4. Aussagenlogische Operationen

4.1. Einführung

In Kapitel 3 haben Sie arithmetische Operationen kennengelernt. Sie haben damit gesehen, wie man aus gegebenen Zahlenwerten weitere Zahlenwerte berechnen kann. Genauso wichtig wie dieses Zahlenrechnen sind die folgenden Operationen:

- Vergleiche von Zahlenwerten (und auch von Werten anderer Typen). Diese Vergleiche liefern Wahrheitswerte:
  - Beispiele:
    – Der Vergleich „5>0“ liefert den Wahrheitswert „wahr“ (englisch: „true“).
    – Der Vergleich „4<3“ liefert den Wahrheitswert „falsch“ (englisch: „false“).

  Aufgabe 1: Welchen Wahrheitswert liefert der Vergleich „7.5/2.5>3.5“?
  (Die Lösung der Aufgaben stehen am Ende des Kapitels.)
- Aussagenlogische Verknüpfungen von Aussagen (also von Sätzen, die jeweils einen Wahrheitswert besitzen). Diese Verknüpfungen liefern wiederum Wahrheitswerte:
  - Beispiele:
    – Die Und-Verknüpfung der beiden Aussagen
    „5>0“ (Wahrheitswert „true“) und „4<3“ (Wahrheitswert „false“)
    liefert den Wahrheitswert „false“.
    [Man beachte: Vergleiche wie 5>0 liefern Wahrheitswerte. Sie sind damit Aussagen im Sinne der Aussagenlogik und können daher durch aussagenlogische Operationen miteinander verknüpft werden.]
    – Die Oder-Verknüpfung
    „Berlin liegt in Frankreich“ (Wahrheitswert „falsch“)
    oder „Berlin liegt in Deutschland“ (Wahrheitswert „true“)
    liefert den Wahrheitswert „true“.

  Aufgabe 2: Welchen Wahrheitswert liefert die Verknüpfung „7%2 hat den Wert 1“ und „9%3 hat den Wert 2“?

4.2. Aussagenlogik allgemein

Die Grundlage für die Arbeit mit Wahrheitswerten ist die Aussagenlogik. Die Aussagenlogik beschäftigt sich mit Aussagen, also mit Sätzen, die einen Wahrheitswert besitzen. Es gibt zwei Wahrheitswerte; der Wertebereich, der der Aussagenlogik zugrundeliegt, ist also eine zweielementige Menge: { wahr, falsch } oder (mit den englischen Bezeichnungen) { true, false }.

Aussagenlogische Funktionen bilden Wahrheitswerte auf Wahrheitswerte ab. Mit diesen Funktionen kann man also Aussagen miteinander verknüpfen (also aus vorhandenen Aussagen neue Aussagen bilden) und den Wahrheitswert dieser neuen Aussagen aus den Wahrheitswerten der vorhandenen Aussagen ableiten.

Die einfachste aussagenlogische Funktion ist die Negation, also der „Nicht“-Operator.

- Beispiele:
  – Die Aussage „5>3“ hat den Wahrheitswert „false“. Die Negation dieser Aussage, nämlich „NICHT 5>3“ (oder anders formuliert: „5≤3“), hat den Wahrheitswert „true“.
  – Die Aussage „Berlin liegt in Deutschland“ hat den Wahrheitswert „true“. Die Negation dieser Aussage, nämlich „Berlin liegt nicht in Deutschland“, hat den Wahrheitswert „false“.

Aussagenlogische Funktionen lassen sich durch Wahrheitstafeln definieren. Das sind Tabellen, die für alle möglichen Parameterwerte der Funktion den zugehörigen Funktionswert angeben. Besonders einfach ist dies für die Negation, die eine „einstellige“ Funktion ist, also nur einen Parameter hat:

x	$\neg x$, $\overline{x}$
false	true
true	false

Die Einträge in der linken Spalte sind die möglichen Parameterwerte (also alle Wahrheitswerte, die bei einer Aussage auftreten können), die Einträge in der rechten Spalte definieren die zugeordneten Funktionswerte (also die entsprechenden Wahrheitswerte der Negation der Aussage). Die Überschrift der rechten Spalte sind zwei alternative Schreibweisen für die Negation einer Aussage x (bzw. ihres Wahrheitswerts). Man bezeichnet Operatoren wie $\neg$ und $\overline{\ }$ auch als (aussagelogische) Junktoren.

