roteiro-07/ex01-01.h

```h
#ifndef MATRIZ_H
#define MATRIZ_H

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

#define MAX 100

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

void zeraMatriz(Matriz* mat) {
    int i, j;
    if (mat != NULL) {
        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);
        *mat = NULL;
    }
}

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;
}

```

```c
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("\t%d", mat->dados[i][j]);
        printf("\n");
    }
    printf("\n");
}

#endif
```

roteiro-07/ex01-02.h

```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) {
    if (mat == NULL) return;

    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);
        *mat = NULL;
    }
}

int preencheAleatorio(Matriz* mat, int ini, int fim) {
    if (mat == NULL) return 0;
    srand(time(NULL));
    int i, j;
```

```c
    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");
}

#endif
```

roteiro-07/ex01.c

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

/*
 *
 *  O ARQUIVO FUNCIONA PARA AS DUAS IMPLEMENTACOES DE MATRIZES
 *  BASTA IMPORTAR APENAS A BIBLIOTECA DESEJADA
 *
 *  EX01-01 = MATRIZ SEQUENCIAL ESTATICA
 *  EX01-02 = MATRIZ SEQUENCIAL DINAMICA
 *
 */

// #include "ex01-01.h"
#include "ex01-02.h"

enum {
    EXIT = 0,
    CREATE,
    FILL,
    INSERT,
    QUERY,
    PRINT,
    RESET,
    DESTROY
} Options;

int getOption() {
    int option;

    printf("\n====================\n");
    printf("(%d) Criar\n", CREATE);
    printf("(%d) Preencher aleatoriamente\n", FILL);
    printf("(%d) Inserir\n", INSERT);
    printf("(%d) Consultar\n", QUERY);
    printf("(%d) Imprimir\n", PRINT);
    printf("(%d) Zerar\n", RESET);
    printf("(%d) Destruir\n", DESTROY);
    printf("(%d) Sair\n", EXIT);
    printf("====================\n");
    printf("Operacao: ");

    scanf("%d", &option);
    printf("\n");

    return option;
}

int runMenu() {
    Matriz* matriz = NULL;
    int exit = 0, item, pri, linhas, colunas, min, max;

```

```c
    do {
        switch (getOption()) {
            case CREATE:
                if (matriz != NULL) {
                    destroiMatriz(&matriz);
                    printf("Matriz apagada");
                }

                printf("Insira as dimensoes\n");
                printf("Linhas: ");
                scanf("%d", &linhas);
                printf("Colunas: ");
                scanf("%d", &colunas);

                matriz = criaMatriz(linhas, colunas);
                break;

            case FILL:
                printf("Insira os limites dos valores\n");
                printf("Min: ");
                scanf("%d", &min);
                printf("Max: ");
                scanf("%d", &max);

                preencheAleatorio(matriz, min, max);
                break;

            case INSERT:
                printf("Elemento para inserir: ");
                scanf("%d", &item);
                printf("Linha (inicio 0): ");
                scanf("%d", &linhas);
                printf("Colunas (inicio 0): ");
                scanf("%d", &colunas);

                if (insereElem(matriz, item, linhas, colunas)) {
                    printf("Inserir(%d) (%d, %d)", item, linhas, colunas);
                } else {
                    printf("Nao foi possivel inserir(%d)", item);
                }
                break;

            case QUERY:
                printf("Posicao para consultar");
                printf("Linha (inicio 0): ");
                scanf("%d", &linhas);
                printf("Colunas (inicio 0): ");
                scanf("%d", &colunas);

                if (consultaElem(matriz, &item, linhas, colunas)) {
                    printf("Elemento = %d\n", item);
                } else {
                    printf("Nao foi consultar a matriz nessa posicao");
                }
                break;
```

```c

            case PRINT:
                imprime(matriz);
                break;

            case RESET:
                zeraMatriz(matriz);
                printf("Matriz zerada");
                break;

            case DESTROY:
                destroiMatriz(&matriz);
                printf("Matriz destruida");
                break;

            case EXIT:
                if (matriz != NULL) {
                    destroiMatriz(&matriz);
                }
                printf("Programa encerrado");
                exit = 1;
                break;

            default:
                printf("Opcao desconhecida, tente novamente");
        }
        printf("\n");
    } while (!exit);
}

int main() {
    runMenu();
    return 0;
}
```

```
gabriel.meira.2004@lab-02-07-04:~/Desktop/ufsj/UFSJ-LabProg2/roteiro-07$ ./ex01

