Termination

Korrektheit und Termination

Schleifeninvarianten $\Rightarrow$ Korrektheit jeder Iteration,
Obergrenze für Anzahl der Iterationen $\Rightarrow$ **Termination**

Notwendig: jede Iteration bringt uns näher zum Ziel

Berechnung einer Obergrenze für Anzahl der Iterationen

ähnelt Aufwandsabschätzung, aber Unterschiede im Detail
(Subtraktion gut, Division schlecht)

Termination versus Aufwandsabschätzung

Binäre Suche: Abbruch wenn $low > high$; pro Iteration:
$low = middle + 1$ oder $high = middle - 1$
wobei $middle = (low + high)/2$

Termination: vor Änderung stets $low \leq middle \leq high$
Iteration verringert $high - low$ um mindestens 1
Obergrenze: $high - low == elems.length$

Aufwand: jede Iteration halbiert $high - low$ (ungefähr)
daher logarithmischer Aufwand
wenn $high == low$ ändert Halbierung nichts
Halbierung für Termination nicht ausreichend

Fortschritt und Termination

jede Iteration entspricht einer Zahl in einer Reihe
(z.B. $j - i$ wobei i bzw. j sich in jeder Iteration ändert)

es muss eine Grenze geben (z.B. $j - i \geq 0$)

jede Iteration muss Zahl näher an die Grenze bringen
(z.B. $j - i$ wird in jeder Iteration um mindestens 1 kleiner)

der Fortschritt darf einen Mindestwert nicht unterschreiten
(nicht unbegrenzt verkleinern, z.B. nicht stets halbieren)

Überlegungen zur Termination beim Programmieren

Testen

Testen und Softwarequalität

Testfälle können nicht alles abdecken

gefundene Fehler $\approx$ fixer Anteil an vorhandenen Fehlern

1 Fehler ausgebessert $\rightarrow$ 10 Fehler eingebaut

Fehler in jeder Softwareentwicklungsphase

nicht alle Fehler werden sichtbar (aber für wen?)

seltene Fehler sind großes Sicherheitsrisiko (Eindringen)

„Fehler“ können Absicht sein (Eindringen ermöglichen)

Testen und statisches Verstehen

Testen kann Problemstellen aufdecken

aber Qualitätssteigerung nur durch statisches Verstehen
(z.B. Code-Reviews), nicht durch Korrektur gefundener Fehler

Testfälle zur Spezifikation verwendbar

Aufgabe: Generische Methoden

Such Sie in Gruppen zu zwei bis drei Personen Antworten auf folgende Frage:

Was ist besser:
Wenn man beim Testen viele oder wenige Fehler aufdeckt?

Zeit: 2 Minuten

Teststufen

Unittest

Integrationstest

Systemtest

Abnahmetest

Testmethoden

Black-Box-Test (Testen am Ende)

White-Box-Test (Programm und Testfälle gleichzeitig)

Grey-Box-Test (Testfälle zuerst)

Testarten

Funktionaler Test

Nichtfunktionaler Test

Schnittstellentest

Oberflächentest

Stresstest (z.B. Crasch- oder Lasttest)

Sicherheitstest

Regressionstest

...

Laufzeitmessungen

Einflüsse auf Laufzeiten

Latenz (Einweglatenz, Round-Trip-Time)

Bandbreite

feingranularer Parallelismus

Thread-Parallelismus

Rechnerbelastung, Betriebssystem, Compiler, Interpreter, Hardware

...

Datenmenge

Häufiger **Fehler:** Hochrechnen auf andere Datenmenge

PK

TU WIEN

Echtzeitsysteme

Zeitüberschreitung = Fehler

harte versus weiche Echtzeitsysteme

harte Echtzeitsysteme häufig sicherheitskritisch

Laufzeitmessungen + Beweisverfahren, oft mit Redundanz

Java dafür kaum geeignet, eher C