

## Typprüfung (Compiler / Laufzeit)

In C und Java müssen Variablen und Methodenergebnisse durch Typangaben erläutert werden. Welche der folgenden Aussagen sind korrekt und welche nicht:

- 1) Der Compiler prüft ob ein Programm hinsichtlich der Datentypen korrekt ist.
- 2) Der Cast (A)x verwandelt das in x referierte Objekt in ein Objekt vom Typ x.
- 3) Nach den Befehlen: `double x = 1.3; (int) x;` steht in x die ganze Zahl 1.
- 4) Die Zuweisung `a = "abc";` ist nur dann korrekt, wenn die Variable a den Typ String hat.

Angenommen wir haben die folgenden Programmelemente:

```
public interface I { ... }
public abstract class A implements I{ ... }
public class B extends A{ ... }
public enum E { ... }
```

welche der folgenden Anweisungen sind (mit Sicherheit) falsch:

- 1) `I vi;`
- 2) `A va;`
- 3) `B vb;`
- 4) `E ve;`
- 5) `vi = new I();` falsch (kein Objekt vom Interface)
- 6) `va = new A();` falsch (kein Objekt von abstrakt. Klasse)
- 7) `vb = new B();`
- 8) `ve = new E();` falsch (Enum-Konstruktor ist private)
- 9) `vi = va;`
- 10) `vi = vb;`
- 11) `vi = ve;` falsch (keine Unterklasse)
- 12) `ve = vb;` " "
- 13) `va = vi;` falsch (keine Unterklasse, aber Cast mögl)
- 14) `va = vb;`
- 15) `va = ve;` falsch (keine Unterklasse)
- 16) `vb = vi;` falsch (keine Unterklasse, aber Cast mögl)
- 17) `vb = va;` falsch (keine Unterklasse, aber Cast mögl)
- 18) `vb = ve;` falsch (keine Unterklasse)
- 19) `ve = vi;` falsch (keine Unterklasse)
- 20) `ve = va;` falsch (keine Unterklasse)
- 21) `ve = vb;` falsch (keine Unterklasse)
- 22) `vb = (B) vi;` Laufzeit entscheidet
- 23) `vb = (B) va;` Laufzeit entscheidet
- 24) `vb = (B) ve;` falsch (keine Unterklasse)

Noch eine Frage zu casts

wir haben zwei Variablen:

```
double x = 1.4;
Object y = "abc";
```

in x steht das Bitmuster für 1.4;

in y steht das Bitmuster für die Adresse von "abc", z.B. 01fa2e34

was steht nach

```
int i = (int) x;  
String j = (String) y;
```

in x, y, i, j?

x,y auf jeden Fall unverändert

(Variablen der rechten Seite bleiben unverändert – Ausnahme i++/++i)

i enthält das Bitmuster für die ganze Zahl 1, also 0000...0001

j enthält exakt die gleiche Referenz wie y (auch das Stringobjekt selbst bleibt unverändert)

[www.gm.fh-koeln.de/ehses/ap/examples/Uebung\\_04.pdf](http://www.gm.fh-koeln.de/ehses/ap/examples/Uebung_04.pdf)

Was ist eine anonyme Klasse?

*Eine anonyme Klasse ist eine (innere) Klasse ohne Namen.*

## Rekursion

Was macht folgende Methode (Vorbedingung, Nachbedingung?) ?

```
int s(int a, int b) {  
    return b == 0 ? 0 : a + s(a, b-1);  
}
```

*Vorbedingung:  $b \geq 0$*

*Nachbedingung: Ergebnis ==  $a * b$*

## O-Notation

Angenommen ein Algorithmus ist in  $O(N^3)$  und für  $N = 1000$  ergibt sich eine Laufzeit von 0,3 sec.

Wie groß ist die erwartete Laufzeit für  $N = 10000$ ?

Desgleichen für  $O(\log N)$ ,  $O(N \log N)$ ,  $O(N)$ ,  $O(N^2)$  ..

*Beispiel  $O(N \log N)$ , sehr ausführlich*

*$t(N) = a N \log N + \dots$  // Rest spielt für große  $N$  keine Rolle,  $a = \text{Konstante}$*

*$t(1000) = a 1000 \log(1000) = 3000 a$  ( $\log = \log_{10}$ , da egal)*

*$t(10000) = a 10000 \log(10000) = 40000 a$*

*aus  $t(1000) = 0,3 \text{ sec}$  folgt:  $a = 0,0001 \text{ sec}$*

*damit  $t(10000) = 0,0001 \text{ sec} * 40000 = 4 \text{ sec}$*

*anderer Weg: ohne  $O(N)$  heißt,  $t(10000) = 3 \text{ sec}$ . Log-Faktor bewirkt, dass*

*Ergebnis =  $3 \text{ sec} * \log(10000)/\log(1000) = 4 \text{ sec}$ .*

# Sortieren

Betrachten Sie folgende Zahlenfolge

23 21 14 4 5 18 19 1 13

Sortieren Sie die Zahlenfolge manuell mit dem Sortierverfahren "Direktes Einfügen". Schreiben Sie jede Veränderung der Zahlenfolge auf, so dass die einzelnen Schritte nachvollziehbar sind. Trennen Sie dabei den sortierten linken von dem unsortierten rechten Teil der Folge durch einen senkrechten Strich und kennzeichnen Sie im unsortierten Teil das in einem Sortierschritt einzufügende Element durch unterstreichen oder einkreisen.

