada com sucesso.\n");
                break;

            case 3:
                printf("Elemento a ser inserido: ");
                scanf("%d", &elem);
                int l;
                printf("Linha a ser inserido: ");
                scanf("%d", &l);
                int c;
                printf("Coluna a ser inserido: ");
                scanf("%d", &c);
                if(insereElemento(mat, elem, l, c)) printf("Elemento inserido com sucesso.
```

1.2 > C main.c > ...

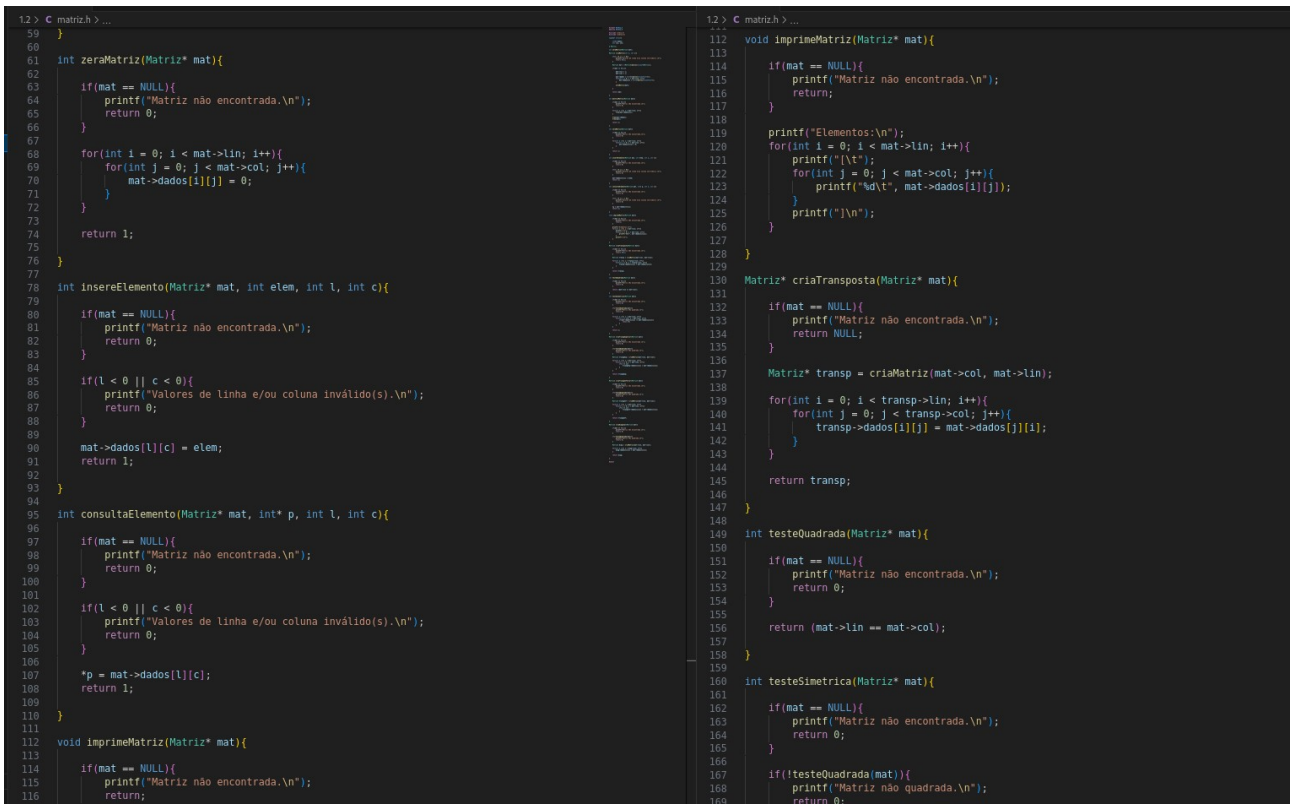
```c
            case 3:
                printf("Elemento a ser inserido: ");
                scanf("%d", &elem);
                int l;
                printf("Linha a ser inserido: ");
                scanf("%d", &l);
                int c;
                printf("Coluna a ser inserido: ");
                scanf("%d", &c);
                if(insereElemento(mat, elem, l, c)) printf("Elemento inserido com sucesso.
                \n");
                break;

            case 4:
                printf("Linha a ser consultada: ");
                scanf("%d", &l);
                printf("Coluna a ser consultada: ");
                scanf("%d", &c);
                if(consultaElemento(mat, p, l, c)) printf("Elemento: %d.\n", *p);
                break;

            case 5:
                imprimeMatriz(mat);
                break;

            case 6:
                transp = criaTransposta(mat);
                imprimeMatriz(transp);
                destroiMatriz(transp);
                break;

            case 7:
                if(testeQuadrada(mat)){
                    printf("Matriz quadrada.\n");
                }
                else{
                    printf("Matriz não quadrada.\n");
                }
                break;

            case 8:
                if(testeSimetrica(mat)){
                    printf("Matriz simétrica.\n");
                }
                else{
                    printf("Matriz não simétrica.\n");
                }
                break;

            case 9:
                triangSup = criaTriangSuperior(mat);
                imprimeMatriz(triangSup);
                destroiMatriz(triangSup);
                break;

            case 10:
                triangInf = criaTriangInferior(mat);
```

1.2 > C main.c > main()

