

roteiro-07/ex01-01.h

```
1  #ifndef MATRIZ_H
2  #define MATRIZ_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     int i, j;
17     if (mat != NULL) {
18         for (i = 0; i < mat->lin; i++)
19             for (j = 0; j < mat->col; j++)
20                 mat->dados[i][j] = 0;
21     }
22 }
23
24 Matriz* criaMatriz(int l, int c) {
25     Matriz* mat;
26     mat = (Matriz*)malloc(sizeof(Matriz));
27     if (mat != NULL) {
28         if (l <= 0 || c <= 0 || l > MAX || c > MAX) {
29             printf("Valores invalidos, matriz nao criada!\n");
30             return NULL;
31         }
32         mat->lin = l;
33         mat->col = c;
34         zeraMatriz(mat);
35     }
36     return mat;
37 }
38
39 void destroiMatriz(Matriz** mat) {
40     if (*mat != NULL) {
41         free(*mat);
42         *mat = NULL;
43     }
44 }
45
46 int preencheAleatorio(Matriz* mat, int ini, int fim) {
47     if (mat == NULL) return 0;
48     srand(time(NULL));
49     int i, j;
50     for (i = 0; i < mat->lin; i++)
51         for (j = 0; j < mat->col; j++)
52             mat->dados[i][j] = ini + rand() % (fim - ini + 1);
53     return 1;
54 }
55
```

```
56 int insereElem(Matriz* mat, int elem, int l, int c) {
57     if (mat == NULL) 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) return 0;
68     if (l < 0 || c < 0 || l >= mat->lin || c >= mat->col) {
69         printf("Valores invalidos, elem nao existe!\n");
70         return 0;
71     }
72     *p = mat->dados[l][c];
73     return 1;
74 }
75
76 void imprime(Matriz* mat) {
77     if (mat == NULL) return;
78     int i, j;
79     printf("Matriz %d x %d:\n", mat->lin, mat->col);
80     for (i = 0; i < mat->lin; i++) {
81         for (j = 0; j < mat->col; j++)
82             printf("\t%d", mat->dados[i][j]);
83         printf("\n");
84     }
85     printf("\n");
86 }
87
88 #endif
```

roteiro-07/ex01-02.h

```
1  #ifndef MATRIZDIN_H
2  #define MATRIZDIN_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     if (mat == NULL) return;
15
16     int i, j;
17     for (i = 0; i < mat->lin; i++)
18         for (j = 0; j < mat->col; j++)
19             mat->dados[i][j] = 0;
20 }
21
22 Matriz* criaMatriz(int l, int c) {
23     Matriz* mat;
24     mat = (Matriz*)malloc(sizeof(Matriz));
25     if (mat != NULL) {
26         if (l <= 0 || c <= 0) {
27             printf("Valores invalidos, matriz nao criada!\n");
28             return NULL;
29         }
30         int i;
31         mat->lin = l;
32         mat->col = c;
33         mat->dados = (int**)malloc(l * sizeof(int*));
34         for (i = 0; i < l; i++)
35             mat->dados[i] = (int*)malloc(c * sizeof(int));
36         zeraMatriz(mat);
37     }
38     return mat;
39 }
40
41 void destroiMatriz(Matriz** mat) {
42     if (*mat != NULL) {
43         int i;
44         for (i = 0; i < (*mat)->lin; i++)
45             free((*mat)->dados[i]);
46         free((*mat)->dados);
47         free(*mat);
48         *mat = NULL;
49     }
50 }
51
52 int preencheAleatorio(Matriz* mat, int ini, int fim) {
53     if (mat == NULL) return 0;
54     srand(time(NULL));
55     int i, j;
```

```

56     for (i = 0; i < mat->lin; i++)
57         for (j = 0; j < mat->col; j++)
58             mat->dados[i][j] = ini + rand() % (fim - ini + 1);
59     return 1;
60 }
61
62 int insereElem(Matriz* mat, int elem, 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 inserido!\n");
66         return 0;
67     }
68     mat->dados[l][c] = elem;
69     return 1;
70 }
71
72 int consultaElem(Matriz* mat, int* p, int l, int c) {
73     if (mat == NULL) return 0;
74     if (l < 0 || c < 0 || l > mat->lin || c > mat->col) {
75         printf("Valores invalidos, elem nao existe!\n");
76         return 0;
77     }
78     *p = mat->dados[l][c];
79     return 1;
80 }
81
82 void imprime(Matriz* mat) {
83     if (mat == NULL) return;
84     int i, j;
85     printf("Matriz %d x %d:\n", mat->lin, mat->col);
86     for (i = 0; i < mat->lin; i++) {
87         for (j = 0; j < mat->col; j++)
88             printf("%d ", mat->dados[i][j]);
89         printf("\n");
90     }
91     printf("\n");
92 }
93
94 #endif

