

MATRIZES

Prof. Muriel Mazzetto
Algoritmos 2

Estruturas de dados

2

- Modo de armazenar e manipular dados na memória de um computador.
- Estruturas de dados homogêneas:
 - ▣ Mesmo tipo de dado agrupado.
 - ▣ Variável indexada (utilizam índices para diferenciar).
 - ▣ Vetores, Strings e Matrizes.
- Estrutura de dados heterogêneas:
 - ▣ Diferentes dados agrupados.
 - ▣ Registros (structs em C).

Conceito de vetor

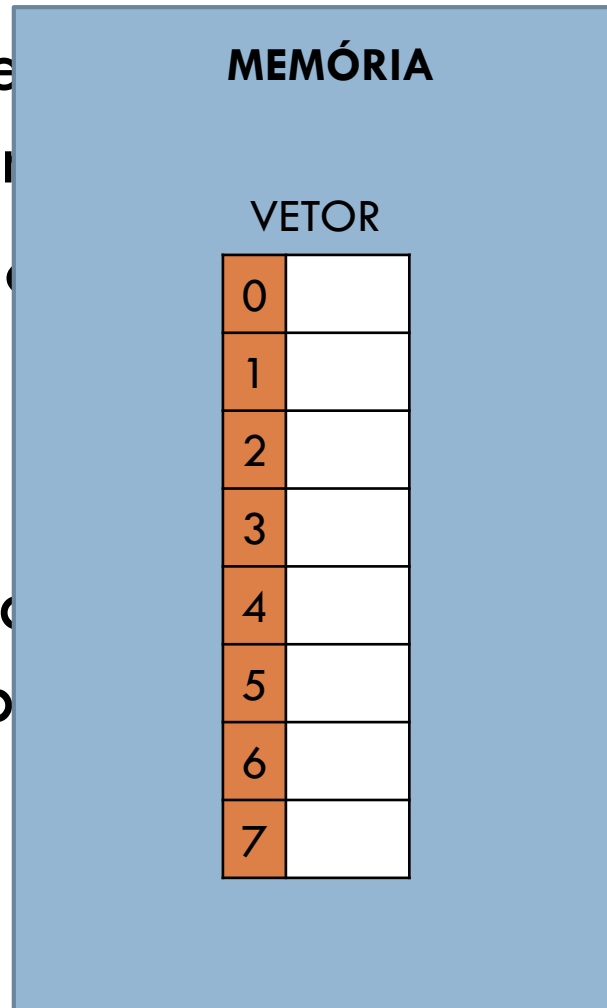
3

- ❑ Declarar um vetor equivale a reservar um espaço na memória temporariamente.
 - ❑ Espaço dividido de acordo com o tipo de dado.
 - ❑ Vários valores do mesmo tipo.
-
- ❑ Agrupamento de variáveis do mesmo tipo, identificadas por índices.

Conceito de vetor

4

- ❑ Declarar um vetor na memória tem o objetivo de reservar um espaço
- ❑ Espaço dividido em células, cada uma com um tipo de dado.
- ❑ Vários valores
- ❑ Agrupamento de elementos com o mesmo tipo, identificadas por índices



Conceito de matriz

5

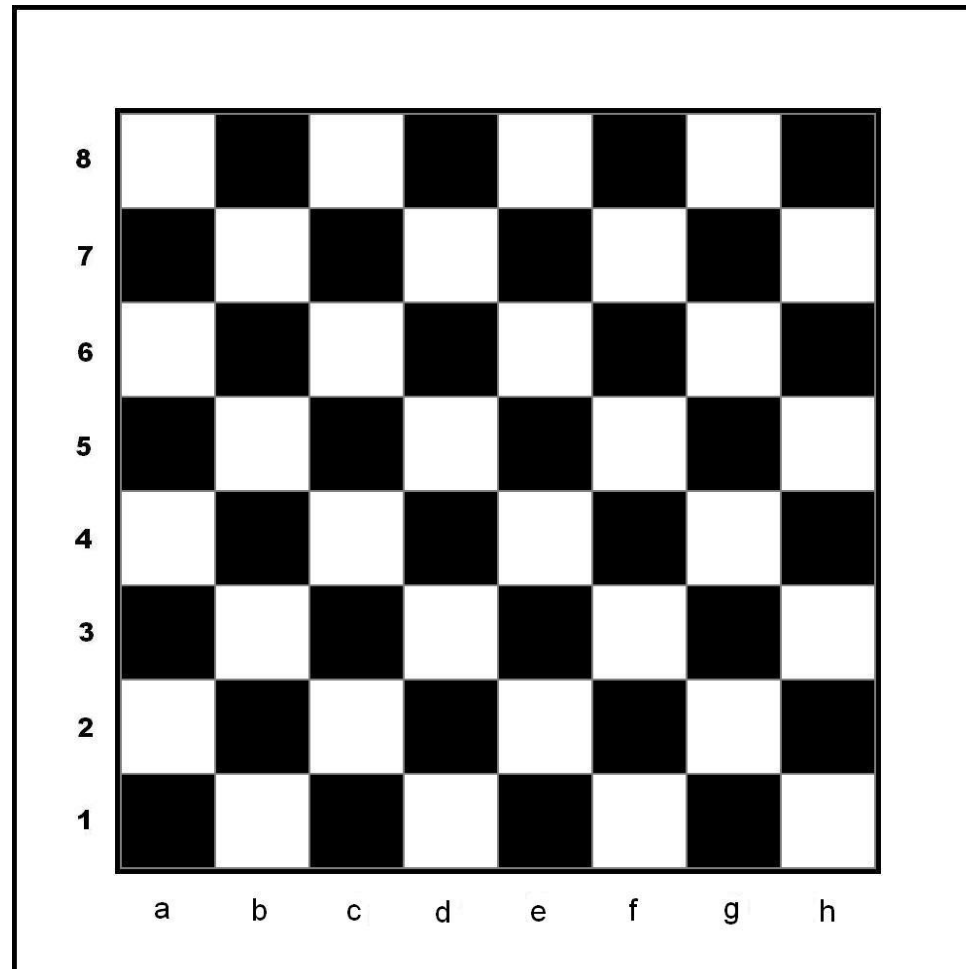
- ❑ Declarar uma matriz equivale a reservar um espaço na memória temporariamente.
- ❑ Espaço dividido de acordo com o tipo de dado.
- ❑ Vários valores do mesmo tipo.

- ❑ Vetor bidimensional.
- ❑ Conjunto de vetores.
- ❑ Possui relação espacial (coordenada).