```c
                else{
                    printf("Matriz não simétrica.\n");
                }
                break;

            case 9:
                triangSup = criaTriangSuperior(mat);
                imprimeMatriz(triangSup);
                destroiMatriz(triangSup);
                break;

            case 10:
                triangInf = criaTriangInferior(mat);
                imprimeMatriz(triangInf);
                destroiMatriz(triangInf);
                break;

            case 11:
                diag = criaDiagonal(mat);
                imprimeMatriz(diag);
                destroiMatriz(diag);
                break;

        }

        printf("------------------------------------------------\n");

    } while(n != 12);

    printf("------------------------------------------------\n");
    free(p);

    return 0;

}
```

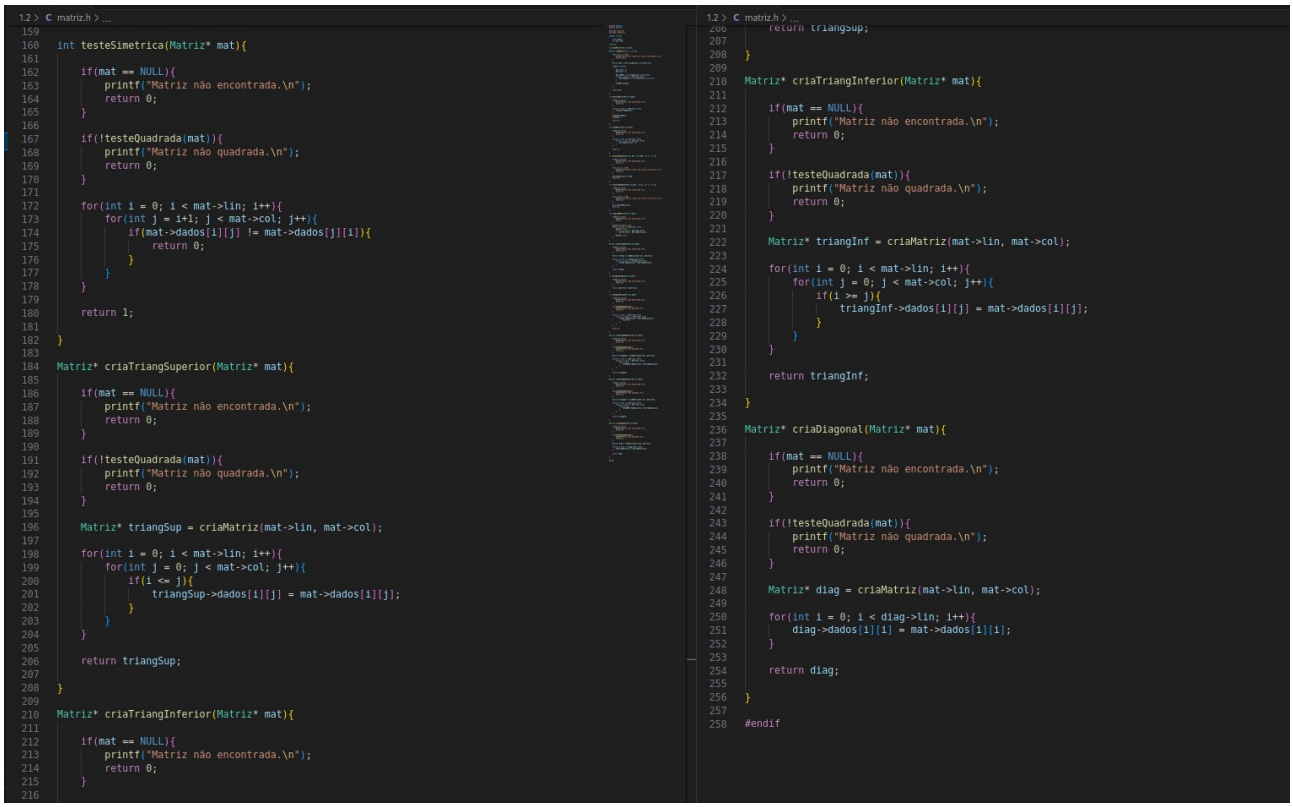
1.2 > C matriz.h > ...

```c
#ifndef MATRIZ_H
#define MATRIZ_H

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

typedef struct{

    int** dados;
    int lin, col;

} Matriz;

int zeraMatriz(Matriz* mat);

Matriz* criaMatriz(int l, int c){

    if(l < 0 || c < 0){
        printf("Valores de linha e/ou coluna inválido(s).\n");
        return NULL;
    }

    Matriz* mat = (Matriz*)malloc(sizeof(Matriz));

    if(mat != NULL){

        mat->lin = l;
        mat->col = c;

        mat->dados = (int**)malloc(sizeof(int*));
        for(int i = 0; i < mat->lin; i++){
            mat->dados[i] = (int*)malloc(sizeof(int));
        }

        zeraMatriz(mat);

    }

    return mat;

}

int destroiMatriz(Matriz* mat){

    if(mat == NULL){
        printf("Matriz não encontrada.\n");
        return 0;
    }

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

    free(mat->dados);
    free(mat);

    return 1;
```



```
1.2 > C matriz.h > ...
}

int zeraMatriz(Matriz* mat){

    if(mat == NULL){
        printf("Matriz não encontrada.\n");
        return 0;
    }

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

    return 1;

}

int insereElemento(Matriz* mat, int elem, int l, int c){

    if(mat == NULL){
        printf("Matriz não encontrada.\n");
        return 0;
    }

    if(l < 0 || c < 0){
        printf("Valores de linha e/ou coluna inválido(s).\n");
        return 0;
    }