Die wichtigsten zweistelligen aussagenlogischen Funktionen (also Funktionen mit zwei Wahrheitswert-Parametern) sind die Und- und die Oder-Verknüpfung („Konjunktion“ bzw. „Disjunktion“ genannt). Sie leiten also aus den Wahrheitswerten zweier Aussagen den Wahrheitswert der „Verknüpfung“ dieser Aussagen ab – einer neuen Aussage, die aus den beiden bestehenden Aussagen gebildet wird. Die entsprechenden Operatoren/Junktoren sind

- $\wedge$ für die Konjunktion und
- $\vee$ für die Disjunktion.

Die Wahrheitstafeln für diese zweistelligen Funktionen definieren die Funktionswerte für alle vier möglichen Kombinationen von Parameterwerten (wieder: Parameterwerte links, zugehörige Funktionswerte rechts):

x	y	$x \wedge y$	$x \vee y$
false	false	false	false
false	true	false	true
true	false	false	true
true	true	true	true

Aufgabe 3: Welchen Wahrheitswert liefert die Verknüpfung „$4<2 \wedge 3>0$“?

Aufgabe 4: Welchen Wahrheitswert liefert die Verknüpfung „$4<2 \vee 3>0$“?

4.3. Aussagenlogik in Java: Typ und Operatoren

Java definiert einen eigenen Typ für Wahrheitswerte. Er heißt `boolean`. Die Werte des Typs `boolean` sind `true` und `false`.

- Beispiel: Deklaration und Initialisierung zweier `boolean`-Variablen
  `boolean a=true, b=true;`

In Java können die Operationen ausgeführt werden, die in der Einführung genannt wurden – nämlich Vergleiche und aussagenlogische Verknüpfungen. Hierfür gibt es entsprechende Operatoren:

- Relationale Operatoren dienen zum Vergleich von Werten und liefern Werte vom Typ `boolean`:

Operator	Bedeutung	Beispielausdrücke
`>`, `>=`	größer, größer gleich	`x>0`, `x-1>=y`
`<`, `<=`	kleiner, kleiner gleich	`x<0`, `x+2<=y`
`==`	gleich	`x==0`, `x+3==y`
`!=`	ungleich	`x!=0`, `x+y!=z`

  - In den Beispielausdrücken bezeichnen `x`, `y` und `z` (Zahl-)Variablen
    - Auswertung einiger Beispielausdrücke:
      – Hat die Variable `x` den Wert `5`, so hat der Ausdruck `x>0` den Wert `true`.
      – Hat die Variable `x` den Wert `0`, so hat der Ausdruck `x>0` den Wert `false`.
      – Hat die Variable `x` den Wert `1` und die Variable `y` den Wert `2`, so hat der Ausdruck `x+2<=y` den Wert `false`.
      – Hat die Variable `x` den Wert `1` und die Variable `y` den Wert `4`, so hat der Ausdruck `x+3==y` den Wert `true`.
      – Hat die Variable `x` den Wert `1`, so hat der Ausdruck `x!=0` den Wert `true`.
  - Achtung: Der Operator, mit dem man die Gleichheit prüft, ist das doppelte Gleichheitszeichen `==`. Das einfache Gleichheitszeichen `=` ist nämlich schon für Zuweisungen reserviert.
    - Beispiele:
      – `x=0` weist der Variablen `x` den Wert `0` zu.
      – `x==0` prüft, ob die Variable `x` den Wert `0` hat, und liefert entsprechend `true` oder `false`.
  - Bei der Auswertung eines Ausdrucks gelten die folgenden Prioritäten:
    - `<`, `>`, `<=`, `>=` binden stärker als `==` und `!=`. In einem Ausdruck werden also die Teilausdrücke mit den Operatoren `<`, `>`, `<=`, `>=` zuerst ausgewertet.
      - Beispiel:
        Im Ausdruck `4 < 3 == 5 >= 7` werden zuerst die Operatoren `<` und `>=` angewendet,

also die Teilausdrücke `4<3` und `5>=7` ausgewertet, und die Resultate in den Gesamtausdruck eingesetzt. Der Gesamtausdruck vereinfacht sich damit zu `false==false`. Dieser Ausdruck hat den Wert `true`, da auf beiden Seiten des `==` derselbe Wert steht.

