DAP2 Praktikum für ETIT und IKT, SoSe 2018, Kurzaufgabe K2

Fällig am 30.4. um 18:00

Die Regeln für Programmieren und die Punktevergabe gelten wie bei Aufgabe K1.

Kurzaufgabe 2 (2 Punkte)

- Verwenden Sie den BubbleSort-Algorithmus vom letzten Mal

  ```
  template < class T > void BubbleSort(vector<T> &v)
  ```

  und schreiben Sie ein Programm `k2`, das `BubbleSort` benutzt und die nachfolgenden Eigenschaften hat.
- Der Aufruf erfolgt mit

  ```
  k2 Duration
  ```

  wobei `Duration` eine positive Fließkommazahl ist. `Duration` ist die Vorgabe der Laufzeit in Sekunden von `BubbleSort`. Das Programm `k2` soll selbständig die Feldgröße `n` für `BubbleSort` ermitteln, die annähernd zu dieser Laufzeit führt.
- Der Algorithmus soll zweistufig ablaufen. In einem ersten Schritt wird grob eine untere und eine obere Schranke für die Laufzeit ermittelt. In einem zweiten Schritt wird per Intervallhalbierung das Ergebnis verbessert.
- `k2` versucht im ersten Schritt zunächst eine Feldgröße `n=1`, die nachfolgend immer weiter verdoppelt wird. Es füllt jeweils einen `vector` mit `n` zufälligen `double` Zahlen zwischen `0` und `1` und bestimmt dann die die Laufzeit für die Sortierung. Die Feldgröße `n` wird so lange verdoppelt, bis die gemessene Laufzeit `Duration` übersteigt.
- Im zweiten Schritt wird mit dieser maximalen Feldgröße und der halben Feldgröße als Startwert eine binäre Suche („Intervallhalbierung“) für die Feldgröße `n` gestartet, die so lange läuft, bis die aktuell gemessene Laufzeit der Vorgabe `Duration` bis auf 0,001 Sekunde entspricht. Die so ermittelte Feldgröße wird am Ende ausgegeben.
- Geben Sie die Feldgröße und die Laufzeiten während des Laufs von `k2` aus, um die Funktion nachzuvollziehen.
- Ein Beispiellauf von `k2` sollte genau in dieser Form formatiert sein und so aussehen:

```
> k2  2

Initial Search...

 Length | Duration
--------+---------
      1 | 0.0000
      2 | 0.0000
      4 | 0.0000
      8 | 0.0000
     16 | 0.0000
     32 | 0.0000
     64 | 0.0000
    128 | 0.0000
```

```
   256 | 0.0000
   512 | 0.0000
  1024 | 0.0000
  2048 | 0.0310
  4096 | 0.1090
  8192 | 0.2820
 16384 | 1.0620
 32768 | 4.5470

Starting Iteration...

   Left |       n |   Right | MeasuredTime | DeltaTime
--------+---------+---------+--------------+----------
  16384 |   24576 |   32768 |    2.6090    |   0.6090
  16384 |   20480 |   24576 |    1.8910    |  -0.1090
  20480 |   22528 |   24576 |    3.0930    |   1.0930
  20480 |   21504 |   22528 |    2.4380    |   0.4380
  20480 |   20992 |   21504 |    2.2190    |   0.2190
  20480 |   20736 |   20992 |    1.7810    |  -0.2190
  20736 |   20864 |   20992 |    2.0940    |   0.0940

Choosing n=20864 @ 2.0940 Seconds.
```

Hinweise:

- Verwenden Sie den STL-Container `vector` aus `#include <vector>` zum Abspeichern der Zufallszahlen vom Typ `double`. Beachten Sie insbesondere `vector::resize()`und verwandte Methoden.
- Aufgrund der Rechnerumgebung schwanken die Messungen der Zeitmessungen. Entwerfen Sie geeignete Abbruchbedingungen, so dass die binäre Suche zwar nicht immer das optimale Ergebnis liefert (die Abweichung kann unter Umständen dann größer als 0,001 Sekunden sein) aber zumindest sicher terminiert.