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

}

int consultaElemento(Matriz* mat, int* p, int l, int c){

    if(mat == NULL){
        printf("Matriz não encontrada.\n");
        return 0;
    }

    if(l < 0 || c < 0){
        printf("Valores de linha e/ou coluna inválido(s).\n");
        return 0;
    }

    *p = mat->dados[l][c];
    return 1;

}

void imprimeMatriz(Matriz* mat){

    if(mat == NULL){
        printf("Matriz não encontrada.\n");
        return;
```

```
1.2 > C matriz.h > ...
void imprimeMatriz(Matriz* mat){

    if(mat == NULL){
        printf("Matriz não encontrada.\n");
        return;
    }

    printf("Elementos:\n");
    for(int i = 0; i < mat->lin; i++){
        printf("[\t");
        for(int j = 0; j < mat->col; j++){
            printf("%d\t", mat->dados[i][j]);
        }
        printf("]\n");
    }

}

Matriz* criaTransposta(Matriz* mat){

    if(mat == NULL){
        printf("Matriz não encontrada.\n");
        return NULL;
    }

    Matriz* transp = criaMatriz(mat->col, mat->lin);

    for(int i = 0; i < transp->lin; i++){
        for(int j = 0; j < transp->col; j++){
            transp->dados[i][j] = mat->dados[j][i];
        }
    }

    return transp;

}

int testeQuadrada(Matriz* mat){

    if(mat == NULL){
        printf("Matriz não encontrada.\n");
        return 0;
    }

    return (mat->lin == mat->col);

}

int testeSimetrica(Matriz* mat){

    if(mat == NULL){
        printf("Matriz não encontrada.\n");
        return 0;
    }

    if(!testeQuadrada(mat)){
        printf("Matriz não quadrada.\n");
        return 0;
```



```
1.2 > C matriz.h > ...

int testeSimetrica(Matriz* mat){

    if(mat == NULL){
        printf("Matriz não encontrada.\n");
        return 0;
    }

    if(!testeQuadrada(mat)){
        printf("Matriz não quadrada.\n");
        return 0;
    }

    for(int i = 0; i < mat->lin; i++){
        for(int j = i+1; j < mat->col; j++){
            if(mat->dados[i][j] != mat->dados[j][i]){
                return 0;
            }
        }
    }

    return 1;

}

Matriz* criaTriangSuperior(Matriz* mat){

    if(mat == NULL){
        printf("Matriz não encontrada.\n");
        return 0;
    }

    if(!testeQuadrada(mat)){
        printf("Matriz não quadrada.\n");
        return 0;
    }

    Matriz* triangSup = criaMatriz(mat->lin, mat->col);

    for(int i = 0; i < mat->lin; i++){
        for(int j = 0; j < mat->col; j++){
            if(i <= j){
                triangSup->dados[i][j] = mat->dados[i][j];
            }
        }
    }

    return triangSup;

}

Matriz* criaTriangInferior(Matriz* mat){

    if(mat == NULL){
        printf("Matriz não encontrada.\n");
        return 0;
    }
```

```
1.2 > C matriz.h > ...
    return triangSup;

}

Matriz* criaTriangInferior(Matriz* mat){

    if(mat == NULL){
        printf("Matriz não encontrada.\n");
        return 0;
    }

    if(!testeQuadrada(mat)){
        printf("Matriz não quadrada.\n");
        return 0;
    }

    Matriz* triangInf = criaMatriz(mat->lin, mat->col);

    for(int i = 0; i < mat->lin; i++){
        for(int j = 0; j < mat->col; j++){
            if(i >= j){
                triangInf->dados[i][j] = mat->dados[i][j];
            }
        }
    }

    return triangInf;

}

Matriz* criaDiagonal(Matriz* mat){

    if(mat == NULL){
        printf("Matriz não encontrada.\n");
        return 0;
    }

    if(!testeQuadrada(mat)){
        printf("Matriz não quadrada.\n");
        return 0;
    }

    Matriz* diag = criaMatriz(mat->lin, mat->col);

    for(int i = 0; i < diag->lin; i++){
        diag->dados[i][i] = mat->dados[i][i];
    }

    return diag;

}

#endif
```

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

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

2.1

2.1 > C main.c > main()

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

int main(){

    int n, tam, i, j, elem;
    int* p = (int*)malloc(sizeof(int));
    MFaixa* mat;

    printf("-----------------------------------------------\n");
    printf("Matriz de Faixa (Tridiagonal):\n");
    printf("1 - Criar Matriz;\n");
    printf("2 - Zerar Matriz;\n");
    printf("3 - Inserir elemento;\n");
    printf("4 - Consultar elemento;\n");
    printf("5 - Imprimir Matriz;\n");
    printf("6 - Destruir Matriz;\n");
    printf("7 - Sair.\n");
    printf("-----------------------------------------------\n");
    do{

        scanf("%d", &n);

        switch(n){

            case 1:
                printf("Tamanho da Matriz: ");
                scanf("%d", &tam);
                mat = criaMatriz(tam);
                if(mat != NULL) printf("Matriz criada com sucesso.\n");
                break;

            case 2:
                if(zeraMatriz(mat)) printf("Matriz zerada com sucesso.\n");
                break;

            case 3:
                printf("Elemento a ser inserido: ");
                scanf("%d", &elem);
                printf("Linha a ser inserido: ");
                scanf("%d", &i);
                printf("Coluna a ser inserido: ");
                scanf("%d", &j);
                if(insereElemento(mat, elem, i, j)) printf("Elemento inserido com sucesso.
                \n");
                break;

            case 4:
                printf("Linha a ser consultada: ");
                scanf("%d", &i);
                printf("Coluna a ser consultada: ");
                scanf("%d", &j);
                printf("Elemento: %d\n", consultaElemento(mat, i, j));
                break;

            case 5:
                imprimeMatriz(mat);
                break;

            case 6:
                if(destroiMatriz(mat)) printf("Matriz destruída com sucesso.\n");
                break;

        }

        printf("-----------------------------------------------\n");

    } while(n != 7);

    printf("-----------------------------------------------\n");
    free(p);
    return 0;

}
```

2.1 > C matriz.h > ...

