

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

```
1 #ifndef MATRIZ_ESTATICA_H
2 #define MATRIZ_ESTATICA_H
3
4 #include <stdio.h>
5 #include <stdlib.h>
6 #include <time.h>
7
8 #define MAX 100
9
10 typedef struct {
11     int dados[MAX][MAX];
12     int lin, col;
13 } Matriz;
14
15 void zeraMatriz(Matriz *mat);
16
17 Matriz *criaMatriz(int l, int c);
18
19 void destroiMatriz(Matriz *mat);
20
21 int preencheAleatorio(Matriz *mat, int ini, int fim);
22
23 int insereElem(Matriz *mat, int elem, int l, int c);
24
25 int consultaElem(Matriz *mat, int *p, int l, int c);
26
27 void imprime(Matriz *mat);
28
29 #endif
```

roteiro_7/matriz_estatica.c

```
1 #include "../matriz_estatica.h"
2
3 void zeraMatriz(Matriz *mat) {
4     int i, j;
5     for (i = 0; i < mat->lin; i++)
6         for (j = 0; j < mat->col; j++)
7             mat->dados[i][j] = 0;
8 }
9
10 Matriz *criaMatriz(int l, int c) {
11     Matriz *mat;
12     mat = (Matriz *)malloc(sizeof(Matriz));
13     if (mat != NULL) {
14         if (l <= 0 || c <= 0 || l > MAX || c > MAX) {
15             printf("Valores invalidos, matriz nao criada!\n");
16             return NULL;
17         }
18         mat->lin = l;
19         mat->col = c;
20         zeraMatriz(mat);
21     }
22     return mat;
23 }
24
25 void destroiMatriz(Matriz *mat) {
26     if (mat != NULL)
27         free(mat);
28     mat = NULL;
29 }
30
31 int preencheAleatorio(Matriz *mat, int ini, int fim) {
32     if (mat == NULL)
33         return 0;
34     srand(time(NULL));
```

```

35     int i, j;
36     for (i = 0; i < mat->lin; i++)
37         for (j = 0; j < mat->col; j++)
38             mat->dados[i][j] = ini + rand() % (fim - ini + 1);
39     return 1;
40 }
41
42 int insereElem(Matriz *mat, int elem, int l, int c) {
43     if (mat == NULL)
44         return 0;
45     if (l < 0 || c < 0 || l >= mat->lin || c >= mat->col) {
46         printf("Valores invalidos, elem nao inserido!\n");
47         return 0;
48     }
49     mat->dados[l][c] = elem;
50     return 1;
51 }
52
53 int consultaElem(Matriz *mat, int *p, int l, int c) {
54     if (mat == NULL)
55         return 0;
56     if (l < 0 || c < 0 || l >= mat->lin || c >= mat->col) {
57         printf("Valores invalidos, elem nao existe!\n");
58         return 0;
59     }
60     *p = mat->dados[l][c];
61     return 1;
62 }
63
64 void imprime(Matriz *mat) {
65     if (mat == NULL)
66         return;
67     int i, j;
68     printf("Matriz %d x %d:\n", mat->lin, mat->col);
69     for (i = 0; i < mat->lin; i++) {
70         for (j = 0; j < mat->col; j++)
71             printf("\t%d", mat->dados[i][j]);
72         printf("\n");
73     }
74     printf("\n");
75 }

```

roteiro_7/1-1.c

```

1  #include "./matriz_estatica.h"
2
3  #define MAX_OPTIONS 8
4
5  enum options {
6      CRIAR = 0,
7      PREENCHER,
8      ZERAR,
9      INSERIR,
10     CONSULTAR,
11     IMPRIMIR,
12     DESTRUIR,
13     SAIR,
14 };
15
16 int get_option() {
17     int opt = -1;
18     do {
19         printf("\tOperacoes\n");
20         printf("[%d] Criar matriz, ", CRIAR);
21         printf("[%d] Preencher aleatoriamente, ", PREENCHER);
22         printf("[%d] Zerar matriz, ", ZERAR);
23         printf("[%d] Inserir elemento, ", INSERIR);
24         printf("[%d] Consultar elemento, ", CONSULTAR);
25         printf("[%d] Imprimir matriz, ", IMPRIMIR);
26         printf("[%d] Destruir matriz, ", DESTRUIR);
27         printf("[%d] Sair do programa \n", SAIR);
28
29         printf("\nInsira a opção desejada: ");
30         scanf("%d", &opt);
31         printf("\n");
32
33         if (opt < 0 || opt >= MAX_OPTIONS) {
34             printf("Opção escolhida inválida!\n");
35         }
36
37     } while (opt < 0 || opt >= MAX_OPTIONS);
38 }