```

roteiro-07/ex01.c

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  /*
5   *
6   * 0 ARQUIVO FUNCIONA PARA AS DUAS IMPLEMENTACOES DE MATRIZES
7   * BASTA IMPORTAR APENAS A BIBLIOTECA DESEJADA
8   *
9   * EX01-01 = MATRIZ SEQUENCIAL ESTATICA
10  * EX01-02 = MATRIZ SEQUENCIAL DINAMICA
11  *
12  */
13
14 // #include "ex01-01.h"
15 #include "ex01-02.h"
16
17 enum {
18     EXIT = 0,
19     CREATE,
20     FILL,
21     INSERT,
22     QUERY,
23     PRINT,
24     RESET,
25     DESTROY
26 } Options;
27
28 int getOption() {
29     int option;
30
31     printf("\n=====\\n");
32     printf("(%) Criar\\n", CREATE);
33     printf("(%) Preencher aleatoriamente\\n", FILL);
34     printf("(%) Inserir\\n", INSERT);
35     printf("(%) Consultar\\n", QUERY);
36     printf("(%) Imprimir\\n", PRINT);
37     printf("(%) Zerar\\n", RESET);
38     printf("(%) Destruir\\n", DESTROY);
39     printf("(%) Sair\\n", EXIT);
40     printf("=====\\n");
41     printf("Operacao: ");
42
43     scanf("%d", &option);
44     printf("\\n");
45
46     return option;
47 }
48
49 int runMenu() {
50     Matriz* matriz = NULL;
51     int exit = 0, item, pri, linhas, colunas, min, max;
52 }
```

```

53 do {
54     switch (getOption()) {
55         case CREATE:
56             if (matriz != NULL) {
57                 destroiMatriz(&matriz);
58                 printf("Matriz apagada");
59             }
60
61             printf("Insira as dimensoes\n");
62             printf("Linhas: ");
63             scanf("%d", &linhas);
64             printf("Colunas: ");
65             scanf("%d", &colunas);
66
67             matriz = criaMatriz(linhas, colunas);
68             break;
69
70         case FILL:
71             printf("Insira os limites dos valores\n");
72             printf("Min: ");
73             scanf("%d", &min);
74             printf("Max: ");
75             scanf("%d", &max);
76
77             preencheAleatorio(matriz, min, max);
78             break;
79
80         case INSERT:
81             printf("Elemento para inserir: ");
82             scanf("%d", &item);
83             printf("Linha (inicio 0): ");
84             scanf("%d", &linhas);
85             printf("Colunas (inicio 0): ");
86             scanf("%d", &colunas);
87
88             if (insereElem(matriz, item, linhas, colunas)) {
89                 printf("Inserir(%d) (%d, %d)", item, linhas, colunas);
90             } else {
91                 printf("Nao foi possivel inserir(%d)", item);
92             }
93             break;
94
95         case QUERY:
96             printf("Posicao para consultar");
97             printf("Linha (inicio 0): ");
98             scanf("%d", &linhas);
99             printf("Colunas (inicio 0): ");
100             scanf("%d", &colunas);
101
102             if (consultaElem(matriz, &item, linhas, colunas)) {
103                 printf("Elemento = %d\n", item);
104             } else {
105                 printf("Nao foi consultar a matriz nessa posicao");
106             }
107             break;

```

