Universidade Federal de São João del Rei
Departamento de Ciência da Computação
Curso de Ciência da Computação

Roteiro 7

Adélson de Oliveira Carmo Júnior
212050019

1 Matrizes comuns

1.1 Matriz Sequencial Estática

Código

```c
#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 *);
Matriz* criaMatriz(int, int);
void destroiMatriz(Matriz *);
int preencheAleatorio(Matriz *, int,
    int);
int insereElem(Matriz *, int, int, int);
int consultaElem(Matriz *, int *, int,
    int);
void imprime(Matriz *);
int e_matrizQuadrada(Matriz *);
Matriz* criaTriangularSup(Matriz *);
Matriz* criaTriangularInf(Matriz *);
Matriz* criaDiagonal(Matriz *);
int e_Simetrica(Matriz *);
Matriz* criaTransposta(Matriz *);

#endif
```

codigos/questao11/questao11.h

```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include "questao11.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;
    }
    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");
}

//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)==0){
        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)==0){
        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)==0){
        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)==0){
        printf("Matriz nao
```

```
            Quadrada!\n");
129         return 0;
130     }
131     int i, j;
132     for(i=0; i<mat->lin; i++)
133         for(j=i+1; j<mat->col; j++)
134             if(mat->dados[i][j] !=
                    mat->dados[j][i])
135                 return 0;
136     return 1;
137 }
138
139 Matriz* criaTransposta(Matriz* mat){
140     if(mat == NULL) return NULL;
141     Matriz* t = criaMatriz(mat->col,
            mat->lin);
142     int i, j;
143     for(i=0; i<mat->lin; i++)
144         for(j=0; j<mat->col; j++)
145             t->dados[j][i] =
                    mat->dados[i][j];
146     return t;
147 }
```

codigos/questao11/questao11.c

```
 1 #include <stdio.h>
 2 #include <stdlib.h>
 3 #include <time.h>
 4 #include "questao11.h"
 5
 6 int main(){
 7     int *elemento = (int
           *)malloc(sizeof(int));
 8     Matriz *matriz = criaMatriz(4, 4);
 9     preencheAleatorio(matriz, 0, 9);
10     imprime(matriz);
11
12     insereElem(matriz, 6, 0, 2);
13     consultaElem(matriz, elemento, 0,
           2);
14     printf("O elemento na posicao
           [0][2] da matriz eh: %d\n\n",
           *elemento);
15
16     if(e_matrizQuadrada(matriz)==1)
17         printf("A matriz eh
               quadrada!\n");
18     else
19         printf("A matriz nao eh
               quadrada!\n");
20
21     Matriz *mts =
           criaTriangularSup(matriz);
22     imprime(mts);
23
24     Matriz *mti =
           criaTriangularInf(matriz);
25     imprime(mti);
26
27     Matriz *md = criaDiagonal(matriz);
28     imprime(md);
29
30     Matriz *mt = criaTransposta(matriz);
31     imprime(mt);
32
33     destroiMatriz(matriz);
34
35     return 0;
36 }
```

codigos/questao11/main.c

```
1 all: questao11.o
2   gcc questao11.o main.c -o main
3
4 questao11.o: questao11.h questao11.c
5   gcc -c questao11.c
6
7 clean:
8   rm -f questao11.o main
```

codigos/questao11/Makefile

Saída

```
Matriz 4 x 4:
        8       2       5       5
        0       4       4       8
        1       8       4       5
        6       6       9       9

O elemento na posicao [0][2] da matriz eh: 6

A matriz eh quadrada!
Matriz 4 x 4:
        8       2       6       5
        0       4       4       8
        0       0       4       5
        0       0       0       9

Matriz 4 x 4:
        8       0       0       0
        0       4       0       0
        1       8       4       0
        6       6       9       9
```

Figura 1: Questão 1.1 - Saida 1

```
Matriz 4 x 4:
        8       0       0       0
        0       4       0       0
        0       0       4       0
        0       0       0       9

Matriz 4 x 4:
        8       0       1       6
        2       4       8       6
        6       4       4       9
        5       8       5       9
```

