UFSJ - Ciências da Computação

Laboratório de Programação 2

Roteiro 7

Nome: Geraldo Arthur Detomi

1.1) TAD: Matriz Sequencial Estática

roteiro_7/matriz_estatica.h

```h
#ifndef MATRIZ_ESTATICA_H
#define MATRIZ_ESTATICA_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);

Matriz *criaMatriz(int l, int c);

void destroiMatriz(Matriz *mat);

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

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

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

void imprime(Matriz *mat);

#endif
```

roteiro_7/matriz_estatica.c

```c
#include "./matriz_estatica.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);
  mat = NULL;
}

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

```c
  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");
}
```

roteiro_7/1-1.c

```c
#include "./matriz_estatica.h"

#define MAX_OPTIONS 8

enum options {
  CRIAR = 0,
  PREENCHER,
  ZERAR,
  INSERIR,
  CONSULTAR,
  IMPRIMIR,
  DESTRUIR,
  SAIR,
};

int get_option() {
  int opt = -1;
  do {
    printf("\tOperacoes\n");
    printf("[%d] Criar matriz, ", CRIAR);
    printf("[%d] Preencher aleatoriamente, ", PREENCHER);
    printf("[%d] Zerar matriz, ", ZERAR);
    printf("[%d] Inserir elemento, ", INSERIR);
    printf("[%d] Consultar elemento, ", CONSULTAR);
    printf("[%d] Imprimir matriz, ", IMPRIMIR);
    printf("[%d] Destruir matriz, ", DESTRUIR);
    printf("[%d] Sair do programa \n", SAIR);

    printf("\nInsira a opção desejada: ");
    scanf("%d", &opt);
    printf("\n");

    if (opt < 0 || opt >= MAX_OPTIONS) {
      printf("Opção escolhida inválida!\n");
    }

  } while (opt < 0 || opt >= MAX_OPTIONS);

```

```c
  return opt;
}

int get_valor(char *msg) {
  int value;

  printf("%s", msg);
  scanf("%d", &value);

  return value;
}

int main() {
  int opt, valor, linha, coluna;

  Matriz *matrix = NULL;

  do {
    opt = get_option();

    switch (opt) {
    case CRIAR:
      printf("Executando comando...\n");

      printf("Insira as dimensões da matriz:\n");
      linha = get_valor("Linhas: ");
      coluna = get_valor("Colunas: ");

      if (matrix == NULL) {
        matrix = criaMatriz(linha, coluna);
      } else {
        destroiMatriz(matrix);
        matrix = NULL;
        matrix = criaMatriz(linha, coluna);
      }

      printf("Matriz criada com sucesso!\n");
      break;
    case PREENCHER:
      printf("Executando comando...\n");

      if (matrix == NULL) {
        printf("Matriz não incializada impossível realizar operação..\n");
        break;
      }

      printf("Insira os limites dos valores a serem preenchidos\n");

      int min, max;

      min = get_valor("Min: ");
      max = get_valor("Max: ");

      preencheAleatorio(matrix, min, max);

      printf("Matriz preenchida com sucesso!\n");

      break;
    case ZERAR:
      printf("Executando comando...\n");

      if (matrix == NULL) {
        printf("Matriz não incializada impossível realizar operação..\n");
        break;
      }

      zeraMatriz(matrix);

      printf("Matriz zerada com sucesso!\n");

      break;
    case INSERIR:
      printf("Executando comando...\n");

      if (matrix == NULL) {
        printf("Matriz não incializada impossível realizar operação..\n");
        break;
      }