```

```
39     return opt;
40 }
41
42 int get_valor(char *msg) {
43     int value;
44
45     printf("%s", msg);
46     scanf("%d", &value);
47
48     return value;
49 }
50
51 int main() {
52     int opt, valor, linha, coluna;
53
54     Matriz *matrix = NULL;
55
56     do {
57         opt = get_option();
58
59         switch (opt) {
60             case CRIAR:
61                 printf("Executando comando...\n");
62
63                 printf("Insira as dimensões da matriz:\n");
64                 linha = get_valor("Linhas: ");
65                 coluna = get_valor("Colunas: ");
66
67                 if (matrix == NULL) {
68                     matrix = criaMatriz(linha, coluna);
69                 } else {
70                     destroiMatriz(matrix);
71                     matrix = NULL;
72                     matrix = criaMatriz(linha, coluna);
73                 }
74
75                 printf("Matriz criada com sucesso!\n");
76                 break;
77             case PREENCHER:
78                 printf("Executando comando...\n");
79
80                 if (matrix == NULL) {
81                     printf("Matriz não inicializada impossível realizar operação..\n");
82                     break;
83                 }
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 inicializada 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 inicializada 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 inicializada 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 inicializada 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@arthurdetomi-System-Product-Name ~/Documents/UFSJ-Graduacao/UFSJ-2025_1/Lab_Prog_2/roteiro_7] on mainxxx 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

```

1 | #ifndef MATRIZ_DINAMICA_H
2 | #define MATRIZ_DINAMICA_H
3 |
4 | #include <stdio.h>
5 | #include <stdlib.h>
6 | #include <time.h>
7 |
8 | typedef struct {
9 |     int **dados;
10 |    int lin, col;
11 | } Matriz;
12 |
13 | void zeraMatriz(Matriz *mat);
14 |
15 | Matriz *criaMatriz(int l, int c);
16 |
17 | void destroiMatriz(Matriz *mat);
18 |
19 | int preencheAleatorio(Matriz *mat, int ini, int fim);
20 |
21 | int insereElem(Matriz *mat, int elem, int l, int c);
22 |
23 | int consultaElem(Matriz *mat, int *p, int l, int c);
24 |
25 | void imprime(Matriz *mat);
26 |
27 | #endif

```

roteiro_7/matriz_dinamica.c

```

1 | #include "matriz_dinamica.h"
2 |
3 | void zeraMatriz(Matriz *mat) {
4 |     int i, j;
5 |     for (i = 0; i < mat->lin; i++)
6 |         for (j = 0; j < mat->col; j++)
7 |             mat->dados[i][j] = 0;
8 | }
9 |
10 | Matriz *criaMatriz(int l, int c) {
11 |     Matriz *mat;

```

```
12 mat = (Matriz *)malloc(sizeof(Matriz));
13 if (mat != NULL) {
14     if (l <= 0 || c <= 0) {
15         printf("Valores invalidos, matriz nao criada!\n");
16         return NULL;
17     }
18     int i;
19     mat->lin = l;
20     mat->col = c;
21     mat->dados = (int **)malloc(l * sizeof(int *));
22     for (i = 0; i < l; i++)
23         mat->dados[i] = (int *)malloc(c * sizeof(int));
24     zeraMatriz(mat);
25 }
26 return mat;
27 }
28
29 void destroiMatriz(Matriz *mat) {
30     if (mat != NULL) {
31         int i;
32         for (i = 0; i < mat->lin; i++) {
33             free(mat->dados[i]);
34             mat->dados[i] = NULL;
35         }
36
37         free(mat->dados);
38         mat->dados = NULL;
39         free(mat);
40         mat = NULL;
41     }
42 }
43
44 int preencheAleatorio(Matriz *mat, int ini, int fim) {
45     if (mat == NULL)
46         return 0;
47     srand(time(NULL));
48     int i, j;
49     for (i = 0; i < mat->lin; i++)
50         for (j = 0; j < mat->col; j++)
51             mat->dados[i][j] = ini + rand() % (fim - ini + 1);
52     return 1;
53 }
54
55 int insereElem(Matriz *mat, int elem, int l, int c) {
56     if (mat == NULL)
57         return 0;
58     if (l < 0 || c < 0 || l > mat->lin || c > mat->col) {
59         printf("Valores invalidos, elem nao inserido!\n");
60         return 0;
61     }
62     mat->dados[l][c] = elem;
63     return 1;
64 }
65
66 int consultaElem(Matriz *mat, int *p, int l, int c) {
67     if (mat == NULL)
68         return 0;
69     if (l < 0 || c < 0 || l > mat->lin || c > mat->col) {
70         printf("Valores invalidos, elem nao existe!\n");
71         return 0;
72     }
73     *p = mat->dados[l][c];
74     return 1;
75 }
76
77 void imprime(Matriz *mat) {
78     if (mat == NULL)
79         return;
80     int i, j;
81     printf("Matriz %d x %d:\n", mat->lin, mat->col);
82     for (i = 0; i < mat->lin; i++) {
83         for (j = 0; j < mat->col; j++)
84             printf("%d ", mat->dados[i][j]);
85         printf("\n");
86     }
87     printf("\n");
88 }
```

roteiro_7/1-2.c