- Ein Ausdruck mit mehreren gleichen Operatoren wird von links nach rechts ausgewertet.
  - Beispiel:
    Im Ausdruck `false == true == false` wird zunächst der linke Teil (also `false==true`) ausgewertet. Mit dem resultierenden Wert `false` wird dann weitergearbeitet, also der vereinfachte Ausdruck `false==false` ausgewertet, was für den Gesamtausdruck den Wert `true` liefert.
- Wie bei Zahlenausdrücken kann man durch Klammerung mit `()` eine andere Auswertereihenfolge festlegen.

- Achtung: Die Teilausdrücke, die auf der rechten und der linken Seite eines Vergleichsoperators stehen, müssen kompatible Typen haben (also z.B. beide von einem Zahlentyp oder beide vom Typ `boolean` sein).
  - Beispiele:
    – Ist `x` eine `int`- und `y` eine `double`-Variable, so ist der Vergleich `x<y` zulässig.
    – Ist `x` eine `int`- und `y` eine `boolean`-Variable, so ist der Vergleich `x==y` nicht zulässig.
    – Sind `x`, `y` und `z` Zahlenvariablen, so ist der Ausdruck `x<=y<=z` nicht zulässig. Die Auswertung des linken Teilausdrucks `x<=y` liefert nämlich einen `boolean`-Wert, der nicht (wie es das rechte `<=` tut) mit dem Zahlenwert `z` verglichen werden darf.
    [Wie ein korrekter Ausdruck aussieht, der prüft, ob y zwischen x und z liegt, wird in der Übung besprochen.]
- Auf `boolean`-Werte oder -Teilausdrücke dürfen nur die Operatoren `==` und `!=` angewendet werden.
  - Beispiel: Der Vergleich `false<true` ist nicht zulässig.

Aufgabe 5: Die int-Variable x habe den Wert 5, die int-Variable y habe den Wert 10. Welche Werte haben dann die folgenden Ausdrücke? `x+y<=15` `x*y>60` `x+y==7` `x+y!=7`

- Aussagenlogische Operatoren werden auf Werte des Typs `boolean` angewendet und liefern auch Werte dieses Typs. Die Operatoren in Java realisieren die drei aussagenlogischen Funktionen, die im vorigen Abschnitt eingeführt wurden:

Operator	Bedeutung	Beispielausdrücke
`!`	Negation ¬	`!a, !(y>x)`
`&&`	Konjunktion ∧	`a&&b, y>=x && y<=z, x==y && x==z`
`||`	Disjunktion ∨	`a||b, y<x || y>z, x==y || x==z`

- In den Beispielausdrücken bezeichnen `a` und `b` `boolean`-Variablen und `x`, `y` und `z` Variablen eines Zahlentyps.
  - Auswertung einiger Beispielausdrücke:
    – Hat die Variable `a` den Wert `true`, so hat der Ausdruck `!a` den Wert `false`.
    – Hat die Variable `a` den Wert `true` und die Variable `b` den Wert `false`, so hat der Ausdruck `a&&b` den Wert `false`.
    – Haben die Variablen `x`, `y` und `z` alle den Wert `1`, so hat der Ausdruck `x==y&&x==z` den Wert `true`.
    – Haben die Variablen `x`, `y` und `z` alle verschiedene Werte, so hat der Ausdruck `x==y||x==z` den Wert `false`.
- Achtung: Konjunktion und Disjunktion boolescher Ausdrücke wird jeweils durch ein Doppelzeichen (`&&` bzw. `||`) ausgedrückt. Was das einfache `&` bzw. `|` bedeutet, wird im folgenden Abschnitt erläutert.
- Bei der Auswertung eines Ausdrucks gelten die folgenden Prioritäten:
  - `!` bindet stärker als `&&`, `&&` bindet stärker als `||`.
  - ! bindet stärker als relationale Operatoren, relationale Operatoren binden stärker als `&&` und `||`.
    - Beispiel: Im Ausdruck `x==y&&x==z` werden zunächst die Vergleichsoperationen und danach, auf ihren Resultaten, die `&&`-Operation ausgeführt.
  - Ein Ausdruck mit mehreren gleichen Operatoren wird von links nach rechts ausgewertet.
  - Durch Klammerung mit `()` kann man eine andere Auswertereihenfolge festlegen.