```

108
109     case PRINT:
110         imprime(matriz);
111         break;
112
113     case RESET:
114         zeraMatriz(matriz);
115         printf("Matriz zerada");
116         break;
117
118     case DESTROY:
119         destroiMatriz(&matriz);
120         printf("Matriz destruida");
121         break;
122
123     case EXIT:
124         if (matriz != NULL) {
125             destroiMatriz(&matriz);
126         }
127         printf("Programa encerrado");
128         exit = 1;
129         break;
130
131     default:
132         printf("Opcao desconhecida, tente novamente");
133     }
134     printf("\n");
135 } while (!exit);
136 }
137
138 int main() {
139     runMenu();
140     return 0;
141 }

```

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

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

Insira as dimensoes
Linhas: 4
Colunas: 6

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

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

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

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

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

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

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

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

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

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

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

Matriz destruida

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

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


roteiro-07/ex02-01.h

```
1  #ifndef MATRIZFAIXA_H
2  #define MATRIZFAIXA_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     int i;
17     for (i = 0; i < mf->tam; i++) {
18         mf->diagonal[i] = 0;
19         if (i < mf->tam - 1) {
20             mf->superior[i] = 0;
21             mf->inferior[i] = 0;
22         }
23     }
24 }
25
26 Matriz* criaMatriz(int t) {
27     Matriz* mf;
28     mf = (Matriz*)malloc(sizeof(Matriz));
29     if (mf != NULL) {
30         if (t <= 1) {
31             printf("Dimensao deve ser > 1, matriz nao criada!");
32             return NULL;
33         }
34         mf->tam = t;
35         mf->diagonal = (int*)malloc(t * sizeof(int));
36         mf->superior = (int*)malloc((t - 1) * sizeof(int));
37         mf->inferior = (int*)malloc((t - 1) * sizeof(int));
38         if (mf->diagonal == NULL || mf->superior == NULL || mf->inferior ==
39 NULL)
40             return NULL;
41         zeraMatriz(mf);
42     }
43     return mf;
44 }
45
46 void destroiMatriz(Matriz** mf) {
47     if (*mf != NULL) {
48         free((*mf)->diagonal);
49         free((*mf)->superior);
50         free((*mf)->inferior);
51         free(*mf);
52         *mf = NULL;
53     }
54 }
```

```

55 int preencheAleatorio(Matriz* mf, int ini, int fim) {
56     if (mf == NULL) return 0;
57     srand(time(NULL));
58     int i;
59     for (i = 0; i < mf->tam; i++) {
60         mf->diagonal[i] = ini + rand() % (fim - ini + 1);
61         if (i < mf->tam - 1) {
62             mf->superior[i] = ini + rand() % (fim - ini + 1);
63             mf->inferior[i] = ini + rand() % (fim - ini + 1);
64         }
65     }
66     return 1;
67 }
68
69 int insereElem(Matriz* mf, int elem, int i, int j) {
70     if (mf == NULL) return 0;
71     if (i < 0 || j < 0 || i >= mf->tam || j >= mf->tam) {
72         printf("Valores invalidos, elem nao inserido!\n");
73         return 0;
74     }
75     if (i == j)
76         mf->diagonal[i] = elem;
77     else if (i + 1 == j)
78         mf->superior[i] = elem;
79     else if (i == j + 1)
80         mf->inferior[j] = elem;
81     else {
82         printf("Indices fora da faixa, elem nao inserido!\n");
83         return 0;
84     }
85     return 1;
86 }
87
88 int consultaElem(Matriz* mf, int i, int j) {
89     if (mf == NULL) return 0;
90     if (i < 0 || j < 0 || i >= mf->tam || j >= mf->tam) {
91         printf("Valores invalidos, elem inexistente!\n");
92         return 0;
93     }
94     if (i == j)
95         return mf->diagonal[i];
96     else if (i + 1 == j)
97         return mf->superior[i];
98     else if (i == j + 1)
99         return mf->inferior[j];
100     else
101         return 0;
102 }
103
104 void imprimeFaixaVetores(Matriz* mf) {
105     if (mf == NULL) return;
106     int i;
107     printf("Matriz Faixa, Tam: %d x %d:\n", mf->tam, mf->tam);
108     printf("Diagonal = [");
109     for (i = 0; i < mf->tam; i++)
110         printf("%d ", mf->diagonal[i]);
111     printf("]\n");

```

