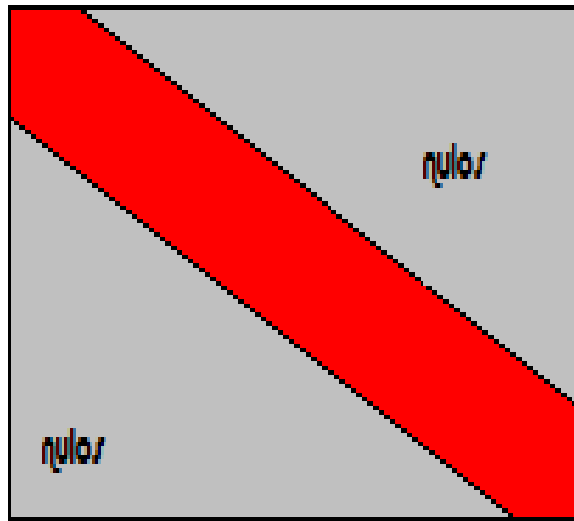


## Matrizes Especiais

São aquelas em que seus elementos se concentram de forma especial, de modo a ser possível lançar mão de esquemas alternativos de armazenamento visando a otimização em termos de espaço ou de velocidade de acesso.

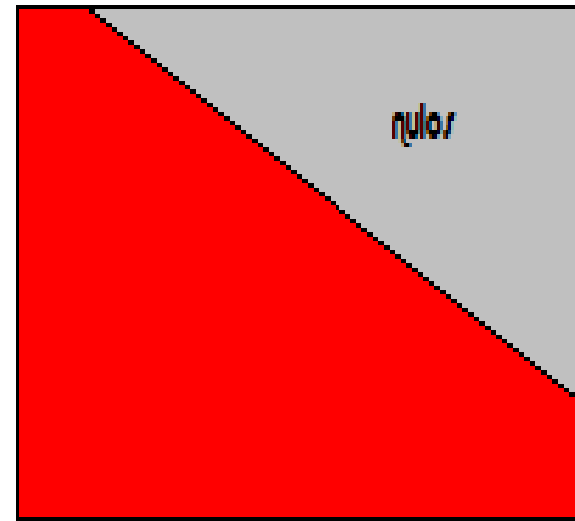
Exemplos:

Diagonal



Se  $i \neq j$  então  $M_{ij} = \text{nulo}$

Triangular Inferior



Se  $i < j$  então  $M_{ij} = \text{nulo}$

# Matrizes Especiais

Simétrica

X	A	B	C	D	E
A	X	L	F	G	H
B	L	X	M	I	J
C	F	M	X	N	K
D	G	I	N	X	P
E	H	J	K	P	X

$$M_{ij} = M_{ji}$$

Anti-Simétrica

1	-5	-6	-7
5	2	-8	-15
6	8	3	8
7	15	-8	4

$$\text{Se } i \neq j \text{ então } M_{ij} = -M_{ji}$$

Na medida em que pode-se inferir um valor em função de uma propriedade geral da matriz, este valor não necessita ser armazenado.

## Matrizes Especiais

Com base no que foi apresentado, defina um TAD para representar uma matriz diagonal de inteiros, o qual contempla as operações de criação, inicialização, impressão e consulta de um determinado elemento da matriz. Implemente o TAD em questão na linguagem C.

```
typedef struct  
{  
    unsigned int ordem;  
    int *elementos;  
}MAT_DIAG;  
void criar_matriz (MAT_DIAG *, int);  
void inicializar_matriz (MAT_DIAG *);  
void imprimir_matriz (MAT_DIAG *);  
int retornar_elemento_matriz (MAT_DIAG *, int,  
    int);
```

```
void criar_matriz (MAT_DIAG *m, int d)
{
    m->ordem = d;
    m->elementos = (int *) malloc (sizeof(int) *d);
    if (!m->elementos)
    {
        printf("Nao foi possivel reservar memoria para
a matriz!\n");
        exit(1);
    }
}
```

```
void inicializar_matriz (MAT_DIAG *m)
{
    int i;
    printf ("\nEntre com os elementos da diagonal
principal da matriz:\n");
    for (i=0;i < m->ordem;i++)
    {
        printf ("matriz[%d][%d]: ",i+1,i+1);
        scanf ("%d", m->elementos+i);
    }
}
```

```
void imprimir_matriz (MAT_DIAG *m)
{
    int i, j;
    for (i=0;i<m->ordem;i++)
    {
        printf("| ");
        for (j=0;j<m->ordem;j++)
            printf("%05d ", retornar_elemento_matriz (m,
i, j));
        printf("\n");
    }
}
```

```
int retornar_elemento_matriz (MAT_DIAG *m, int i,  
    int j)  
{  
    if (i==j)  
        return (m->elementos[i]);  
    else  
        return (0);  
}
```



