Unit 17

Dr. Günter Kolousek

21. Juli 2015

Lege wiederum ein Verzeichnis an. Nennes es `29_unit17`! In diesem Verzeichnis sollen alle Dateien der jeweiligen Einheit abgelegt werden.

1 Schulübungen

Weiter mit dem Sortieren und im Speziellen dem Insertion-Algorithmus!

1. Testen wir zuerst nochmals die Laufzeit: Modifiziere zuerst das Programm aus Einheit 16 so, dass nicht nur 1000 sondern auch 2000 und 4000 Zahlen zu sortieren sind und passe die Ausgabe an!

   Was kann man hier erkennen? Bei einer Verdopplung der zu sortierenden Elemente kommt es zu…

   Laufzeiten gemessen auf einem Acer AS1810TZ Subnotebook. Bei mir kam es zu folgender Ausgabe auf dem Subnotebook:

   ```
   1000 Elemente
   bubble1: 0.617578983307s
   bubble2: 0.384479999542s
   bubble3: 0.401995897293s
   bubble3/sortiert: 0.000426054000854s
   selection: 0.178674936295s
   selection/sortiert: 0.176262140274s

   2000 Elemente
   bubble1: 2.53244400024s
   bubble2: 1.61540794373s
   bubble3: 1.66196107864s
   bubble3/sortiert: 0.000874042510986s
   selection: 0.707681894302s
   selection/sortiert: 0.709786891937s
   ```

```
4000 Elemente
bubble1: 10.1481890678s
bubble2: 6.4617459774s
bubble3: 6.67881298065s
bubble3/sortiert: 0.00176692008972s
selection: 2.87358307838s
selection/sortiert: 2.84905791283s
```

2. Schreibe jetzt eine neue Funktion `insertion(lst)` in einem Modul `unit17`, die eine Liste mit dem Insertion-Sort Algorithmus sortiert. Wie der funktioniert? Lies weiter!

   Die prinzipielle Idee ist folgende:

   a) Lege eine neue Ergebnisliste mit dem ersten Element der zu sortierenden Liste an.

   b) Gehe alle Elemente der zu sortierenden Liste von Position 2 bis zum Ende durch und füge das aktuelle Element in der Ergebnisliste ein.

   Das Verfahren heißt deshalb Insertion-Sort, weil jedes Element der ursprünglichen Liste in einer Ergebnisliste an der richtigen Stelle eingefügt wird.

   Hier folgt die genauere Beschreibung des Algorithmus:

```
Für jedes Element "curr" von der zweiten bis zur letzten Position:
  Wert "val" mit dem Wert an der Stelle "curr" belegen
  Index "i" mit "curr" belegen
  Endlosschleife
    Wenn i == 0, dann Schleife beenden
   Wenn Wert an Stelle i - 1 kleiner gleich "val", dann Schleife beenden
    Liste an der Position i mit Wert von Position i - 1 belegen
    i dekrementieren
  Liste an der Position i mit "val" belegen
```

   Schreibe eine Funktion `insertion(lst)`, die diesen Algorithmus implementiert.

3. Teste wiederum die Laufzeiten analog zu dem Testen der Laufzeit des Selection-Sort Verfahrens!

   Es sind die Laufzeiten für jeweils 1000, 2000 und 4000 zufällige Elemente und jeweils auch für eine schon aufsteigend sortierte Liste zu ermitteln und auszugeben.

   Laufzeiten gemessen auf einem Acer AS1810TZ Subnotebook. Bei mir kommt es zu folgender Ausgabe auf dem Subnotebook:

```
1000 Elemente
insertion: 0.213353872299s
insertion/sortiert: 0.00111818313599s

2000 Elemente
insertion: 0.868803977966s
insertion/sortiert: 0.00175404548645s

4000 Elemente
insertion: 3.58962988853s
insertion/sortiert: 0.000869035720825s
```

Vergleich jetzt die Laufzeiten mit den Laufzeiten von Selection-Sort. Was fällt auf?

Wann wird man Bubble-Sort, wann Selection-Sort und wann Insertion-Sort verwenden?