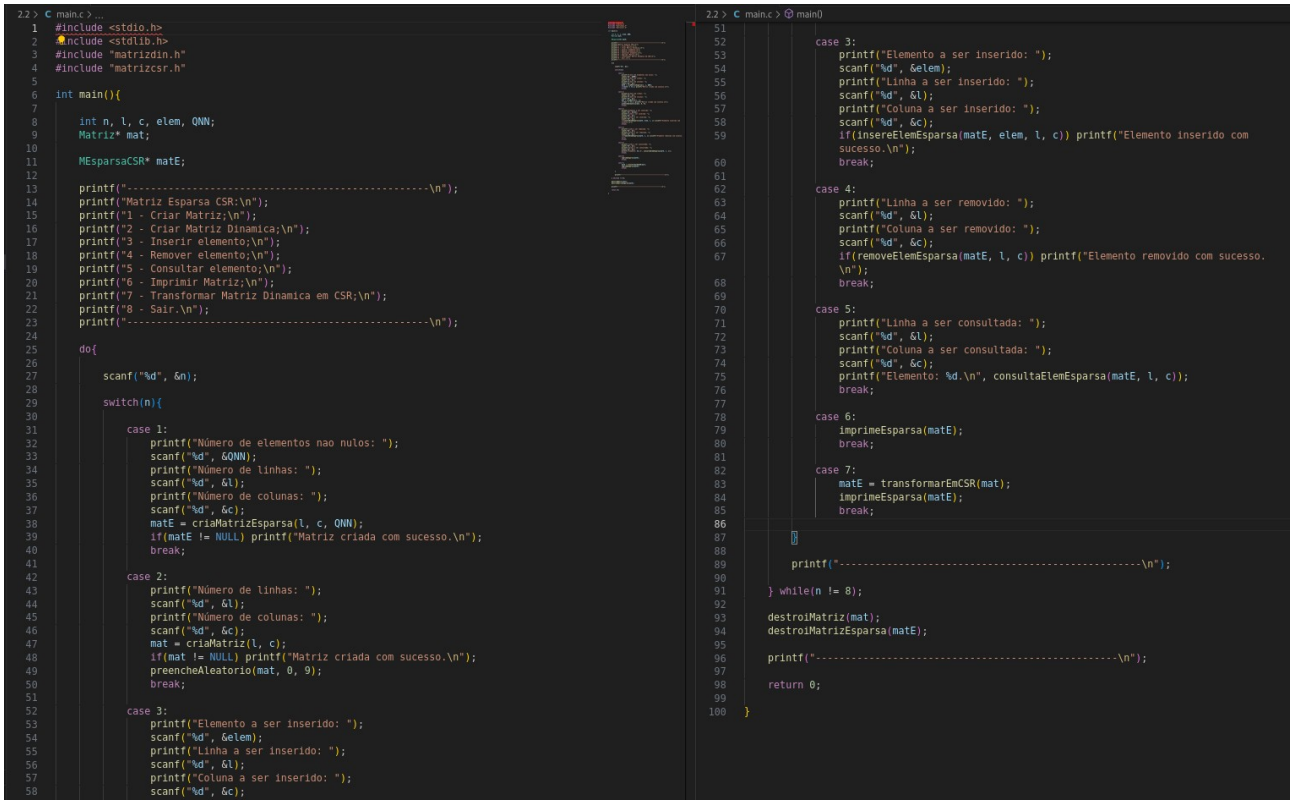
```c
#ifndef MATRIZ_H
#define MATRIZ_H

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

typedef struct{

    int* diagonal;
    int* superior;
    int* inferior;
    int tam;

} MFaixa;

int zeraMatriz(MFaixa* mat);

MFaixa* criaMatriz(int tam){

    MFaixa* mat = (MFaixa*)malloc(sizeof(MFaixa));

    if(mat != NULL){

        if(tam <= 1){
            printf("ERRO: Tamanho da matriz inválido.\n");
            return NULL;
        }

        mat->tam = tam;
        mat->diagonal = (int*)malloc(tam * sizeof(int));
        mat->superior = (int*)malloc((tam-1) * sizeof(int));
        mat->inferior = (int*)malloc((tam-1) * sizeof(int));

        if(mat->diagonal == NULL || mat->superior == NULL || mat->inferior == NULL){
            return NULL;
        }

        zeraMatriz(mat);

    }

    return mat;

}

int destroiMatriz(MFaixa* mat){

    if(mat == NULL){
        printf("ERRO: Matriz não encontrada.\n");
        return 0;
    }

    free(mat->diagonal);
    free(mat->superior);
    free(mat->inferior);
    free(mat);

    return 1;

}

int zeraMatriz(MFaixa* mat){

    if(mat == NULL){
        printf("ERRO: Matriz não encontrada.\n");
        return 0;
    }

    int i;
    for(i = 0; i < mat->tam-1; i++){
        mat->diagonal[i] = 0;
        mat->superior[i] = 0;
        mat->inferior[i] = 0;
    }

    mat->diagonal[i] = 0;

    return 1;

}

int insereElemento(MFaixa* mat, int elem, int i, int j){

    if(mat == NULL){
        printf("ERRO: Matriz não encontrada.\n");
        return 0;
    }

    if(i == j){
        mat->diagonal[i] = elem;
    }

    else if(i+1 == j){
        mat->superior[i] = elem;
    }

    else if(i == j+1){
        mat->inferior[j] = elem;
    }
```

```c
}

int insereElemento(MFaixa* mat, int elem, int i, int j){

    if(mat == NULL){
        printf("ERRO: Matriz não encontrada.\n");
        return 0;
    }

    if(i == j){
        mat->diagonal[i] = elem;
    }

    else if(i+1 == j){
        mat->superior[i] = elem;
    }

    else if(i == j+1){
        mat->inferior[j] = elem;
    }

    else{
        printf("ERRO: Posição do elemento fora dos limites da matriz.\n");
        return 0;
    }

    return 1;

}

int consultaElemento(MFaixa* mat, int i, int j){

    if(mat == NULL){
        printf("ERRO: Matriz não encontrada.\n");
        return 0;
    }

    if(i == j){
        return mat->diagonal[i];
    }

    else if(i+1 == j){
        return mat->superior[i];
    }

    else if(i == j+1){
        return mat->inferior[j];
    }

    else{
        printf("ERRO: Posição do elemento fora dos limites da matriz.\n");
        return 0;
    }

}

void imprimeMatriz(MFaixa* mat){
```

