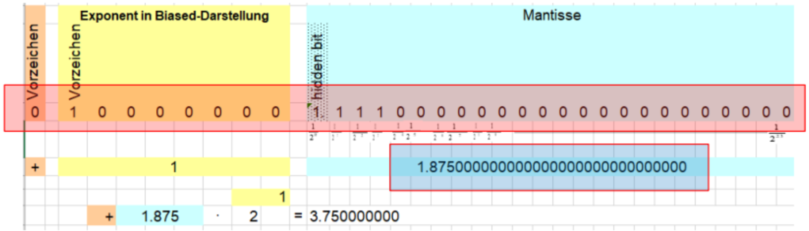
C# type/keyword	Approximate range	Precision	Size
`float`	$\pm 1.5 \times 10^{-45}$ to $\pm 3.4 \times 10^{38}$	~6-9 digits	4 bytes
`double`	$\pm 5.0 \times 10^{-324}$ to $\pm 1.7 \times 10^{308}$	~15-17 digits	8 bytes
`decimal`	$\pm 1.0 \times 10^{-28}$ to $\pm 7.9228 \times 10^{28}$	28-29 digits	16 bytes



Codierung	Einsatz-gebiet	Vorteile	Nachteile
Biased (und Exzess)	Gleitkomma-darstellung (für Exponent)	Einfache Anwendung	Nur resistent für Addition und Subtraktion
Zweier-komplement	Operationen auf Stufe ALU	Vorzeichen ist wegen hoch-wertigstem Bit klar (1 → negativ) Resistent für alle Grundoperationen	Leicht kompliziertere Bildung

Formel: $$R = \frac{\textit{nicht genutzte Kombinationen}}{\textit{alle möglichen Kombinationen}}$$

Der 1 aus 10 Code hat also eine Redundanz von 99%

Der Hamming-Abstand eines Codes gibt an, wie viele Bits man **mindestens ändern** muss, um ein anderes (gültiges) Zeichen desselben Codes zu erhalten.

Tendenziell gilt: Je grösser die Redundanz, desto grösser ist der mögliche Hamming-Abstand eines Codes.

Variante	Gerade Anzahl Einsen	Ungerade Anzahl Einsen
Even Parity	Paritätsbit: **0**	Paritätsbit: **1**
Odd Parity	Paritätsbit: **1**	Paritätsbit: **0**

Originale Nachricht: 1001 0000

Vorgegebener Rahmen, welcher die Plätze mit 2er-Potenznummern reserviert hat:

Platz	12	11	10	9	8	7	6	5	4	3	2	1
Data												

Schritt 1: Nachricht in Rahmen «abfüllen»:

Platz	12	11	10	9	8	7	6	5	4	3	2	1
Data	1	0	0	1		0	0	0		0		

Schritt 2: Alle Positionsnummern mit einer Eins werden XOR-verknüpft.

Position 12: 1100
Position 9: 1001
XOR-Verknüpfung: 0101

Schritt 3: Die XOR-Verknüpfung in die reservierten Plätze «abfüllen»:

Platz	12	11	10	9	8	7	6	5	4	3	2	1
Data	1	0	0	1	0	0	0	0	1	0	0	1

Decodieren auf Empfängerseite

XOR-Verknüpfung aller Positionen mit einer Eins:

Position 12: 1100
Position 9: 1001
Position 4: 0100
Position 1: 0001
XOR-Verknüpfung: 0000

Beispiel Fehlerkorrektur

Platz	12	11	10	9	8	7	6	5	4	3	2	1
Data	1	0	0	1	0	0	1	0	1	0	0	1

Durch Fehlübermittlung wurde an Stelle 6 eine Eins empfangen…

Position 12: 1100
Position 9: 1001
Position 6: 0110
Position 4: 0100
Position 1: 0001
XOR-Verknüpfung: 0110 → Fehler auf Position 6!