      printf("Insira a posição para inserir o elemento na matriz:\n");

```

```
120       linha = get_valor("Linha :");
121       coluna = get_valor("Coluna :");
122       valor = get_valor("Valor :");
123
124       if (insereElem(matrix, valor, linha, coluna)) {
125         printf("Valor inserido com sucesso!\n");
126       }
127
128       break;
129     case CONSULTAR:
130       printf("Executando comando...\n");
131
132       if (matrix == NULL) {
133         printf("Matriz não incializada impossível realizar operação..\n");
134         break;
135       }
136
137       printf("Insira a posição do elemento a ser consultado:\n");
138
139       linha = get_valor("Linha :");
140       coluna = get_valor("Coluna :");
141
142       if (consultaElem(matrix, &valor, linha, coluna)) {
143         printf("matriz[%d][%d] = %d\n", linha, coluna, valor);
144         printf("Elemento consultado com sucesso!");
145       }
146       break;
147     case IMPRIMIR:
148       printf("Executando comando...\n");
149
150       if (matrix == NULL) {
151         printf("Matriz não incializada impossível realizar operação..\n");
152         break;
153       }
154
155       imprime(matrix);
156
157       printf("Matriz imprimida com sucesso!\n");
158
159       break;
160     case DESTRUIR:
161       printf("Executando comando...\n");
162
163       destroiMatriz(matrix);
164       matrix = NULL;
165
166       printf("Matriz destruida com sucesso!\n");
167
168       break;
169     case SAIR:
170       if (matrix != NULL) {
171         destroiMatriz(matrix);
172         matrix = NULL;
173       }
174
175       printf("Finalizando programa! Até mais!\n");
176       break;
177     }
178
179   } while (opt != SAIR);
180
181   return 0;
182 }
```

Saída do terminal:

```
arthurdetomi at arthurdetomi-System-Product-Name in ~/Documents/UFSJ-Graduacao/UFSJ-2025_1/Lab_Prog_2/roteiro_7 on main*** 25-05-12 - 21:15:15
└─.: ./1-1.out
        Operacoes
[0] Criar matriz, [1] Preencher aleatoriamente, [2] Zerar matriz, [3] Inserir elemento, [4] Consultar elemento, [5] Imprimir matriz, [6] Destruir matriz,

Insira a opção desejada: 0

Executando comando...
Insira as dimensões da matriz:
Linhas: 3
Colunas: 3
Matriz criada com sucesso!
        Operacoes
[0] Criar matriz, [1] Preencher aleatoriamente, [2] Zerar matriz, [3] Inserir elemento, [4] Consultar elemento, [5] Imprimir matriz, [6] Destruir matriz,

Insira a opção desejada: 1

Executando comando...
Insira os limites dos valores a serem preenchidos
Min: 0
Max: 5
Matriz preenchida com sucesso!
        Operacoes
[0] Criar matriz, [1] Preencher aleatoriamente, [2] Zerar matriz, [3] Inserir elemento, [4] Consultar elemento, [5] Imprimir matriz, [6] Destruir matriz,

Insira a opção desejada: 5

Executando comando...
Matriz 3 x 3:
        5       2       1
        1       3       0
        1       1       3

Matriz imprimida com sucesso!
        Operacoes
[0] Criar matriz, [1] Preencher aleatoriamente, [2] Zerar matriz, [3] Inserir elemento, [4] Consultar elemento, [5] Imprimir matriz, [6] Destruir matriz,

Insira a opção desejada: 4

Executando comando...
Insira a posição do elemento a ser consultado:
Linha :2
Coluna :1
matriz[2][1] = 1
Elemento consultado com sucesso!        Operacoes
[0] Criar matriz, [1] Preencher aleatoriamente, [2] Zerar matriz, [3] Inserir elemento, [4] Consultar elemento, [5] Imprimir matriz, [6] Destruir matriz,

Insira a opção desejada: 7

Finalizando programa! Até mais!
```

1.2) TAD: Matriz Sequencial Dinâmica

roteiro_7/matriz_dinamica.h

```c
#ifndef MATRIZ_DINAMICA_H
#define MATRIZ_DINAMICA_H

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

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

void zeraMatriz(Matriz *mat);

Matriz *criaMatriz(int l, int c);

void destroiMatriz(Matriz *mat);

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

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

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

void imprime(Matriz *mat);

#endif
```

roteiro_7/matriz_dinamica.c

```c
#include "./matriz_dinamica.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;
```

```c
  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]);
      mat->dados[i] = NULL;
    }