Conceito de matriz

6

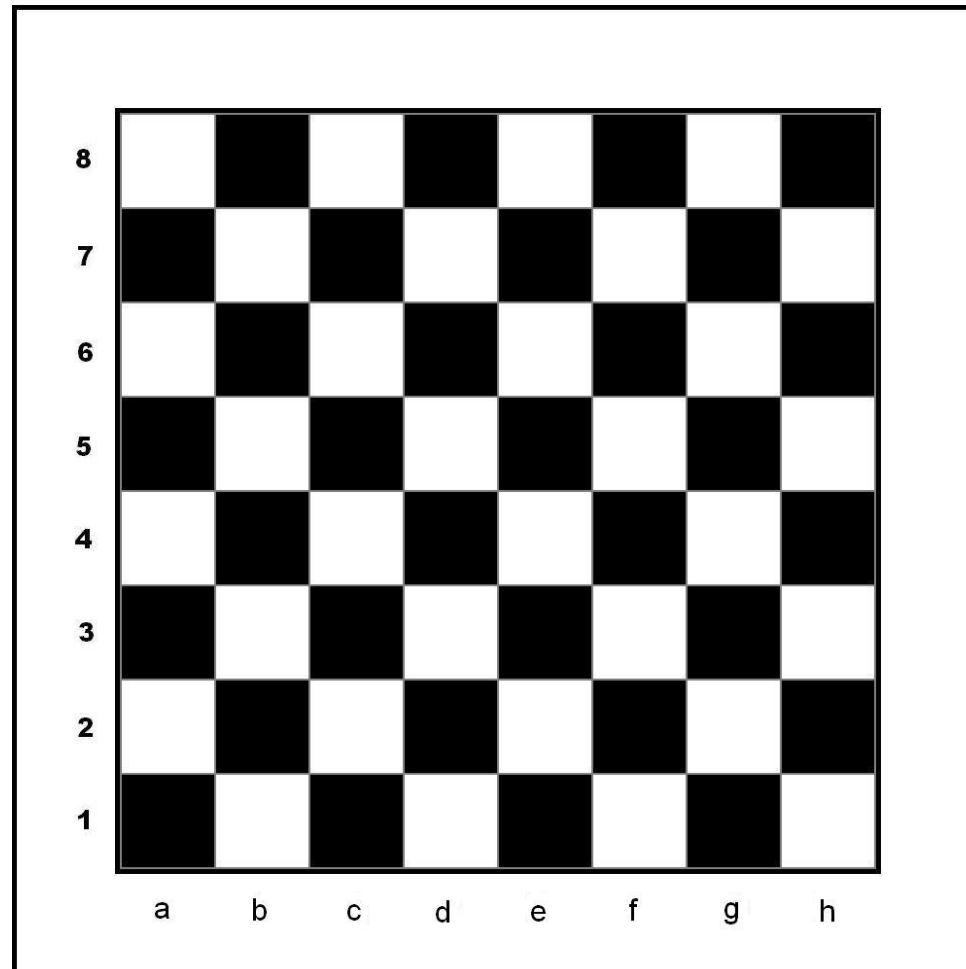
- Tabuleiro de xadrez:



Conceito de matriz

7

□ Tabuleiro de xadrez:

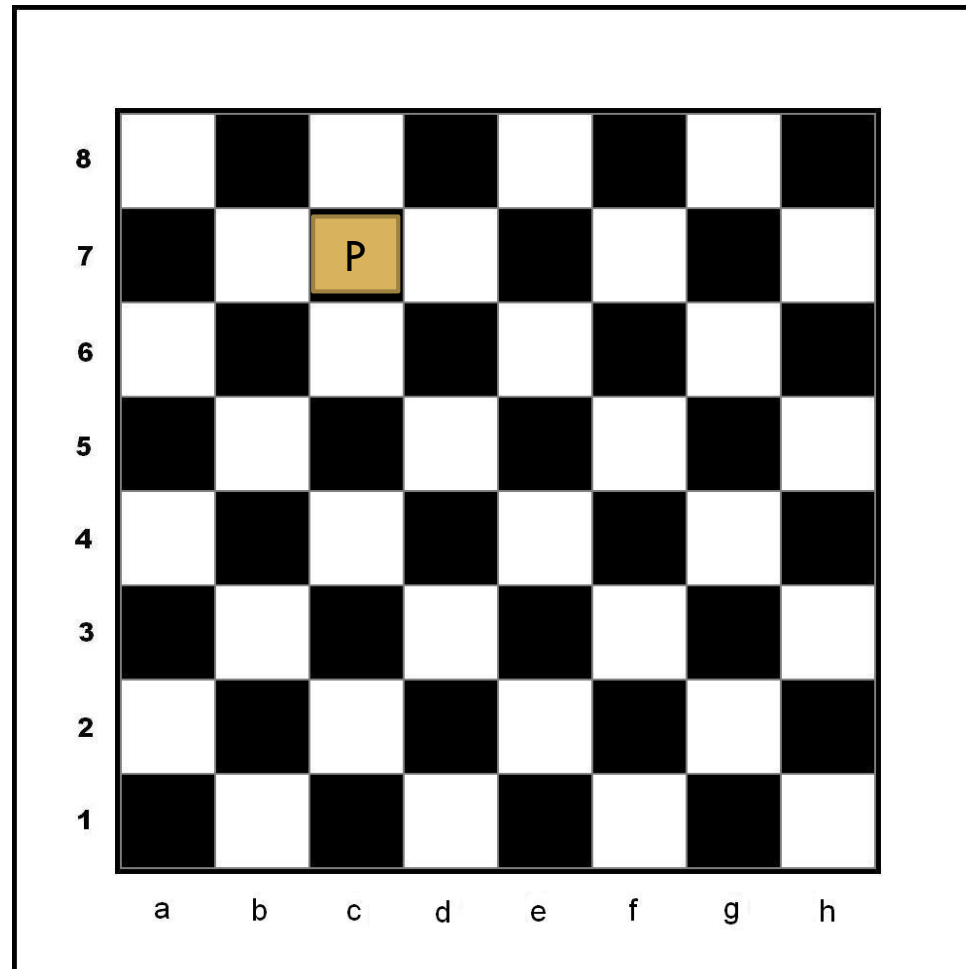


Conceito de matriz

8

- Tabuleiro de xadrez:

[7] [C]

