



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

## Roteiro 7

Rodrigo José Zonzin  
212050002

### 1

```
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 #include "matrizEstatica.h"
5
6
7
8 // Inicializa uma matriz com linhas e colunas especificadas
9 Matriz* criaMatriz(int linhas, int colunas){
10     Matriz *matriz = (Matriz*) malloc(sizeof(Matriz));
11
12     matriz->linhas = linhas;
13     matriz->colunas = colunas;
14     for(int i = 0; i < linhas; i++){
15         for(int j = 0; j < colunas; j++){
16             matriz->data[i][j] = 0; // Pode ser outro valor padr o se preferir
17         }
18     }
19
20     return matriz;
21 }
22
23 // Define o valor de um elemento da matriz na posi o (linha, coluna)
24 void definirElemento(Matriz *matriz, int linha, int coluna, int valor){
25     if(linha >= 0 && linha < matriz->linhas && coluna >= 0 && coluna <
        matriz->colunas){
26         matriz->data[linha][coluna] = valor;
27     } else{
28         printf("Posi o inv lida\n");
29     }
30 }
31
32 // Obt m o valor de um elemento da matriz na posi o (linha, coluna)
33 int obterElemento(Matriz *matriz, int linha, int coluna){
34     if(linha >= 0 && linha < matriz->linhas && coluna >= 0 && coluna <
        matriz->colunas){
35         return matriz->data[linha][coluna];
36     } else{
37         printf("Posi o inv lida\n");
38         return 0; // Pode retornar outro valor padr o se preferir
39     }
```

```

40 }
41
42 // Imprime a matriz
43 void imprimirMatriz(Matriz *matriz){
44     for(int i = 0; i < matriz->linhas; i++){
45         for(int j = 0; j < matriz->colunas; j++){
46             printf("%d ", matriz->data[i][j]);
47         }
48         printf("\n");
49     }
50 }
51
52 Matriz* transposta(Matriz* original){
53     int linhas = original->colunas;
54     int colunas = original->linhas;
55
56     Matriz* resultado = criaMatriz(linhas, colunas);
57
58     for(int i = 0; i < original->linhas; i++) {
59         for(int j = 0; j < original->colunas; j++) {
60             resultado->data[j][i] = original->data[i][j];
61         }
62     }
63
64     return resultado;
65 }
66
67 int main(){
68     Matriz *minhaMatriz = criaMatriz(3, 3);
69
70
71     definirElemento(minhaMatriz, 0, 1, 1);
72     definirElemento(minhaMatriz, 1, 1, 2);
73     definirElemento(minhaMatriz, 2, 2, 3);
74
75     imprimirMatriz(minhaMatriz);
76
77     int valor = obterElemento(minhaMatriz, 1, 1);
78     printf("Valor na posi o (1, 1): %d\n", valor);
79
80     Matriz *minhaTransposta = transposta(minhaMatriz);
81     imprimirMatriz(minhaTransposta);
82
83     return 0;
84 }

```

../ex11/matrizEstatica.c

```

1 #define _MATRIZESTATICA_H
2
3 #define MAX_TAM 100
4
5 struct matriz{
6     int data[MAX_TAM][MAX_TAM];
7     int linhas;
8     int colunas;
9 };
10

```

```
11 typedef struct matriz Matriz;
    ../ex11/matrizEstatica.h
```

## 2

```
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 struct matriz {
5     int **data;
6     int linhas;
7     int colunas;
8 };
9
10 typedef struct matriz Matriz;
11
12 Matriz *criaMatriz(int linhas, int colunas) {
13     Matriz *mat = (Matriz *)malloc(sizeof(Matriz));
14     if (mat == NULL) {
15         perror("Erro ao alocar a matriz");
16         exit(1);
17     }
18
19     mat->linhas = linhas;
20     mat->colunas = colunas;
21
22     mat->data = (int **)malloc(linhas * sizeof(int *));
23     if (mat->data == NULL) {
24         perror("Erro ao alocar as linhas da matriz");
25         exit(1);
26     }
27
28     for (int i = 0; i < linhas; i++) {
29         mat->data[i] = (int *)malloc(colunas * sizeof(int));
30         if (mat->data[i] == NULL) {
31             perror("Erro ao alocar as colunas da matriz");
32             exit(1);
33         }
34     }
35
36     return mat;
37 }
38
39 void preencherMatriz(Matriz *mat, int *valores) {
40     for (int i = 0; i < mat->linhas; i++) {
41         for (int j = 0; j < mat->colunas; j++) {
42             mat->data[i][j] = valores[i * mat->colunas + j];
43         }
44     }
45 }
46
47 void imprimirMatriz(Matriz *mat) {
48     for (int i = 0; i < mat->linhas; i++) {
49         for (int j = 0; j < mat->colunas; j++) {
50             printf("%d ", mat->data[i][j]);
51         }
52         printf("\n");
53     }
54 }
```

```

54 }
55
56 void liberarMatriz(Matriz *mat) {
57     for (int i = 0; i < mat->linhas; i++) {
58         free(mat->data[i]);
59     }
60     free(mat->data);
61     free(mat);
62 }
63
64 Matriz* transposta(Matriz* original){
65     int linhas = original->colunas;
66     int colunas = original->linhas;
67
68     Matriz* resultado = criaMatriz(linhas, colunas);
69
70     for(int i = 0; i < original->linhas; i++) {
71         for(int j = 0; j < original->colunas; j++) {
72             resultado->data[j][i] = original->data[i][j];
73         }
74     }
75
76     return resultado;
77 }
78
79 int main() {
80     int valores[] = {1, 2, 3, 4, 5, 6};
81     int linhas = 2;
82     int colunas = 3;
83
84     Matriz *minhaMatriz = criaMatriz(linhas, colunas);
85     preencherMatriz(minhaMatriz, valores);
86
87     printf("Matriz:\n");
88     imprimirMatriz(minhaMatriz);
89
90     Matriz *minhaTransposta = transposta(minhaMatriz);
91     imprimirMatriz(minhaTransposta);
92
93     liberarMatriz(minhaMatriz);
94     liberarMatriz(minhaTransposta);
95
96     return 0;
97 }