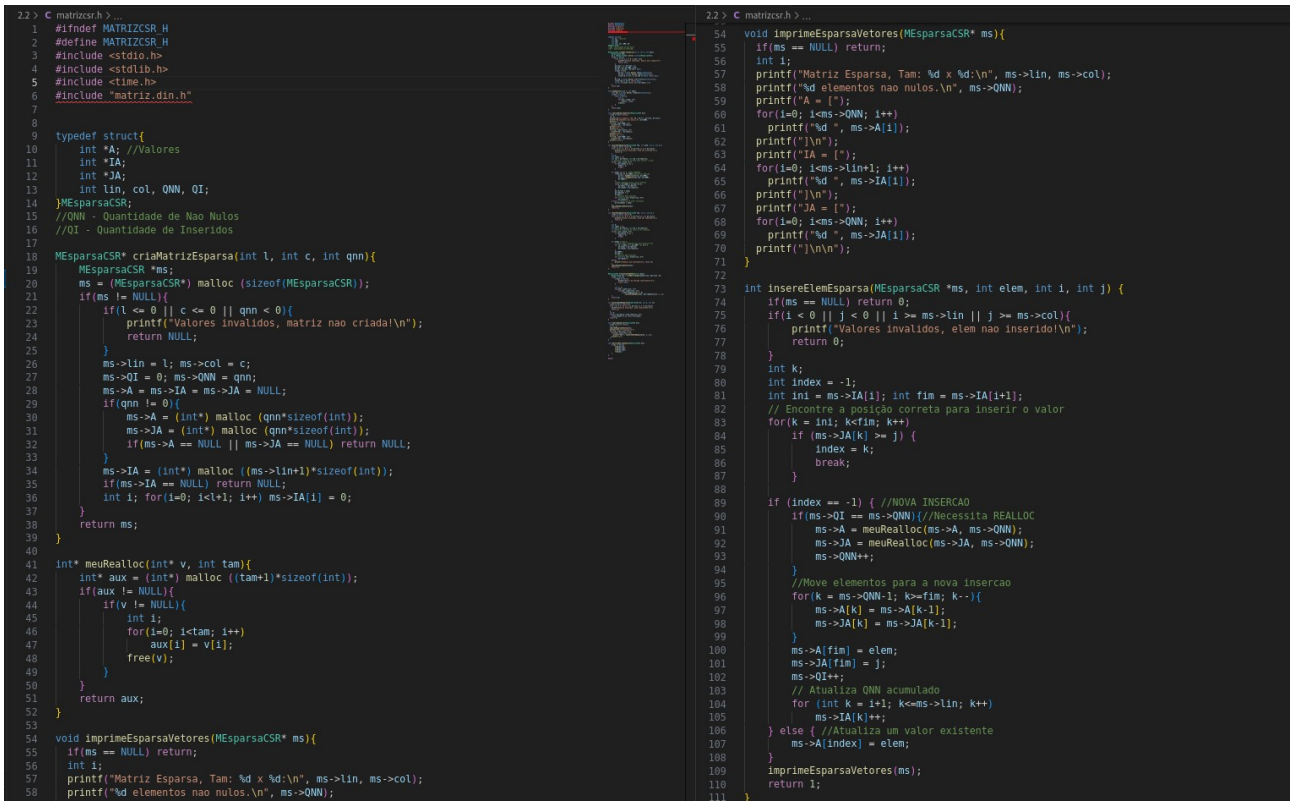
```c
        printf("ERRO: Posição do elemento fora dos limites da matriz.\n");
        return 0;
    }

}

void imprimeMatriz(MFaixa* mat){

    if(mat == NULL){
        printf("ERRO: Matriz não encontrada.\n");
    }

    for(int i = 0; i < mat->tam; i++){
        for(int j = 0; j < mat->tam; j++){

            if(i == j){
                printf("%d\t", mat->diagonal[i]);
            }

            else if(i+1 == j){
                printf("%d\t", mat->superior[i]);
            }

            else if(i == j+1){
                printf("%d\t", mat->inferior[j]);
            }

            else{
                printf("0\t");
            }

        }
        printf("\n");
    }

}

#endif
```

```
[lucascosta@fedora 2.1]$ gcc main.c -o main
[lucascosta@fedora 2.1]$ ./main
-----------------------------------------------
Matriz de Faixa (Tridiagonal):
1 - Criar Matriz;
2 - Zerar Matriz;
3 - Inserir elemento;
4 - Consultar elemento;
5 - Imprimir Matriz;
6 - Destruir Matriz;
7 - Sair.
-----------------------------------------------

1
Tamanho da Matriz: 3
Matriz criada com sucesso.
-----------------------------------------------
3
Elemento a ser inserido: 5
Linha a ser inserido: 1
Coluna a ser inserido: 2
Elemento inserido com sucesso.
-----------------------------------------------
3
Elemento a ser inserido: 15
Linha a ser inserido:
0
Coluna a ser inserido: 1
Elemento inserido com sucesso.
-----------------------------------------------
4
Linha a ser consultada: 0
Coluna a ser consultada: 1
Elemento: 15
-----------------------------------------------
5
0       15      0
0       0       5
0       0       0
-----------------------------------------------
2
Matriz zerada com sucesso.
-----------------------------------------------
5
0       0       0
0       0       0
0       0       0
-----------------------------------------------
6
Matriz destruída com sucesso.
-----------------------------------------------
7
-----------------------------------------------
-----------------------------------------------
[lucascosta@fedora 2.1]$
```