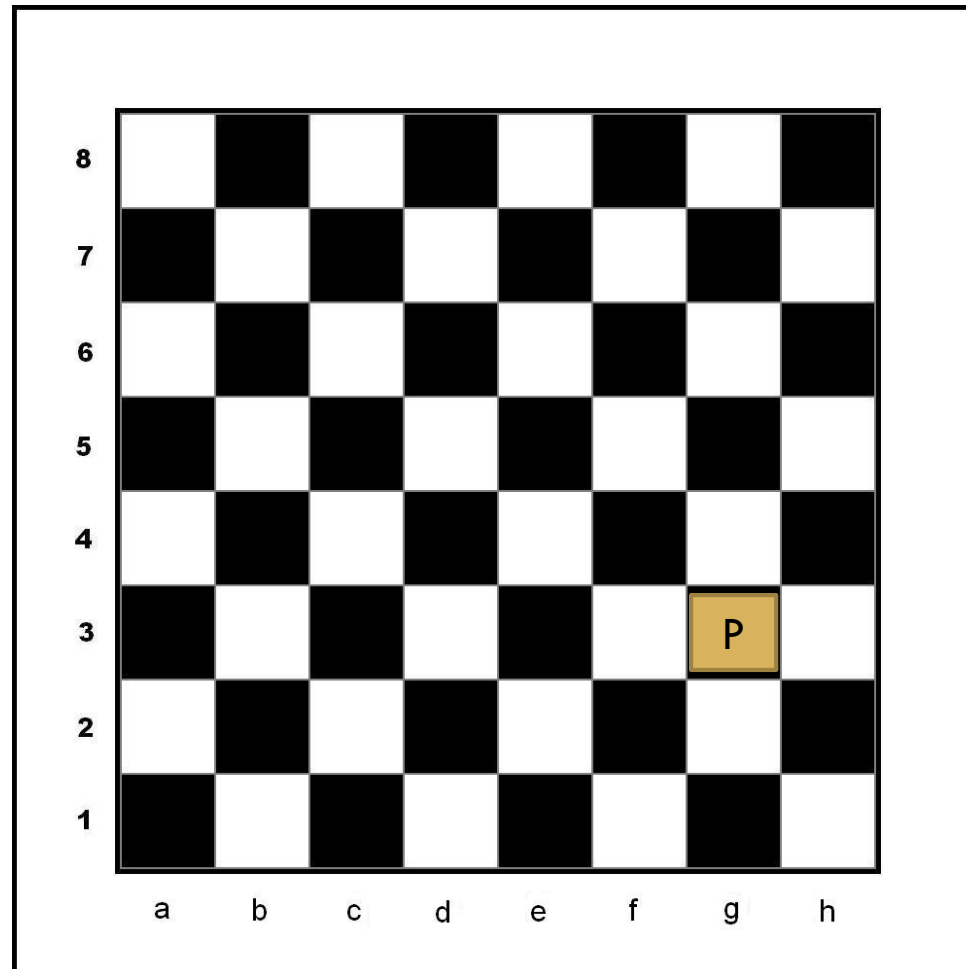


Conceito de matriz

9

- Tabuleiro de xadrez:

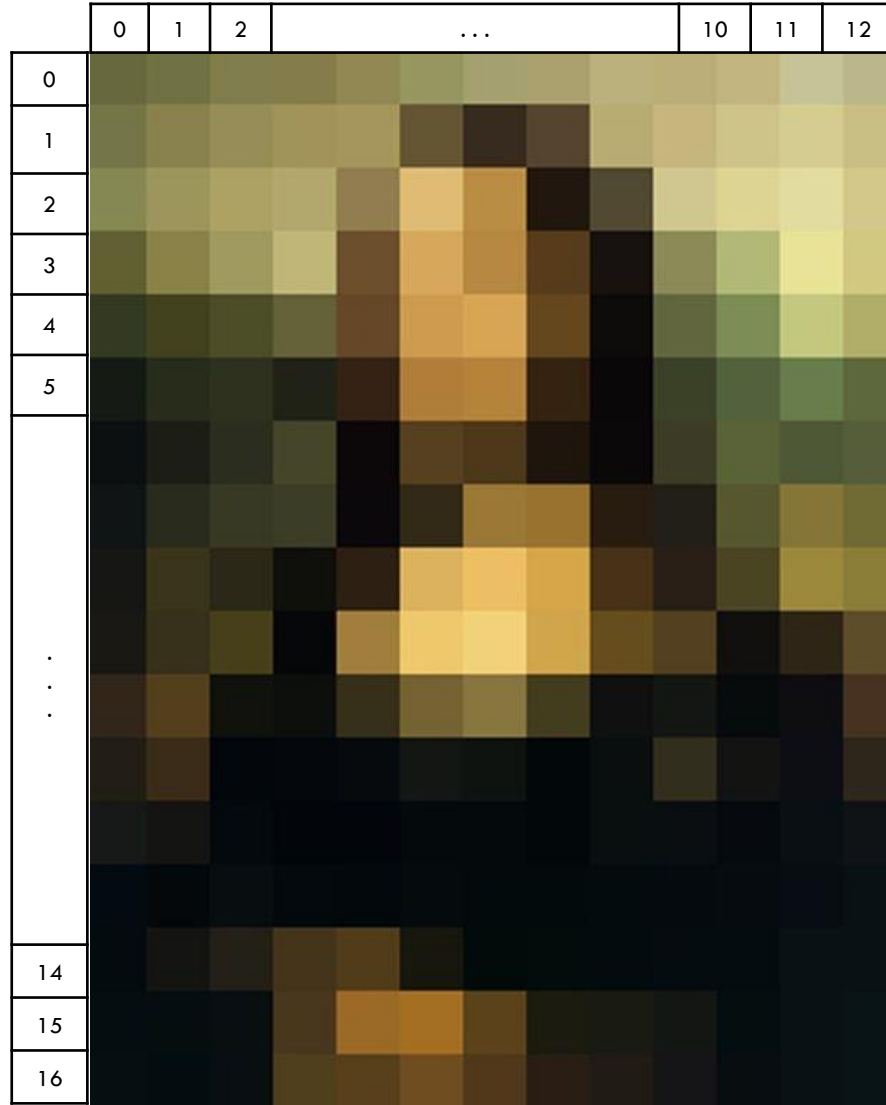
[3] [G]



Conceito de matriz

10

□ Imagem:

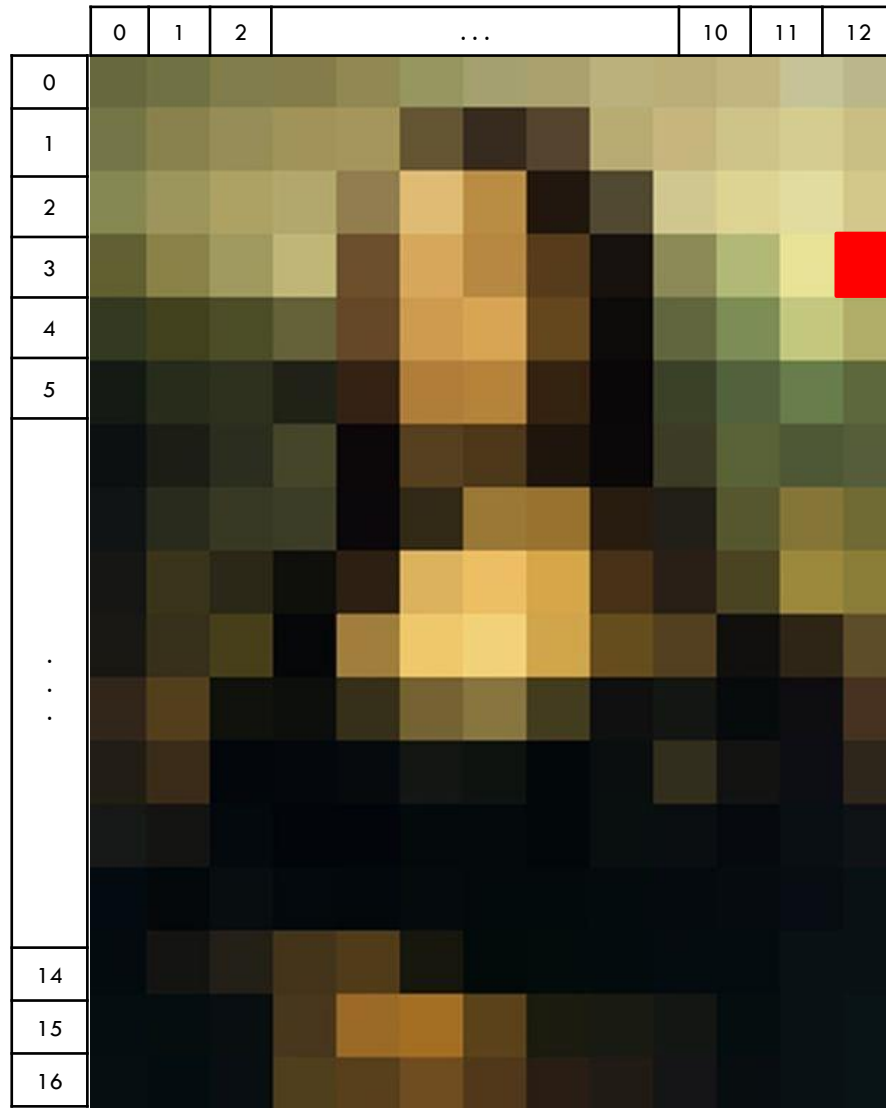


Conceito de matriz

11

□ Imagem:

[3] [1 2]



Conceito de matriz

12

- A sintaxe para declarar uma matriz em C é:

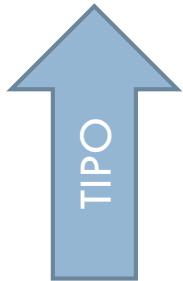
Tipo Nome[qtd_linhas][qtd_colunas];

Conceito de matriz

13

- A sintaxe para declarar uma matriz em C é:

Tipo Nome[qtd_linhas][qtd_colunas];



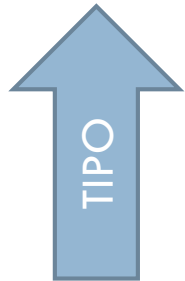
Tipo de dado que
cada espaço da
matriz irá
armazenar.

Conceito de matriz

14

- A sintaxe para declarar uma matriz em C é:

`Tipo Nome[qtd_linhas][qtd_colunas];`

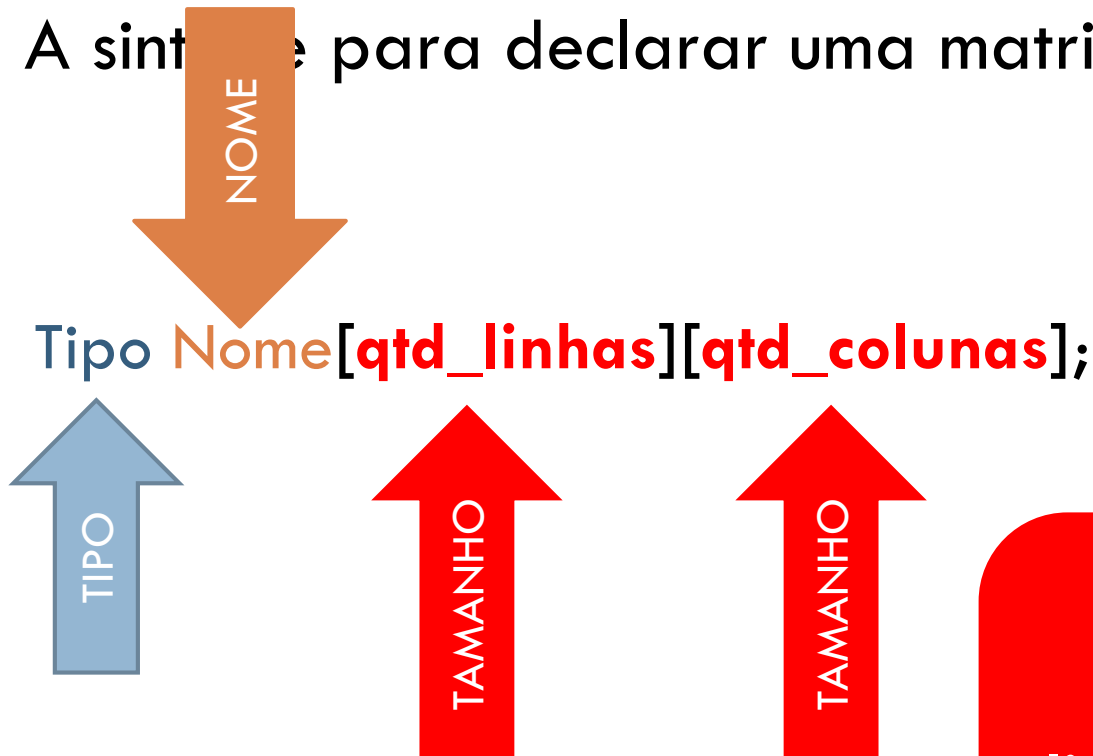


Nome que será
utilizado para
utilizar a variável.

Conceito de matriz

15

- A sintaxe para declarar uma matriz em C é:


Tipo Nome[qtd_linhas][qtd_colunas];

TAMANHO:
quantidade de
linhas e de **colunas**
que a matriz terá
(tabela).

Conceito de matriz

16

□ Exemplo:

□ `int Matriz[7][6];`

Conceito de matriz

17

□ Exemplo:

▣ `int Matriz[7][6];`

- 7 linhas;

- Cada linha

com 6 colunas;

Conceito de matriz

18

□ Exemplo:

▣ `int Matriz[7][6];`

- 7 linhas;

- Cada linha

com 6 colunas;

- Possui 7x6 espaços de armazenamento.

MEMÓRIA						
Matriz						
	0	1	2	3	4	5
0						
1						
2						
3						
4						
5						
6						

Conceito de matriz

19

□ Exemplo:

□ `int Matriz[7][6];`

- Cada linha pode ser considerada como um vetor.

MEMÓRIA						
Matriz						
	0	1	2	3	4	5
0						
1						
2						
3						
4						
5						
6						

Conceito de matriz

20

□ Exemplo:

□ `int Matriz[7][6];`

□ Os índices vão
de 0 até
TAMANHO - 1

MEMÓRIA						
Matriz						
	0	1	2	3	4	5
0						
1						
2						
3						
4						
5						
6						

Conceito de matriz

21

□ Exemplo:

□ `int Matriz[7][6];`

□ Cada posição é acessada através do conjunto de índices.

□ Ex: `Matriz[1][2]`

MEMÓRIA						
Matriz						
	0	1	2	3	4	5
0						
1						
2						
3						
4						
5						
6						

Conceito de matriz

22

□ Exemplo:

□ `int Matriz[7][6];`

□ Cada posição armazena um dado do **tipo declarado**.

□ Ex: `Matriz[1][2] = 25;`

MEMÓRIA						
Matriz						
	0	1	2	3	4	5
0						
1			25			
2						
3						
4						
5						
6						

Conceito de matriz

23

Exemplo:

▣ `int Matriz[7][6];`

▣ Cada posição armazena um dado do **tipo declarado**.

▣ Ex: `Matriz[6][5] = 7;`

MEMÓRIA						
Matriz						
	0	1	2	3	4	5
0						
1						
2						
3						
4						
5						
6						7

Conceito de matriz

24

□ Exemplo:

□ **float** Mat[6][5];

□ Cada posição armazena um dado do **tipo declarado**.