```
1 | #include "../matriz_dinamica.h"
2 |
```

```
3 | #define MAX_OPTIONS 8
4 |
5 | enum options {
6 |     CRIAR = 0,
7 |     PREENCHER,
8 |     ZERAR,
9 |     INSERIR,
10 |    CONSULTAR,
11 |    IMPRIMIR,
12 |    DESTRUIR,
13 |    SAIR,
14 | };
15 |
16 | int get_option() {
17 |     int opt = -1;
18 |     do {
19 |         printf("\tOperacoes\n");
20 |         printf("[%d] Criar matriz, ", CRIAR);
21 |         printf("[%d] Preencher aleatoriamente, ", PREENCHER);
22 |         printf("[%d] Zerar matriz, ", ZERAR);
23 |         printf("[%d] Inserir elemento, ", INSERIR);
24 |         printf("[%d] Consultar elemento, ", CONSULTAR);
25 |         printf("[%d] Imprimir matriz, ", IMPRIMIR);
26 |         printf("[%d] Destruir matriz, ", DESTRUIR);
27 |         printf("[%d] Sair do programa \n", SAIR);
28 |
29 |         printf("\nInsira a opção desejada: ");
30 |         scanf("%d", &opt);
31 |         printf("\n");
32 |
33 |         if (opt < 0 || opt >= MAX_OPTIONS) {
34 |             printf("Opção escolhida inválida!\n");
35 |         }
36 |
37 |     } while (opt < 0 || opt >= MAX_OPTIONS);
38 |
39 |     return opt;
40 | }
41 |
42 | int get_valor(char *msg) {
43 |     int value;
44 |
45 |     printf("%s", msg);
46 |     scanf("%d", &value);
47 |
48 |     return value;
49 | }
50 |
51 | int main() {
52 |     int opt, valor, linha, coluna;
53 |
54 |     Matriz *matrix = NULL;
55 |
56 |     do {
57 |         opt = get_option();
58 |
59 |         switch (opt) {
60 |             case CRIAR:
61 |                 printf("Executando comando...\n");
62 |
63 |                 printf("Insira as dimensões da matriz:\n");
64 |                 linha = get_valor("Linhas: ");
65 |                 coluna = get_valor("Colunas: ");
66 |
67 |                 if (matrix == NULL) {
68 |                     matrix = criaMatriz(linha, coluna);
69 |                 } else {
70 |                     destroiMatriz(matrix);
71 |                     matrix = NULL;
72 |                     matrix = criaMatriz(linha, coluna);
73 |                 }
74 |
75 |                 printf("Matriz criada com sucesso!\n");
76 |                 break;
77 |             case PREENCHER:
78 |                 printf("Executando comando...\n");
79 |
80 |                 if (matrix == NULL) {
81 |                     printf("Matriz não inicializada impossível realizar operação...\n");
82 |                     break;
83 |                 }
```

```
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 inicializada 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 inicializada 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 inicializada 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 inicializada 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)

```