2.2



```c
#include <stdio.h>
#include <stdlib.h>
#include "matrizdin.h"
#include "matrizcsr.h"

int main(){

    int n, l, c, elem, QNN;
    Matriz* mat;

    MEsparsaCSR* matE;

    printf("-----------------------------------------------------\n");
    printf("Matriz Esparsa CSR:\n");
    printf("1 - Criar Matriz;\n");
    printf("2 - Criar Matriz Dinamica;\n");
    printf("3 - Inserir elemento;\n");
    printf("4 - Remover elemento;\n");
    printf("5 - Consultar elemento;\n");
    printf("6 - Imprimir Matriz;\n");
    printf("7 - Transformar Matriz Dinamica em CSR;\n");
    printf("8 - Sair.\n");
    printf("-----------------------------------------------------\n");

    do{

        scanf("%d", &n);

        switch(n){

            case 1:
                printf("Número de elementos nao nulos: ");
                scanf("%d", &QNN);
                printf("Número de linhas: ");
                scanf("%d", &l);
                printf("Número de colunas: ");
                scanf("%d", &c);
                matE = criaMatrizEsparsa(l, c, QNN);
                if(matE != NULL) printf("Matriz criada com sucesso.\n");
                break;

            case 2:
                printf("Número de linhas: ");
                scanf("%d", &l);
                printf("Número de colunas: ");
                scanf("%d", &c);
                mat = criaMatriz(l, c);
                if(mat != NULL) printf("Matriz criada com sucesso.\n");
                preencheAleatorio(mat, 0, 9);
                break;

            case 3:
                printf("Elemento a ser inserido: ");
                scanf("%d", &elem);
                printf("Linha a ser inserido: ");
                scanf("%d", &l);
                printf("Coluna a ser inserido: ");
                scanf("%d", &c);
                if(insereElemEsparsa(matE, elem, l, c)) printf("Elemento inserido com
                sucesso.\n");
                break;

            case 4:
                printf("Linha a ser removido: ");
                scanf("%d", &l);
                printf("Coluna a ser removido: ");
                scanf("%d", &c);
                if(removeElemEsparsa(matE, l, c)) printf("Elemento removido com sucesso.
                \n");
                break;

            case 5:
                printf("Linha a ser consultada: ");
                scanf("%d", &l);
                printf("Coluna a ser consultada: ");
                scanf("%d", &c);
                printf("Elemento: %d.\n", consultaElemEsparsa(matE, l, c));
                break;

            case 6:
                imprimeEsparsa(matE);
                break;

            case 7:
                matE = transformarEmCSR(mat);
                imprimeEsparsa(matE);
                break;

        }

        printf("-----------------------------------------------------\n");

    } while(n != 8);

    destroiMatriz(mat);
    destroiMatrizEsparsa(matE);

    printf("-----------------------------------------------------\n");

    return 0;

}
```



```c
#ifndef MATRIZCSR_H
#define MATRIZCSR_H
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include "matriz.din.h"


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){
        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;
  printf("Matriz Esparsa, Tam: %d x %d:\n", ms->lin, ms->col);
  printf("%d elementos nao nulos.\n", ms->QNN);
```

```c
void imprimeEsparsaVetores(MEsparsaCSR* ms){
  if(ms == NULL) return;
  int i;
  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");
  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;
}
```

2.2 > C matrizcsr.h > ...

```
}

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");
```

2.2 > C matrizcsr.h > ...

```
    }
    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
```

2.2 > C matrizdin.h > ...

