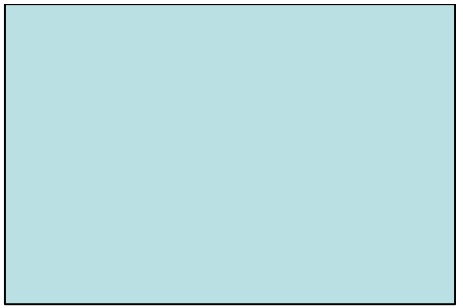
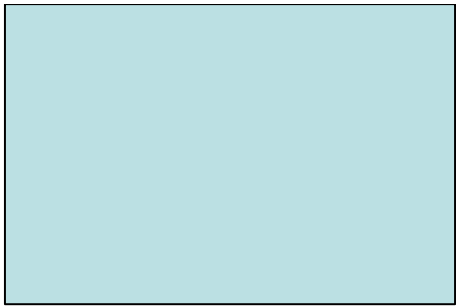
Vetores – Exercícios



Implemente as operações definidas anteriormente para o TAD MATRIZ considerando a nova estrutura apresentada.

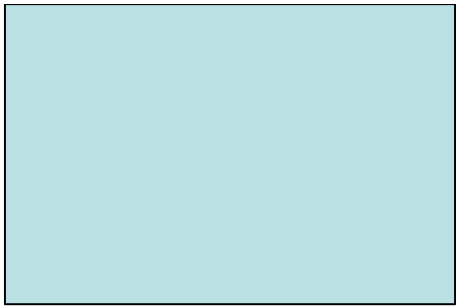
UNIVASF



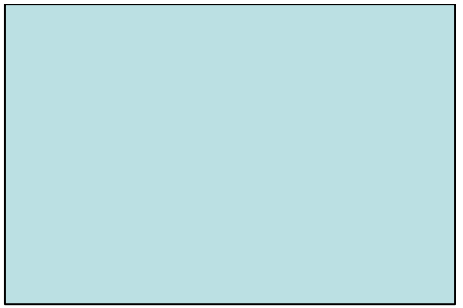
```
typedef struct {
   int nl;
   int nc;
   int *elementos;
}MATRIZ;
void criar_matriz (int, int, MATRIZ *);
void inicializar_matriz (MATRIZ *);
void imprimir_matriz (MATRIZ *);
void somar_matrizes (MATRIZ *, MATRIZ *, MATRIZ *);
void subtrair_matrizes (MATRIZ *, MATRIZ *, MATRIZ *);
void multiplicar_matrizes (MATRIZ *, MATRIZ *, MATRIZ *);
```

UNIVASF

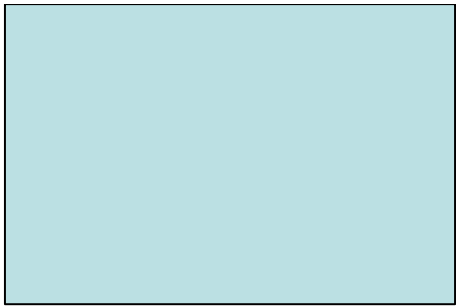
```
void criar_matriz (int nl, int nc, MATRIZ *m) {
  int i;
  m->elementos = (int *) malloc (sizeof(int)*nl*nc);
  m->elementos = (int *) calloc (sizeof(int),nl*nc);
  if (!m->elementos)  {
     printf("Nao foi possivel reservar memoria para a matriz!\
   n");
     exit(1);
  }
  m->nl = nl;
  m->nc = nc;
  for (i=0; i<nl*nc; i++)
     m->elementos[i] = 0;
}
```



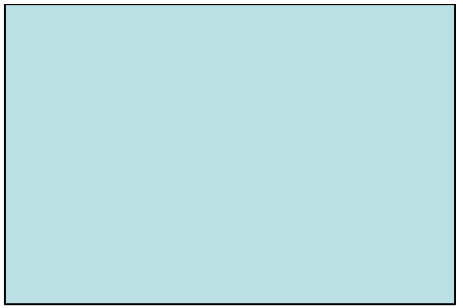
```
void criar_matriz (int nl, int nc, MATRIZ *m)
{
   m->elementos = (int *) malloc (sizeof(int)*nl*nc);
   if (!m->elementos)
   {
      printf("Nao foi possivel reservar memoria para a
    matriz!\n");
      exit(1);
   }
   m->nl = nl;
   m->nc = nc;
   memset (m->elementos, 0, nl*nc);
}
```