Figura 2: Questão 1.1 - Saida 2

1.2 Matriz Sequencial Dinamica

Código

```c
#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 *);
Matriz* criaMatriz(int, int);
void destroiMatriz(Matriz *);
int preencheAleatorio(Matriz *, int,
    int);
int insereElem(Matriz *, int, int, int);
int consultaElem(Matriz *, int *, int,
    int);
void imprime(Matriz *);

//Matrizes Quadradas e propriedades

int e_matrizQuadrada(Matriz *);
Matriz* criaTriangularSup(Matriz *);
Matriz* criaTriangularInf(Matriz *);
Matriz* criaDiagonal(Matriz *);
int e_Simetrica(Matriz *);
Matriz* criaTransposta(Matriz *);

#endif
```

codigos/questao12/questao12.h

```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include "questao12.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;
```

```
23          mat->col = c;
24          mat->dados = (int**) malloc
                (l*sizeof(int*));
25          for(i=0; i<l; i++)
26              mat->dados[i] = (int*)
                    malloc (c*sizeof(int));
27          zeraMatriz(mat);
28      }
29      return mat;
30  }
31
32  void destroiMatriz(Matriz* mat){
33      if(mat != NULL){
34          int i;
35          for(i=0; i<mat->lin; i++)
36              free(mat->dados[i]);
37          free(mat->dados);
38          free(mat);
39      }
40  }
41
42  int preencheAleatorio(Matriz* mat, int
        ini, int fim){
43      if(mat == NULL) return 0;
44      srand(time(NULL));
45      int i, j;
46      for(i=0; i<mat->lin; i++)
47          for(j=0; j<mat->col; j++)
48              mat->dados[i][j] = ini +
                    rand() % (fim-ini + 1);
49      return 1;
50  }
51
52  int insereElem(Matriz* mat, int elem,
        int l, int c){
53      if(mat == NULL) return 0;
54      if(l < 0 || c < 0 || l > mat->lin
            || c > mat->col){
55          printf("Valores invalidos, elem
                nao inserido!\n");
56          return 0;
57      }
58      mat->dados[l][c] = elem;
59      return 1;
60  }
61
62  int consultaElem(Matriz* mat, int *p,
        int l, int c){
63      if(mat == NULL) return 0;
64      if(l < 0 || c < 0 || l > mat->lin
            || c > mat->col){
65          printf("Valores invalidos, elem
                nao existe!\n");
66          return 0;
67      }
68      *p = mat->dados[l][c];
69      return 1;
70  }
71
72  void imprime(Matriz* mat){
73      if(mat == NULL) return;
74      int i, j;
75      printf("Matriz %d x %d:\n",
            mat->lin, mat->col);
76      for(i=0; i<mat->lin; i++){
77          for(j=0; j<mat->col; j++)
78              printf("%d ",
                    mat->dados[i][j]);
79          printf("\n");
80      }
81      printf("\n");
82  }
83
84  //Matrizes Quadradas e propriedades
85
86  int e_matrizQuadrada(Matriz *mat){
87      if(mat == NULL) return 0;
88      return (mat->lin == mat->col);
89  }
90
91  Matriz* criaTriangularSup(Matriz* mat){
92      if(mat == NULL) return NULL;
93      if(!e_matrizQuadrada(mat)){
94          printf("Matriz nao
                Quadrada!\n");
95          return NULL;
96      }
97      int i, j;
98      Matriz* ts = criaMatriz(mat->lin,
            mat->col);
99      for(i=0; i<mat->lin; i++)
100         for(j=0; j<mat->col; j++)
101             if(i <= j)
102                 ts->dados[i][j] =
                        mat->dados[i][j];
103     return ts;
104 }
105
106 Matriz* criaTriangularInf(Matriz* mat){
107     if(mat == NULL) return NULL;
108     if(!e_matrizQuadrada(mat)){
109         printf("Matriz nao
                Quadrada!\n");
110         return NULL;
111     }
112     int i, j;
113     Matriz* ti = criaMatriz(mat->lin,
            mat->col);
114     for(i=0; i<mat->lin; i++)
115         for(j=0; j<mat->col; j++)
116             if(i >= j)
117                 ti->dados[i][j] =
                        mat->dados[i][j];
118     return ti;
119 }
120
121 Matriz* criaDiagonal(Matriz* mat){
122     if(mat == NULL) return NULL;
```

```c
    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;
}
```

codigos/questao12/questao12.c

```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include "questao12.h"

int main(){
    int *elemento = (int *)malloc(sizeof(int));
    Matriz *matriz = criaMatriz(4, 4);
    preencheAleatorio(matriz, 0, 9);
    imprime(matriz);

    insereElem(matriz, 6, 0, 2);
    consultaElem(matriz, elemento, 0, 2);
    printf("O elemento na posicao [0][2] da matriz eh: %d\n\n", *elemento);

    if(e_matrizQuadrada(matriz)==1)
        printf("A matriz eh quadrada!\n");
    else
        printf("A matriz nao eh quadrada!\n");

    Matriz *mts = criaTriangularSup(matriz);
    imprime(mts);

    Matriz *mti = criaTriangularInf(matriz);
    imprime(mti);

    Matriz *md = criaDiagonal(matriz);
    imprime(md);

    Matriz *mt = criaTransposta(matriz);
    imprime(mt);

    destroiMatriz(matriz);

    return 0;
}
```