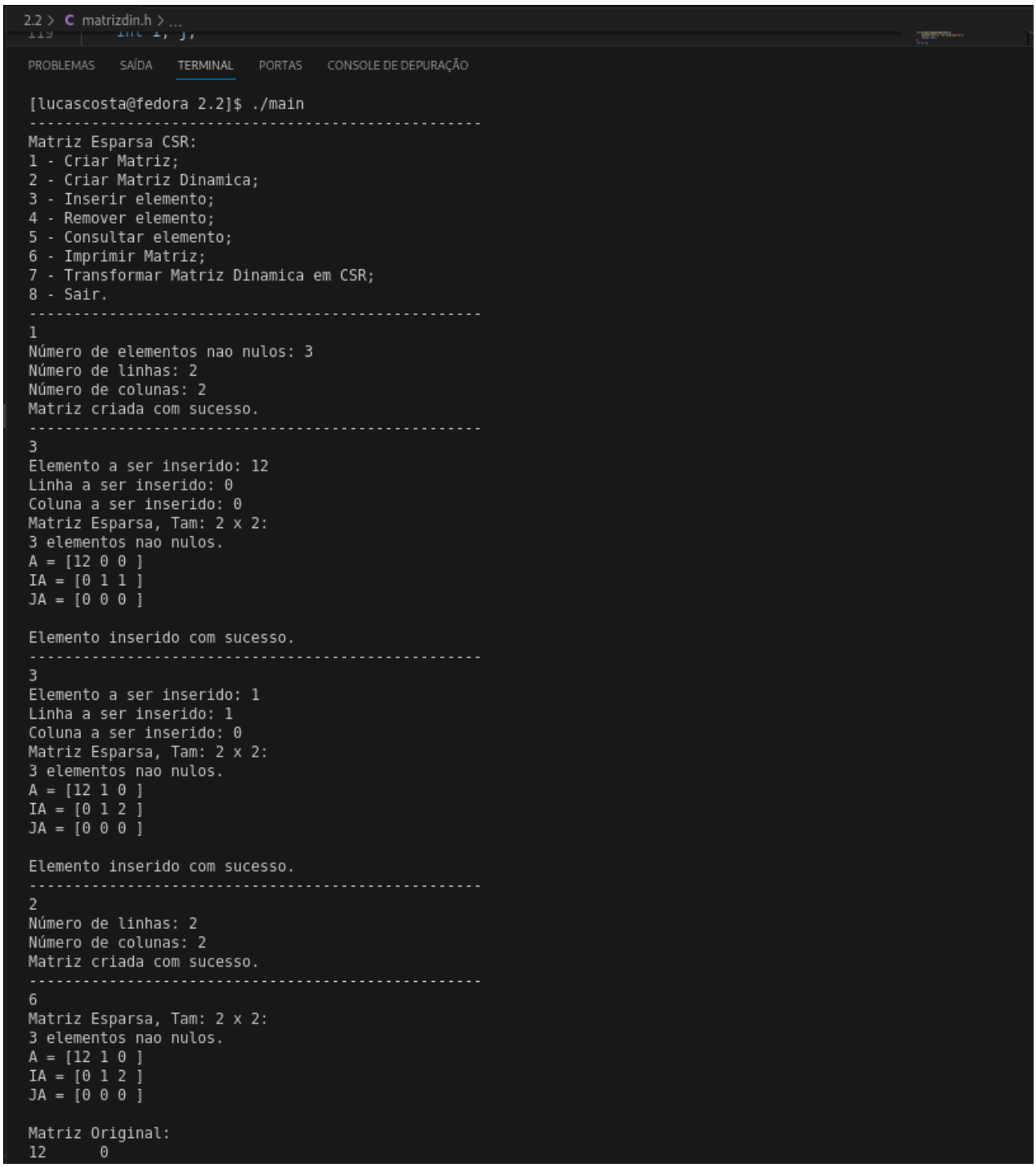
```
#ifndef MATRIZDIN_H
#define MATRIZDIN_H

#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){
        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;
}