```

68     mf->superior[i] = elem;
69     else if (i == j + 1)
70     mf->inferior[j] = elem;
71     else {
72         printf("Indices fora da faixa, elem nao inserido!\n");
73         return 0;
74     }
75     return 1;
76 }
77
78 int consultaElem(Matriz *mf, int i, int j) {
79     if (mf == NULL)
80         return 0;
81     if (i < 0 || j < 0 || i >= mf->tam || j >= mf->tam) {
82         printf("Valores invalidos, elem inexistente!\n");
83         return 0;
84     }
85     if (i == j)
86         return mf->diagonal[i];
87     else if (i + 1 == j)
88         return mf->superior[i];
89     else if (i == j + 1)
90         return mf->inferior[j];
91     else
92         return 0;
93 }
94
95 void imprimeFaixaVetores(Matriz *mf) {
96     if (mf == NULL)
97         return;
98     int i;
99     printf("Matriz Faixa, Tam: %d x %d:\n", mf->tam, mf->tam);
100    printf("Diagonal = [");
101    for (i = 0; i < mf->tam; i++)
102        printf("%d ", mf->diagonal[i]);
103    printf("]\n");
104    printf("Superior = [");
105    for (i = 0; i < mf->tam - 1; i++)
106        printf("%d ", mf->superior[i]);
107    printf("]\n");
108    printf("Inferior = [");
109    for (i = 0; i < mf->tam - 1; i++)
110        printf("%d ", mf->inferior[i]);
111    printf("]\n\n");
112 }
113
114 void imprime(Matriz *mf) {
115     if (mf == NULL)
116         return;
117     int i, j;
118     imprimeFaixaVetores(mf);
119     printf("Matriz Original:\n");
120     for (i = 0; i < mf->tam; i++) {
121         for (j = 0; j < mf->tam; j++)
122             printf("%d\t", consultaElem(mf, i, j));
123         printf("\n");
124     }
125 }

```

roteiro_7/2-1.c

```

1  #include "./matriz_faixa.h"
2
3  #define MAX_OPTIONS 9
4
5  enum options {
6      CRIAR = 0,
7      PREENCHER,
8      ZERAR,
9      INSERIR,
10     CONSULTAR,
11     IMPRIMIR,
12     IMPRIMIR_VETORES,
13     DESTRUIR,
14     SAIR,
15 };
16
17 int get_option() {
18     int opt = -1;
19     do {
20         printf("\t0peracoes\n");
21         printf("[%d] Criar matriz, ", CRIAR);

```

```
22     printf("[%d] Preencher aleatoriamente, ", PREENCHER);
23     printf("[%d] Zerar matriz, ", ZERAR);
24     printf("[%d] Inserir elemento, ", INSERIR);
25     printf("[%d] Consultar elemento, ", CONSULTAR);
26     printf("[%d] Imprimir matriz, ", IMPRIMIR);
27     printf("[%d] Imprimir faixa de vetores matriz, ", IMPRIMIR_VETORES);
28     printf("[%d] Destruir matriz, ", DESTRUIR);
29     printf("[%d] Sair do programa \n", SAIR);
30
31     printf("\nInsira a opção desejada: ");
32     scanf("%d", &opt);
33     printf("\n");
34
35     if (opt < 0 || opt >= MAX_OPTIONS) {
36         printf("Opção escolhida inválida!\n");
37     }
38
39 } while (opt < 0 || opt >= MAX_OPTIONS);
40
41 return opt;
42 }
43
44 int get_valor(char *msg) {
45     int value;
46
47     printf("%s", msg);
48     scanf("%d", &value);
49
50     return value;
51 }
52
53 int main() {
54     int opt, valor, linha, coluna;
55
56     Matriz *matrix = NULL;
57
58     do {
59         opt = get_option();
60
61         switch (opt) {
62             case CRIAR:
63                 printf("Executando comando...\n");
64
65                 printf("Insira as dimensões da matriz faixa:\n");
66                 linha = get_valor("Diagonal: ");
67
68                 if (matrix == NULL) {
69                     matrix = criaMatriz(linha);
70                 } else {
71                     destroiMatriz(&matrix);
72                     matrix = NULL;
73                     matrix = criaMatriz(linha);
74                 }
75
76                 printf("Matriz criada com sucesso!\n");
77                 break;
78             case PREENCHER:
79                 printf("Executando comando...\n");
80
81                 if (matrix == NULL) {
82                     printf("Matriz não inicializada impossível realizar operação...\n");
83                     break;
84                 }
85
86                 printf("Insira os limites dos valores a serem preenchidos\n");
87
88                 int min, max;
89
90                 min = get_valor("Min: ");
91                 max = get_valor("Max: ");
92
93                 preencheAleatorio(matrix, min, max);
94
95                 printf("Matriz preenchida com sucesso!\n");
96
97                 break;
98             case ZERAR:
99                 printf("Executando comando...\n");
100
101                 if (matrix == NULL) {
102                     printf("Matriz não inicializada impossível realizar operação...\n");
```

