Technische Universität München
Fakultät für Informatik
Rechnerarchitektur-Praktikum
SS 2015

DICHTEMESSUNG

SPEZIFIKATION

Bearbeitet von:

Mahdi Sellami
Niklas Rosenstein
Christoph Pflüger

07.06.2015

INHALT

1. Aufgabenkurzbeschreibung

Es soll ein Assembler Programm zur Berechnung von Dichtewerten anhand einer Eingabedatei entwickelt werden. Dabei wird die Eingabedatei von einem "C" Rahmenprogramm eingelesen und von dort das eingebundene Assemblerprogramm aufgerufen. Für Teilaufgabe 1 soll zu jedem Wertepaar ein Dichtewert berechnet und in einen Buffer geschrieben werden. In Teilaufgabe 2 soll aus den berechneten Werten der Kleinste, Größte und der Mittelwert gefunden werden, wobei die 10 größten und 10 kleinsten Werte vernachlässigt werden sollen.

2. Lösungsansätze

Die Lösung von Teilaufgabe 1 ist eindeutig und nicht auf mehrere Wege lösbar. Für Teilaufgabe 2 hingegen wurden zwei Lösungsansätze ausgearbeitet. **Lösungsansatz A** verwendet mehrere Suchen nach dem kleinsten bzw. größten Wert innerhalb eines Intervalls wobei der erste Suchlauf nicht auf ein Minimum bzw. Maximum begrenzt ist. **Lösungsansatz B** hingegen verwendet einen Sortieralgorithmus.

2.1. Lösungsansatz A

Dieser Lösungsansatz verwendet wiederholte Intervalsuche um den *n*-kleinsten Wert zu ermitteln. Die Laufzeit beträgt stets $O(n^2)$. Die Abwandlung um den *n*-größten Wert zu ermitteln ist trivial und wird nicht weiter erörtert. Nachdem mithilfe dieses Algorithmus die untere und obere Schranke gefunden wurden, müssen bei der Berechnung des Mittelwerts lediglich alle Werte außerhalb dieses Intervals vernachlässigt werden.

Algorithmus:

```
// Finde kleinsten Wert in values
Initialisiere min mit values[0]
Für jeden Wert x in values
    Wenn x < min
        min = x
// Finde 10. Kleinsten Wert in values
Durchlaufe 9 Schleifen
    // Finde nächst größeren Wert zu min
    Initialisiere tmp mit null
    Für jeden Wert x in values
        Wenn x > min und (tmp === null oder x < tmp)
            tmp = x
    // Untere Schranke für nächsten Durchlauf aktualisieren
    min = tmp
Ergebniswert min
```

Deklarationen:

- ***min***: Aktueller kleinster Wert
- ***tmp***: Aktueller kleinster Wert größer *min*
- ***x***: Aktueller Wert in Durchläufen aller Eingabewerte
- ***values***: (Eingabe) Eingabewerte

2.2. Lösungsansatz B

Dieser Lösungsansatz verwendet den "*Insertion Sort*" Sortieralgorithmus um die Dichtewerte in aufsteigende Reihenfolge zu bringen. Anschließend kann der *(n-1)*-te Wert aus der sortierten Werteliste als untere Schranke verwendet werden. Die obere Schranke lässt sich in der selben Weise durch Indizierung von der rechten Seite der Liste ermitteln. Bei der anschließenden Berechnung des Mittelwerts können die ersten und letzten *n* Werte ausgelassen werden.
Die Laufzeit des "Insertion Sort" Algorithmus beträgt im besten Fall $O(n)$ und im schlechtesten und durchschnittlichen Fall $O(n^2)$.

3. Bewertung der Ansätze

Lösungsansatz A		**Lösungsansatz B**	
Vorteile	**Nachteile**	**Vorteile**	**Nachteile**
• Einfache Implementierung • Konstante Schleifenzahl *n-1*	• Zweifache Implementierung notwendig (kleinster und größter Wert)	• Best Case Performance • Keine weiteren Vergleiche bei Mittelwertberechnung • Einmalige Ausführung	• Komplizierte Implementierung

4. Entscheidungswahl

Da die zweifache Implementierung desselben Algorithmus sehr Fehleranfällig ist und bei Korrekturen beide in Synchronisation gehalten werden sollten, wählen wir den Lösungsansatz B.