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

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");
}
```

roteiro_7/1-2.c

```c
#include "./matriz_dinamica.h"

```

```c
#define MAX_OPTIONS 8

enum options {
  CRIAR = 0,
  PREENCHER,
  ZERAR,
  INSERIR,
  CONSULTAR,
  IMPRIMIR,
  DESTRUIR,
  SAIR,
};

int get_option() {
  int opt = -1;
  do {
    printf("\tOperacoes\n");
    printf("[%d] Criar matriz, ", CRIAR);
    printf("[%d] Preencher aleatoriamente, ", PREENCHER);
    printf("[%d] Zerar matriz, ", ZERAR);
    printf("[%d] Inserir elemento, ", INSERIR);
    printf("[%d] Consultar elemento, ", CONSULTAR);
    printf("[%d] Imprimir matriz, ", IMPRIMIR);
    printf("[%d] Destruir matriz, ", DESTRUIR);
    printf("[%d] Sair do programa \n", SAIR);

    printf("\nInsira a opção desejada: ");
    scanf("%d", &opt);
    printf("\n");

    if (opt < 0 || opt >= MAX_OPTIONS) {
      printf("Opção escolhida inválida!\n");
    }

  } while (opt < 0 || opt >= MAX_OPTIONS);

  return opt;
}

int get_valor(char *msg) {
  int value;

  printf("%s", msg);
  scanf("%d", &value);

  return value;
}

int main() {
  int opt, valor, linha, coluna;

  Matriz *matrix = NULL;

  do {
    opt = get_option();

    switch (opt) {
    case CRIAR:
      printf("Executando comando...\n");

      printf("Insira as dimensões da matriz:\n");
      linha = get_valor("Linhas: ");
      coluna = get_valor("Colunas: ");

      if (matrix == NULL) {
        matrix = criaMatriz(linha, coluna);
      } else {
        destroiMatriz(matrix);
        matrix = NULL;
        matrix = criaMatriz(linha, coluna);
      }

      printf("Matriz criada com sucesso!\n");
      break;
    case PREENCHER:
      printf("Executando comando...\n");

      if (matrix == NULL) {
        printf("Matriz não incializada impossível realizar operação..\n");
        break;
      }
```

```
84
85        printf("Insira os limites dos valores a serem preenchidos\n");
86
87        int min, max;
88
89        min = get_valor("Min: ");
90        max = get_valor("Max: ");
91
92        preencheAleatorio(matrix, min, max);
93
94        printf("Matriz preenchida com sucesso!\n");
95
96        break;
97      case ZERAR:
98        printf("Executando comando...\n");
99
100       if (matrix == NULL) {
101         printf("Matriz não incializada impossível realizar operação..\n");
102         break;
103       }
104
105       zeraMatriz(matrix);
106
107       printf("Matriz zerada com sucesso!\n");
108
109       break;
110     case INSERIR:
111       printf("Executando comando...\n");
112
113       if (matrix == NULL) {
114         printf("Matriz não incializada impossível realizar operação..\n");
115         break;
116       }
117
118       printf("Insira a posição para inserir o elemento na matriz:\n");
119
120       linha = get_valor("Linha :");
121       coluna = get_valor("Coluna :");
122       valor = get_valor("Valor :");
123
124       if (insereElem(matrix, valor, linha, coluna)) {
125         printf("Valor inserido com sucesso!\n");
126       }
127
128       break;
129     case CONSULTAR:
130       printf("Executando comando...\n");
131
132       if (matrix == NULL) {
133         printf("Matriz não incializada impossível realizar operação..\n");
134         break;
135       }
136
137       printf("Insira a posição do elemento a ser consultado:\n");
138
139       linha = get_valor("Linha :");
140       coluna = get_valor("Coluna :");
141
142       if (consultaElem(matrix, &valor, linha, coluna)) {
143         printf("matriz[%d][%d] = %d\n", linha, coluna, valor);
144         printf("Elemento consultado com sucesso!");
145       }
146       break;
147     case IMPRIMIR:
148       printf("Executando comando...\n");
149
150       if (matrix == NULL) {
151         printf("Matriz não incializada impossível realizar operação..\n");
152         break;
153       }
154
155       imprime(matrix);
156
157       printf("Matriz imprimida com sucesso!\n");
158
159       break;
160     case DESTRUIR:
161       printf("Executando comando...\n");
162
163       destroiMatriz(matrix);
164       matrix = NULL;
```

```
165
166       printf("Matriz destruida com sucesso!\n");
167
168       break;
169     case SAIR:
170       if (matrix != NULL) {
171         destroiMatriz(matrix);
172         matrix = NULL;
173       }
174
175       printf("Finalizando programa! Até mais!\n");
176       break;
177     }
178
179   } while (opt != SAIR);
180
181   return 0;
182 }
```

Saída do terminal:

```
└─ .:: ./1-2.out
       Operacoes
[0] Criar matriz, [1] Preencher aleatoriamente, [2] Zerar matriz, [3] Inserir elemento, [4] Consultar elemento, [5] Imprimir matriz, [6] Destruir matriz

Insira a opção desejada: 0

Executando comando...
Insira as dimensões da matriz:
Linhas: 2
Colunas: 2
Matriz criada com sucesso!
       Operacoes
[0] Criar matriz, [1] Preencher aleatoriamente, [2] Zerar matriz, [3] Inserir elemento, [4] Consultar elemento, [5] Imprimir matriz, [6] Destruir matriz

Insira a opção desejada: 1

Executando comando...
Insira os limites dos valores a serem preenchidos
Min: 0
Max: 6
Matriz preenchida com sucesso!
       Operacoes
[0] Criar matriz, [1] Preencher aleatoriamente, [2] Zerar matriz, [3] Inserir elemento, [4] Consultar elemento, [5] Imprimir matriz, [6] Destruir matriz

Insira a opção desejada: 5

Executando comando...
Matriz 2 x 2:
1 2
6 4

Matriz imprimida com sucesso!
       Operacoes
[0] Criar matriz, [1] Preencher aleatoriamente, [2] Zerar matriz, [3] Inserir elemento, [4] Consultar elemento, [5] Imprimir matriz, [6] Destruir matriz

Insira a opção desejada: 3

Executando comando...
Insira a posição para inserir o elemento na matriz:
Linha :0
Coluna :1
Valor :2
Valor inserido com sucesso!
       Operacoes
[0] Criar matriz, [1] Preencher aleatoriamente, [2] Zerar matriz, [3] Inserir elemento, [4] Consultar elemento, [5] Imprimir matriz, [6] Destruir matriz

Insira a opção desejada: 7

Finalizando programa! Até mais!
```

2.1) TAD: Matriz de Faixa (Tridiagonal)

roteiro_7/matriz_faixa.h

```
1  #ifndef MATRIZ_FAIXA_H
2  #define MATRIZ_FAIXA_H
3
4  #include <stdio.h>
5  #include <stdlib.h>
6  #include <time.h>
7
8  typedef struct {
9    int *diagonal;
10   int *superior;
11   int *inferior;
12   int tam;
13 } Matriz;
14
15 void zeraMatriz(Matriz *mf);
16
17 Matriz *criaMatriz(int t);
18
19 void destroiMatriz(Matriz **mf);
```

```
20
21 int preencheAleatorio(Matriz *mf, int ini, int fim);
22
23 int insereElem(Matriz *mf, int elem, int i, int j);
24
25 int consultaElem(Matriz *mf, int i, int j);
26
27 void imprimeFaixaVetores(Matriz *mf);
28
29 void imprime(Matriz *mf);
30
31 #endif
```

roteiro_7/matriz_faixa.c

```
1  #include "./matriz_faixa.h"
2
3  void zeraMatriz(Matriz *mf) {
4    int i;
5    for (i = 0; i < mf->tam; i++) {
6      mf->diagonal[i] = 0;
7      if (i < mf->tam - 1) {
8        mf->superior[i] = 0;
9        mf->inferior[i] = 0;
10     }
11   }
12 }
13
14 Matriz *criaMatriz(int t) {
15   Matriz *mf;
16   mf = (Matriz *)malloc(sizeof(Matriz));
17   if (mf != NULL) {
18     if (t <= 1) {
19       printf("Dimensao deve ser > 1, matriz nao criada!");
20       return NULL;
21     }
22     mf->tam = t;
23     mf->diagonal = (int *)malloc(t * sizeof(int));
24     mf->superior = (int *)malloc((t - 1) * sizeof(int));
25     mf->inferior = (int *)malloc((t - 1) * sizeof(int));
26     if (mf->diagonal == NULL || mf->superior == NULL || mf->inferior == NULL)
27       return NULL;
28     zeraMatriz(mf);
29   }
30   return mf;
31 }
32
33 void destroiMatriz(Matriz **mf) {
34   if (*mf != NULL) {
35     free((*mf)->diagonal);
36     free((*mf)->superior);
37     free((*mf)->inferior);
38     free(*mf);
39     *mf = NULL;
40   }
41 }
42
43 int preencheAleatorio(Matriz *mf, int ini, int fim) {
44   if (mf == NULL)
45     return 0;
46   srand(time(NULL));
47   int i;
48   for (i = 0; i < mf->tam; i++) {
49     mf->diagonal[i] = ini + rand() % (fim - ini + 1);
50     if (i < mf->tam - 1) {
51       mf->superior[i] = ini + rand() % (fim - ini + 1);
52       mf->inferior[i] = ini + rand() % (fim - ini + 1);
53     }
54   }
55   return 1;
56 }
57
58 int insereElem(Matriz *mf, int elem, int i, int j) {
59   if (mf == NULL)
60     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)
66     mf->diagonal[i] = elem;
67   else if (i + 1 == j)
```