====================
(1) Criar
(2) Preencher aleatoriamente
(3) Inserir
(4) Consultar
(5) Imprimir
(6) Zerar
(7) Destruir
(0) Sair
====================
Operacao: 1

Insira as dimensoes
Linhas: 4
Colunas: 6


====================
(1) Criar
(2) Preencher aleatoriamente
(3) Inserir
(4) Consultar
(5) Imprimir
(6) Zerar
(7) Destruir
(0) Sair
====================
Operacao: 2

Insira os limites dos valores
Min: -10
Max: 10


====================
(1) Criar
(2) Preencher aleatoriamente
(3) Inserir
(4) Consultar
(5) Imprimir
(6) Zerar
(7) Destruir
(0) Sair
====================
Operacao: 4

Posicao para consultarLinha (inicio 0): 3
Colunas (inicio 0): 2
Elemento = -4


====================
(1) Criar
(2) Preencher aleatoriamente
(3) Inserir
(4) Consultar
(5) Imprimir
(6) Zerar
(7) Destruir
(0) Sair
====================
Operacao: 5

Matriz 4 x 6:
4 8 -7 4 -8 8
-7 -2 -1 -9 9 -7
10 4 -3 6 -1 -3
-1 -6 -4 -6 9 -7



====================
(1) Criar
(2) Preencher aleatoriamente
(3) Inserir
(4) Consultar
(5) Imprimir
(6) Zerar
(7) Destruir
(0) Sair
====================
Operacao: 3

Elemento para inserir: 23
Linha (inicio 0): 3
Colunas (inicio 0): 2
Inserir(23) (3, 2)

====================
(1) Criar
(2) Preencher aleatoriamente
(3) Inserir
(4) Consultar
(5) Imprimir
(6) Zerar
(7) Destruir
(0) Sair
====================
Operacao: 5

Matriz 4 x 6:
4 8 -7 4 -8 8
-7 -2 -1 -9 9 -7
10 4 -3 6 -1 -3
-1 -6 23 -6 9 -7



====================
(1) Criar
(2) Preencher aleatoriamente
(3) Inserir
(4) Consultar
(5) Imprimir
(6) Zerar
(7) Destruir
(0) Sair
====================
Operacao: 7

Matriz destruida

====================
(1) Criar
(2) Preencher aleatoriamente
(3) Inserir
(4) Consultar
(5) Imprimir
(6) Zerar
(7) Destruir
(0) Sair
====================
Operacao: 0

Programa encerrado
gabriel.meira.2004@lab-02-07-04:~/Desktop/ufsj/UFSJ-LabProg2/roteiro-07$
```

roteiro-07/ex02-01.h

```c
#ifndef MATRIZFAIXA_H
#define MATRIZFAIXA_H

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

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

void zeraMatriz(Matriz* 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;
        }
    }
}

Matriz* criaMatriz(int t) {
    Matriz* mf;
    mf = (Matriz*)malloc(sizeof(Matriz));
    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(Matriz** mf) {
    if (*mf != NULL) {
        free((*mf)->diagonal);
        free((*mf)->superior);
        free((*mf)->inferior);
        free(*mf);
        *mf = NULL;
    }
}

```

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

```c
    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 imprime(Matriz* 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");
    }
}

#endif
```

roteiro-07/ex02-01.c

```c
#include "ex02-01.h"

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

enum {
    EXIT = 0,
    CREATE,
    FILL,
    INSERT,
    QUERY,
    PRINT,
    RESET,
    DESTROY
} Options;

int getOption() {
    int option;

    printf("\n====================\n");
    printf("(%d) Criar\n", CREATE);
    printf("(%d) Preencher aleatoriamente\n", FILL);
    printf("(%d) Inserir\n", INSERT);
    printf("(%d) Consultar\n", QUERY);
    printf("(%d) Imprimir\n", PRINT);
    printf("(%d) Zerar\n", RESET);
    printf("(%d) Destruir\n", DESTROY);
    printf("(%d) Sair\n", EXIT);
    printf("====================\n");
    printf("Operacao: ");

    scanf("%d", &option);
    printf("\n");