Aufgabe 6: Die int-Variable x habe den Wert 5, die int-Variable y habe den Wert 10. Welche Werte haben dann die folgenden Ausdrücke? !(x>y) x<y||x>y x<y&&x>y

- Wie die Beispiele in der Tabelle zeigen, werden aussagenlogische Operatoren insbesondere in Kombination mit Vergleichsoperatoren benutzt. Zudem kann der Wert eines aussagenlogischen Ausdrucks einer `boolean`-Variablen zugewiesen werden.
  - Beispiel:
    ```
    boolean imIntervall = (zahl>=10 && zahl<=20);
    ```
- Eine besondere Eigenschaft aussagenlogischer Operationen in Java ist die Kurzschlussauswertung: Ein `boolean`-Ausdruck wird von links nach rechts nur so weit ausgewertet, bis das Ergebnis feststeht. Konkret heißt das:
  - Hat im Ausdruck `A||B` der erste Teilausdruck `A` den Wert `true`, so ist der Wert des gesamten Ausdrucks in jedem Fall `true` – unabhängig vom Wert von `B`. Java „spart sich“ also die Auswertung von `B`, führt also die Operationen von `B` nicht aus.
  - Hat im Ausdruck `A&&B` der erste Teilausdruck `A` den Wert `false`, so ist der Wert des gesamten Ausdrucks in jedem Fall `false` – unabhängig vom Wert von `B`. Auch hier wird `B` also nicht ausgewertet.

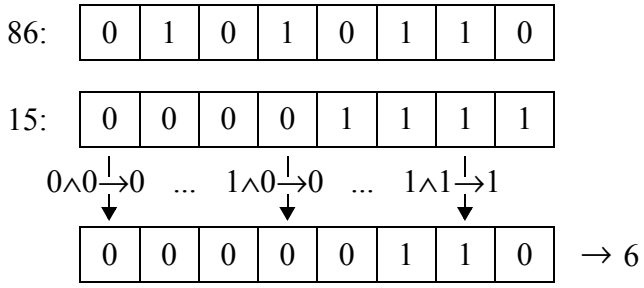
- Nützlich ist die Kurzschlussauswertung zum Beispiel, wenn man bei einer Division sicherstellen möchte, dass der Divisor ungleich `0` ist. Ein entsprechender Ausdruck sieht dann so aus: `y!=0 && x/y>1`. Ist hier `y` gleich `0`, dann wird der zweite Teilausdruck nicht ausgewertet. Es wird also nicht durch `y` dividiert, und das Programm stürzt nicht ab.

Aufgabe 7: Wie weit wertet Java den Ausdruck 4<5 && 5<6 && 6<3 && 5<2 && 6<7 aus?

4.4. Aussagenlogik in Java: Bitweise Operatoren

Aussagenlogische Operationen können in Java nicht nur auf Werten des Typs `boolean` ausgeführt werden, sondern auch auf einzelnen Bits von Zahlenwerten. Hierbei wird (in der Binärdarstellung der Zahl) die Ziffer 0 als false und die Ziffer 1 als true interpretiert.