```c
    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");
  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");
  }
}
```

roteiro_7/2-1.c

```c
#include "./matriz_faixa.h"

#define MAX_OPTIONS 9

enum options {
  CRIAR = 0,
  PREENCHER,
  ZERAR,
  INSERIR,
  CONSULTAR,
  IMPRIMIR,
  IMPRIMIR_VETORES,
  DESTRUIR,
  SAIR,
};

int get_option() {
  int opt = -1;
  do {
    printf("\tOperacoes\n");
    printf("[%d] Criar matriz, ", CRIAR);
```

```c
    printf("[%d] Preencher aleatoriamente, ", PREENCHER);
    printf("[%d] Zerar matriz, ", ZERAR);
    printf("[%d] Inserir elemento, ", INSERIR);
    printf("[%d] Consultar elemento, ", CONSULTAR);
    printf("[%d] Imprimir matriz, ", IMPRIMIR);
    printf("[%d] Imprimir faixa de vetores matriz, ", IMPRIMIR_VETORES);
    printf("[%d] Destruir matriz, ", DESTRUIR);
    printf("[%d] Sair do programa \n", SAIR);

    printf("\nInsira a opção desejada: ");
    scanf("%d", &opt);
    printf("\n");

    if (opt < 0 || opt >= MAX_OPTIONS) {
      printf("Opção escolhida inválida!\n");
    }

  } while (opt < 0 || opt >= MAX_OPTIONS);

  return opt;
}

int get_valor(char *msg) {
  int value;

  printf("%s", msg);
  scanf("%d", &value);

  return value;
}

int main() {
  int opt, valor, linha, coluna;

  Matriz *matrix = NULL;

  do {
    opt = get_option();

    switch (opt) {
    case CRIAR:
      printf("Executando comando...\n");

      printf("Insira as dimensões da matriz faixa:\n");
      linha = get_valor("Diagonal: ");

      if (matrix == NULL) {
        matrix = criaMatriz(linha);
      } else {
        destroiMatriz(&matrix);
        matrix = NULL;
        matrix = criaMatriz(linha);
      }

      printf("Matriz criada com sucesso!\n");
      break;
    case PREENCHER:
      printf("Executando comando...\n");

      if (matrix == NULL) {
        printf("Matriz não incializada impossível realizar operação..\n");
        break;
      }

      printf("Insira os limites dos valores a serem preenchidos\n");

      int min, max;

      min = get_valor("Min: ");
      max = get_valor("Max: ");

      preencheAleatorio(matrix, min, max);

      printf("Matriz preenchida com sucesso!\n");

      break;
    case ZERAR:
      printf("Executando comando...\n");

      if (matrix == NULL) {
        printf("Matriz não incializada impossível realizar operação..\n");
```

