Mengen & Bitvektoren

Blagoy Genov

03.11.2011

Binärdarstellung von 16-bit Zahlen

```
short a = 96;
```

Binärdarstellung:

Bit-Nr.:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bitwert:	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0

entspricht der Menge $\{5, 6\}$

```
short b = 48;
```

Binärdarstellung:

Bit-Nr.:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bitwert:	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0

entspricht der Menge $\{4, 5\}$

Vereinigung (1)

Vereinigung zweier Mengen (mathematisch)

$$A \cup B := \{x \mid x \in A \vee x \in B\}$$

$A := \{5, 6\}$
$B := \{4, 5\}$
$A \cup B = \{5, 6\} \cup \{4, 5\} = \{4, 5, 6\}$

Vereinigung (2)

Vereinigung zweier Mengen (java)

```
short a = 96;
short b = 48;


short c = (short)( a | b);
```

a	0000000001100000
∪	
b	0000000000110000
c	0000000001110000

Durchschnitt (1)

Durchschnitt zweier Mengen (mathematisch)

$A \cap B := \{x \mid x \in A \wedge x \in B\}$

$A := \{5, 6\}$
$B := \{4, 5\}$
$A \cap B = \{5, 6\} \cap \{4, 5\} = \{5\}$

Durchschnitt (2)

Durchschnitt zweier Mengen (java)

```
short a = 96;
short b = 48;

short c = (short)( a & b );
```

a	0000000001100000
∩	
b	0000000000110000
c	0000000000100000

Weitere Mengenoperationen

Differenz:

$A \setminus B := \{x \mid x \in A \wedge x \notin B\}$

$A := \{5, 6\}$
$B := \{4, 5\}$
$A \setminus B = \{5, 6\} \setminus \{4, 5\} = \{6\}$

Kardinalität von Mengen:

$|A|$ - Anzahl der Elemente in A.

$|A| = |\{5, 6\}| = 2$

Diese Operationen sind nicht direkt in Java unterstützt!

Kardinalität

Einfacher Algorithmus:

Sei x eine Ganzzahl vom Typ `short`. Die Kardinalität der durch x repräsentierten Menge lässt sich folgendermaßen ausrechnen:

- Setze die Kardinalität initial auf 0.
- Für jedes Bit n in x:
  1. Lese den Wert von n (0 oder 1).
  2. Wenn $n = 1$, erhöhe die Kardinalität um 1.

Kardinalität (notwendige Java-Konstrukte)(1)

For-Schleife

Wiederhole einen Block von Anweisungen *n*-mal.

```
for ( int index = 0; index < n; index++ ) {
      ...
}
```

1. Initialisierung (einmal am Anfang): `int index = 0`
2. Bedingung (vor jeder Iteration): `index < n`
3. Aktualisierung (nach jeder Iteration): `index++`

Kardinalität (notwendige Java-Konstrukte)(2)

Bedingte Anweisung

Wenn a gleich b, führe den Anweisungsblock A aus. Ansonsten führe den den Anweisungsblock B aus.

```
if ( a == b ) {
      Block A
} else {
      Block B
}
```

Bitweise Verschiebungen

Verschieben nach links:

```
short a = 1;

short c = (short)( a << 5 );
```

a	0000000000000001 << 5
c	0000000000100000

Verschieben nach rechts:

```
short a = 32;

short c = (short)( a >>> 1 );
```

a	0000000000100000 >>> 1
c	0000000000010000

Komplement

Invertieren aller Bits:

```
short a = 1;

short c = (short)( ~a );
```

a	~0000000000000001
c	1111111111111110