## Matrizes Especiais

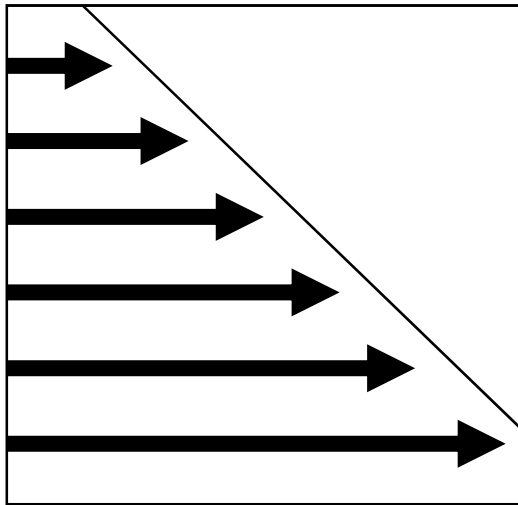
Outra matriz especial é a triangular.

A matriz triangular concentra seus elementos significativos ou da diagonal para baixo, caso em que  $i < j$  implica em um valor nulo (triangular inferior); ou da diagonal principal para cima, caso em que  $i > j$  implica em um valor nulo (triangular superior).

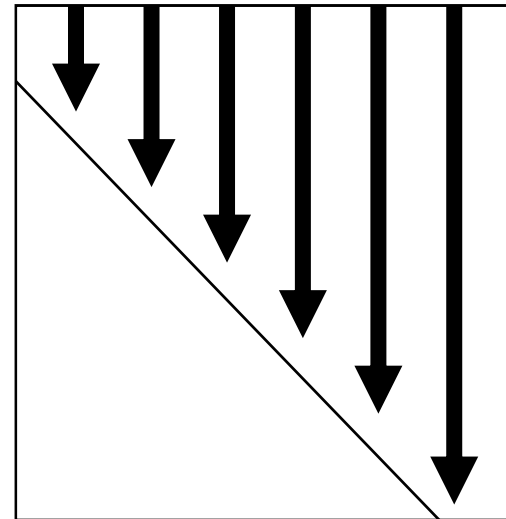
Pode-se também armazená-la em um vetor, mapeando seus elementos não nulos conforme os esquemas esboçados nas figuras a seguir:

# Matrizes Especiais

## Esquemas de armazenamento



triangular inferior



triangular superior

Isto é, os elementos não nulos são armazenados em sequência desde a borda até a diagonal, linha a linha num caso e coluna a coluna noutro.

## Matrizes Especiais

Com base no que foi apresentado, em se tratando de uma matriz triangular inferior de ordem  $n$ , qual o número máximo de elementos diferentes de zero nessa matriz?

$$\sum_{i=1}^n i = (1 + n) \cdot \frac{n}{2}$$

ou

$$\sum_{i=1}^n i = n + \sum_{i=1}^{n-1} i = n + \frac{n \cdot (n - 1)}{2}$$

## Matrizes Especiais

Logo, os elementos podem ser localizados no vetor pelas seguintes funções de mapeamento:

- triangular inferior:  $fm(i, j) = j + i * (i - 1) / 2$

- triangular superior:  $fm(i, j) = i + j * (j - 1) / 2$

## Matrizes Especiais

Com base no que foi apresentado, defina um TAD para representar uma matriz triangular superior de inteiros, o qual contempla as operações de criação, inicialização, impressão e consulta de um determinado elemento da matriz. Implemente o TAD em questão na linguagem C.