```
        break;
      }

      zeraMatriz(matrix);

      printf("Matriz zerada com sucesso!\n");

      break;
    case INSERIR:
      printf("Executando comando...\n");

      if (matrix == NULL) {
        printf("Matriz não incializada impossível realizar operação..\n");
        break;
      }

      printf("Insira a posição para inserir o elemento na matriz:\n");

      linha = get_valor("Linha :");
      coluna = get_valor("Coluna :");
      valor = get_valor("Valor :");

      if (insereElem(matrix, valor, linha, coluna)) {
        printf("Valor inserido com sucesso!\n");
      }

      break;
    case CONSULTAR:
      printf("Executando comando...\n");

      if (matrix == NULL) {
        printf("Matriz não incializada impossível realizar operação..\n");
        break;
      }

      printf("Insira a posição do elemento a ser consultado:\n");

      linha = get_valor("Linha :");
      coluna = get_valor("Coluna :");
      valor = consultaElem(matrix, linha, coluna);

      if (valor) {
        printf("matriz[%d][%d] = %d\n", linha, coluna, valor);
        printf("Elemento consultado com sucesso!");
      }

      break;
    case IMPRIMIR:
      printf("Executando comando...\n");

      if (matrix == NULL) {
        printf("Matriz não incializada impossível realizar operação..\n");
        break;
      }

      imprime(matrix);

      printf("Matriz imprimida com sucesso!\n");

      break;
    case IMPRIMIR_VETORES:
      printf("Executando comando...\n");

      if (matrix == NULL) {
        printf("Matriz não incializada impossível realizar operação..\n");
        break;
      }

      imprimeFaixaVetores(matrix);

      printf("Faixa de vetores imprimida com sucesso!\n");

      break;
    case DESTRUIR:
      printf("Executando comando...\n");

      destroiMatriz(&matrix);
      matrix = NULL;

      printf("Matriz destruida com sucesso!\n");

```

```
184       break;
185     case SAIR:
186       if (matrix != NULL) {
187         destroiMatriz(&matrix);
188         matrix = NULL;
189       }
190
191       printf("Finalizando programa! Até mais!\n");
192       break;
193     }
194
195   } while (opt != SAIR);
196
197   return 0;
198 }
```

Saída do terminal:

```
└─:: ./2-1.out
       Operacoes
[0] Criar matriz, [1] Preencher aleatoriamente, [2] Zerar matriz, [3] Inserir elemento, [4] Consultar elemento, [5] Imprimir matriz, [6] Imprimir faixa de vetores matriz, [7] De

Insira a opção desejada: 0

Executando comando...
Insira as dimensões da matriz faixa:
Diagonal: 4
Matriz criada com sucesso!
       Operacoes
[0] Criar matriz, [1] Preencher aleatoriamente, [2] Zerar matriz, [3] Inserir elemento, [4] Consultar elemento, [5] Imprimir matriz, [6] Imprimir faixa de vetores matriz, [7] De

Insira a opção desejada: 5

Executando comando...
Matriz Faixa, Tam: 4 x 4:
Diagonal = [0 0 0 0 ]
Superior = [0 0 0 ]
Inferior = [0 0 0 ]

Matriz Original:
0       0       0       0
0       0       0       0
0       0       0       0
0       0       0       0
Matriz imprimida com sucesso!
       Operacoes
[0] Criar matriz, [1] Preencher aleatoriamente, [2] Zerar matriz, [3] Inserir elemento, [4] Consultar elemento, [5] Imprimir matriz, [6] Imprimir faixa de vetores matriz, [7] De

Insira a opção desejada: 1

Executando comando...
Insira os limites dos valores a serem preenchidos
Min: 0
Max: 5
Matriz preenchida com sucesso!
       Operacoes
[0] Criar matriz, [1] Preencher aleatoriamente, [2] Zerar matriz, [3] Inserir elemento, [4] Consultar elemento, [5] Imprimir matriz, [6] Imprimir faixa de vetores matriz, [7] De

Insira a opção desejada: 6

Executando comando...
Matriz Faixa, Tam: 4 x 4:
Diagonal = [3 2 5 3 ]
Superior = [3 4 4 ]
Inferior = [2 0 5 ]

Faixa de vetores imprimida com sucesso!
       Operacoes
[0] Criar matriz, [1] Preencher aleatoriamente, [2] Zerar matriz, [3] Inserir elemento, [4] Consultar elemento, [5] Imprimir matriz, [6] Imprimir faixa de vetores matriz, [7] De

Insira a opção desejada: 7

Executando comando...
Matriz destruida com sucesso!
       Operacoes
[0] Criar matriz, [1] Preencher aleatoriamente, [2] Zerar matriz, [3] Inserir elemento, [4] Consultar elemento, [5] Imprimir matriz, [6] Imprimir faixa de vetores matriz, [7] De
```

2.2) o TAD: Matriz Esparsa CSR

roteiro_7/matriz_esparsa_csr.h