```
103     break;
104 }
105
106 zeraMatriz(matrix);
107
108 printf("Matriz zerada com sucesso!\n");
109
110 break;
111 case INSERIR:
112     printf("Executando comando...\n");
113
114     if (matrix == NULL) {
115         printf("Matriz não inicializada impossível realizar operação..\n");
116         break;
117     }
118
119     printf("Insira a posição para inserir o elemento na matriz:\n");
120
121     linha = get_valor("Linha :");
122     coluna = get_valor("Coluna :");
123     valor = get_valor("Valor :");
124
125     if (insereElem(matrix, valor, linha, coluna)) {
126         printf("Valor inserido com sucesso!\n");
127     }
128
129     break;
130 case CONSULTAR:
131     printf("Executando comando...\n");
132
133     if (matrix == NULL) {
134         printf("Matriz não inicializada impossível realizar operação..\n");
135         break;
136     }
137
138     printf("Insira a posição do elemento a ser consultado:\n");
139
140     linha = get_valor("Linha :");
141     coluna = get_valor("Coluna :");
142     valor = consultaElem(matrix, linha, coluna);
143
144     if (valor) {
145         printf("matriz[%d][%d] = %d\n", linha, coluna, valor);
146         printf("Elemento consultado com sucesso!");
147     }
148
149     break;
150 case IMPRIMIR:
151     printf("Executando comando...\n");
152
153     if (matrix == NULL) {
154         printf("Matriz não inicializada impossível realizar operação..\n");
155         break;
156     }
157
158     imprime(matrix);
159
160     printf("Matriz imprimida com sucesso!\n");
161
162     break;
163 case IMPRIMIR_VETORES:
164     printf("Executando comando...\n");
165
166     if (matrix == NULL) {
167         printf("Matriz não inicializada impossível realizar operação..\n");
168         break;
169     }
170
171     imprimeFaixaVetores(matrix);
172
173     printf("Faixa de vetores imprimida com sucesso!\n");
174
175     break;
176 case DESTRUIR:
177     printf("Executando comando...\n");
178
179     destroiMatriz(&matrix);
180     matrix = NULL;
181
182     printf("Matriz destruida com sucesso!\n");
183
```

```
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:

```

C:\3 .\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

```

```

23 int insereElem(Matriz *ms, int elem, int i, int j);
24
25 int removeElem(Matriz *ms, int i, int j);
26
27 int consultaElem(Matriz *ms, int i, int j);
28
29 void imprime(Matriz *ms);
30
31 void destroiMatriz(Matriz **ms);
32
33 #endif

```

roteiro_7/matriz_esparsa_csr.c

```

1  #include "matriz_esparsa_csr.h"
2
3  Matriz *criaMatriz(int l, int c, int qnn) {
4      Matriz *ms;
5      ms = (Matriz *)malloc(sizeof(Matriz));
6      if (ms != NULL) {
7          if (l <= 0 || c <= 0 || qnn < 0) {
8              printf("Valores invalidos, matriz nao criada!\n");
9              return NULL;
10         }
11         ms->lin = l;
12         ms->col = c;
13         ms->QI = 0;
14         ms->QNN = qnn;
15         ms->A = ms->IA = ms->JA = NULL;
16         if (qnn != 0) {
17             ms->A = (int *)malloc(qnn * sizeof(int));
18             ms->JA = (int *)malloc(qnn * sizeof(int));
19             if (ms->A == NULL || ms->JA == NULL)
20                 return NULL;
21         }
22         ms->IA = (int *)malloc((ms->lin + 1) * sizeof(int));
23         if (ms->IA == NULL)
24             return NULL;
25         int i;
26         for (i = 0; i < l + 1; i++)
27             ms->IA[i] = 0;
28     }
29     return ms;
30 }
31
32 int *meuRealloc(int *v, int tam) {
33     int *aux = (int *)malloc((tam + 1) * sizeof(int));
34     if (aux != NULL) {
35         if (v != NULL) {
36             int i;
37             for (i = 0; i < tam; i++)
38                 aux[i] = v[i];
39             free(v);
40         }
41     }
42     return aux;
43 }
44
45 void imprimeVetores(Matriz *ms) {
46     if (ms == NULL)
47         return;
48     int i = 0;
49     printf("Matriz Esparsa, Tam: %d x %d:\n", ms->lin, ms->col);
50     printf("%d elementos nao nulos.\n", ms->QNN);
51     printf("A = [");
52     for (i = 0; i < ms->QNN; i++)
53         printf("%d ", ms->A[i]);
54     printf("]\n");
55     printf("IA = [");
56     for (i = 0; i < ms->lin + 1; i++)
57         printf("%d ", ms->IA[i]);
58     printf("]\n");
59     printf("JA = [");
60     for (i = 0; i < ms->QNN; i++)
61         printf("%d ", ms->JA[i]);
62     printf("]\n\n");
63 }
64
65 int insereElem(Matriz *ms, int elem, int i, int j) {
66     if (ms == NULL)
67         return 0;
68     if (i < 0 || j < 0 || i >= ms->lin || j >= ms->col) {

