1. Estrutura de Dados Homogênea Bidimensional: Matriz

Uma *matriz* é uma coleção homogênea bidimensional, com elementos distribuídos em linhas e colunas. Se A é uma matriz $m \times n$, então suas linhas são indexadas de 0 a $m-1$ e suas colunas de 0 a $n-1$. Para acessar um elemento de A, escrevemos $A[i][j]$, sendo i o número da linha e j o número da coluna que o elemento ocupa (Figura 1).

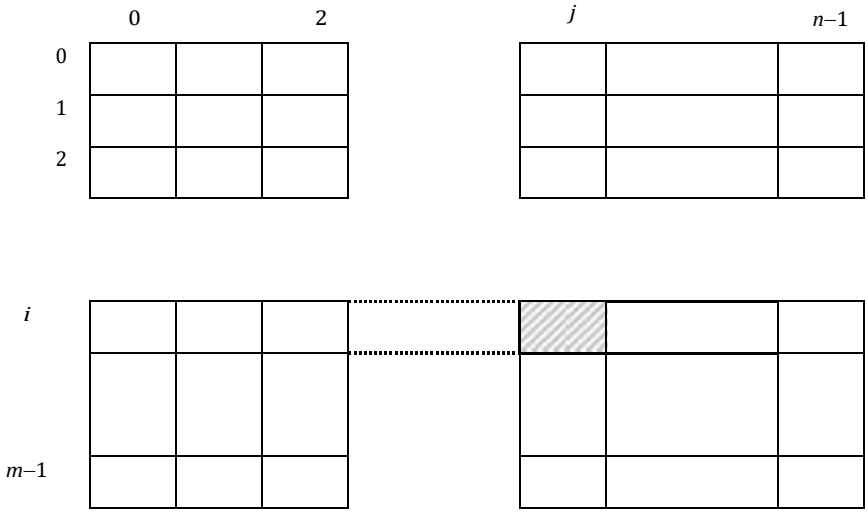

Figura 1 – Ilustração de uma matriz bidimensional.

- **Exemplo 1.** Uma matriz 3×4 de números inteiros pode ser declarada como:

```
int A[3][4];
```

  - Interpretação: essa declaração cria um vetor A cujos elementos $A[0]$, $A[1]$ e $A[2]$ são vetores contendo cada um deles 4 elementos do tipo *int*.

Acesso aos elementos: É comum utilizar estruturas de repetição para manipular uma matriz. Para tanto, índices são utilizados para controlar linhas e colunas: um para linhas e outro para colunas.

- **Exemplo 2.** Como controlar as coordenadas para os elementos de uma matriz.

```
1.  #include <stdio.h>
2.  #define m 3
3.  #define n 4
4.  int main()
5.  {
6.        int A[m][n], cont=0;
```

```
7.      for(int i=0; i<m; i++) //Um laço para acessar as linhas da matriz A
8.      {
9.          putchar('\n');
10.         for(int j=0; j<n; j++)//Para cada linha i, um laço para acessar
            as colunas correspondentes
11.         {   printf("[%d][%d] ", i, j);
12.             A[i][j]=cont;
13.             cont++;
14.         }
15.     }
16.     printf("\n\nConteúdo da matriz A\n");
17.     for(int i=0; i<m; i++) //Um laço para acessar as linhas da matriz A
18.     {
19.         putchar('\n');
20.         for(int j=0; j<n; j++)//Para cada linha i, um laço para acessar
    as colunas correspondentes
21.             printf("[%d][%d]->%d ", i, j, A[i][j]);
22.     }
23.     return 0;
24. }
```

Saídas produzidas pelo programa (primeiros dois laços, das linhas 7 a 15):

```
[0][0] [0][1] [0][2] [0][3]
[1][0] [1][1] [1][2] [1][3]
[2][0] [2][1] [2][2] [2][3]
```

Saídas produzidas pelo programa (últimos dois laços, das linhas 17 a 22):

```
[0][0]->0 [0][1]->1 [0][2]->2 [0][3]->3
[1][0]->4 [1][1]->5 [1][2]->6 [1][3]->7
[2][0]->8 [2][1]->9 [2][2]->10 [2][3]->11
```

- Ou seja, as coordenadas e cada um dos elementos da matriz A com dimensões 3×4.

Um dos usos mais comuns de matrizes em C é quando precisamos armazenar uma coleção de *strings*. Como uma *string* é um vetor, devemos criar um vetor cujos elementos também sejam vetores.

- **Exemplo 3.** Uma matriz de caracteres usada como um vetor de strings.

```
1. #include <stdio.h>
2. #define l 5
3. #define m 20
4. int main()
5. {
6.      char n[l][m];
7.      int i;
8.      for(i=0; i<l; i++)
9.      {
10.          printf("Digite o %do. nome: ", i+1);
11.          scanf(" %[^\n]s", n[i]);
12.     }
13.     for(i=0; i<l; i++)
14.          printf("\n%do. nome digitado: %s", i+1,n[i]);
15.     //system("PAUSE");
16.     return 0;
17. }
```

As instruções apresentadas nas linhas 11 e 14 permitem tratar a matriz como um vetor de *strings*, visto que somente as linhas são controladas pelos laços. Para cada linha, as colunas são preenchidas, de forma automática, a partir de cada item digitado pelo usuário.