```
1  #ifndef MATRIZ_ESPARSA_CSR
2  #define MATRIZ_ESPARSA_CSR
3
4  #include <stdio.h>
5  #include <stdlib.h>
6  #include <time.h>
7
8  typedef struct {
9    int *A; // Valores
10   int *IA;
11   int *JA;
12   int lin, col, QNN, QI;
13 } Matriz;
14 // QNN - Quantidade de Nao Nulos
15 // QI - Quantidade de Inseridos
16
17 Matriz *criaMatriz(int l, int c, int qnn);
18
19 int *meuRealloc(int *v, int tam);
20
21 void imprimeVetores(Matriz *ms);
22
```

```c
int insereElem(Matriz *ms, int elem, int i, int j);

int removeElem(Matriz *ms, int i, int j);

int consultaElem(Matriz *ms, int i, int j);

void imprime(Matriz *ms);

void destroiMatriz(Matriz **ms);

#endif
```

roteiro_7/matriz_esparsa_csr.c

```c
#include "matriz_esparsa_csr.h"

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;
}

void imprimeVetores(Matriz *ms) {
  if (ms == NULL)
    return;
  int i = 0;
  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;
}

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 -1;
  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 -1;
}

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;
    ms = NULL;
  }
}
```

roteiro_7/2-2.c

```
#include "./matriz_esparsa_csr.h"

#define MAX_OPTIONS 8

enum options {
  CRIAR = 0,
  INSERIR,
  REMOVER,
  CONSULTAR,
  IMPRIMIR,
  IMPRIMIR_VETORES,
  DESTRUIR,
  SAIR,
};

int get_option() {
  int opt = -1;
  do {
    printf("\tOperacoes\n");
    printf("[%d] Criar matriz, ", CRIAR);
    printf("[%d] Inserir elemento, ", INSERIR);
    printf("[%d] Remover elemento, ", REMOVER);
    printf("[%d] Consultar elemento, ", CONSULTAR);
    printf("[%d] Imprimir matriz, ", IMPRIMIR);
    printf("[%d] Imprimir vetores, ", IMPRIMIR_VETORES);
    printf("[%d] Destruir matriz, ", DESTRUIR);
    printf("[%d] Sair do programa \n", SAIR);

    printf("\nInsira a opção desejada: ");
    scanf("%d", &opt);
    printf("\n");

    if (opt < 0 || opt >= MAX_OPTIONS) {
      printf("Opção escolhida inválida!\n");
    }

  } while (opt < 0 || opt >= MAX_OPTIONS);

  return opt;
}

int get_valor(char *msg) {
  int value;

  printf("%s", msg);
  scanf("%d", &value);

```

```c
  return value;
}

int main() {
  int opt, valor, linha, coluna;

  Matriz *matrix = NULL;

  do {
    opt = get_option();

    switch (opt) {
    case CRIAR:
      printf("Executando comando...\n");

      printf("Insira as dimensões da matriz:\n");
      linha = get_valor("Linhas: ");
      coluna = get_valor("Colunas: ");

      if (matrix == NULL) {
        matrix = criaMatriz(linha, coluna, 0);
      } else {
        destroiMatriz(&matrix);
        matrix = NULL;
        matrix = criaMatriz(linha, coluna, 0);
      }

      printf("Matriz criada com sucesso!\n");
      break;
    case INSERIR:
      printf("Executando comando...\n");

      if (matrix == NULL) {
        printf("Matriz não incializada impossível realizar operação..\n");
        break;
      }

      printf("Insira a posição para inserir o elemento na matriz:\n");

      linha = get_valor("Linha :");
      coluna = get_valor("Coluna :");
      valor = get_valor("Valor :");

      if (insereElem(matrix, valor, linha, coluna)) {
        printf("Valor inserido com sucesso!\n");
      }

      break;
    case REMOVER:
      printf("Executando comando...\n");

      if (matrix == NULL) {
        printf("Matriz não incializada impossível realizar operação..\n");
        break;
      }

      linha = get_valor("Linha: ");
      coluna = get_valor("Coluna: ");