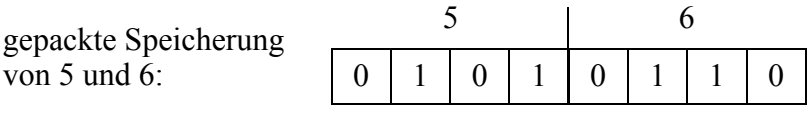
- Beispiel: Bitweise ∧-Verknüpfung der Zahlen 86 und 15 (vom Typ `byte`):



  Wie die Grafik zeigt, werden jeweils zwei Bits betrachtet, die bei den zu verknüpfenden Zahlen an derselben Position stehen. Diese Bits werden per ∧ verknüpft. Das Ergebnis (0 oder 1) wird an derselben Position der Resultat-Zahl eingesetzt. Die bitweise ∧-Verknüpfung von 86 und 15 liefert also den Wert 6.

Das Beispiel zeigt nur die prinzipielle Vorgehensweise, verdeutlicht aber nicht, wozu die bitweisen Operationen nützlich sind. Nützlich sind sie beispielsweise beim Umgang mit gepackt gespeicherten Zahlen, also Zahlen, die mit möglichst geringem Speicheraufwand abgelegt werden sollen. Eine gepackte Speicherung ist dann möglich, wenn die Zahlenwerte klein sind, wenn also ihre Binärdarstellung nur wenige Bits lang ist.

- Beispiel: Die Zahlen 5 und 6 können gemeinsam in den acht Bits einer `byte`-Variablen abgelegt werden – die 5 in den vorderen vier Bits, die 6 in den hinteren vier Bits:

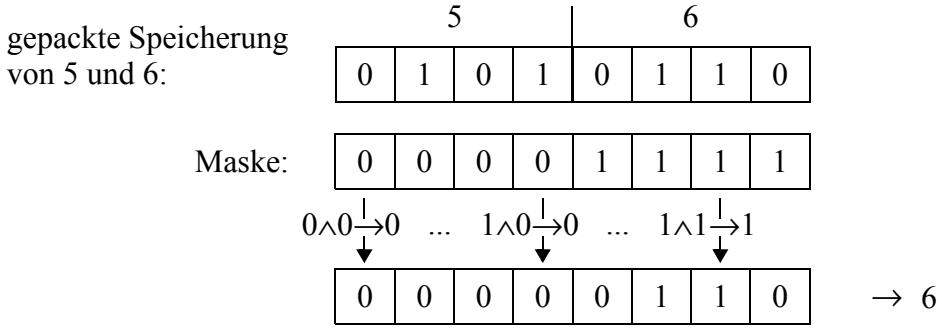


  Man benötigt hier also nur halb so viel Speicherplatz, als wenn man die Zahlen in zwei `byte`-Variablen ablegt.

Die bitweise ∧-Verknüpfung kann nun dazu benutzt werden, um eine der gepackt gespeicherten Zahlen zu extrahieren (also „herauszuholen“). Bei der Verknüpfung benutzt man eine „Maske“.

Eine solche Maske ist eine Bitkombination, die an genau den Positionen den Wert 1 hat, die man extrahieren möchte.

- Beispiel: Extraktion der Zahl, die in den hinteren vier Bits gespeichert ist



Die vordere Zahl wird extrahiert, indem man die Bits zunächst um vier Positionen nach rechts schiebt (also einen Rechtsshift durchführt) und dann mit der Maske verknüpft.

Die folgende Tabelle zeigt die bitweisen aussagenlogischen Operatoren von Java:

Operator	Bedeutung	Beispiel	Ergebnis des Bsp.ausdrucks
`&`	bitweises UND	`99 & 240`	$\texttt{0..001100011}_2$ $\texttt{\& 0..011110000}_2$ $\texttt{== 0..001100000}_2$ $\texttt{== 96}_{10}$
`|`	bitweises ODER	`28 | 1`	$\texttt{0..000011100}_2$ $\texttt{\| 0..000000001}_2$ $\texttt{== 0..000011101}_2$ $\texttt{== 29}_{10}$
`^`	bitweises EXKL. ODER	`29 ^ 1`	$\texttt{0..000011101}_2$ $\texttt{\| 0..000000001}_2$ $\texttt{== 0..000011100}_2$ $\texttt{== 28}_{10}$
~	bitweises NICHT	`~29`	$\texttt{\~{} 0..000011101}_2$ $\texttt{== 1..111100010}_2$ $\texttt{== -30}_{10}$