□ Ex: Mat[5][4] = 0.4;

MEMÓRIA

Mat

	0	1	2	3	4
0					
1					
2					
3					
4					
5					0.4

Conceito de matriz

25

Exemplo:

▣ **float** Mat[6][5];

▣ printf("%f", Mat[5][4]);

▣ scanf("%f", &Mat[5][4]);

MEMÓRIA

Mat

	0	1	2	3	4
0					
1					
2					
3					
4					
5					0.4

Conceito de matriz

26

□ Exemplo:

□ **float** Mat[6][5];

- Mat[0][0] = 1;
- Mat[0][1] = 2.3;
- Mat[3][3] = 4.7;
- Mat[1][4] = 4;
- Mat[4][3] = 7.4;
- Mat[5][0] = 17;
- Mat[2][2] = 22.2;
- Mat[5][2] = 10.8;
- Mat[1][1] = 78;

MEMÓRIA					
Mat					
	0	1	2	3	4
0	1.0	2.3			
1		78.0			4.0
2			22.2		
3				4.7	
4				7.4	
5	17.0		10.8		

Questionário

27

- 1) Defina o que é uma matriz.
- 2) Escreva os códigos para:
 - ▣ A) Alocar uma matriz de char de 4 linhas e 3 colunas.
 - ▣ B) Inserir o valor 'A' na linha 0 coluna 2, e '\0' na linha 3 e coluna 2.
 - ▣ C) Desenhe a representação dessa matriz após inserir os valores da letra B.
 - ▣ D) Escreva as operações scanf()/printf() para ler/imprimir o valor de uma posição [i][j] dessa matriz.

Conceito de matriz

28

□ Inicialização de matriz:

▣ `int matriz[2][4] = {{17, 33, 21, 15}, {13, 81, 97, 67}}`;

Conceito de matriz

29

□ Inicialização de matriz:

□ `int matriz[2][4] = {{17, 33, 21, 15}, {13, 81, 97, 67}};`

Primeira linha (`matriz[0][]`)

Segunda linha (`matriz[1][]`)

Conceito de matriz

30

□ Inicialização de matriz:

▣ `char mes[12][4] = {"jan", "fev", "mar", "abr", "mai", "jun", "jul", "ago", "set", "out", "nov", "dez"};`

	0	1	2	3
0	j	a	n	\0
1	f	e	v	\0
2	m	a	r	\0
3	a	b	r	\0
4	m	a	i	\0
...	...			
11	d	e	z	\0

Percorrendo um vetor

31

```
#include <stdio.h>

int main(void)
{
    //quantidade de posições do vetor
    int tam = 10;

    //declaração do vetor
    int vetor[tam];

    //variável auxiliar para índices
    int i;
    |
    //leitura do teclado para cada índice
    //percorrendo o vetor, variando o índice i de 0 até tam-1
    for(i = 0; i < tam; i++)
    {
        scanf("%d", &vetor[i]);
    }

    //impressão de cada índice
    for(i = 0; i < tam; i++)
    {
        printf("%d ", vetor[i]);
    }

    return 0;
}
```

MEMÓRIA

VETOR

0	
1	
2	
3	
4	
5	
6	
.	...

Percorrendo um vetor

32

```
#include <stdio.h>

int main(void)
{
    //quantidade de posições do vetor
    int tam = 10;

    //declaração do vetor
    int vetor[tam];

    //variável auxiliar para índices
    int i;

    //leitura do teclado para cada índice
    //percorrendo o vetor, variando o índice i de 0 até tam-1
    for(i = 0; i < tam; i++)
    {
        scanf("%d", &vetor[i]);
    }

    //impressão de cada índice
    for(i = 0; i < tam; i++)
    {
        printf("%d ", vetor[i]);
    }

    return 0;
}
```



MEMÓRIA

VETOR

0	10
1	20
2	30
3	40
4	50
5	60
6	70
.	...

Percorrendo um vetor

33

```
#include <stdio.h>

int main(void)
{
    //quantidade de posições do vetor
    int tam = 10;

    //declaração do vetor
    int vetor[tam];

    //variável auxiliar para índices
    int i;
    |
    //leitura do teclado para cada índice
    //percorrendo o vetor, variando o índice i de 0 até tam-1
    for(i = 0; i < tam; i++)
    {
        scanf("%d", &vetor[i]);
    }

    //impressão de cada índice
    for(i = 0; i < tam; i++)
    {
        printf("%d ", vetor[i]);
    }

    return 0;
}
```

i	0
---	---

MEMÓRIA

VETOR

0	10
1	20
2	30
3	40
4	50
5	60
6	70
.	...

Percorrendo um vetor

34

```
#include <stdio.h>

int main(void)
{
    //quantidade de posições do vetor
    int tam = 10;

    //declaração do vetor
    int vetor[tam];

    //variável auxiliar para índices
    int i;

    //leitura do teclado para cada índice
    //percorrendo o vetor, variando o índice i de 0 até tam-1
    for(i = 0; i < tam; i++)
    {
        scanf("%d", &vetor[i]);
    }

    //impressão de cada índice
    for(i = 0; i < tam; i++)
    {
        printf("%d ", vetor[i]);
    }

    return 0;
}
```

i	1
---	---

MEMÓRIA

VETOR

0	10
1	20
2	30
3	40
4	50
5	60
6	70
.	...

Percorrendo um vetor

35

```
#include <stdio.h>

int main(void)
{
    //quantidade de posições do vetor
    int tam = 10;

    //declaração do vetor
    int vetor[tam];

    //variável auxiliar para índices
    int i;

    //leitura do teclado para cada índice
    //percorrendo o vetor, variando o índice i de 0 até tam-1
    for(i = 0; i < tam; i++)
    {
        scanf("%d", &vetor[i]);
    }

    //impressão de cada índice
    for(i = 0; i < tam; i++)
    {
        printf("%d ", vetor[i]);
    }

    return 0;
}
```

i	2
---	---

MEMÓRIA

VETOR

0	10
1	20
2	30
3	40
4	50
5	60
6	70
.	...

Percorrendo um vetor

36

```
#include <stdio.h>

int main(void)
{
    //quantidade de posições do vetor
    int tam = 10;

    //declaração do vetor
    int vetor[tam];

    //variável auxiliar para índices
    int i;

    //leitura do teclado para cada índice
    //percorrendo o vetor, variando o índice i de 0 até tam-1
    for(i = 0; i < tam; i++)
    {
        scanf("%d", &vetor[i]);
    }

    //impressão de cada índice
    for(i = 0; i < tam; i++)
    {
        printf("%d ", vetor[i]);
    }

    return 0;
}
```

i	3
---	---

MEMÓRIA

VETOR

0	10
1	20
2	30
3	40
4	50
5	60
6	70
.	...