```

```
69     printf("Valores invalidos, elem nao inserido!\n");
70     return 0;
71 }
72 int k;
73 int index = -1;
74 int ini = ms->IA[i];
75 int fim = ms->IA[i + 1];
76 // Encontre a posição correta para inserir o valor
77 for (k = ini; k < fim; k++)
78     if (ms->JA[k] >= j) {
79         index = k;
80         break;
81     }
82
83 if (index == -1) {          // NOVA INSERCAO
84     if (ms->QI == ms->QNN) { // Necessita REALLOC
85         ms->A = meuRealloc(ms->A, ms->QNN);
86         ms->JA = meuRealloc(ms->JA, ms->QNN);
87         ms->QNN++;
88     }
89     // Move elementos para a nova insercao
90     for (k = ms->QNN - 1; k >= fim; k--) {
91         ms->A[k] = ms->A[k - 1];
92         ms->JA[k] = ms->JA[k - 1];
93     }
94     ms->A[fim] = elem;
95     ms->JA[fim] = j;
96     ms->QI++;
97     // Atualiza QNN acumulado
98     for (int k = i + 1; k <= ms->lin; k++)
99         ms->IA[k]++;
100 } else { // Atualiza um valor existente
101     ms->A[index] = elem;
102 }
103 imprimeVetores(ms);
104 return 1;
105 }
106
107 int removeElem(Matriz *ms, int i, int j) {
108     if (ms == NULL)
109         return 0;
110     if (i < 0 || j < 0 || i >= ms->lin || j >= ms->col) {
111         printf("Valores invalidos, elem nao removido!\n");
112         return 0;
113     }
114
115     int k;
116     int index = -1;
117     int ini = ms->IA[i];
118     int fim = ms->IA[i + 1];
119     // Encontre a posição do valor a ser removido
120     for (k = ini; k < fim; k++)
121         if (ms->JA[k] == j) {
122             index = k;
123             break;
124         }
125
126     if (index != -1) {
127         // Move todos elementos uma posição para tras
128         for (k = index; k < ms->QNN - 1; k++) {
129             ms->A[k] = ms->A[k + 1];
130             ms->JA[k] = ms->JA[k + 1];
131         }
132         ms->QNN--;
133         ms->QI--;
134         // Atualiza QNN acumulado
135         for (int k = i + 1; k <= ms->lin; k++)
136             ms->IA[k]--;
137     } else {
138         printf("Elemento nao existente\n");
139         return 0;
140     }
141     imprimeVetores(ms);
142     return 1;
143 }
144
145 int consultaElem(Matriz *ms, int i, int j) {
146     if (ms == NULL)
147         return -1;
148     if (i < 0 || j < 0 || i >= ms->lin || j >= ms->col) {
149         printf("Valores invalidos, elem inexistente!\n");
```



```

150     return 0;
151 }
152 int k;
153 for (k = ms->IA[i]; k < ms->IA[i + 1]; k++)
154     if (ms->JA[k] == j)
155         return ms->A[k];
156 return -1;
157 }
158
159 void imprime(Matriz *ms) {
160     if (ms == NULL)
161         return;
162     int i, j;
163     imprimeVetores(ms);
164     printf("Matriz Original:\n");
165     for (i = 0; i < ms->lin; i++) {
166         for (j = 0; j < ms->col; j++)
167             printf("%d\t", consultaElem(ms, i, j));
168         printf("\n");
169     }
170 }
171
172 void destroiMatriz(Matriz **ms) {
173     if (*ms != NULL) {
174         free((*ms)->A);
175         free((*ms)->IA);
176         free((*ms)->JA);
177         free(*ms);
178         *ms = NULL;
179         ms = NULL;
180     }
181 }