- Das exklusive Oder, das in der dritten Zeile genannt wird, liefert ein true genau dann, wenn genau einer der beiden Operanden true (der andere also false) ist. Die Verknüpfung „true EXKL. ODER true“ liefert also false – im Gegensatz zur „normalen“ ODER-Verknüpfung.
- Weitere Beispiele:
  – Maskierungsoperation auf einer `byte`-Variablen `var`, um die hinteren vier Bits zu extrahieren: `var&15` oder (besser lesbar mit Hexadezimaldarstellung) `var&0xF`
  – Maskierungsoperation auf einer `byte`-Variablen `var`, um die vorderen vier Bits zu extrahieren: `(var>>4)&0xF`

  Aufgabe 8: Welchen Wert hat der Ausdruck (31>>2)&9?

Bitweise aussagenlogische Operationen spielen auch in der Digitaltechnik eine wichtige Rolle. Näheres dazu erfahren Sie in diesem Fach.

4.5. Aussagenlogik in C

In C gibt es, im Gegensatz zu Java, keinen eigenen Typen für Wahrheitswerte. Wahrheitswerte werden in C durch Zahlenwerte dargestellt. Dabei steht eine 0 für false, alle Zahlenwerte ungleich 0 (insbesondere 1) stehen für true. In C muss also der Programmierer darauf achten, dass gut zwischen „tatsächlichen" Zahlen und Zahlen, die Wahrheitswerte darstellen, unterschieden wird, also Werte nicht falsch interpretiert werden. Der Java-Compiler würde dagegen solche Fehler erkennen.

4.6. Lernergebnisse dieses Kapitels

Nach diesem Kapitel muss man mindestens wissen,

- wie die Wahrheitstafeln von NICHT, UND und ODER aussehen,
- was der Unterschied zwischen einer aussagenlogischen Operation und einer bitweisen Operation ist und
- was eine Kurzschlussauswertung ist.

In Java muss man jetzt mindestens

- Variablen des Typs `boolean` deklarieren und initialisieren können,
- Vergleichsoperatoren, aussagenlogische Operatoren und Bitoperatoren anwenden können und insbesondere
- die Operatoren == und = jeweils richtig anwenden können.

Aufgabe 9: Laden Sie das Programmbeispiel http://www.nt.th-koeln.de/vogt/dv/java/Daten/Boole.java herunter, führen Sie es aus und vollziehen Sie seine Operationen im Quellcode nach. Verändern und erweitern Sie es nach Ihren eigenen Ideen.

4.7. Lösung der Aufgaben

Aufgabe 1: false (denn 7.5/2.5 ist 3.0, also kleiner als 3.5)

Aufgabe 2: false (denn 7%2 hat zwar den Wert 0, aber 9%3 hat den Wert 0, also nicht 2)

Aufgabe 3: false (denn 4<2 hat den Wahrheitswert false, 3>0 hat den Wahrheitswert true und gemäß der zweiten Zeile der Wahrheitstafel hat die Und-Verknüpfung dieser beiden Aussagen den Wahrheitswert false)

Aufgabe 4: true (denn 4<2 hat den Wahrheitswert false, 3>0 hat den Wahrheitswert true und gemäß der zweiten Zeile der Wahrheitstafel hat die Oder-Verknüpfung dieser beiden Aussagen den Wahrheitswert true)

Aufgabe 5: x+y<=15 hat den Wert true, x*y>60 hat den Wert false, x+y==7 hat den Wert false, x+y!=7 hat den Wert true.

Aufgabe 6: !(x>y) hat den Wert true, x<y||x>y hat den Wert true, x<y&&x>y hat den Wert false.

Aufgabe 7: bis einschließlich 6<3.

Aufgabe 8: 1 (denn: 31 hat die Binärdarstellung 0..011111. Ein Rechtsshift um zwei Stellen liefert 0..0111. Eine bitweise UND-Verknüpfung mit 0..01001, der Binärdarstellung von 9, liefert dann schließlich 1.)