    return option;
}

int runMenu() {
    Matriz* matriz = NULL;
    int exit = 0, item, pri, linhas, colunas, min, max;

    do {
        switch (getOption()) {
            case CREATE:
                if (matriz != NULL) {
                    destroiMatriz(&matriz);
                    printf("Matriz apagada");
                }

                printf("Insira as dimensoes\n");
                printf("Diagonal: ");
                scanf("%d", &linhas);

                matriz = criaMatriz(linhas);
                break;

```

```
57              case FILL:
58                  printf("Insira os limites dos valores\n");
59                  printf("Min: ");
60                  scanf("%d", &min);
61                  printf("Max: ");
62                  scanf("%d", &max);
63
64                  preencheAleatorio(matriz, min, max);
65                  break;
66
67              case INSERT:
68                  printf("Elemento para inserir: ");
69                  scanf("%d", &item);
70                  printf("Linha (inicio 0): ");
71                  scanf("%d", &linhas);
72                  printf("Colunas (inicio 0): ");
73                  scanf("%d", &colunas);
74
75                  if (insereElem(matriz, item, linhas, colunas)) {
76                      printf("Inserir(%d) (%d, %d)", item, linhas, colunas);
77                  } else {
78                      printf("Nao foi possivel inserir(%d)", item);
79                  }
80                  break;
81
82              case QUERY:
83                  printf("Posicao para consultar");
84                  printf("Linha (inicio 0): ");
85                  scanf("%d", &linhas);
86                  printf("Colunas (inicio 0): ");
87                  scanf("%d", &colunas);
88
89                  if (item = consultaElem(matriz, linhas, colunas)) {
90                      printf("Elemento = %d\n", item);
91                  } else {
92                      printf("Nao foi consultar a matriz nessa posicao");
93                  }
94                  break;
95
96              case PRINT:
97                  imprime(matriz);
98                  break;
99
100             case RESET:
101                 zeraMatriz(matriz);
102                 printf("Matriz zerada");
103                 break;
104
105             case DESTROY:
106                 destroiMatriz(&matriz);
107                 printf("Matriz destruida");
108                 break;
109
110             case EXIT:
111                 if (matriz != NULL) {
112                     destroiMatriz(&matriz);
113                 }
114                 printf("Programa encerrado");
115                 exit = 1;
```

```c
                break;

            default:
                printf("Opcao desconhecida, tente novamente");
        }
        printf("\n");
    } while (!exit);
}

int main() {
    runMenu();
    return 0;
}
```

```
Terminal
gabriel.meira.2004@lab-02-07-04:~/Desktop/ufsj/UFSJ-LabProg2/roteiro-07$ ./ex02-01

====================
(1) Criar
(2) Preencher aleatoriamente
(3) Inserir
(4) Consultar
(5) Imprimir
(6) Zerar
(7) Destruir
(0) Sair
====================
Operacao: 1

Insira as dimensoes
Diagonal: 5


====================
(1) Criar
(2) Preencher aleatoriamente
(3) Inserir
(4) Consultar
(5) Imprimir
(6) Zerar
(7) Destruir
(0) Sair
====================
Operacao:
2

Insira os limites dos valores
Min: -10
Max: 10


====================
(1) Criar
(2) Preencher aleatoriamente
(3) Inserir
(4) Consultar
(5) Imprimir
(6) Zerar
(7) Destruir
(0) Sair
====================
Operacao: 5

Matriz Faixa, Tam: 5 x 5:
Diagonal = [-3 3 -10 9 -4 ]
Superior = [-3 1 6 7 ]
Inferior = [4 -3 -8 -2 ]

Matriz Original:
-3      -3      0       0       0
4       3       1       0       0
0       -3      -10     6       0
0       0       -8      9       7
0       0       0       -2      -4


====================
(1) Criar
(2) Preencher aleatoriamente
(3) Inserir
(4) Consultar
(5) Imprimir
(6) Zerar
(7) Destruir
(0) Sair
====================
Operacao: 3

Elemento para inserir: 23
Linha (inicio 0): 1
Colunas (inicio 0): 1
Inserir(23) (1, 1)

====================
(1) Criar
(2) Preencher aleatoriamente
(3) Inserir
(4) Consultar
(5) Imprimir
(6) Zerar
(7) Destruir
(0) Sair
====================
Operacao: 4