Durch Setzen des Wertes 0 auf Position 6 wird der Fehler behoben…

Mathematik: 1001_2 1001_{10} 1001_{16}

Informatik: 0b1001 0d1001 0x1001

Umrechnen von ... ...nach	Binär	Dezimal	Hexadezimal
Binär		Binär-Geld oder Algorithmus	4er-Blöcke machen
Dezimal	Stellenwerte aufaddieren		Stellenwerte aufaddieren
Hexadezimal	4er-Blöcke zusammenfassen	Hexa-Geld oder Algorithmus	

Operator	Symbol nach Norm A	Symbol nach Norm B
AND	&	AND
OR	≥1	OR
XOR	=1	XOR
NOT		NOT
NAND	&	NAND

Beispiel: 234 $_{(10)}$ → x $_{(2)}$

```
234 ÷ 2 = 117 Rest 0      1 1 1 0 1 0 1 0
117 ÷ 2 =  58 Rest 1
 58 ÷ 2 =  29 Rest 0
 29 ÷ 2 =  14 Rest 1
 14 ÷ 2 =   7 Rest 0
  7 ÷ 2 =   3 Rest 1
  3 ÷ 2 =   1 Rest 1
  1 ÷ 2 =   0 Rest 1
```

Or

A	B	Ausdruck (A||B)
1	1	1
1	0	1
0	1	1
0	0	0

AND

A	B	Ausdruck (A&B)
1	1	1
1	0	0
0	1	0
0	0	0

NAND

A	B	Ausdruck !(A&B)
1	1	0
1	0	1
0	1	1
0	0	1

XOR

A	B	Ausdruck A#B
1	1	0
1	0	1
0	1	1
0	0	0

Dezimal:

Division

```
7 0 : 5 = 1 4
5
2 0
2 0
0 0
```

Binär:

```
        1 0 0 0 1 1 0 : 1 0 1 = 1 1 1 0
          1 0 1
behalte 1 1 1
        0 0 1 1 1
            1 0 1
behalte
              1 0 1
              1 0 1
behalte
              0 0 0 0
                    0 0
                      0
```

Multiplikation

Dezimal:

```
        1 1 * 1 1
              1 1
            1 1
Behalte
            1 2 1
```

Binär:

```
        1 0 1 1 * 1 0 1 1
                  1 0 1 1
                1 0 1 1
              0 0 0 0
            1 0 1 1
Behalte       1 1 1 1
            1 1 1 1 0 0 1
```

Adition

Dezimal:

```
            9
            7
Behalte   1
          1 6
```

Binär:

```
          1 0 0 1
            1 1 1
Behalte 1 1 1 1
        1 0 0 0 0
```

Subtraktion

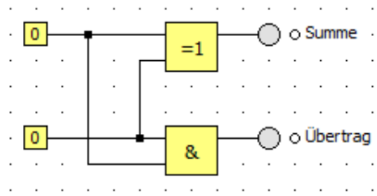
Dezimal:

```
          9
          7
Behalte
          2
```

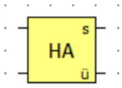
Binär:

```
        1 0 0 1
          1 1 1
Behalte 1 1
        0 0 1 0
```

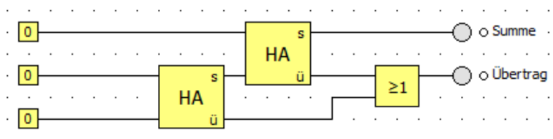
- Als erstes wird ein Addierer für einstellige Binärzahlen benötigt:



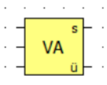
Diese Schaltung wird zusammengefasst als **Halbaddierer** bezeichnet.



Um mehrstellige Binärzahlen addieren zu können, benötigt man eine Schaltung, welche (analog zu der schriftlichen Binär-Addition) die Überträge der letzten Stelle mit einbezieht:



Diese Schaltung wird zusammengefasst als **Volladdierer** bezeichnet.



...gesetz		
Kommutativ-	A&B = B&A	A||B = B||A
Assoziativ-	(A&B)&C = A&(B&C)	(A||B)||C = A||(B||C)
Idempotenz-	A&A = A	A||A = A
Distributiv-	A&(B||C) = (A&B)||(A&C)	A||(B&C) = (A||B)&(A||C)
Neutralitäts-	A&1 = A	A||0 = A
Extremal-	A&0 = 0	A||1 = 1
Doppelnegation-	!(!A) = A	
De Morgan	!(A&B) = !A||!B	!(A||B) = !A&!B
Dualitäts-	!0 = 1	!1 = 0
Absorptions-	A||(A&B) = A	A&(A||B) = A