codigos/questao12/main.c

```
all: questao12.o
  gcc questao12.o main.c -o main

questao12.o: questao12.h questao12.c
  gcc -c questao12.c

clean:
  rm -f questao12.o main
```

codigos/questao12/Makefile

Saída

```
Matriz 4 x 4:
9 2 0 1
0 5 8 2
8 3 3 7
8 2 1 9

O elemento na posicao [0][2] da matriz eh: 6

A matriz eh quadrada!
Matriz 4 x 4:
9 2 6 1
0 5 8 2
0 0 3 7
0 0 0 9
```

Figura 3: Questão 1.1 - Saida 1

```
Matriz 4 x 4:
9 0 0 0
0 5 0 0
8 3 3 0
8 2 1 9

Matriz 4 x 4:
9 0 0 0
0 5 0 0
0 0 3 0
0 0 0 9

Matriz 4 x 4:
9 0 8 8
2 5 3 2
6 8 3 1
1 2 7 9
```

Figura 4: Questão 1.1 - Saida 2

2 Matrizes especiais

2.1 Matriz de Faixa (Tridiagonal)

Código

```c
#ifndef MFAIXA_H
#define MFAIXA_H

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

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

void zeraMatriz(MFaixa *);
MFaixa* criaMatriz(int);
void destroiMatriz(MFaixa *);
int preencheAleatorio(MFaixa *, int,
    int);
int insereElem(MFaixa *, int, int, int);
int consultaElem(MFaixa *, int, int);
void imprimeFaixaVetores(MFaixa *);
void imprimeFaixa(MFaixa *);

/*Funcao extra*/
void aguardaLimpa();

#endif
```

codigos/questao21/questao21.h

```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include "questao21.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));
```

```
20      if(mf != NULL){
21          if(t <= 1){
22              printf("Dimensao deve ser > 1, matriz nao criada!");
23              return NULL;
24          }
25          mf->tam = t;
26          mf->diagonal = (int*) malloc (t*sizeof(int));
27          mf->superior = (int*) malloc ((t-1)*sizeof(int));
28          mf->inferior = (int*) malloc ((t-1)*sizeof(int));
29          if(mf->diagonal == NULL || mf->superior == NULL || mf->inferior == NULL)
30            return NULL;
31          zeraMatriz(mf);
32      }
33      return mf;
34  }
35
36  void destroiMatriz(MFaixa* mf){
37      if(mf != NULL){
38          free(mf->diagonal);
39          free(mf->superior);
40          free(mf->inferior);
41          free(mf);
42      }
43  }
44
45  int preencheAleatorio(MFaixa* mf, int ini, int fim){
46      if(mf == NULL) return 0;
47      srand(time(NULL));
48      int i;
49      for(i=0; i<mf->tam; i++){
50          mf->diagonal[i] = ini + rand() % (fim-ini + 1);
51          if(i < mf->tam -1){
52              mf->superior[i] = ini + rand() % (fim-ini + 1);
53              mf->inferior[i] = ini + rand() % (fim-ini + 1);
54          }
55      }
56      return 1;
57  }
58
59  int insereElem(MFaixa* mf, int elem, int i, int j){
60    if(mf == NULL) return 0;
61    if(i < 0 || j < 0 || i >= mf->tam || j >= mf->tam){
62      printf("Valores invalidos, elem nao inserido!\n");
63      return 0;
64    }
65    if(i == j) mf->diagonal[i] = elem;
66    else if(i + 1 == j) mf->superior[i] = elem;
67    else if(i == j + 1) mf->inferior[j] = elem;
68    else{
69      printf("Indices fora da faixa, elem nao inserido!\n");
70      return 0;
71    }
72    return 1;
73  }
74
75  int consultaElem(MFaixa* mf, int i, int j){
76    if(mf == NULL) return 0;
77    if(i < 0 || j < 0 || i >= mf->tam || j >= mf->tam){
78      printf("Valores invalidos, elem inexistente!\n");
79      return 0;
80    }
81    if(i == j) return mf->diagonal[i];
82    else if(i + 1 == j) return mf->superior[i];
83    else if(i == j + 1) return mf->inferior[j];
84    else return 0;
85  }
86
87  void imprimeFaixaVetores(MFaixa* mf){
88    if(mf == NULL) return;
89    int i;
90    printf("Matriz Faixa, Tam: %d x %d:\n", mf->tam, mf->tam);
91    printf("Diagonal = [");
92    for(i=0; i<mf->tam; i++)
93      printf("%d ", mf->diagonal[i]);
94    printf("]\n");
95    printf("Superior = [");
96    for(i=0; i<mf->tam-1; i++)
97      printf("%d ", mf->superior[i]);
98    printf("]\n");
99    printf("Inferior = [");
100   for(i=0; i<mf->tam-1; i++)
101     printf("%d ", mf->inferior[i]);
102   printf("]\n\n");
103 }
104
105 void imprimeFaixa(MFaixa* mf){
106   if(mf == NULL) return;
107   int i, j;
108   imprimeFaixaVetores(mf);
109   printf("Matriz Original:\n");
110   for(i=0; i<mf->tam; i++){
111     for(j=0; j<mf->tam; j++)
112       printf("%d\t", consultaElem(mf, i, j));
113     printf("\n");
114   }
```

```c
}

/* Funcao para esperar resposta do usuario e depois limpar*/
void aguardaLimpa(){
    getchar();
    printf("\n\nAperte qualquer tecla para continuar\n");
    getchar();
    system("clear");
}
```

