

```
typedef struct  
{  
    unsigned int ordem;  
    int *elementos;  
}MT;  
void criar_matriz (MT *, int);  
void inicializar_matriz (MT *);  
void imprimir_matriz (MT *);  
int retorna_elemento_matriz (MT *, int, int);  
int fm (int, int);
```

```

void criar_matriz (MT *m, int d)
{
    m->elementos = (int *) malloc (sizeof(int)*
(d+1)*d/2);
    if (m->elementos)
        m->ordem = d;
    else
    {
        printf("\n\nImpossivel armazenar tal
matriz com a memoria disponivel!\n\n");
        exit(1);
    }
}

```

```

void inicializar_matriz (MT *m)
{
    int i,j;
    printf ("\nEntre com os elementos da matriz
triangular superior:\n");
    for (i=0;i < m->ordem;i++)
        for (j=0;j < m->ordem;j++)
            if (j>=i)
                {
                    printf ("m[%d][%d]: ",i+1,j+1);
                    scanf ("%d", m->elementos+fm(i, j));
                }
    }

```

```

}

```

```
void imprimir_matriz (MT *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 (MT *m, int i, int j)  
{  
    if (i<=j)  
        return (m->elementos[fm(i, j)]);  
    else  
        return (0);  
}
```

```
int fm (int i, int j)  
{  
    return (i + (j+1)*j/2);  
}
```

Matrizes Especiais

Cabe também destacar as matrizes simétricas e anti-simétrica.

As matrizes simétrica e anti-simétrica são aquelas em que $M_{ij} == M_{ji}$ e $M_{ij} == -M_{ji}$ (para $i \neq j$), respectivamente. Como pode-se observar, com base no que foi apresentado, estas matrizes podem ser tratadas como casos particulares de matrizes triangulares: conhecendo-se os valores acima ou abaixo da diagonal, pode-se inferir os demais.

Matrizes Especiais

Com base no que foi apresentado, defina um TAD para representar uma matriz simétrica de caracteres, 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;  
    char *elementos;  
}MAT;  
void criar_matriz (MAT *, int);  
void inicializar_matriz (MAT *);  
void imprimir_matriz (MAT *);  
char retornar_elemento_matriz (MAT *, int,  
    int);  
int fm (int, int);
```

```
void criar_matriz (MAT *m, int d)  
{  
    m->elementos = (char *) malloc  
(sizeof(char)* (d+1)*d/2);  
    if (m->elementos)  
        m->ordem = d;  
    else  
    {  
        printf("\n\nImpossivel armazenar tal  
matriz com a memoria disponivel!\n\n");  
        exit(1);  
    }  
}
```

```

void inicializar_matriz (MAT *m) {
    int i,j;
    printf ("\nEntre com os elementos da matriz
    simetrica de caracteres:\n");
    for (i=0;i < m->ordem;i++)
        for (j=0;j < m->ordem;j++)
            if (j>=i)
                {
                    printf ("m[%d][%d]: ",i+1,j+1);
                    setbuf (stdin, NULL); /*para limpar o buffer*/
                    /* fflush (stdin); ou __fpurge (stdin); */
                    scanf ("%c", m->elementos+fm(i, j));
                }
}

```

```

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

```

```
char retornar_elemento_matriz (MAT *m, int  
    i, int j)  
{  
    if (i<=j)  
        return (m->elementos[fm(i, j)]);  
    else  
        return (m->elementos[fm(j, i)]);  
}
```

```
int fm (int i, int j)  
{  
    return (i + (j+1)*j/2);  
}
```

Matrizes Especiais

Com base no que foi apresentado, defina um TAD para representar uma matriz anti-simétrica de floats, 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.

Matrizes Tridiagonal

Matriz tridiagonal, é uma denominação do Algoritmo de Thomas. O qual é um método algébrico que consiste numa simplificação da Eliminação Gaussiana para resolução de sistemas de equações tridiagonais.

Uma matriz tridiagonal é uma matriz quadrada onde apenas os elementos da diagonal principal e das que estão acima e abaixo da mesma são não nulos.

Logo, torna-se um desperdício computacional armazenar os zeros.

Matrizes Esparsas

Em uma matriz esparsa, a maioria dos elementos é igual a zero, sendo, aproximadamente, apenas 30% dos seus valores significativos. Para esse tipo de estrutura também armazenam-se apenas os valores significativos.

Exemplo:

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 6 & 0 \\ 0 & -3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 4 & 0 \\ 5 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Com base no que foi apresentado, proponha uma estrutura de dados para armazenar os valores significativos de uma matriz esparsa.

Uma possibilidade seria a denominada *matriz associada de índices e valores*, consiste em se armazenar apenas os não nulos junto a seus índices, num esquema como o que segue, referente a uma matriz bidimensional $p \times q$, mas que pode ser facilmente generalizado.

N_Ele	<u>n</u>			L1	<u>n_l</u>		L2	<u>n_c</u>		max
IND	i1	i2	i3	...	in					
	j1	j2	j3	...	jn					
VAL	v1	v2	v3	...	vn					

Matrizes Esparsas

Os índices são guardados paralelamente aos valores. Há espaço na matriz de índices IND e no vetor de valores VAL para max elementos, sendo max uma percentagem de $p \cdot q$. Até um certo momento podem ter sido usados apenas n desses espaços. Por isso a presença de uma variável N_Ele, que informa este número. Para validação das operações de consulta e atribuição, são utilizados dos inteiros L1 e L2, que representam, respectivamente, os limites de linhas e colunas.

Matrizes Esparsas

Para uma melhor compreensão, analisaremos a representação da matriz esparsa do slide 78 através da estrutura proposta.

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 6 & 0 \\ 0 & -3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 4 & 0 \\ 5 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Desta forma teríamos:

N_Ele	5			L1	5		L2	6		
IND	0	1	2	3	3					
	4	1	4	0	2					
VAL	6	-3	4	5	1					

Matrizes Esparsas

Com base na estrutura de dados proposta, implemente, na linguagem C, o TAD matriz esparsa, o qual contempla as operações de criação de uma matriz, atribuição e consulta de um determinado elemento da matriz.