- **Desafio 1.** Codifique um programa para ler uma matriz quadrada de ordem *n* e exibir apenas os elementos da diagonal principal.

2. Inicialização de Matrizes

Se lembrarmos de que uma matriz nada mais é que um vetor, cujos elementos são vetores, a inicialização não tem grandes novidades.

Por exemplo, quando uma matriz é instanciada, o número de linhas pode ser omitido. Nesse caso, o compilador determina essa dimensão contando os elementos fornecidos na lista de valores iniciais.

- **Exemplo 4.** Inicializando e exibindo um labirinto.

```
#include <stdio.h>
#define d 10
int main()
{
  int lab[][d] = {
  {1,1,1,1,1,1,1,1,1,1},
  {0,0,1,0,0,0,1,0,1,1},
  {1,0,1,0,1,0,1,0,1,1},
  {1,0,1,0,1,0,0,0,0,1},
  {1,0,1,1,1,0,1,1,0,1},
  {1,0,0,0,0,0,1,0,1,1},
  {1,0,1,0,0,1,1,0,1,1},
  {1,0,0,1,0,1,0,0,0,1},
  {1,0,1,1,0,0,0,1,0,0},
  {1,1,1,1,1,1,1,1,1,1}
  }, i, j;

  for(i=0; i<d; i++)
  {
        putchar('\n');
        for(j=0; j<d; j++)
              putchar(lab[i][j] ? 35 : 32);
  }
  //system("PAUSE");
  return 0;
}
```

– A saída do programa é:

```
> clang-7 -pthread -lm -o main main.c
> ./main

##########
  #   # ##
# # # # ##
# # #    #
# ### ## #
#     # ##
# #  ## ##
#  # #   #
# ##   #
##########>
```

– Note que cada um dos 10 elementos da matriz é um vetor de 10 inteiros.

- **Exemplo 5.** Inicializando e exibindo um menu de opções.

```
#include <stdio.h>
int main()
{
   char menu[][7] = { "abrir", "editar", "salvar", "sair"};
   int i;
   for(i=0; i<4; i++)
      puts(menu[i]);
   return 0;
}
```

– Note que a matriz *menu* é usada no programa como um vetor de *strings*:

	0	1	2	3	4	5	6
0	a	b	r	i	r	\0	
1	e	d	i	t	a	r	\0
2	s	a	l	v	a	r	\0
3	s	a	i	r	\0		

Figura 2 – *A matriz menu indicada no exemplo 5.*

Exemplo 6. Inicializando parcialmente uma matriz, as demais posições são definidas automaticamente pelo compilador como 0.

```
#include <stdio.h>
#define n 2
#define m 3
int main()
{
 int A[n][m]={{9,6,4},};
       printf("\n\t\tValores matriz A:\n ");
 for (int i=0; i<n; i++)
 {     for(int j=0; j<m; j++)
             printf("\t[%d]", A[i][j]);
       printf("\n");
 }
 return 0;
}
```

```
clang-7 -pthread -lm -o main main.c
./main

      Valores matriz A:
   [9] [6] [4]
   [0] [0] [0]
```

Exemplo 7. Inicializando parcialmente uma matriz, as demais posições são definidas automaticamente pelo compilador como 0.

```
#include <stdio.h>
#define n 2
#define m 3
int main()
{
	int A[][m]={{},{}};
	printf("\n\t\tValores matriz A:\n ");
	for (int i=0; i<n; i++)
	{	for(int j=0; j<m; j++)
			printf("\t[%d]", A[i][j]);
		printf("\n");
	}
	return 0;
}
```

```
clang-7 -pthread -lm -o main main.c
./main

        Valores matriz A:
    [0] [0] [0]
    [0] [0] [0]
```

Exemplo 8. **Atenção, não é possível** definir uma matriz sem definir explicitamente o número de colunas.

```
#include <stdio.h>
#define n 2
#define m 3
int main()
{
	int A[2][]={{3,1,4},{}};
	printf("\n\t\tValores matriz A:\n ");
	for (int i=0; i<n; i++)
	{	for(int j=0; j<m; j++)
			printf("\t[%d]", A[i][j]);
		printf("\n");
	}
	return 0;
}
```

```
clang-7 -pthread -lm -o main main.c
main.c:7:7: error: array has incomplete element type 'int []'
        int A[][]={{3,1,4},{}};
              ^
1 error generated.
exit status 1
```

Atenção: As análises e complmentos de códigos por compiliadores têm limites, não padronizados. Logo, não assuma que o compilador realizará todas as definições necessárias para garantir a execução correta do seu código, como inicializações ou mesmo definições implícitas de dimensões.

Exercício (Teste 2). Crie um programa para inicializar e exibir uma matriz representandoum tabuleiro para o "*jogo da velha*", conforme a seguir:

```
   | o | x
---+---+---
 o | x | o
---+---+---
 x |   |
```

Referência

- SALES, André Barros de; AMVAME-NZE, Georges Daniel. Linguagem C: roteiro de experimentos para aulas práticas [recurso eletrônico]. Florianópolis: UFSC, 2016. Disponível em: <http://repositorio.unb.br/handle/10482/21540>.
  - ❑ Páginas 119 a 130.
  - ❑ Realizar os Experimentos e Atividades de Fixação
  - ❑ Complementar com leitura das páginas 135 a 141.