codigos/questao21/questao21.c

```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include "questao21.h"

int main(){
    MFaixa *faixa;
    int escolha, elemento, tamanho, x, y;

    do{
        printf("Digite o que deseja fazer:\n1- Zerar a faixa\n2- Criar a faixa\n"
            "3- Destruir a faixa\n4- Prencher aleatoriamente\n5- Inserir elemento\n"
            "6- Consultar elemento\n7- Imprimir faixa vetores\n8- Imprimir faixa\n0- Sair\n");
        scanf("%d",&escolha);

        switch (escolha){
        case 0:
            break;

        case 1:
            zeraMatriz(faixa);
            printf("Faixa zerada!\n");
            aguardaLimpa();
            break;

        case 2:
            printf("Digite o tamanho da faixa\n");
            scanf("%d", &tamanho);
            faixa = criaMatriz(tamanho);
            printf("Faixa criada!");
            aguardaLimpa();
            break;

        case 3:
            destroiMatriz(faixa);
            printf("Faixa destruida!");
            aguardaLimpa();
            break;

        case 4:
            preencheAleatorio(faixa, 0, 9);
            printf("Faixa preenchida aleatoriamente!");
            aguardaLimpa();
            break;

        case 5:
            printf("Digite o elemento que deseja inserir, assim como sua posicao\n");
            scanf("%d %d %d", &elemento, &x, &y);
            insereElem(faixa, elemento, x, y);
            aguardaLimpa();
            break;

        case 6:
            printf("O elemento na posicao [2][3] da faixa eh: %d", consultaElem(faixa, 2, 3));
            aguardaLimpa();
            break;

        case 7:
            imprimeFaixaVetores(faixa);
            aguardaLimpa();
            break;

        case 8:
            imprimeFaixa(faixa);
            aguardaLimpa();
            break;

        default:
            printf("Algo de errado nao esta certo!");
            break;
        }
    }while(escolha!=0);

    return 0;
}
```

codigos/questao21/main.c

```
all: questao21.o
  gcc questao21.o main.c -o main

questao21.o: questao21.h questao21.c
  gcc -c questao21.c

clean:
  rm -f questao21.o main
```

codigos/questao21/Makefile

Saída

```
Digite o que deseja fazer:
1- Zerar a faixa
2- Criar a faixa
3- Destruir a faixa
4- Prencher aleatoriamente
5- Inserir elemento
6- Consultar elemento
7- Imprimir faixa vetores
8- Imprimir faixa
0- Sair
2
Digite o tamanho da faixa
5
Faixa criada!
```

Figura 5: Questão 2.1 - Cria matriz

```
Digite o que deseja fazer:
1- Zerar a faixa
2- Criar a faixa
3- Destruir a faixa
4- Prencher aleatoriamente
5- Inserir elemento
6- Consultar elemento
7- Imprimir faixa vetores
8- Imprimir faixa
0- Sair
4
Faixa preenchida aleatoriamente!
```