Posicao para consultarLinha (inicio 0): 1
Colunas (inicio 0): 1
Elemento = 23


====================
(1) Criar
(2) Preencher aleatoriamente
(3) Inserir
(4) Consultar
(5) Imprimir
(6) Zerar
(7) Destruir
(0) Sair
====================
Operacao: 5

Matriz Faixa, Tam: 5 x 5:
Diagonal = [-3 23 -10 9 -4 ]
Superior = [-3 1 6 7 ]
Inferior = [4 -3 -8 -2 ]

Matriz Original:
-3      -3      0       0       0
4       23      1       0       0
0       -3      -10     6       0
0       0       -8      9       7
0       0       0       -2      -4


====================
(1) Criar
(2) Preencher aleatoriamente
(3) Inserir
(4) Consultar
(5) Imprimir
(6) Zerar
(7) Destruir
(0) Sair
====================
Operacao: 7

Matriz destruida

====================
(1) Criar
(2) Preencher aleatoriamente
(3) Inserir
(4) Consultar
(5) Imprimir
(6) Zerar
(7) Destruir
(0) Sair
====================
Operacao: 0

Programa encerrado
gabriel.meira.2004@lab-02-07-04:~/Desktop/ufsj/UFSJ-LabProg2/roteiro-07$
```

roteiro-07/ex02-02.h

```c
#ifndef MESPARSACSR_H
#define MESPARSACSR_H

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

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

Matriz* criaMatriz(int l, int c, int qnn) {
    Matriz* ms;
    ms = (Matriz*)malloc(sizeof(Matriz));
    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;
}

```

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

int insereElem(Matriz* 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;
    }
    imprimeVetores(ms);
    return 1;
```

```c
}

int removeElem(Matriz* 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;
    }
    imprimeVetores(ms);
    return 1;
}

int consultaElem(Matriz* 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 imprime(Matriz* ms) {
    if (ms == NULL) return;
    int i, j;
    imprimeVetores(ms);
    printf("Matriz Original:\n");
    for (i = 0; i < ms->lin; i++) {
        for (j = 0; j < ms->col; j++)
```

```
            printf("%d\t", consultaElem(ms, i, j));
        printf("\n");
    }
}

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

#endif
```

