

# K1 Info 1.+2. Gruppe - 1. Stunde

Notiztitel

14.09.2012

Inhalt:

- Vorstellen

- Übersicht / Formatik K1 Themen

a) Theorie (Datenverarbeitung, Programmieren)

b) Gruppen / Java: Einstieg in eine  
Prog. Sprache

c) Report zum Interview

d) Projekt: wird nach a)-c) folgt.

- GFS

- allg. Stundenablauf
- Start!

## Codierung von Daten im PC

### **Wie werden Informationen übertragen? Wie macht das der PC?**

Das waren die zwei Einstiegsfragen in die erste Unterrichtseinheit. Angenommen, wir wollten eine Nachricht übermitteln, ohne direkt zu sprechen.

Ich möchte dir beispielsweise deine Note mitteilen. Dich interessiert aber erst einmal, ob du unter 5 Punkten stehst oder nicht, da du Info sowieso nicht abrechnen möchtest. Dann wäre eine einfache Möglichkeit, eine Taschenlampe zu nehmen und vorher folgendes zu vereinbaren:

- Taschenlampe, grünes Licht: 5 Punkte oder besser („Daumen hoch“)
- Taschenlampe, rotes Licht: unter 5 Punkten („Daumen runter“)

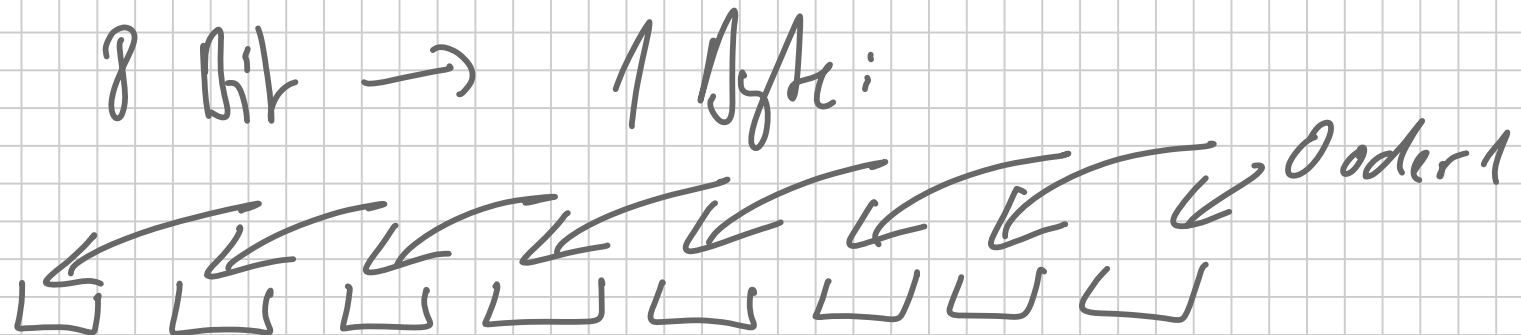
Der PC bietet eine ähnliche Möglichkeit. Über Leitungen kann man ein entsprechendes „binäres“ („Null oder Eins“; „An oder Aus“) Signal schicken. Es kann an einer festen Stelle im Rechner eine Spannung anliegen oder eben nicht. Damit ist es prinzipiell möglich, in einem Rechner Daten zu speichern. Allerdings nur 0-1-Daten (**Bits**) und noch keine komplizierten (bspw. formatierten) Texte!

Um dieses Problem zu umgehen, werden Pakete von diesen 0-1-Daten gesendet. In unserem anschaulichen Beispiel willst du vielleicht doch die genaue Note wissen. Wir einigen uns auf eine Folge von 4 Blinkern. Wenn wir alle Notenpunkte binär entwickeln, dann finden wir:

0 Notenpunkte =  $(0000)_2$  bzw. 1 Notenpunkt =  $(0001)_2$  bzw. 2 Notenpunkte =  $(0010)_2$  bzw.  
3 Notenpunkte =  $(0011)_2$  bzw. 4 Notenpunkte =  $(0100)_2$  bzw. 5 Notenpunkte =  $(0101)_2$  bzw.  
6 Notenpunkte =  $(0110)_2$  bzw. 7 Notenpunkte =  $(0111)_2$  bzw. 8 Notenpunkte =  $(1000)_2$  bzw.  
9 Notenpunkte =  $(1001)_2$  bzw. 10 Notenpunkte =  $(1010)_2$  bzw. 11 Notenpunkte =  $(1011)_2$  bzw.  
12 Notenpunkte =  $(1100)_2$  bzw. 13 Notenpunkte =  $(1101)_2$  bzw.  
14 Notenpunkte =  $(1110)_2$  bzw. 15 Notenpunkte =  $(1111)_2$ !

Grünes Licht ist eine 1, rotes Licht die Null. Wir blinken also für 11 Notenpunkte grün, rot, grün, grün. Übrigens könnte man auch einfach umgekehrt blinken, aber wir legen uns halt von links nach rechts fest.

Im PC geht's genauso, er bekommt jetzt Datenpakete zu 8 Bits geschickt, was einem **Byte** entspricht. Damit lassen sich bereits 256 Dinge (wegen  $2^8=256!$ ) kodieren.