Figura 6: Questão 2.1 - Preenche aleatoriamente

```
Digite o que deseja fazer:
1- Zerar a faixa
2- Criar a faixa
3- Destruir a faixa
4- Prencher aleatoriamente
5- Inserir elemento
6- Consultar elemento
7- Imprimir faixa vetores
8- Imprimir faixa
0- Sair
8
Matriz Faixa, Tam: 5 x 5:
Diagonal = [7 2 1 4 4 ]
Superior = [8 1 2 9 ]
Inferior = [2 8 8 6 ]

Matriz Original:
7       8       0       0       0
2       2       1       0       0
0       8       1       2       0
0       0       8       4       9
0       0       0       6       4
```

Figura 7: Questão 2.1 - Imprime

```
Digite o que deseja fazer:
1- Zerar a faixa
2- Criar a faixa
3- Destruir a faixa
4- Prencher aleatoriamente
5- Inserir elemento
6- Consultar elemento
7- Imprimir faixa vetores
8- Imprimir faixa
0- Sair
1
Faixa zerada!
```

Figura 8: Questão 2.1 - Zera matriz

```
Digite o que deseja fazer:
1- Zerar a faixa
2- Criar a faixa
3- Destruir a faixa
4- Prencher aleatoriamente
5- Inserir elemento
6- Consultar elemento
7- Imprimir faixa vetores
8- Imprimir faixa
0- Sair
7
Matriz Faixa, Tam: 5 x 5:
Diagonal = [0 0 0 0 0 ]
Superior = [0 0 0 0 ]
Inferior = [0 0 0 0 ]
```

Figura 9: Questão 2.1 - Imprime os vetores

```
Digite o que deseja fazer:
1- Zerar a faixa
2- Criar a faixa
3- Destruir a faixa
4- Prencher aleatoriamente
5- Inserir elemento
6- Consultar elemento
7- Imprimir faixa vetores
8- Imprimir faixa
0- Sair
5
Digite o elemento que deseja inserir, assim como sua posicao
4
2
3
```

Figura 10: Questão 2.1 - Insere elemento

```
Digite o que deseja fazer:
1- Zerar a faixa
2- Criar a faixa
3- Destruir a faixa
4- Prencher aleatoriamente
5- Inserir elemento
6- Consultar elemento
7- Imprimir faixa vetores
8- Imprimir faixa
0- Sair
6
O elemento na posicao [2][3] da faixa eh: 4
```

Figura 11: Questão 2.1 - Consulta a posição [2][3]

```
Digite o que deseja fazer:
1- Zerar a faixa
2- Criar a faixa
3- Destruir a faixa
4- Prencher aleatoriamente
5- Inserir elemento
6- Consultar elemento
7- Imprimir faixa vetores
8- Imprimir faixa
0- Sair
3
Faixa destruida!
```

Figura 12: Questão 2.1 - Destroi matriz

2.2 Matriz Esparsa CSR

Nesta seção foi usado o código da Matriz Sequencial Dinamica, então para o funcionamento dele é necessário também usar a biblioteca do tópico 1.2. Código

```c
#ifndef MESPARSACSR_H
#define MESPARSACSR_H

#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include "questao12.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, int, int);
int* meuRealloc(int *, int);
void imprimeEsparsaVetores(MEsparsaCSR *);
int insereElemEsparsa(MEsparsaCSR *, int, int, int);
int removeElemEsparsa(MEsparsaCSR *, int, int);
MEsparsaCSR* transformarEmCSR(Matriz *);
int consultaElemEsparsa(MEsparsaCSR *, int, int);
void imprimeEsparsa(MEsparsaCSR *);
void destroiMatrizEsparsa(MEsparsaCSR *);

/*Funcao extra*/
void aguardaLimpa();
```

```
30
31 #endif
```

