`integer` - Interne Darstellung

- Darstellung als Bit-Vektor der Länge 32 = 4 Bytes.
- Interpretation als normale Dezimalzahl - aber halt, was ist mit negativen Zahlen?
- Ausweg: 31 Bits für die Zahl und 1 Bit für das Vorzeichen, unterschiedliche Möglichkeiten
- Achtung, größte Zahl ist $2^{31} - 1 = 2\,147\,483\,647$.

integer - Darstellung negativer Zahlen I

intToBits() gibt die Binärdarstellung eines Integers in umgekehrter Reihenfolge aus. Dabei schreibt R eine 0 als '00' und eine 1 als '01':

```
intToBits(41)
```

```
##  [1] 01 00 00 01 00 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
## [24] 00 00 00 00 00 00 00 00
```

```
rev(intToBits(2^31-1))
```

```
##  [1] 00 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01
## [24] 01 01 01 01 01 01 01 01
```

```
rev(intToBits(0))
```

```
##  [1] 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
## [24] 00 00 00 00 00 00 00 00
```

Das letzte Bit (das erste in der umgekehrten Reihenfolge) gibt das Vorzeichen an. Dabei bedeutet offensichtlich '00' positiv.

integer - Darstellung negativer Zahlen II

```
rev(intToBits(-41))
```

```
##  [1] 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01
## [24] 01 01 01 00 01 00 01 01 01
```

```
rev(intToBits(-(2^31-1)))
```

```
##  [1] 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
## [24] 00 00 00 00 00 00 00 00 01
```

Was ist jetzt hier passiert? Warum werden die negativen Zahlen nicht genauso dargestellt wie die positiven, nur mit einer '01' als letztem (ersten) Bit?

R verwendet zur Darstellung von negativen Zahlen das sogenannte 'Zweierkomplement'.

integer - Darstellung negativer Zahlen III

Wie bekommt man das Zweierkomplement?

- Man negiert alle Bits (inklusive des Vorzeichen-Bits),
- Man addiert eine 1 auf die negierten Bits.

Warum das ganze?

- Rechenoperationen werden so einfacher. Beispielweise kann eine Subtraktion zweier Zahlen als Addition mit einer negativen Zahl betrachtet werden und der 'Algorithmus der schriftlichen Addition' lässt sich anwenden.
- Die Null hat keine zwei Darstellungen mehr. Es gibt keine '+0' und '-0'.

integer - Darstellung negativer Zahlen IV

Damit gibt es jetzt jeweils $2^{31} - 1$ positive und negative Zahlen und eine 0. Eine Bit-Kombination ist nun noch nicht vergeben, die der 'negativen Null', sprich das Vorzeichen-Bit ist '01' (negativ) und die restlichen Bits sind '00'.

Welcher Wert könnte diese Bit-Kombination haben?

integer - Darstellung negativer Zahlen IV

Damit gibt es jetzt jeweils $2^{31} - 1$ positive und negative Zahlen und eine 0. Eine Bit-Kombination ist nun noch nicht vergeben, die der 'negativen Null', sprich das Vorzeichen-Bit ist '01' (negativ) und die restlichen Bits sind '00'.

Welcher Wert könnte diese Bit-Kombination haben?

Es ist das `NA` - der fehlende Wert:

```
rev(intToBits(NA))
```

```
##  [1] 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
## [24] 00 00 00 00 00 00 00 00
```

integer - Kleines Beispiel I

Zur besseren Veranschaulichung arbeitet dieses Beispiel mit 4 Bit statt der üblichen 32 Bit - es würde mit 32 Bit analog laufen und zum selben Ergebnis führen.

Aufgabe: Rechne $7 - 6$:

- Zunächst sei bemerkt, dass $7 - 6 = 7 + (-6)$ gilt.
- Die Binärdarstellung der 7 lautet '0111' und die der 6 '0110'.
- Bilde nun das Zweierkomplement der 6:
  - Negierung der Bits: '1001',
  - Addition von 1: '1001' + '0001' = '1010'.
- Rechne nun:

$$\begin{array}{cccccc} & & 0 & 1 & 1 & 1 \\ + & & 1 & 0 & 1 & 0 \\ & {\scriptstyle 1} & {\scriptstyle 1} & {\scriptstyle 1} & & \\ \hline & 1 & 0 & 0 & 0 & 1 \end{array}$$

integer - Kleines Beispiel II

- Nun haben wir 5 Bits statt der erlaubten 4. Das erste Bit wird deswegen gestrichen (bzw. vom Computer überhaupt nicht berechnet).
- Das Ergebnis der Addition lautet also '0001'.
- Somit haben wir ein Ergebnis mit positivem Vorzeichen und der Binärdarstellung '001', welche der 1 in der Dezimaldarstellung entstpricht.
- Das Ergebnis von $7 - 6$ lautet also $+1$.