Sortieren durch Einfügen (InsertionSort)

Verfahren:

(a) Man betrachtet eine Teilliste, die nur das erste Element der Liste enthält. Diese Teilliste ist somit bereits sortiert.

(b) Das der bereits sortierten Teilliste unmittelbar nachfolgende Element der Liste wird in diese sortierte Teilliste eingeordnet. Dies kann dadurch geschehen, dass man die Elemente solange durchmustert, bis man ein (erstes) Element gefunden hat, dessen Schlüssel größer ist. Genau an dieser Stelle wird das neue Element eingesetzt, d.h. die nachfolgenden Elemente der sortierten Teilliste werden um je eine Stelle verschoben. Gibt es keinen solchen größeren Schlüssel, dann bleibt das neue Element an seiner ursprünglichen Stelle stehen.

(c) Nach Schritt (b) hat sich die sortierte Teilliste um eine Element vergrößert. Gibt es ein nachfolgendes Element, dann wird mit (b) fortgesetzt, ansonsten ist die Sortierung beendet.

Mittlere Anzahl der Vergleiche $\approx n^2 / 4$ (n = Anzahl der Listenelemente)

Pseudokode:

```
algorithm InsertionSort(F)      // E/A-Parameter F: zu sortierende Folge F der Länge n
   for i:=1 to n-1 do {
       m := F[i];                  // zu merkendes Element
       j := i;
       while (j > 0) do {
          if (F[j-1] >= m) then {
            /* Verschiebe F[j-1] eine Position nach rechts */
            F[j] := F[j-1];
            j := j - 1
          }
          else  break;
       }
       F[j] := m                   // zeigt auf Einfügeposition
   }
}
```

```
/* Sortiert aufsteigend die n Elemente des int-Arrays v in C++ */
void insertionsort(int v[], unsigned long n) {
    for(unsigned long i = 1UL; i<n; i++){
        int m=v[i];
        unsigned long j=i;
        while(j>0UL){
            if(v[j-1UL]>=m){ v[j]=v[j-1UL]; j--; }
            else break;
        }
        v[j]=m;
    }
}
```

Sortieren durch Auswahl (Selection Sort)

Verfahren:

(a) Suchen des Elementes mit dem kleinsten Schlüssels in der Liste

(b) Tauschen des ersten Listenelementes mit dem Element mit dem kleinsten Schlüssel

(c) Man betrachtet nun die Liste ohne dem ersten Element als die Liste, die noch sortiert werden muss. Diese um ein Element kürzere Liste wird mit dem gleichen Verfahren behandelt (ab Schritt (a)), es sei denn, sie enthält nur noch ein Element. Im letzteren Fall ist die Gesamtsortierung abgeschlossen.

Anzahl der Vergleiche $\approx n^2 / 4$ (n .... Anzahl der Listenelemente)

```
algorithm SelectionSort( F ){   // E/A-Parameter F: zu sortierende Folge F der Länge n
      p := n;
      while( p > = 0) do {
            g := Index des größten Elementes aus F im Bereich  0 ... p-1
            Vertausche Werte von F[p] und F[g];
            p := p-1;
      }
}
```

```
/* Sortiert aufsteigend die n Elemente des int-Arrays v C++ */
void selectionsort(int v[], unsigned long n) {
    unsigned long p=n-1UL;
    while(p>=0UL){
        unsigned long g = 0UL;
        for(unsigned long i = 1UL; i<=p; i++)
            if(v[i]>v[g]) g=i;
        int temp=v[g];
        v[g]=v[p];
        v[p]=temp;
        if(p==0UL) break;
        p--;
    }
}
```