codigos/questao22/questao22.h

```
 1 #include <stdio.h>
 2 #include <stdlib.h>
 3 #include <time.h>
 4 #include "questao22.h"
 5
 6
 7 MEsparsaCSR* criaMatrizEsparsa(int l,
       int c, int qnn){
 8     MEsparsaCSR *ms;
 9     ms = (MEsparsaCSR*) malloc
           (sizeof(MEsparsaCSR));
10     if(ms != NULL){
11         if(l <= 0 || c <= 0 || qnn < 0){
12             printf("Valores invalidos,
                   matriz nao criada!\n");
13             return NULL;
14         }
15         ms->lin = l; ms->col = c;
16         ms->QI = 0; ms->QNN = qnn;
17         ms->A = ms->IA = ms->JA = NULL;
18         if(qnn != 0){
19             ms->A = (int*) malloc
                   (qnn*sizeof(int));
20             ms->JA = (int*) malloc
                   (qnn*sizeof(int));
21             if(ms->A == NULL || ms->JA
                   == NULL) return NULL;
22         }
23         ms->IA = (int*) malloc
               ((ms->lin+1)*sizeof(int));
24         if(ms->IA == NULL) return NULL;
25         int i; for(i=0; i<l+1; i++)
               ms->IA[i] = 0;
26     }
27     return ms;
28 }
29
30 int* meuRealloc(int* v, int tam){
31     int* aux = (int*) malloc
           ((tam+1)*sizeof(int));
32     if(aux != NULL){
33         if(v != NULL){
34             int i;
35             for(i=0; i<tam; i++)
36                 aux[i] = v[i];
37             free(v);
38         }
39     }
40     return aux;
41 }
42
43 void imprimeEsparsaVetores(MEsparsaCSR*
       ms){
44   if(ms == NULL) return;
45   int i, j;
46   printf("Matriz Esparsa, Tam: %d x
         %d:\n", ms->lin, ms->col);
47   printf("%d elementos nao nulos.\n",
         ms->QNN);
48   printf("A = [");
49   for(i=0; i<ms->QNN; i++)
50     printf("%d ", ms->A[i]);
51   printf("]\n");
52   printf("IA = [");
53   for(i=0; i<ms->lin+1; i++)
54     printf("%d ", ms->IA[i]);
55   printf("]\n");
56   printf("JA = [");
57   for(i=0; i<ms->QNN; i++)
58     printf("%d ", ms->JA[i]);
59   printf("]\n\n");
60 }
61
62 int insereElemEsparsa(MEsparsaCSR *ms,
       int elem, int i, int j) {
63     if(ms == NULL) return 0;
64     if(i < 0 || j < 0 || i >= ms->lin
           || j >= ms->col){
65         printf("Valores invalidos, elem
               nao inserido!\n");
66         return 0;
67     }
68     int k;
69     int index = -1;
70     int ini = ms->IA[i]; int fim =
           ms->IA[i+1];
71     for(k = ini; k<fim; k++)
72         if (ms->JA[k] >= j) {
73             index = k;
74             break;
75         }
76
77     if (index == -1) {
78         if(ms->QI == ms->QNN){
79             ms->A = meuRealloc(ms->A,
                   ms->QNN);
80             ms->JA = meuRealloc(ms->JA,
                   ms->QNN);
81             ms->QNN++;
82         }
83         for(k = ms->QNN-1; k>=fim; k--){
84             ms->A[k] = ms->A[k-1];
85             ms->JA[k] = ms->JA[k-1];
```

```
86          }
87          ms->A[fim] = elem;
88          ms->JA[fim] = j;
89          ms->QI++;
90          for (int k = i+1; k<=ms->lin;
                k++)
91              ms->IA[k]++;
92      } else {
93          ms->A[index] = elem;
94      }
95      imprimeEsparsaVetores(ms);
96      return 1;
97  }
98
99  int removeElemEsparsa(MEsparsaCSR *ms,
        int i, int j) {
100     if(ms == NULL) return 0;
101     if(i < 0 || j < 0 || i >= ms->lin
            || j >= ms->col){
102         printf("Valores invalidos, elem
                nao removido!\n");
103         return 0;
104     }
105
106     int k;
107     int index = -1;
108     int ini = ms->IA[i]; int fim =
            ms->IA[i+1];
109     for(k = ini; k<fim; k++)
110         if (ms->JA[k] == j) {
111             index = k;
112             break;
113         }
114
115     if (index != -1) {
116         for (k = index; k < ms->QNN -
                1; k++) {
117             ms->A[k] = ms->A[k+1];
118             ms->JA[k] = ms->JA[k+1];
119         }
120         ms->QNN--;
121         ms->QI--;
122         for (int k = i+1; k<=ms->lin;
                k++)
123             ms->IA[k]--;
124     }else{
125         printf("Elemento nao
                existente\n"); return 0;
126     }
127     imprimeEsparsaVetores(ms);
128     return 1;
129 }
130
131
132 MEsparsaCSR* transformarEmCSR(Matriz*
        mat){
133     MEsparsaCSR *ms =
            criaMatrizEsparsa(mat->lin,
            mat->col, 0);
134     if(ms != NULL){
135         if(mat == NULL){
136             printf("Matriz de entrada
                    inexistente!\n");
137             return NULL;
138         }
139
140         int i, j;
141         for(i=0; i<mat->lin; i++)
142             for(j=0; j<mat->col; j++)
143                 if(mat->dados[i][j] !=
                        0)
144                     insereElemEsparsa(ms,
                            mat->dados[i][j],
                            i, j);
145     }
146     return ms;
147 }
148
149 int consultaElemEsparsa(MEsparsaCSR*
        ms, int i, int j){
150   if(ms == NULL) return 0;
151   if(i < 0 || j < 0 || i >= ms->lin ||
          j >= ms->col){
152     printf("Valores invalidos, elem
            inexistente!\n");
153     return 0;
154   }
155   int k;
156   for(k = ms->IA[i]; k<ms->IA[i+1]; k++)
157     if(ms->JA[k] == j) return ms->A[k];
158   return 0;
159 }
160
161 void imprimeEsparsa(MEsparsaCSR* ms){
162   if(ms == NULL) return;
163   int i, j;
164   imprimeEsparsaVetores(ms);
165   printf("Matriz Original:\n");
166   for(i=0; i<ms->lin; i++){
167     for(j=0; j<ms->col; j++)
168       printf("%d\t",
              consultaElemEsparsa(ms, i, j));
169     printf("\n");
170   }
171 }
172
173 void destroiMatrizEsparsa(MEsparsaCSR*
        ms){
174     if(ms != NULL){
175         free(ms->A);
176         free(ms->IA);
177         free(ms->JA);
178         free(ms);
179     }
180 }
181
182 /* Funcao para esperar resposta do
        usuario e depois limpar*/
```

```c
void aguardaLimpa(){
    getchar();
    printf("\n\nAperte qualquer tecla para continuar\n");
    getchar();
    system("clear");
}
```