Percorrendo uma matriz

37

MEMÓRIA					
MATRIZ					
	0	1	2	3	4
0	10	20	30	40	50
1	60	70	80	90	100
2	110	120	130	140	150
3	160	170	180	190	200
4	210	220	230	240	250
5	260	270	280	290	300

Percorrendo uma matriz

38

MEMÓRIA					
MATRIZ					
	0	1	2	3	4
0	10	20	30	40	50
1	60	70	80	90	100
2	110	120	130	140	150
3	160	170	180	190	200
4	210	220	230	240	250
5	260	270	280	290	300

Percorrendo uma matriz

39

```
for(j = 0; j < qtd_colunas; j++)  
{  
    printf("%d", matriz[0][j]);  
}
```

MEMÓRIA					
MATRIZ					
	0	1	2	3	4
0	10	20	30	40	50
1	60	70	80	90	100
2	110	120	130	140	150
3	160	170	180	190	200
4	210	220	230	240	250
5	260	270	280	290	300

Percorrendo uma matriz

40

Índice da linha **fixado em 0**.

```
for(j = 0; j < qtd_colunas; j++)  
{  
    printf("%d", matriz[0][j]);  
}
```

MEMÓRIA					
MATRIZ					
	0	1	2	3	4
0	10	20	30	40	50
1	60	70	80	90	100
2	110	120	130	140	150
3	160	170	180	190	200
4	210	220	230	240	250
5	260	270	280	290	300

Percorrendo uma matriz

41

Tamanho do vetor **0** é a
quantidade de colunas.

```
for(j = 0; j < qtd_colunas; j++)  
{  
    printf("%d", matriz[0][j]);  
}
```

MEMÓRIA					
MATRIZ					
	0	1	2	3	4
0	10	20	30	40	50
1	60	70	80	90	100
2	110	120	130	140	150
3	160	170	180	190	200
4	210	220	230	240	250
5	260	270	280	290	300

Percorrendo uma matriz

42

```
for(j = 0; j < qtd_colunas; j++)  
{  
    printf("%d", matriz[0][j]);  
}
```

MEMÓRIA					
MATRIZ					
	0	1	2	3	4
0	10	20	30	40	50
1	60	70	80	90	100
2	110	120	130	140	150
3	160	170	180	190	200
4	210	220	230	240	250
5	260	270	280	290	300

Percorrendo uma matriz

43

```
for(j = 0; j < qtd_colunas; j++)  
{  
    printf("%d", matriz[1][j]);  
}
```

MEMÓRIA					
MATRIZ					
	0	1	2	3	4
0	10	20	30	40	50
1	60	70	80	90	100
2	110	120	130	140	150
3	160	170	180	190	200
4	210	220	230	240	250
5	260	270	280	290	300

Percorrendo uma matriz

44

```
for(j = 0; j < qtd_colunas; j++)  
{  
    printf("%d", matriz[2][j]);  
}
```

MEMÓRIA					
MATRIZ					
	0	1	2	3	4
0	10	20	30	40	50
1	60	70	80	90	100
2	110	120	130	140	150
3	160	170	180	190	200
4	210	220	230	240	250
5	260	270	280	290	300

Percorrendo uma matriz

45

```
for(j = 0; j < qtd_colunas; j++)  
{  
    printf("%d", matriz[3][j]);  
}
```

MEMÓRIA					
MATRIZ					
	0	1	2	3	4
0	10	20	30	40	50
1	60	70	80	90	100
2	110	120	130	140	150
3	160	170	180	190	200
4	210	220	230	240	250
5	260	270	280	290	300

Percorrendo uma matriz

46

```
for(j = 0; j < qtd_colunas; j++)  
{  
    printf("%d", matriz[4][j]);  
}
```

MEMÓRIA					
MATRIZ					
	0	1	2	3	4
0	10	20	30	40	50
1	60	70	80	90	100
2	110	120	130	140	150
3	160	170	180	190	200
4	210	220	230	240	250
5	260	270	280	290	300

Percorrendo uma matriz

47

```
for(j = 0; j < qtd_colunas; j++)  
{  
    printf("%d", matriz[5][j]);  
}
```

MEMÓRIA					
MATRIZ					
	0	1	2	3	4
0	10	20	30	40	50
1	60	70	80	90	100
2	110	120	130	140	150
3	160	170	180	190	200
4	210	220	230	240	250
5	260	270	280	290	300

Percorrendo uma matriz

48

```
for(i = 0; i < qtd_linhas; i++)  
{  
    for(j = 0; j < qtd_colunas; j++)  
    {  
        printf("%d", matriz[i][j]);  
    }  
}
```

MEMÓRIA					
MATRIZ					
	0	1	2	3	4
0	10	20	30	40	50
1	60	70	80	90	100
2	110	120	130	140	150
3	160	170	180	190	200
4	210	220	230	240	250
5	260	270	280	290	300

Percorrendo uma matriz

49

```
for(i = 0; i < qtd_linhas; i++)  
{  
    for(j = 0; j < qtd_colunas; j++)  
    {  
        printf("%d", matriz[i][j]);  
    }  
}
```

MEMÓRIA					
MATRIZ					
	0	1	2	3	4
0	10	20	30	40	50
1	60	70	80	90	100
2	110	120	130	140	150
3	160	170	180	190	200
4	210	220	230	240	250
5	260	270	280	290	300

Percorrendo uma matriz

50

<i>i</i>	
<i>j</i>	

```
for(i = 0; i < qtd_linhas; i++)
{
    for(j = 0; j < qtd_colunas; j++)
    {
        printf("%d", matriz[i][j]);
    }
}
```

MEMÓRIA					
MATRIZ					
	0	1	2	3	4
0	10	20	30	40	50
1	60	70	80	90	100
2	110	120	130	140	150
3	160	170	180	190	200
4	210	220	230	240	250
5	260	270	280	290	300

Percorrendo uma matriz

51

<i>i</i>	0
<i>j</i>	

```
for(i = 0; i < qtd_linhas; i++)  
{  
    for(j = 0; j < qtd_colunas; j++)  
    {  
        printf("%d", matriz[i][j]);  
    }  
}
```

MEMÓRIA					
MATRIZ					
	0	1	2	3	4
0	10	20	30	40	50
1	60	70	80	90	100
2	110	120	130	140	150
3	160	170	180	190	200
4	210	220	230	240	250
5	260	270	280	290	300

Percorrendo uma matriz

52

<i>i</i>	0
<i>j</i>	0

```
for(i = 0; i < qtd_linhas; i++)  
{  
    for(j = 0; j < qtd_colunas; j++)  
    {  
        printf("%d", matriz[i][j]);  
    }  
}
```

MEMÓRIA					
MATRIZ					
	0	1	2	3	4
0	10	20	30	40	50
1	60	70	80	90	100
2	110	120	130	140	150
3	160	170	180	190	200
4	210	220	230	240	250
5	260	270	280	290	300

Percorrendo uma matriz

53

<i>i</i>	0
<i>j</i>	1

```
for(i = 0; i < qtd_linhas; i++)  
{  
    for(j = 0; j < qtd_colunas; j++)  
    {  
        printf("%d", matriz[i][j]);  
    }  
}
```

MEMÓRIA					
MATRIZ					
	0	1	2	3	4
0	10	20	30	40	50
1	60	70	80	90	100
2	110	120	130	140	150
3	160	170	180	190	200
4	210	220	230	240	250
5	260	270	280	290	300

Percorrendo uma matriz

54

<i>i</i>	0
<i>j</i>	2

```
for(i = 0; i < qtd_linhas; i++)  
{  
    for(j = 0; j < qtd_colunas; j++)  
    {  
        printf("%d", matriz[i][j]);  
    }  
}
```

MEMÓRIA					
MATRIZ					
	0	1	2	3	4
0	10	20	30	40	50
1	60	70	80	90	100
2	110	120	130	140	150
3	160	170	180	190	200
4	210	220	230	240	250
5	260	270	280	290	300

Percorrendo uma matriz

55

<i>i</i>	0
<i>j</i>	3

```
for(i = 0; i < qtd_linhas; i++)  
{  
    for(j = 0; j < qtd_colunas; j++)  
    {  
        printf("%d", matriz[i][j]);  
    }  
}
```