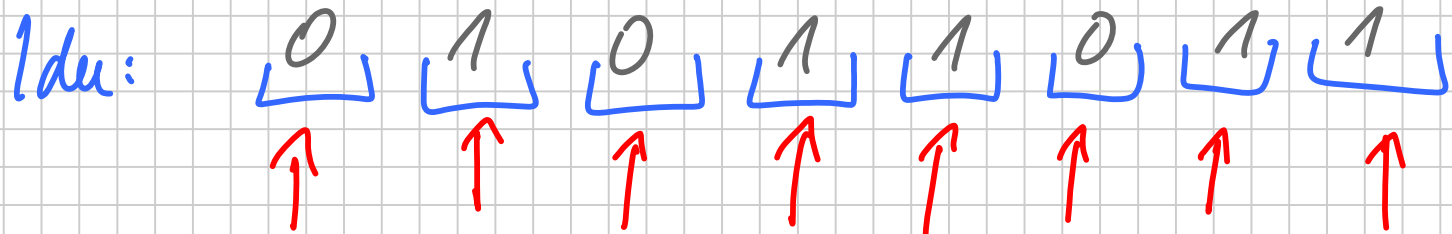


Damit lassen sich 256 versch. Zustände beschreiben:

0000	0000	$\hat{=}$	Zahl 0
0000	0001	$\hat{=}$	Zahl 1
0000	0010	$\hat{=}$	Zahl 2
	⋮		
1111	1111	$\hat{=}$	Zahl 255

Man kann diese Zustände direkt in Zahlen

umrechnen: Bsp. 01011011 ist der ?!?!  
Zustand ?!



Also...

$$64 + 16 + 8 + 2 + 1$$

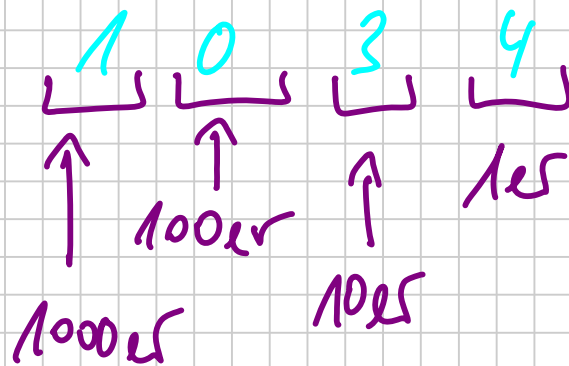
macht...

$$\underline{\underline{91}}$$



Wie so?! Bsp.  $2^3 = 8$ ,  $2^0 = 1$

Das ist so wie beim 10er-System...

