Problemstellung bei Sortieralgorithmen

Generell geht es bei *Sortieralgorithmen* darum gegebene Elemente zu sortieren.

Folgende Fragen sind dazu zunächst zu beantworten:

- der *Typ* der Daten: kann direkt sortiert werden (*primitive Datentypen* wie int), oder muß gemäß eines Attributs, etwa key oder value, sortiert werden?

- Können die *Standardvergleiche* und ausgenutzt werden, oder müssen *spezielle Vergleichsfunktionen* benutzt und ggf. erst programmiert werden?

- Soll *aufsteigend* oder *absteigend* sortiert werden?

- Geschieht das Sortieren *ohne zusätzlichen Speicherbedarf* (”**in place**“), oder auf einer *Kopie* der Datensätze?

- Haben alle *Datensätze* gleichzeitig Platz im Speicher (”**Internes Sortieren**“), oder sind externe Speicher einzubeziehen (”**Externes Sortieren**“)?

- Welche *Komplexität* hat das gewählte Verfahren?

Gemessen werden bei Sortieralgorithmen

- die *Anzahl Vergleiche (C)* sowie
- die *Anzahl Bewegungen von Datensätzen (M)*

Sortieralgorithmus BubbleSort

Bubble Sort ist ein elementares Suchverfahren.

Die Grundidee des Verfahrens ist:

> Vertausche *direkt benachbarte Elemente,* falls diese nicht sortiert sind und laufe dazu max. n-mal über die Liste, um vollständig zu sortieren.
>
> Komplexität: $O(n^2)$

```
// Implementierung BubbleSort
// Sortieren der eingegebenen Records nach dem BubbleSort-Verfahren
// ohne Laufzeitoptimierung durch boole'sche Variable.

for (i=0;i<AnzElemente; i++)
   {
     for(j=(AnzElemente-1); j>i; j--)
                            // Vergleich zweier benachbarter Elemente

       if(RecordCompare(&stARecord[j-1],&stARecord[j]) > 0)
                    RecordExChange(&stARecord[j-1], &stARecord[j]);
   }
```