MEMÓRIA					
MATRIZ					
	0	1	2	3	4
0	10	20	30	40	50
1	60	70	80	90	100
2	110	120	130	140	150
3	160	170	180	190	200
4	210	220	230	240	250
5	260	270	280	290	300

Percorrendo uma matriz

56

<i>i</i>	0
<i>j</i>	4

```
for(i = 0; i < qtd_linhas; i++)  
{  
    for(j = 0; j < qtd_colunas; j++)  
    {  
        printf("%d", matriz[i][j]);  
    }  
}
```

MEMÓRIA					
MATRIZ					
	0	1	2	3	4
0	10	20	30	40	50
1	60	70	80	90	100
2	110	120	130	140	150
3	160	170	180	190	200
4	210	220	230	240	250
5	260	270	280	290	300

Percorrendo uma matriz

57

<i>i</i>	1
<i>j</i>	

```
for(i = 0; i < qtd_linhas; i++)  
{  
    for(j = 0; j < qtd_colunas; j++)  
    {  
        printf("%d", matriz[i][j]);  
    }  
}
```

MEMÓRIA					
MATRIZ					
	0	1	2	3	4
0	10	20	30	40	50
1	60	70	80	90	100
2	110	120	130	140	150
3	160	170	180	190	200
4	210	220	230	240	250
5	260	270	280	290	300

Percorrendo uma matriz

58

<i>i</i>	1
<i>j</i>	0

```
for(i = 0; i < qtd_linhas; i++)  
{  
    for(j = 0; j < qtd_colunas; j++)  
    {  
        printf("%d", matriz[i][j]);  
    }  
}
```

MEMÓRIA					
MATRIZ					
	0	1	2	3	4
0	10	20	30	40	50
1	60	70	80	90	100
2	110	120	130	140	150
3	160	170	180	190	200
4	210	220	230	240	250
5	260	270	280	290	300

Percorrendo uma matriz

59

<i>i</i>	1
<i>j</i>	1

```
for(i = 0; i < qtd_linhas; i++)  
{  
    for(j = 0; j < qtd_colunas; j++)  
    {  
        printf("%d", matriz[i][j]);  
    }  
}
```

MEMÓRIA					
MATRIZ					
	0	1	2	3	4
0	10	20	30	40	50
1	60	70	80	90	100
2	110	120	130	140	150
3	160	170	180	190	200
4	210	220	230	240	250
5	260	270	280	290	300

Percorrendo uma matriz

60

<i>i</i>	1
<i>j</i>	2

```
for(i = 0; i < qtd_linhas; i++)  
{  
    for(j = 0; j < qtd_colunas; j++)  
    {  
        printf("%d", matriz[i][j]);  
    }  
}
```

MEMÓRIA					
MATRIZ					
	0	1	2	3	4
0	10	20	30	40	50
1	60	70	80	90	100
2	110	120	130	140	150
3	160	170	180	190	200
4	210	220	230	240	250
5	260	270	280	290	300

Percorrendo uma matriz

61

<i>i</i>	1
<i>j</i>	3

```
for(i = 0; i < qtd_linhas; i++)  
{  
    for(j = 0; j < qtd_colunas; j++)  
    {  
        printf("%d", matriz[i][j]);  
    }  
}
```

MEMÓRIA					
MATRIZ					
	0	1	2	3	4
0	10	20	30	40	50
1	60	70	80	90	100
2	110	120	130	140	150
3	160	170	180	190	200
4	210	220	230	240	250
5	260	270	280	290	300

Percorrendo uma matriz

62

<i>i</i>	1
<i>j</i>	4

```
for(i = 0; i < qtd_linhas; i++)  
{  
    for(j = 0; j < qtd_colunas; j++)  
    {  
        printf("%d", matriz[i][j]);  
    }  
}
```

MEMÓRIA					
MATRIZ					
	0	1	2	3	4
0	10	20	30	40	50
1	60	70	80	90	100
2	110	120	130	140	150
3	160	170	180	190	200
4	210	220	230	240	250
5	260	270	280	290	300

Percorrendo uma matriz

63

<i>i</i>	2
<i>j</i>	

```
for(i = 0; i < qtd_linhas; i++)  
{  
    for(j = 0; j < qtd_colunas; j++)  
    {  
        printf("%d", matriz[i][j]);  
    }  
}
```

MEMÓRIA					
MATRIZ					
	0	1	2	3	4
0	10	20	30	40	50
1	60	70	80	90	100
2	110	120	130	140	150
3	160	170	180	190	200
4	210	220	230	240	250
5	260	270	280	290	300

Percorrendo uma matriz

64

<i>i</i>	2
<i>j</i>	0

```
for(i = 0; i < qtd_linhas; i++)  
{  
    for(j = 0; j < qtd_colunas; j++)  
    {  
        printf("%d", matriz[i][j]);  
    }  
}
```

MEMÓRIA					
MATRIZ					
	0	1	2	3	4
0	10	20	30	40	50
1	60	70	80	90	100
2	110	120	130	140	150
3	160	170	180	190	200
4	210	220	230	240	250
5	260	270	280	290	300

Percorrendo uma matriz

65

<i>i</i>	2
<i>j</i>	1

```
for(i = 0; i < qtd_linhas; i++)  
{  
    for(j = 0; j < qtd_colunas; j++)  
    {  
        printf("%d", matriz[i][j]);  
    }  
}
```

MEMÓRIA					
MATRIZ					
	0	1	2	3	4
0	10	20	30	40	50
1	60	70	80	90	100
2	110	120	130	140	150
3	160	170	180	190	200
4	210	220	230	240	250
5	260	270	280	290	300

Percorrendo uma matriz

66

<i>i</i>	2
<i>j</i>	2

```
for(i = 0; i < qtd_linhas; i++)  
{  
    for(j = 0; j < qtd_colunas; j++)  
    {  
        printf("%d", matriz[i][j]);  
    }  
}
```

MEMÓRIA					
MATRIZ					
	0	1	2	3	4
0	10	20	30	40	50
1	60	70	80	90	100
2	110	120	130	140	150
3	160	170	180	190	200
4	210	220	230	240	250
5	260	270	280	290	300

Percorrendo uma matriz

67

<i>i</i>	2
<i>j</i>	3

```
for(i = 0; i < qtd_linhas; i++)  
{  
    for(j = 0; j < qtd_colunas; j++)  
    {  
        printf("%d", matriz[i][j]);  
    }  
}
```

MEMÓRIA					
MATRIZ					
	0	1	2	3	4
0	10	20	30	40	50
1	60	70	80	90	100
2	110	120	130	140	150
3	160	170	180	190	200
4	210	220	230	240	250
5	260	270	280	290	300

Percorrendo uma matriz

68

<i>i</i>	2
<i>j</i>	4

```
for(i = 0; i < qtd_linhas; i++)  
{  
    for(j = 0; j < qtd_colunas; j++)  
    {  
        printf("%d", matriz[i][j]);  
    }  
}
```

MEMÓRIA					
MATRIZ					
	0	1	2	3	4
0	10	20	30	40	50
1	60	70	80	90	100
2	110	120	130	140	150
3	160	170	180	190	200
4	210	220	230	240	250
5	260	270	280	290	300