```
112     printf("Superior = [");
113     for (i = 0; i < mf->tam - 1; i++)
114         printf("%d ", mf->superior[i]);
115     printf("]\n");
116     printf("Inferior = [");
117     for (i = 0; i < mf->tam - 1; i++)
118         printf("%d ", mf->inferior[i]);
119     printf("]\n\n");
120 }
121
122 void imprime(Matriz* mf) {
123     if (mf == NULL) return;
124     int i, j;
125     imprimeFaixaVetores(mf);
126     printf("Matriz Original:\n");
127     for (i = 0; i < mf->tam; i++) {
128         for (j = 0; j < mf->tam; j++)
129             printf("%d\t", consultaElem(mf, i, j));
130         printf("\n");
131     }
132 }
133
134 #endif
```

roteiro-07/ex02-01.c

```
1  #include "ex02-01.h"
2
3  #include <stdio.h>
4  #include <stdlib.h>
5
6  enum {
7      EXIT = 0,
8      CREATE,
9      FILL,
10     INSERT,
11     QUERY,
12     PRINT,
13     RESET,
14     DESTROY
15 } Options;
16
17 int getOption() {
18     int option;
19
20     printf("\n=====\\n");
21     printf("(%) Criar\\n", CREATE);
22     printf("(%) Preencher aleatoriamente\\n", FILL);
23     printf("(%) Inserir\\n", INSERT);
24     printf("(%) Consultar\\n", QUERY);
25     printf("(%) Imprimir\\n", PRINT);
26     printf("(%) Zerar\\n", RESET);
27     printf("(%) Destruir\\n", DESTROY);
28     printf("(%) Sair\\n", EXIT);
29     printf("=====\\n");
30     printf("Operacao: ");
31
32     scanf("%d", &option);
33     printf("\\n");
34
35     return option;
36 }
37
38 int runMenu() {
39     Matriz* matriz = NULL;
40     int exit = 0, item, pri, linhas, colunas, min, max;
41
42     do {
43         switch (getOption()) {
44             case CREATE:
45                 if (matriz != NULL) {
46                     destroiMatriz(&matriz);
47                     printf("Matriz apagada");
48                 }
49
50                 printf("Insira as dimensoes\\n");
51                 printf("Diagonal: ");
52                 scanf("%d", &linhas);
53
54                 matriz = criaMatriz(linhas);
55                 break;
56 }
```

```

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

```

```
116         break;
117
118     default:
119         printf("Opcao desconhecida, tente novamente");
120     }
121     printf("\n");
122 } while (!exit);
123 }
124
125 int main() {
126     runMenu();
127     return 0;
128 }
```

```
gabriel.meira.2004@lab-02-07-04:~/Desktop/ufsj/UF5J-LabProg2/roteiro-07$ ./ex02-01

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

Insira as dimensoes
Diagonal: 5

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

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

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

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

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

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

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

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

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

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

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

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

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

Matriz destruida

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

Programa encerrado
gabriel.meira.2004@lab-02-07-04:~/Desktop/ufsj/UF5J-LabProg2/roteiro-07$
```

roteiro-07/ex02-02.h