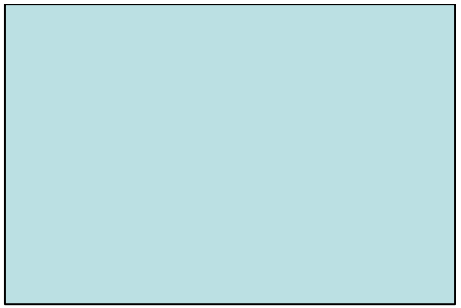
```
void inicializar_matriz (MATRIZ *m)
{
   int i;
   for (i=0; i<m->nl*m->nc; i++)
   {
        printf ("\nEntre com matriz[%d][%d]=",
            (i/(m->nc))+1, i%m->nc+1);
            scanf ("%d", m->elementos+i);
   }
}
```



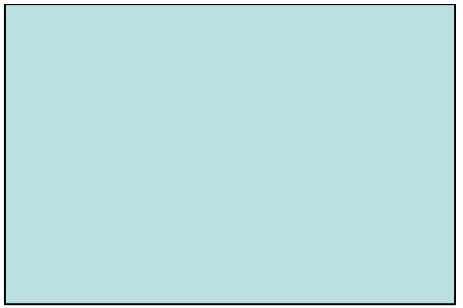
```
void imprimir_matriz (MATRIZ *m)
{
   int i, j;
   for (i=0;i<m->nl;i++)
   {
      printf("\n|");
      for (j=0;j<m->nc;j++)
            printf ("%5d", m->elementos[i*m->nc+j]);
      printf(" |");
   }
}
```



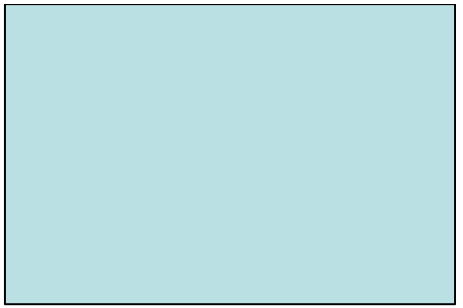
```
void somar_matrizes (MATRIZ *m1, MATRIZ *m2,
MATRIZ *m3) {
   if (m1->nl==m2->nl && m1->nc==m2->nc) {
      int i;
      criar_matriz (m1->nl, m1->nc, m3);
      for (i=0; i<m3->nl*m3->nc; i++)
         m3->elementos[i] = m1->elementos[i] +
m2->elementos[i];
   }
   else {
      printf ("A soma nao eh possivel!\n");
   }
}
```



```
void subtrair_matrizes (MATRIZ *m1,
MATRIZ *m2, MATRIZ *m3) {
   if (m1->nl==m2->nl && m1->nc==m2->nc) {
      int i;
      criar_matriz (m1->nl, m1->nc, m3);
      for (i=0; i<m3->nl*m3->nc; i++)
         m3->elementos[i] = m1->elementos[i] -
m2->elementos[i];
   }
   else {
      printf ("A subtracao nao eh possivel!\n");
   }
}
```



```
void multiplicar_matrizes (MATRIZ *m1, MATRIZ *m2,
MATRIZ *m3) {
  if (m1->nc==m2->nl) {
    int i, j, z;
    criar_matriz (m1->nl, m2->nc, m3);
    for (i=0; i<m3->nl; i++)
      for (j=0; j<m3->nc; j++) {
        m3->elementos[i*m3->nc+j] = 0;
        for (z=0; z<m1->nc; z++)
          m3->elementos[i*m3->nc+j] += m1->elementos[i*m1->nc+z]  * m2-
   >elementos[z*m2->nc+j];
        }
      }
  }
  else {
    printf ("A multiplicacao nao eh possivel!\n");
  }}
```