      if (removeElem(matrix, linha, coluna)) {
        printf("Elemento removido com sucesso!\n");
      }

      break;
    case CONSULTAR:
      printf("Executando comando...\n");

      if (matrix == NULL) {
        printf("Matriz não incializada impossível realizar operação..\n");
        break;
      }

      printf("Insira a posição do elemento a ser consultado:\n");

      linha = get_valor("Linha :");
      coluna = get_valor("Coluna :");
      valor = consultaElem(matrix, linha, coluna);

      if (valor != -1) {
        printf("matriz[%d][%d] = %d\n", linha, coluna, valor);
        printf("Elemento consultado com sucesso!");
```

```
129       } else {
130         printf("Elemento não encontrado!!!\n");
131       }
132 
133       break;
134     case IMPRIMIR:
135       printf("Executando comando...\n");
136 
137       if (matrix == NULL) {
138         printf("Matriz não incializada impossível realizar operação..\n");
139         break;
140       }
141 
142       imprime(matrix);
143 
144       printf("Matriz imprimida com sucesso!\n");
145 
146       break;
147     case IMPRIMIR_VETORES:
148       printf("Executando comando...\n");
149 
150       if (matrix == NULL) {
151         printf("Matriz não inicializada impossível realizar operação..\n");
152         break;
153       }
154 
155       imprimeVetores(matrix);
156 
157       printf("Vetores imprimido com sucesso!\n");
158 
159       break;
160     case DESTRUIR:
161       printf("Executando comando...\n");
162 
163       destroiMatriz(&matrix);
164       matrix = NULL;
165 
166       printf("Matriz destruida com sucesso!\n");
167 
168       break;
169     case SAIR:
170       if (matrix != NULL) {
171         destroiMatriz(&matrix);
172         matrix = NULL;
173       }
174 
175       printf("Finalizando programa! Até mais!\n");
176       break;
177     }
178 
179   } while (opt != SAIR);
180 
181   return 0;
182 }
```

Saída do terminal:

```
└─∴ ./2-2.out
        Operacoes
[0] Criar matriz, [1] Inserir elemento, [2] Remover elemento, [3] Consultar elemento, [4] Imprimir matriz, [5] Imprimir vetores, [6] Destruir matriz, [7] Sair do programa

Insira a opção desejada: 0

Executando comando...
Insira as dimensões da matriz:
Linhas: 3
Colunas: 3
Matriz criada com sucesso!
        Operacoes
[0] Criar matriz, [1] Inserir elemento, [2] Remover elemento, [3] Consultar elemento, [4] Imprimir matriz, [5] Imprimir vetores, [6] Destruir matriz, [7] Sair do programa

Insira a opção desejada: 1

Executando comando...
Insira a posição para inserir o elemento na matriz:
Linha :0
Coluna :1
Valor :2
Matriz Esparsa, Tam: 3 x 3:
1 elementos nao nulos.
A = [2 ]
IA = [0 1 1 1 ]
JA = [1 ]

Valor inserido com sucesso!
        Operacoes
[0] Criar matriz, [1] Inserir elemento, [2] Remover elemento, [3] Consultar elemento, [4] Imprimir matriz, [5] Imprimir vetores, [6] Destruir matriz, [7] Sair do programa

Insira a opção desejada: 4

Executando comando...
Matriz Esparsa, Tam: 3 x 3:
1 elementos nao nulos.
A = [2 ]
IA = [0 1 1 1 ]
JA = [1 ]

Matriz Original:
-1      2       -1
-1      -1      -1
-1      -1      -1
Matriz imprimida com sucesso!
        Operacoes
[0] Criar matriz, [1] Inserir elemento, [2] Remover elemento, [3] Consultar elemento, [4] Imprimir matriz, [5] Imprimir vetores, [6] Destruir matriz, [7] Sair do programa

Insira a opção desejada: 6

Executando comando...
Matriz destruida com sucesso!
        Operacoes
[0] Criar matriz, [1] Inserir elemento, [2] Remover elemento, [3] Consultar elemento, [4] Imprimir matriz, [5] Imprimir vetores, [6] Destruir matriz, [7] Sair do programa

Insira a opção desejada: 7

Finalizando programa! Até mais!
```