codigos/questao22/questao22.c

```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include "questao22.h"
#include "questao12.h"

int main(){
    MEsparsaCSR *matriz_csr, *t_matriz_csr;
    Matriz *matriz;
    int escolha, elemento, linhas, colunas, x, y;

    do{
        printf("Digite o que deseja fazer:\n1- Criar matriz_csr\n2- Imprime os vetores\n"
            "3- Inserir elemento\n4- Remover elemento\n5- Transformar em CSR\n"
            "6- Consultar elemento\n7- Imprimir a matriz_csr\n8- Destruir a matriz_csr\n0- Sair\n");
        scanf("%d",&escolha);

        switch (escolha){
        case 0:
            break;

        case 1:
            printf("Digite as linha e colunas da matriz_csr\n");
            scanf("%d %d", &linhas, &colunas);
            matriz_csr = criaMatrizEsparsa(linhas, colunas, 0);
            printf("Matriz criada!");
            aguardaLimpa();
            break;

        case 2:
            imprimeEsparsaVetores(matriz_csr);
            aguardaLimpa();
            break;

        case 3:
            printf("Digite o elemento que deseja inserir, assim como sua posicao\n");
            scanf("%d %d %d", &elemento, &x, &y);
            insereElemEsparsa(matriz_csr, elemento, x, y);
            aguardaLimpa();
            break;

        case 4:
            printf("Digite as posicoes do elemento que deseja remover\n");
            scanf("%d %d", &x, &y);
            removeElemEsparsa(matriz_csr, x, y);
            aguardaLimpa();
            break;

        case 5:
            matriz = criaMatriz(4,4);
            preencheAleatorio(matriz, 0, 9);
            imprime(matriz);
            t_matriz_csr = transformarEmCSR(matriz);
            imprimeEsparsa(t_matriz_csr);
            aguardaLimpa();
            destroiMatriz(matriz);
            destroiMatrizEsparsa(t_matriz_csr);
            break;

        case 6:
            printf("O elemento na posicao [0][2] da matriz_csr eh: %d", consultaElemEsparsa(matriz_csr, 0, 2));
            aguardaLimpa();
            break;

        case 7:
            imprimeEsparsa(matriz_csr);
            aguardaLimpa();
            break;

        case 8:
            destroiMatrizEsparsa(matriz_csr);
            printf("Matriz destruida!");
```

```
73              aguardaLimpa();
74              break;
75
76          default:
77              printf("Algo de errado nao
                    esta certo!");
78              break;
79          }
80      }while(escolha!=0);
81
82      return 0;
83  }
```

codigos/questao22/main.c

```
1  all: questao12.o questao22.o
2    gcc questao12.o questao22.o main.c -o
         main
3
4  questao12.o: questao12.h questao12.c
5    gcc -c questao12.c
6
7  questao22.o: questao22.h questao22.c
8    gcc -c questao22.c
9
10 clean:
11   rm -f questao12.o questao22.o main
```

codigos/questao22/Makefile

Saída

```
Digite o que deseja fazer:
1- Criar matriz_csr
2- Imprime os vetores
3- Inserir elemento
4- Remover elemento
5- Transformar em CSR
6- Consultar elemento
7- Imprimir a matriz_csr
8- Destruir a matriz_csr
0- Sair
1
Digite as linha e colunas da matriz_csr
4
5
Matriz criada!
```