```
1  #ifndef MESPARSACSR_H
2  #define MESPARSACSR_H
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     Matriz* ms;
19     ms = (Matriz*)malloc(sizeof(Matriz));
20     if (ms != NULL) {
21         if (l <= 0 || c <= 0 || qnn < 0) {
22             printf("Valores invalidos, matriz nao criada!\n");
23             return NULL;
24         }
25         ms->lin = l;
26         ms->col = c;
27         ms->QI = 0;
28         ms->QNN = qnn;
29         ms->A = ms->IA = ms->JA = NULL;
30         if (qnn != 0) {
31             ms->A = (int*)malloc(qnn * sizeof(int));
32             ms->JA = (int*)malloc(qnn * sizeof(int));
33             if (ms->A == NULL || ms->JA == NULL) return NULL;
34         }
35         ms->IA = (int*)malloc((ms->lin + 1) * sizeof(int));
36         if (ms->IA == NULL) return NULL;
37         int i;
38         for (i = 0; i < l + 1; i++) ms->IA[i] = 0;
39     }
40     return ms;
41 }
42
43 int* meuRealloc(int* v, int tam) {
44     int* aux = (int*)malloc((tam + 1) * sizeof(int));
45     if (aux != NULL) {
46         if (v != NULL) {
47             int i;
48             for (i = 0; i < tam; i++)
49                 aux[i] = v[i];
50             free(v);
51         }
52     }
53     return aux;
54 }
55
```



```

56 void imprimeVetores(Matriz* ms) {
57     if (ms == NULL) return;
58     int i, j;
59     printf("Matriz Esparsa, Tam: %d x %d:\n", ms->lin, ms->col);
60     printf("%d elementos nao nulos.\n", ms->QNN);
61     printf("A = [");
62     for (i = 0; i < ms->QNN; i++)
63         printf("%d ", ms->A[i]);
64     printf("]\n");
65     printf("IA = [");
66     for (i = 0; i < ms->lin + 1; i++)
67         printf("%d ", ms->IA[i]);
68     printf("]\n");
69     printf("JA = [");
70     for (i = 0; i < ms->QNN; i++)
71         printf("%d ", ms->JA[i]);
72     printf("]\n\n");
73 }
74
75 int insereElem(Matriz* ms, int elem, int i, int j) {
76     if (ms == NULL) return 0;
77     if (i < 0 || j < 0 || i >= ms->lin || j >= ms->col) {
78         printf("Valores invalidos, elem nao inserido!\n");
79         return 0;
80     }
81     int k;
82     int index = -1;
83     int ini = ms->IA[i];
84     int fim = ms->IA[i + 1];
85     // Encontre a posição correta para inserir o valor
86     for (k = ini; k < fim; k++)
87         if (ms->JA[k] >= j) {
88             index = k;
89             break;
90         }
91
92     if (index == -1) { // NOVA INSERCAO
93         if (ms->QI == ms->QNN) { // Necessita REALLOC
94             ms->A = meuRealloc(ms->A, ms->QNN);
95             ms->JA = meuRealloc(ms->JA, ms->QNN);
96             ms->QNN++;
97         }
98         // Move elementos para a nova insercao
99         for (k = ms->QNN - 1; k >= fim; k--) {
100             ms->A[k] = ms->A[k - 1];
101             ms->JA[k] = ms->JA[k - 1];
102         }
103         ms->A[fim] = elem;
104         ms->JA[fim] = j;
105         ms->QI++;
106         // Atualiza QNN acumulado
107         for (int k = i + 1; k <= ms->lin; k++)
108             ms->IA[k]++;
109     } else { // Atualiza um valor existente
110         ms->A[index] = elem;
111     }
112     imprimeVetores(ms);
113     return 1;

```