23 | 21 14 4 5 18 19 1 13  
21 23 | 14 4 5 18 19 1 13  
14 21 23 | 4 5 18 19 1 13  
4 14 21 23 | 5 18 19 1 13  
4 5 14 21 23 | 18 19 1 13  
4 5 14 18 21 23 | 19 1 13  
4 5 14 18 19 21 23 | 1 13  
1 4 5 14 18 19 21 23 | 13  
1 4 5 13 14 18 19 21 23 |

## Schnelle Sortierverfahren

a) Welche Bedeutung hat die Wahl des Vergleichs-/Pivot-Elements bei Quicksort?

*Mittels des Pivot-Elements wird festgelegt, welche Elemente dem linken und welche dem rechten Feldbereich zugeordnet werden. Idealerweise sind dann beide Bereiche gleich groß.*

b) Welche Strategie kennen Sie, die Wahl des Vergleichs-/Pivot-Elements möglichst günstig zu treffen?

*Zunächst kann man ein beliebiges Element des zu sortierenden Bereichs auswählen. Die Auswahl wird besser, wenn man aus einer Stichprobe den Median bestimmt (aber nicht aus der gesamten Bereich!).*

c) Was ist der schlimmste Fall, der beim Quicksort eintreten kann? Welche Konsequenz hat das für die Laufzeit?

*Im schlimmsten Fall wird stets das kleinste oder größte Element als Pivot-Element gewählt. In diesem Fall ist die Laufzeit  $O(n)$ .*

d) Welche Strategie gibt es, für Quicksort die Laufzeit  $O(n \log n)$  zu garantieren? Begründen Sie Ihre Antwort!

*Keine. Für garantiertes  $O(n \log n)$  ist eine exakte Bestimmung des Medians notwendig. Diese ist selbst schon  $O(n \log n)$ , müsste aber sogar wiederholt durchgeführt werden.*

## Hashing

Hashfunktion (String) = Nummer des ersten Buchstabens % Arraygröße

Buchstabenummerierung:

1	A	11	K	21	U
2	B	12	L	22	V
3	C	13	M	23	W
4	D	14	N	24	X
5	E	15	O	25	Y
6	F	16	P	26	Z
7	G	17	Q		
8	H	18	R		
9	I	19	S		
10	J	20	T		

a) Warum ist diese Hashfunktion nicht gut?

*Die Werte liegen maximal im Bereich 1..26 damit ist die Funktion für große Arrays ungeeignet. Außerdem sind die Anfangsbuchstaben in realen Namen nicht gleich häufig.*

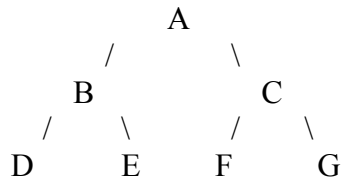
b) Gegeben ist die folgende Hashtabelle (lineare Verschiebung):

	<b>Key</b>	<b>Value</b>
<b>0</b>	Jogi	5
<b>1</b>	Fritz	16
<b>2</b>	Elke	33
<b>3</b>		
<b>4</b>	Nora	4

Da die Tabelle praktisch voll ist, soll Sie vor dem Eintragen von Gerd = 18 auf die Größe 7 erweitert werden. Wie sieht das Ergebnis aus, nachdem Gerd eingetragen wurde?

Wie könnte die Tabelle mit direkter Verkettung (Bucket-Sort) für ein Array der Länge 5 aus (mit Gerd) aussehen?

## Bäume



a) geben Sie den Baum in Preorder-Reihenfolge aus:

**ABDECFG**

b) geben Sie den Baum in Postorder-Reihenfolge aus:

**DEBFGCA**

c) geben Sie den Baum in Inorder-Reihenfolge aus:

**DBEAFCG**

d) Was ist ein entarteter Baum?

**Ein Baum, beim dem jeder Knoten maximal 1 Kind hat == Liste**

Gegeben ist die folgende Deklaration (null = kein Kind):

```
class Knoten {
    int zahl;
    Knoten links, rechts;
}
```

Ergänzen Sie die folgende Methode zur Bestimmung der größten Zahl in einem Baum:

```
// k ist nie null!
static int groessteZahl(Knoten k) {
    int max = k.zahl;
    if (k.links != null) {
        int maxLinks = groessteZahl(k.links);
        if (max < maxLinks) max = maxLinks;
    }
    if (k.rechts != null) {
        int maxRechts = groessteZahl(k.rechts);
        if (max < maxRechts) max = maxRechts;
    }

    return max;
}
```

weitere Aufgabe: schreiben Sie einen effizienteren Algorithmus, der von der Voraussetzung eines sortierten Suchbaums ausgeht.

a) rekursiv, b) iterativ ?

```
static int groessteZahl(Knoten k) {
}
}
```

## Verkettete Liste

Knoten:

```
class Knoten {
    int zahl;
    Knoten next;
}

class Liste {
    Knoten anfang;

    public int anzahl() {
        int c = 0;
        Knoten t = anfang;
        while(t != null) {
            t = t.next;
            c++;
        }
        return c;
    }

    public int max() {
        // ... iterative Fassung
        if (anfang == null) throw new NoSuchElementException();
        int m = anfang.zahl;
        Knoten t = anfang;
        t = t.next; // ist moeglich aber nicht notwendig
        while(t != null) {
            if (m < t.zahl) m = t.zahl;
            t = t.next;
        }
        return m;
    }

    public int anzahl2() {
        return anzahl(anfang);
    }

    private static int anzahl(Knoten k) {
        // rekursive Fassung
        if (k == null) {
            return 0;
        }
        else {
            return 1 + anzahl(k.next);
        }
    }
}
```

Üben Sie auch weitere ähnliche Dinge!