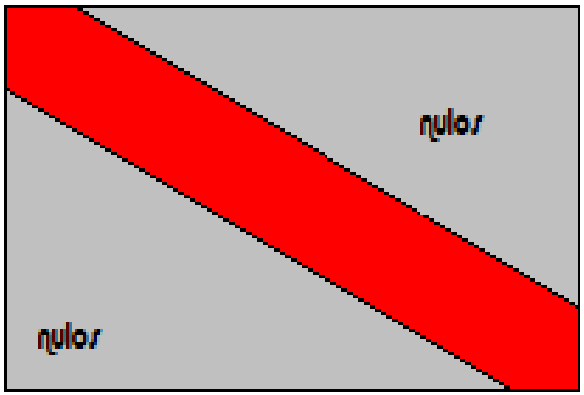


Aplicação da abstração de dados sobre matrizes especiais

Matrizes Especiais

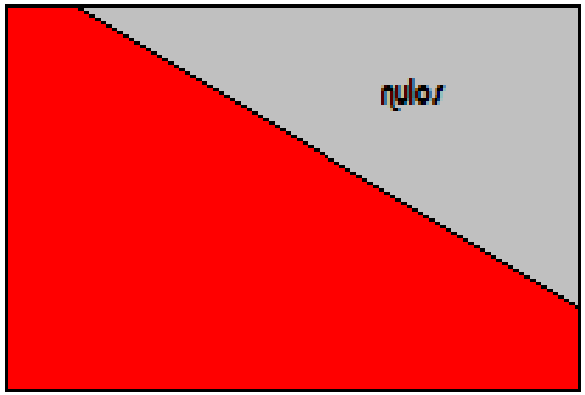
São aquelas em que seus elementos se concentram de forma peculiar, de modo a ser possível lançar mão de esquemas alternativos de armazenamento visando a otimização em termos de espaço. **Exemplos:**

Diagonal



Se i != j então M_{ij} = nulo

Triangular Inferior



Se i < j então M_{ij} = 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

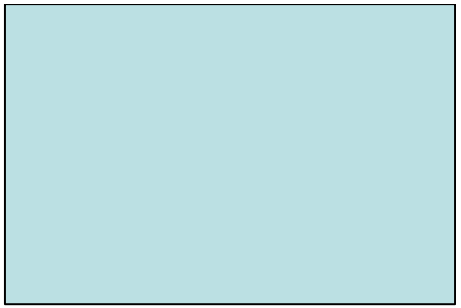
0	-5	-6	-7
5	0	-8	-15
6	8	0	8
7	15	-8	0

Se i != j 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.

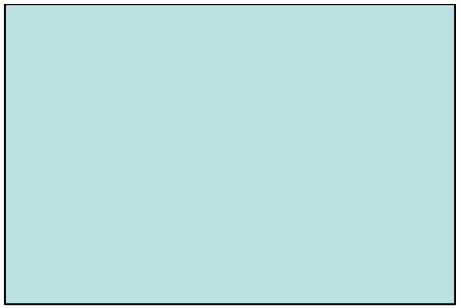
Matrizes Especiais

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.



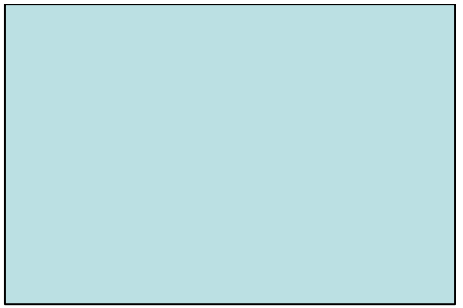
```
typedef struct
{
   unsigned int ordem;
   int *elementos;
}MAT_DIAG;
void criar_matriz (MAT_DIAG *, int);
void inicializar_matriz (MAT_DIAG *);
void imprimir_matriz (MAT_DIAG *);
int retornar_elemento_matriz (MAT_DIAG *, int, int);
```

```
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);
   }
}
```