```

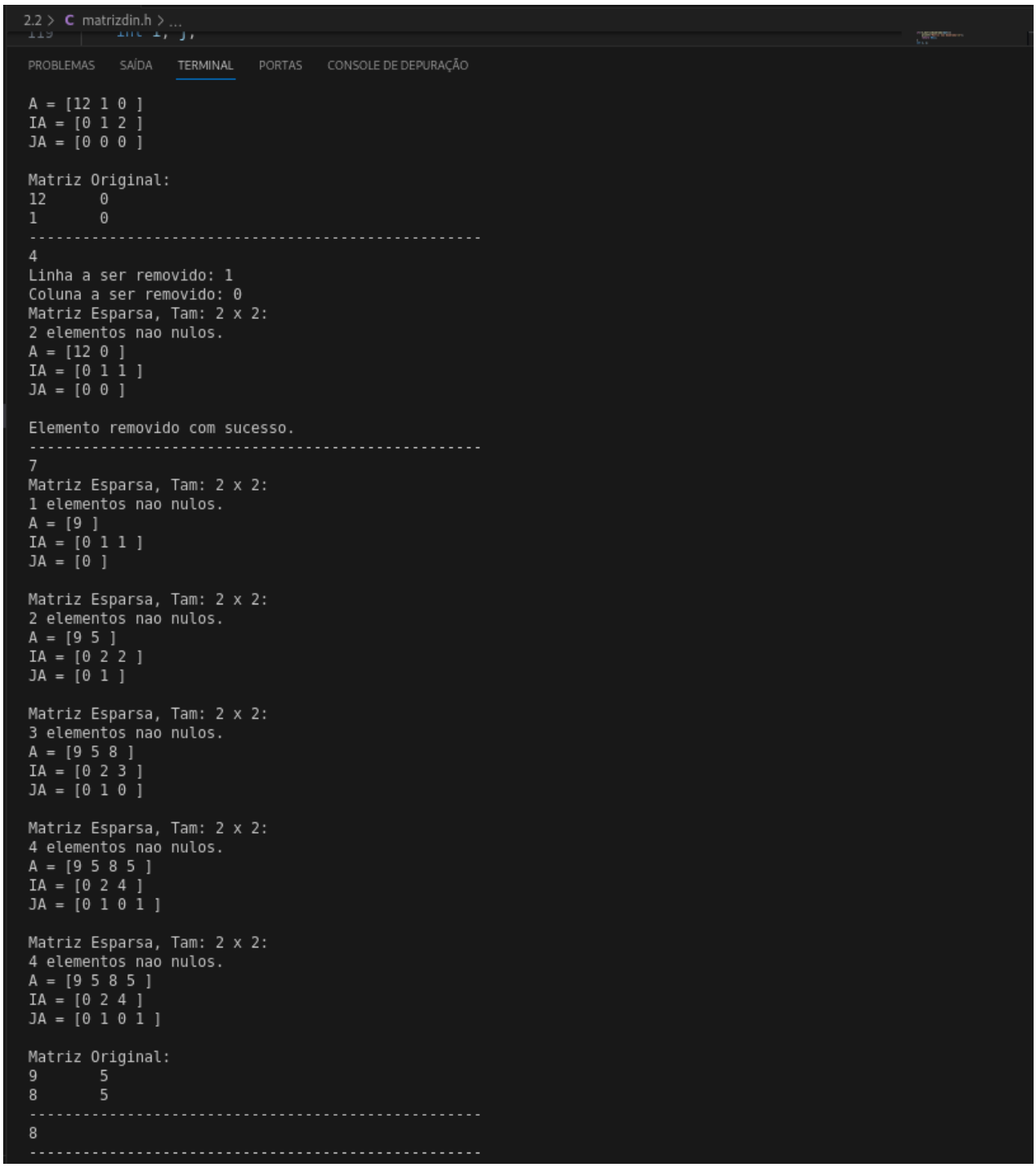
2.2 > C matrizdin.h > ...

```
}

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");
}

//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;
```

2.2 > C matrizdin.h > ...

```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;
    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
```


2.2 > C matrizdin.h > ...
int i, j;
PROBLEMAS SAÍDA TERMINAL PORTAS CONSOLE DE DEPURAÇÃO
[lucascosta@fedora 2.2]$ ./main
Matriz Esparsa CSR:
1 - Criar Matriz;
2 - Criar Matriz Dinamica;
3 - Inserir elemento;
4 - Remover elemento;
5 - Consultar elemento;
6 - Imprimir Matriz;
7 - Transformar Matriz Dinamica em CSR;
8 - Sair.
1
Número de elementos nao nulos: 3
Número de linhas: 2
Número de colunas: 2
Matriz criada com sucesso.
3
Elemento a ser inserido: 12
Linha a ser inserido: 0
Coluna a ser inserido: 0
Matriz Esparsa, Tam: 2 x 2:
3 elementos nao nulos.
A = [12 0 0]
IA = [0 1 1]
JA = [0 0 0]
Elemento inserido com sucesso.
3
Elemento a ser inserido: 1
Linha a ser inserido: 1
Coluna a ser inserido: 0
Matriz Esparsa, Tam: 2 x 2:
3 elementos nao nulos.
A = [12 1 0]
IA = [0 1 2]
JA = [0 0 0]
Elemento inserido com sucesso.
2
Número de linhas: 2
Número de colunas: 2
Matriz criada com sucesso.
6
Matriz Esparsa, Tam: 2 x 2:
3 elementos nao nulos.
A = [12 1 0]
IA = [0 1 2]
JA = [0 0 0]
Matriz Original:
12 0


2.2 > C matrizdin.h > ...
int i, j;
PROBLEMAS SAÍDA TERMINAL PORTAS CONSOLE DE DEPURAÇÃO
A = [12 1 0]
IA = [0 1 2]
JA = [0 0 0]
Matriz Original:
12 0
1 0
4
Linha a ser removido: 1
Coluna a ser removido: 0
Matriz Esparsa, Tam: 2 x 2:
2 elementos nao nulos.
A = [12 0]
IA = [0 1 1]
JA = [0 0]
Elemento removido com sucesso.
7
Matriz Esparsa, Tam: 2 x 2:
1 elementos nao nulos.
A = [9]
IA = [0 1 1]
JA = [0]
Matriz Esparsa, Tam: 2 x 2:
2 elementos nao nulos.
A = [9 5]
IA = [0 2 2]
JA = [0 1]
Matriz Esparsa, Tam: 2 x 2:
3 elementos nao nulos.
A = [9 5 8]
IA = [0 2 3]
JA = [0 1 0]
Matriz Esparsa, Tam: 2 x 2:
4 elementos nao nulos.
A = [9 5 8 5]
IA = [0 2 4]
JA = [0 1 0 1]
Matriz Esparsa, Tam: 2 x 2:
4 elementos nao nulos.
A = [9 5 8 5]
IA = [0 2 4]
JA = [0 1 0 1]
Matriz Original:
9 5
8 5
8