Ordenação pelo Método da Bolha Bubblesort

- Um dos algoritmos mais simples que existem
- Algoritmo:
  - Percorra o vetor inteiro comparando elementos adjacentes (dois a dois)
  - Troque as posições dos elements se eles estiverem fora de ordem
  - Repita os dois passos acima com os primeiros n-1 itens, depois com os primeiros n-2 itens, até que reste apenas um item

- O método ilustrado:

	1	2	3	4	5	6
Chaves iniciais:	O	R	D	E	N	A
i = 6	O	D	E	N	A	**R**
i = 5	D	E	N	A	**O**	R
i = 4	D	E	A	**N**	O	R
i = 3	D	A	**E**	N	O	R
i = 2	A	**D**	E	N	O	R

- As chaves em negrito foram empurradas para o fim do vetor a cada passada : *bolhas*

- Método da Bolha (Bubblesort):

```
void  Bolha (Vetor A; Indice n) {
Indice  i, j;
Item temp;

   for  (i:= n-1; i >= 1; i--)  {
        for (j= 0; j < i ;j++) {
              if  (A[j].chave < A[j+1].chave) {
                 temp = A[j].chave;
                 A[j].chave = A[j+1].chave;
                  A[j+1].chave = temp;
              }
        }
   }
}
```

- Número de comparações entre chaves e movimentações de registros, pior caso:

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

$$M(n) = 3C(n)$$

- Método muito simples, mas custo alto
  - Adequado apenas se arquivo pequeno
  - Ruim se registros muito grandes

- Número de operações não se altera se vetor já está (parcialmente) ordenado
  - Como melhorar?

- Método da Bolha Melhorado: Termina execução quando nenhuma troca é realizada após uma passada pelo vetor

```
#define  TRUE   1
#define  FALSE  0

void  Bolha (Vetor A; Indice n) {
   Indice  i, j;
   Item temp;
   char troca;

   troca = TRUE;
   for  (i:= n-1; (i >= 1) && (troca == TRUE); i--)  {
       troca = FALSE;
       for (j= 0; j < i ;j++) {
            if  (A[j].chave < A[j+1].chave) {
                    temp = A[j].chave;
                    A[j].chave = A[j+1].chave;
                    A[j+1].chave = temp;
                    troca = TRUE;
             }
       }
   }
 }
```