roteiro-07/ex02-02.c

```c
#include "ex02-02.h"

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

enum {
    EXIT = 0,
    CREATE,
    INSERT,
    REMOVE,
    QUERY,
    PRINT,
    DESTROY
} Options;

int getOption() {
    int option;

    printf("\n====================\n");
    printf("(%d) Criar\n", CREATE);
    printf("(%d) Inserir\n", INSERT);
    printf("(%d) Remove\n", REMOVE);
    printf("(%d) Consultar\n", QUERY);
    printf("(%d) Imprimir\n", PRINT);
    printf("(%d) Destruir\n", DESTROY);
    printf("(%d) Sair\n", EXIT);
    printf("====================\n");
    printf("Operacao: ");

    scanf("%d", &option);
    printf("\n");

    return option;
}

int runMenu() {
    Matriz* matriz = NULL;
    int exit = 0, item, pri, linhas, colunas, min, max;

    do {
        switch (getOption()) {
            case CREATE:
                if (matriz != NULL) {
                    destroiMatriz(&matriz);
                    printf("Matriz apagada");
                }

                printf("Insira as dimensoes\n");
                printf("Linhas: ");
                scanf("%d", &linhas);
                printf("Colunas: ");
                scanf("%d", &colunas);

                matriz = criaMatriz(linhas, colunas, 0);
                break;
```

```
56
57              case INSERT:
58                  printf("Elemento para inserir: ");
59                  scanf("%d", &item);
60                  printf("Linha (inicio 0): ");
61                  scanf("%d", &linhas);
62                  printf("Colunas (inicio 0): ");
63                  scanf("%d", &colunas);
64
65                  if (insereElem(matriz, item, linhas, colunas)) {
66                      printf("Inseriu(%d) (%d, %d)", item, linhas, colunas);
67                  } else {
68                      printf("Nao foi possivel inserir(%d)", item);
69                  }
70                  break;
71
72              case REMOVE:
73                  printf("Linha (inicio 0): ");
74                  scanf("%d", &linhas);
75                  printf("Colunas (inicio 0): ");
76                  scanf("%d", &colunas);
77
78                  if (removeElem(matriz, linhas, colunas)) {
79                      printf("Removeu (%d) (%d, %d)", item, linhas, colunas);
80                  } else {
81                      printf("Nao foi possivel remover(%d)", item);
82                  }
83
84                  printf("Matriz zerada");
85                  break;
86
87              case QUERY:
88                  printf("Posicao para consultar");
89                  printf("Linha (inicio 0): ");
90                  scanf("%d", &linhas);
91                  printf("Colunas (inicio 0): ");
92                  scanf("%d", &colunas);
93
94                  if (item = consultaElem(matriz, linhas, colunas)) {
95                      printf("Elemento = %d\n", item);
96                  } else {
97                      printf("Nao foi consultar a matriz nessa posicao");
98                  }
99                  break;
100
101             case PRINT:
102                 imprime(matriz);
103                 break;
104
105             case DESTROY:
106                 destroiMatriz(&matriz);
107                 printf("Matriz destruida");
108                 break;
109
110             case EXIT:
111                 if (matriz != NULL) {
112                     destroiMatriz(&matriz);
113                 }
```

```c
                printf("Programa encerrado");
                exit = 1;
                break;

            default:
                printf("Opcao desconhecida, tente novamente");
        }
        printf("\n");
    } while (!exit);
}

int main() {
    runMenu();
    return 0;
}
```

```
Terminal
gabriel.meira.2004@lab-02-07-04:~/Desktop/ufsj/UFSJ-LabProg2/roteiro-07$ ./ex02-02

====================
(1) Criar
(2) Inserir
(3) Remove
(4) Consultar
(5) Imprimir
(6) Destruir
(0) Sair
====================
Operacao: 1

Insira as dimensoes
Linhas: 7
Colunas: 9


====================
(1) Criar
(2) Inserir
(3) Remove
(4) Consultar
(5) Imprimir
(6) Destruir
(0) Sair
====================
Operacao: 2

Elemento para inserir: 23
Linha (inicio 0): 4
Colunas (inicio 0): 6
Matriz Esparsa, Tam: 7 x 9:
1 elementos nao nulos.
A = [23 ]
IA = [0 0 0 0 0 1 1 1 ]
JA = [6 ]

Inseriu(23) (4, 6)

====================
(1) Criar
(2) Inserir
(3) Remove
(4) Consultar
(5) Imprimir
(6) Destruir
(0) Sair
====================
Operacao: 2

Elemento para inserir: 19
Linha (inicio 0): 2
Colunas (inicio 0): 1
Matriz Esparsa, Tam: 7 x 9:
2 elementos nao nulos.
A = [19 23 ]
IA = [0 0 0 1 1 2 2 2 ]
JA = [1 6 ]

Inseriu(19) (2, 1)

====================
(1) Criar
(2) Inserir
(3) Remove
(4) Consultar
(5) Imprimir
(6) Destruir
(0) Sair
====================
Operacao: 4

Posicao para consultarLinha (inicio 0): 2
Colunas (inicio 0): 1
Elemento = 19


====================
(1) Criar
(2) Inserir
(3) Remove
(4) Consultar
(5) Imprimir
(6) Destruir
(0) Sair
====================
Operacao: 5

Matriz Esparsa, Tam: 7 x 9:
2 elementos nao nulos.
A = [19 23 ]
IA = [0 0 0 1 1 2 2 2 ]
JA = [1 6 ]

Matriz Original:
0       0       0       0       0       0       0       0       0
0       0       0       0       0       0       0       0       0
0       19      0       0       0       0       0       0       0
0       0       0       0       0       0       0       0       0
0       0       0       0       0       0       23      0       0
0       0       0       0       0       0       0       0       0
0       0       0       0       0       0       0       0       0


====================
(1) Criar
(2) Inserir
(3) Remove
(4) Consultar
(5) Imprimir
(6) Destruir
(0) Sair
====================
Operacao: 6

Matriz destruida

====================
(1) Criar
(2) Inserir
(3) Remove
(4) Consultar
(5) Imprimir
(6) Destruir
(0) Sair
====================
Operacao: 0

Programa encerrado
gabriel.meira.2004@lab-02-07-04:~/Desktop/ufsj/UFSJ-LabProg2/roteiro-07$
```