Figura 13: Questão 2.2 - Cria matriz

```
Digite o que deseja fazer:
1- Criar matriz_csr
2- Imprime os vetores
3- Inserir elemento
4- Remover elemento
5- Transformar em CSR
6- Consultar elemento
7- Imprimir a matriz_csr
8- Destruir a matriz_csr
0- Sair
2
Matriz Esparsa, Tam: 4 x 5:
0 elementos nao nulos.
A = []
IA = [0 0 0 0 0 ]
JA = []
```

Figura 14: Questão 2.2 - Imprime vetores

```
Digite o que deseja fazer:
1- Criar matriz_csr
2- Imprime os vetores
3- Inserir elemento
4- Remover elemento
5- Transformar em CSR
6- Consultar elemento
7- Imprimir a matriz_csr
8- Destruir a matriz_csr
0- Sair
3
Digite o elemento que deseja inserir, assim como sua posicao
4
1
1
Matriz Esparsa, Tam: 4 x 5:
1 elementos nao nulos.
A = [4 ]
IA = [0 0 1 1 1 ]
JA = [1 ]
```

Figura 15: Questão 2.2 - Insere elemento

```
Digite o que deseja fazer:
1- Criar matriz_csr
2- Imprime os vetores
3- Inserir elemento
4- Remover elemento
5- Transformar em CSR
6- Consultar elemento
7- Imprimir a matriz_csr
8- Destruir a matriz_csr
0- Sair
7
Matriz Esparsa, Tam: 4 x 5:
3 elementos nao nulos.
A = [4 8 3 ]
IA = [0 0 2 3 3 ]
JA = [1 2 0 ]

Matriz Original:
0       0       0       0       0
0       4       8       0       0
3       0       0       0       0
0       0       0       0       0
```

Figura 16: Questão 2.2 - Imprime

```
Digite o que deseja fazer:
1- Criar matriz_csr
2- Imprime os vetores
3- Inserir elemento
4- Remover elemento
5- Transformar em CSR
6- Consultar elemento
7- Imprimir a matriz_csr
8- Destruir a matriz_csr
0- Sair
4
Digite as posicoes do elemento que deseja
2
0
Matriz Esparsa, Tam: 4 x 5:
2 elementos nao nulos.
A = [4 8 ]
IA = [0 0 2 2 2 ]
JA = [1 2 ]
```

Figura 17: Questão 2.2 - Remove elemento

```
Digite o que deseja fazer:
1- Criar matriz_csr
2- Imprime os vetores
3- Inserir elemento
4- Remover elemento
5- Transformar em CSR
6- Consultar elemento
7- Imprimir a matriz_csr
8- Destruir a matriz_csr
0- Sair
6
O elemento na posicao [0][2] da matriz_csr eh: 0
```

Figura 18: Questão 2.2 - Consulta posição [0][2]

```
Digite o que deseja fazer:
1- Criar matriz_csr
2- Imprime os vetores
3- Inserir elemento
4- Remover elemento
5- Transformar em CSR
6- Consultar elemento
7- Imprimir a matriz_csr
8- Destruir a matriz_csr
0- Sair
5
Matriz 4 x 4:
7 0 3 4
6 5 4 1
1 1 9 6
3 0 7 1
```

Figura 19: Questão 2.2 - Transformação para CSR

```
Matriz Esparsa, Tam: 4 x 4:
14 elementos nao nulos.
A = [7 3 4 6 5 4 1 1 1 9 6 3 7 1 ]
IA = [0 3 7 11 14 ]
JA = [0 2 3 0 1 2 3 0 1 2 3 0 2 3 ]

Matriz Original:
7       0       3       4
6       5       4       1
1       1       9       6
3       0       7       1
```

Figura 20: Questão 2.2 - Transformação para CSR

```
Digite o que deseja fazer:
1- Criar matriz_csr
2- Imprime os vetores
3- Inserir elemento
4- Remover elemento
5- Transformar em CSR
6- Consultar elemento
7- Imprimir a matriz_csr
8- Destruir a matriz_csr
0- Sair
8
Matriz destruida!
```

Figura 21: Questão 2.2 - Destroi matriz