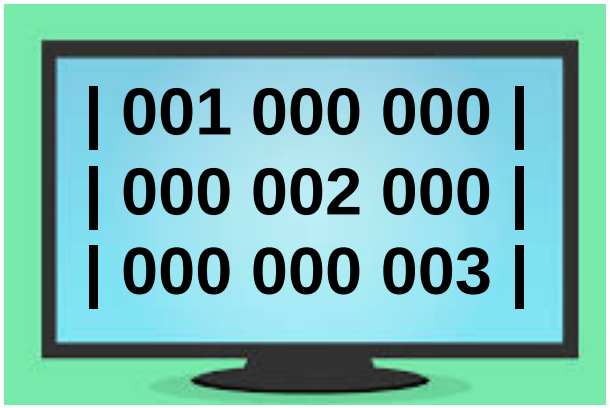


```
| 001 000 000 |
| 000 002 000 |
| 000 000 003 |
```

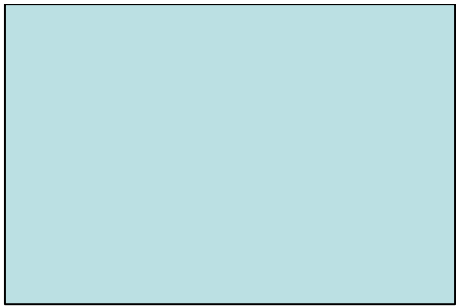
```
void inicializar_matriz (MAT_DIAG *m) {
   int i;
   printf ("\nEntre com os elementos da
   diagonal principal da matriz:\n");
   for (i=0;i < m->ordem;i++)
   {
      printf ("matriz[%d][%d]: ",i+1,i+1);
      scanf ("%d", m->elementos+i);
   }
}
```



```
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("%03d ", 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

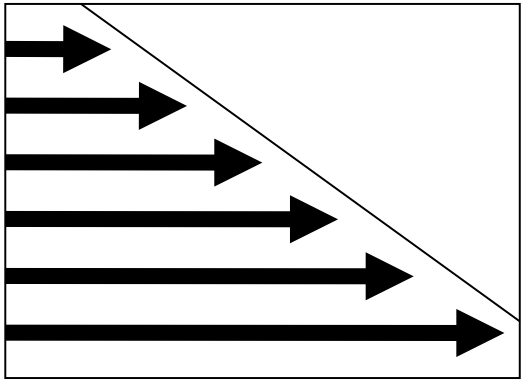
Outra matriz especial é a triangular.

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).

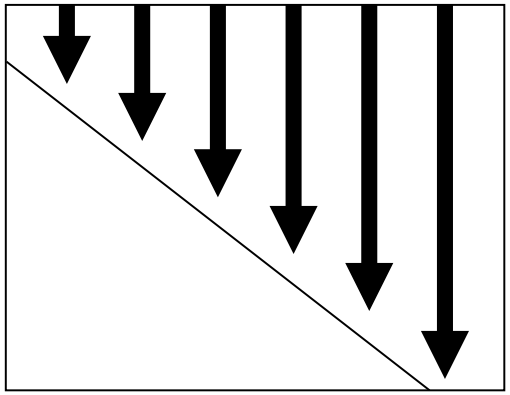
Pode-se também armazená-la em um vetor, mapeando seus elementos não nulos conforme os esquemas esboçados nas figuras a seguir:

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?

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

| 001 000 000 |
| 002 003 000 |
| 004 005 006 |

ou

$$\sum_{i=1}^{n} i = \quad = \quad =$$

Matrizes Especiais

Como os elementos podem ser localiza-dos no vetor?

Vetor = {1, 2, 3, 4, 5, 6}

Matriz =

001	**000**	**000**
002	**003**	**000**
004	**005**	**006**

Matriz[3,2] = Vetor[5] = 5

abstraindo índices iniciando em zero

fm (3, 2) = $\sum_{i=1}^{n} i$ + 3 + $\sum_{i=1}^{n-1} i$ + 5 = $\frac{n \cdot (n-1)}{2}$

através das 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, inicilização, impressão e consulta de um determinado elemento da matriz. Implemente o TAD em questão na linguagem C.