Percorrendo uma matriz

69

<i>i</i>	3
<i>j</i>	

```
for(i = 0; i < qtd_linhas; i++)  
{  
    for(j = 0; j < qtd_colunas; j++)  
    {  
        printf("%d", matriz[i][j]);  
    }  
}
```

MEMÓRIA					
MATRIZ					
	0	1	2	3	4
0	10	20	30	40	50
1	60	70	80	90	100
2	110	120	130	140	150
3	160	170	180	190	200
4	210	220	230	240	250
5	260	270	280	290	300

Percorrendo uma matriz

70

<i>i</i>	3
<i>j</i>	0

```
for(i = 0; i < qtd_linhas; i++)  
{  
    for(j = 0; j < qtd_colunas; j++)  
    {  
        printf("%d", matriz[i][j]);  
    }  
}
```

MEMÓRIA					
MATRIZ					
	0	1	2	3	4
0	10	20	30	40	50
1	60	70	80	90	100
2	110	120	130	140	150
3	160	170	180	190	200
4	210	220	230	240	250
5	260	270	280	290	300

Percorrendo uma matriz

71

<i>i</i>	3
<i>j</i>	1

```
for(i = 0; i < qtd_linhas; i++)  
{  
    for(j = 0; j < qtd_colunas; j++)  
    {  
        printf("%d", matriz[i][j]);  
    }  
}
```

MEMÓRIA					
MATRIZ					
	0	1	2	3	4
0	10	20	30	40	50
1	60	70	80	90	100
2	110	120	130	140	150
3	160	170	180	190	200
4	210	220	230	240	250
5	260	270	280	290	300

Percorrendo uma matriz

72

<i>i</i>	3
<i>j</i>	2

```
for(i = 0; i < qtd_linhas; i++)  
{  
    for(j = 0; j < qtd_colunas; j++)  
    {  
        printf("%d", matriz[i][j]);  
    }  
}
```

MEMÓRIA					
MATRIZ					
	0	1	2	3	4
0	10	20	30	40	50
1	60	70	80	90	100
2	110	120	130	140	150
3	160	170	180	190	200
4	210	220	230	240	250
5	260	270	280	290	300

Percorrendo uma matriz

73

<i>i</i>	3
<i>j</i>	3

```
for(i = 0; i < qtd_linhas; i++)  
{  
    for(j = 0; j < qtd_colunas; j++)  
    {  
        printf("%d", matriz[i][j]);  
    }  
}
```

MEMÓRIA					
MATRIZ					
	0	1	2	3	4
0	10	20	30	40	50
1	60	70	80	90	100
2	110	120	130	140	150
3	160	170	180	190	200
4	210	220	230	240	250
5	260	270	280	290	300

Percorrendo uma matriz

74

<i>i</i>	3
<i>j</i>	4

```
for(i = 0; i < qtd_linhas; i++)  
{  
    for(j = 0; j < qtd_colunas; j++)  
    {  
        printf("%d", matriz[i][j]);  
    }  
}
```

MEMÓRIA					
MATRIZ					
	0	1	2	3	4
0	10	20	30	40	50
1	60	70	80	90	100
2	110	120	130	140	150
3	160	170	180	190	200
4	210	220	230	240	250
5	260	270	280	290	300

Percorrendo uma matriz

75

<i>i</i>	...
<i>j</i>	...

```
for(i = 0; i < qtd_linhas; i++)  
{  
    for(j = 0; j < qtd_colunas; j++)  
    {  
        printf("%d", matriz[i][j]);  
    }  
}
```

MEMÓRIA					
MATRIZ					
	0	1	2	3	4
0	10	20	30	40	50
1	60	70	80	90	100
2	110	120	130	140	150
3	160	170	180	190	200
4	210	220	230	240	250
5	260	270	280	290	300

Percorrendo uma matriz

76

<i>i</i>	5
<i>j</i>	4

```
for(i = 0; i < qtd_linhas; i++)  
{  
    for(j = 0; j < qtd_colunas; j++)  
    {  
        printf("%d", matriz[i][j]);  
    }  
}
```

MEMÓRIA					
MATRIZ					
	0	1	2	3	4
0	10	20	30	40	50
1	60	70	80	90	100
2	110	120	130	140	150
3	160	170	180	190	200
4	210	220	230	240	250
5	260	270	280	290	300

Percorrendo uma matriz

77

<i>i</i>	6
<i>j</i>	

```
for(i = 0; i < qtd_linhas; i++)  
{  
    for(j = 0; j < qtd_colunas; j++)  
    {  
        printf("%d", matriz[i][j]);  
    }  
}
```

Executou para todas as
linhas da matriz.

MEMÓRIA					
MATRIZ					
	0	1	2	3	4
0	10	20	30	40	50
1	60	70	80	90	100
2	110	120	130	140	150
3	160	170	180	190	200
4	210	220	230	240	250
5	260	270	280	290	300

Percorrendo uma matriz

78

```
int main(void)
{
    //quantidade de linhas e de colunas
    int qtd_linhas = 6;
    int qtd_colunas = 5;
    //declaração da matriz
    int matriz[qtd_linhas][qtd_colunas];
    //variáveis auxiliares para índices
    int i, j;

    //leitura do teclado para cada índice
    //percorrendo a matriz, varrendo uma linha por vez
    for(i = 0; i < qtd_linhas; i++)
    {
        for(j = 0; j < qtd_colunas; j++)
        {
            scanf("%d", &matriz[i][j]);
        }
    }
    //impressão de cada índice
    for(i = 0; i < qtd_linhas; i++)
    {
        for(j = 0; j < qtd_colunas; j++)
        {
            printf("%d", matriz[i][j]);
        }
    }
    return 0;
}
```

Exercício

79

- Escreva um código em C que:
 - ▣ Leia dois valores informados pelo usuário;
 - ▣ Declare uma matriz float com o tamanho informado;
 - ▣ Preencha a matriz com valores lidos do teclado;
 - ▣ Imprima os valores da matriz, quebrando linha (`\n`) cada vez que trocar o índice de linha da matriz.

Exercício

80

- Escreva
- Leia
- Decl
- Pree
- Imprim
- cada

```
int main(void)
{
    int qtd_linhas;
    int qtd_colunas;

    printf("Informe a quantidade de LINHAS e de COLUNAS:");
    scanf("%d %d", &qtd_linhas, &qtd_colunas);

    int matriz[qtd_linhas][qtd_colunas];
    int i, j;

    for(i = 0; i < qtd_linhas; i++)
    {
        for(j = 0; j < qtd_colunas; j++)
        {
            scanf("%d", &matriz[i][j]);
        }
    }
    printf("\n=====\\n\\n");
    for(i = 0; i < qtd_linhas; i++)
    {
        for(j = 0; j < qtd_colunas; j++)
        {
            printf("%d ", matriz[i][j]);
        }
        printf("\\n");
    }
    return 0;
}
```

armado;
do;
na (\\n)
triz.

Exercício

81

- Escreva o trecho de código que imprime em tela apenas os valores da diagonal principal da matriz.

MATRIZ					
	0	1	2	3	4
0	10	20	30	40	50
1	60	70	80	90	100
2	110	120	130	140	150
3	160	170	180	190	200
4	210	220	230	240	250