```

../ex12/matriz.c

### 3

```

1 #include <stdio.h>
2 #include <stdlib.h>
3
4 // Definição da estrutura da Matriz de Faixa (Tridiagonal)
5 typedef struct {
6     int n;           // Dimensão da matriz
7     int* diag;        // Elementos da diagonal principal
8     int* sup;         // Elementos da superdiagonal (acima da diagonal principal)
9     int* sub;         // Elementos da subdiagonal (abaixo da diagonal principal)
10 } TridiagonalMatrix;

```

```

11
12 // Função para inicializar uma Matriz de Faixa
13 TridiagonalMatrix* createTridiagonalMatrix(int n) {
14     TridiagonalMatrix* mat = (TridiagonalMatrix*)malloc(sizeof(TridiagonalMatrix));
15     if (mat == NULL) {
16         perror("Erro ao alocar memória para a matriz");
17         exit(1);
18     }
19
20     mat->n = n;
21     mat->diag = (int*)malloc(n * sizeof(int));
22     mat->sup = (int*)malloc((n - 1) * sizeof(int));
23     mat->sub = (int*)malloc((n - 1) * sizeof(int));
24
25     if (mat->diag == NULL || mat->sup == NULL || mat->sub == NULL) {
26         perror("Erro ao alocar memória para os elementos da matriz");
27         exit(1);
28     }
29
30     return mat;
31 }
32
33 // Função para definir um elemento na matriz
34 void setElement(TridiagonalMatrix* mat, int i, int j, int value) {
35     if (i < 0 || i >= mat->n || j < 0 || j >= mat->n) {
36         fprintf(stderr, "Índices fora dos limites da matriz\n");
37         exit(1);
38     }
39
40     if (i == j) {
41         mat->diag[i] = value;
42     } else if (i == j - 1) {
43         mat->sup[i] = value;
44     } else if (i == j + 1) {
45         mat->sub[j] = value;
46     } else {
47         fprintf(stderr, "Elemento fora da banda tridiagonal\n");
48         exit(1);
49     }
50 }
51
52 // Função para obter um elemento da matriz
53 int getElement(const TridiagonalMatrix* mat, int i, int j) {
54     if (i < 0 || i >= mat->n || j < 0 || j >= mat->n) {
55         fprintf(stderr, "Índices fora dos limites da matriz\n");
56         exit(1);
57     }
58
59     if (i == j) {
60         return mat->diag[i];
61     } else if (i == j - 1) {
62         return mat->sup[i];
63     } else if (i == j + 1) {
64         return mat->sub[j];
65     } else {
66         return 0; // Elementos fora da banda tridiagonal são zeros
67     }
68 }
69

```

```

70 void imprimeVetoresMatrix(TridiagonalMatrix* mat){
71     printf("Diagonal: ");
72     for (int i = 0; i < mat->n; i++) {
73         printf("%d ", mat->diag[i]);
74     }
75     printf("\n");
76
77     printf("Superdiagonal: ");
78     for (int i = 0; i < mat->n - 1; i++) {
79         printf("%d ", mat->sup[i]);
80     }
81     printf("\n");
82
83     printf("Subdiagonal: ");
84     for (int i = 0; i < mat->n - 1; i++) {
85         printf("%d ", mat->sub[i]);
86     }
87     printf("\n");
88 }
89
90 // Função para liberar a memória da matriz
91 void freeTridiagonalMatrix(TridiagonalMatrix* mat) {
92     free(mat->diag);
93     free(mat->sup);
94     free(mat->sub);
95     free(mat);
96 }
97
98 int main() {
99     int n = 5;
100     TridiagonalMatrix* mat = createTridiagonalMatrix(n);
101
102     // Preencher a matriz com alguns valores de exemplo
103     for (int i = 0; i < n; i++) {
104         setElement(mat, i, i, 2);
105     }
106     for (int i = 0; i < n - 1; i++) {
107         setElement(mat, i, i + 1, 1);
108         setElement(mat, i + 1, i, 1);
109     }
110
111     imprimeVetoresMatrix(mat);
112
113     // Liberar a memória da matriz
114     freeTridiagonalMatrix(mat);
115
116     return 0;
117 }

```

../ex13/matrizFaixa.c

## 4

Em branco.