```

roteiro_7/2-2.c

```

1  #include "../matriz_esparsa_csr.h"
2
3  #define MAX_OPTIONS 8
4
5  enum options {
6      CRIAR = 0,
7      INSERIR,
8      REMOVER,
9      CONSULTAR,
10     IMPRIMIR,
11     IMPRIMIR_VETORES,
12     DESTRUIR,
13     SAIR,
14 };
15
16 int get_option() {
17     int opt = -1;
18     do {
19         printf("\tOperacoes\n");
20         printf("[%d] Criar matriz, ", CRIAR);
21         printf("[%d] Inserir elemento, ", INSERIR);
22         printf("[%d] Remover elemento, ", REMOVER);
23         printf("[%d] Consultar elemento, ", CONSULTAR);
24         printf("[%d] Imprimir matriz, ", IMPRIMIR);
25         printf("[%d] Imprimir vetores, ", IMPRIMIR_VETORES);
26         printf("[%d] Destruir matriz, ", DESTRUIR);
27         printf("[%d] Sair do programa \n", SAIR);
28
29         printf("\nInsira a opção desejada: ");
30         scanf("%d", &opt);
31         printf("\n");
32
33         if (opt < 0 || opt >= MAX_OPTIONS) {
34             printf("Opção escolhida inválida!\n");
35         }
36
37     } while (opt < 0 || opt >= MAX_OPTIONS);
38
39     return opt;
40 }
41
42 int get_valor(char *msg) {
43     int value;
44
45     printf("%s", msg);
46     scanf("%d", &value);
47

```

```
48     return value;
49 }
50
51 int main() {
52     int opt, valor, linha, coluna;
53
54     Matriz *matrix = NULL;
55
56     do {
57         opt = get_option();
58
59         switch (opt) {
60             case CRIAR:
61                 printf("Executando comando...\n");
62
63                 printf("Insira as dimensões da matriz:\n");
64                 linha = get_valor("Linhas: ");
65                 coluna = get_valor("Colunas: ");
66
67                 if (matrix == NULL) {
68                     matrix = criaMatriz(linha, coluna, 0);
69                 } else {
70                     destroiMatriz(&matrix);
71                     matrix = NULL;
72                     matrix = criaMatriz(linha, coluna, 0);
73                 }
74
75                 printf("Matriz criada com sucesso!\n");
76                 break;
77             case INSERIR:
78                 printf("Executando comando...\n");
79
80                 if (matrix == NULL) {
81                     printf("Matriz não inicializada impossível realizar operação..\n");
82                     break;
83                 }
84
85                 printf("Insira a posição para inserir o elemento na matriz:\n");
86
87                 linha = get_valor("Linha:");
88                 coluna = get_valor("Coluna:");
89                 valor = get_valor("Valor:");
90
91                 if (insereElem(matrix, valor, linha, coluna)) {
92                     printf("Valor inserido com sucesso!\n");
93                 }
94
95                 break;
96             case REMOVER:
97                 printf("Executando comando...\n");
98
99                 if (matrix == NULL) {
100                     printf("Matriz não inicializada impossível realizar operação..\n");
101                     break;
102                 }
103
104                 linha = get_valor("Linha: ");
105                 coluna = get_valor("Coluna: ");
106
107                 if (removeElem(matrix, linha, coluna)) {
108                     printf("Elemento removido com sucesso!\n");
109                 }
110
111                 break;
112             case CONSULTAR:
113                 printf("Executando comando...\n");
114
115                 if (matrix == NULL) {
116                     printf("Matriz não inicializada impossível realizar operação..\n");
117                     break;
118                 }
119
120                 printf("Insira a posição do elemento a ser consultado:\n");
121
122                 linha = get_valor("Linha:");
123                 coluna = get_valor("Coluna:");
124                 valor = consultaElem(matrix, linha, coluna);
125
126                 if (valor != -1) {
127                     printf("matriz[%d][%d] = %d\n", linha, coluna, valor);
128                     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 inicializada 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!
```