```

114 }
115
116 int removeElem(Matriz* ms, int i, int j) {
117     if (ms == NULL) return 0;
118     if (i < 0 || j < 0 || i >= ms->lin || j >= ms->col) {
119         printf("Valores invalidos, elem nao removido!\n");
120         return 0;
121     }
122
123     int k;
124     int index = -1;
125     int ini = ms->IA[i];
126     int fim = ms->IA[i + 1];
127     // Encontre a posição do valor a ser removido
128     for (k = ini; k < fim; k++)
129         if (ms->JA[k] == j) {
130             index = k;
131             break;
132         }
133
134     if (index != -1) {
135         // Move todos elementos uma posição para tras
136         for (k = index; k < ms->QNN - 1; k++) {
137             ms->A[k] = ms->A[k + 1];
138             ms->JA[k] = ms->JA[k + 1];
139         }
140         ms->QNN--;
141         ms->QI--;
142         // Atualiza QNN acumulado
143         for (int k = i + 1; k <= ms->lin; k++)
144             ms->IA[k]--;
145     } else {
146         printf("Elemento nao existente\n");
147         return 0;
148     }
149     imprimeVetores(ms);
150     return 1;
151 }
152
153 int consultaElem(Matriz* ms, int i, int j) {
154     if (ms == NULL) return 0;
155     if (i < 0 || j < 0 || i >= ms->lin || j >= ms->col) {
156         printf("Valores invalidos, elem inexistente!\n");
157         return 0;
158     }
159     int k;
160     for (k = ms->IA[i]; k < ms->IA[i + 1]; k++)
161         if (ms->JA[k] == j) return ms->A[k];
162     return 0;
163 }
164
165 void imprime(Matriz* ms) {
166     if (ms == NULL) return;
167     int i, j;
168     imprimeVetores(ms);
169     printf("Matriz Original:\n");
170     for (i = 0; i < ms->lin; i++) {
171         for (j = 0; j < ms->col; j++)

```

```
172         printf("%d\t", consultaElem(ms, i, j));
173     printf("\n");
174 }
175 }
176
177 void destroiMatriz(Matriz** ms) {
178     if (*ms != NULL) {
179         free((*ms)->A);
180         free((*ms)->IA);
181         free((*ms)->JA);
182         free(*ms);
183         *ms = NULL;
184     }
185 }
186
187 #endif
```

roteiro-07/ex02-02.c

```
1  #include "ex02-02.h"
2
3  #include <stdio.h>
4  #include <stdlib.h>
5
6  enum {
7      EXIT = 0,
8      CREATE,
9      INSERT,
10     REMOVE,
11     QUERY,
12     PRINT,
13     DESTROY
14 } Options;
15
16 int getOption() {
17     int option;
18
19     printf("\n=====\\n");
20     printf("(%) Criar\\n", CREATE);
21     printf("(%) Inserir\\n", INSERT);
22     printf("(%) Remove\\n", REMOVE);
23     printf("(%) Consultar\\n", QUERY);
24     printf("(%) Imprimir\\n", PRINT);
25     printf("(%) Destruir\\n", DESTROY);
26     printf("(%) Sair\\n", EXIT);
27     printf("=====\\n");
28     printf("Operacao: ");
29
30     scanf("%d", &option);
31     printf("\\n");
32
33     return option;
34 }
35
36 int runMenu() {
37     Matriz* matriz = NULL;
38     int exit = 0, item, pri, linhas, colunas, min, max;
39
40     do {
41         switch (getOption()) {
42             case CREATE:
43                 if (matriz != NULL) {
44                     destroiMatriz(&matriz);
45                     printf("Matriz apagada");
46                 }
47
48                 printf("Insira as dimensoes\\n");
49                 printf("Linhas: ");
50                 scanf("%d", &linhas);
51                 printf("Colunas: ");
52                 scanf("%d", &colunas);
53
54                 matriz = criaMatriz(linhas, colunas, 0);
55                 break;
```

```

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

```

```
114         printf("Programa encerrado");
115         exit = 1;
116         break;
117
118         default:
119             printf("Opcao desconhecida, tente novamente");
120     }
121     printf("\n");
122 } while (!exit);
123 }
124
125 int main() {
126     runMenu();
127     return 0;
128 }
```

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

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

Insira as dimensoes
Linhas: 7
Colunas: 9

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

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

Inseriu(23) (4, 6)

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

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

Inseriu(19) (2, 1)

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

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

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

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

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

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

Matriz destruida

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

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