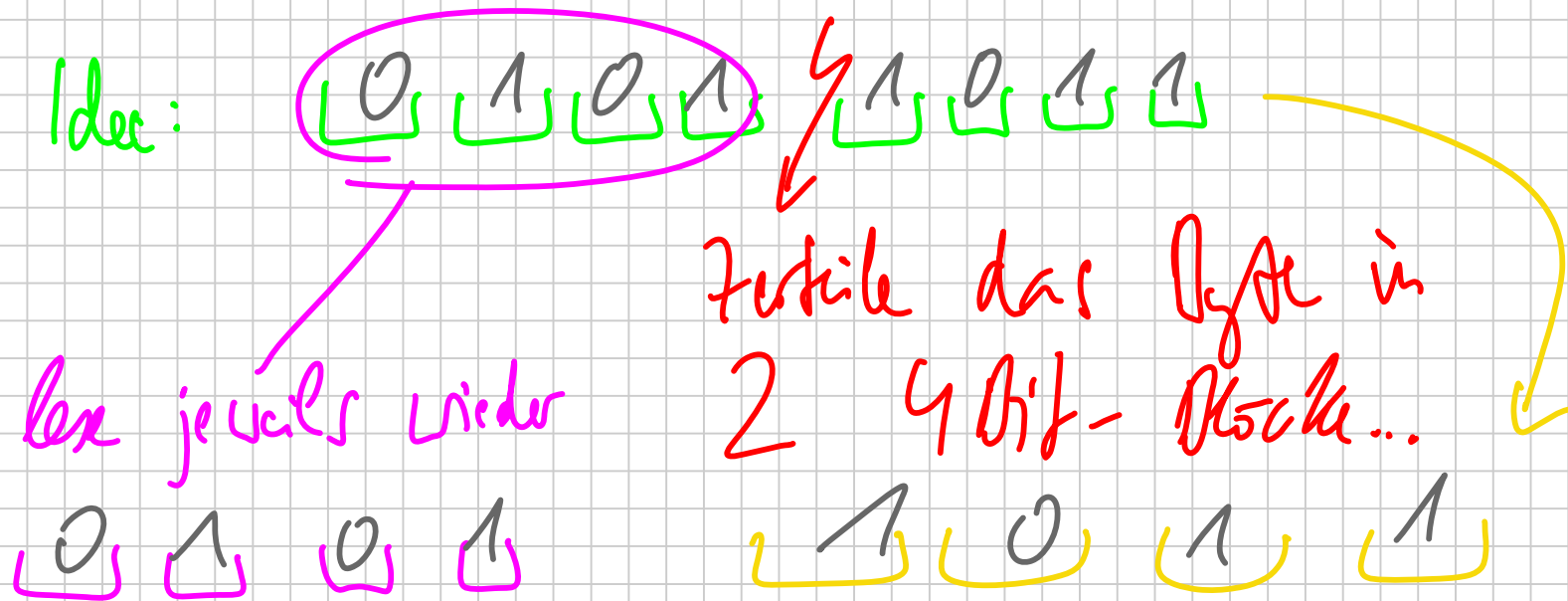


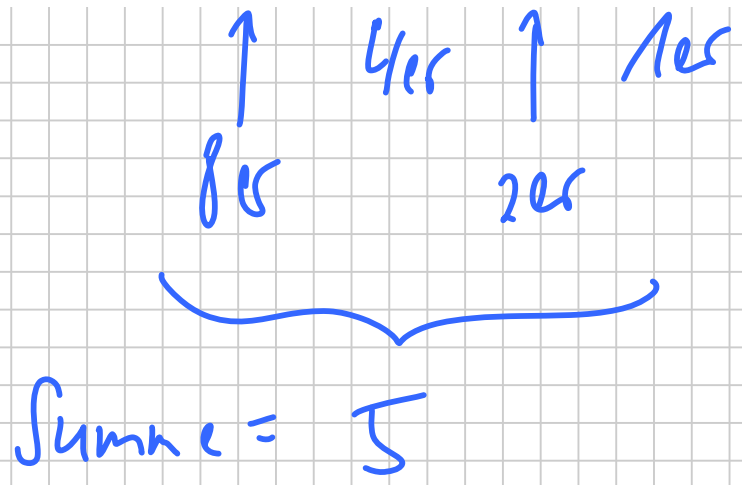
Mit dem ASCII hat man sich auf die Zeichen geeinigt, mit denen man möglichst einfach Texte kodieren kann; es gibt das Leerzeichen, das Komma, aber auch einen Tabulator usw.

Nur binär zu arbeiten, macht sehr lange Zeichenketten nötig. Man wollte das vermeiden und hat sich auf eine Darstellung im 16er-System geeinigt. Wichtig ist, dass 16 eine Zweierpotenz ist!

Diese Entwicklung ist eigentlich eine 2er-Entwicklung... nur kürzer. Da man 0 bis 15 verschieden oft jede Stelle (1er, 16er, 256er wg  $16^2=256$  usw.) besetzen kann, braucht man neben 0...9 auch noch A-F. Sie ersetzen 10-15.

In Farbkodierungen findet man oft #FF0066. Das ist eine **Hexadezimalcodierung** (in „Schlau“ für **16er-System-Codierung**). Ist nicht nötig, erleichtert aber den Alltag.





$$8 + 2 + 1 = 11$$

Nun ist das Hexadezimalsystem ( $16 = 2^4$ )  
 ein System, welches aus dem 2er-System  
 entsteht...

$$\begin{aligned}
 \underbrace{5}_{10} \quad \underbrace{11}_{10} &= 5 \cdot 16 + 11 \\
 \text{Aber } 1_{16} &= 10 + 1 \\
 \uparrow \text{ungewohnt} \text{ 😊} &= 51 \text{ passt auch!}
 \end{aligned}$$

Korrekte Notation:

$$(91)_{10} = (01011011)_2 = (5B)_{16}$$

„Eindeutigkeit“  
im sexagesimal Dezimalsystem

↑  
„B“ ersetzt 11  
(A-F  $\hat{=}$  10-15)

Die modernen PCs haben z.T. riesige Speicher von bspw. 160 GB (mein Netbook). Ein Byte sind 8 Bit, grob also 10 Bit. Giga ist Mega mal 1000, Mega ist eine Million. Giga ist also eine Milliarde. somit haben wir etwa 1600 Milliarden Bits zur Verfügung. Als Vergleich: Wir könnten eine Nachricht mit der Taschenlampe kodieren, für die wir 1600 Milliarden mal blinken müssen. Blinken wir einmal in der Sekunde, kannst du dir vorstellen, wie lange das dauert und die Batterien wären auch bald leer. Daher sind Computer so „mächtig“ und damit sehr nützlich.


Der nächste Schritt ist dann das Verarbeiten der Daten. Wie kann eine gespeicherte Nachricht elektronisch bearbeitet werden? Bzw. sind Zahlen kodiert, wie kann man sie verrechnen?! Auch hier geht alles unglaublich schnell mithilfe moderner Elektronik und der Darstellung in Bits



Wir haben noch über die Farbcodierung gesprochen:

### Farbcodierung

Farben werden mit der RGB-Codierung (red-green-blue) verschlüsselt. Es stehen ein Byte, also 256 Abstufungen für Rot, 256 für Grün und 256 für Blau zur Verfügung. Volle Farbanteile erzeugen weißes Licht, keine Anteile (alles auf 0) bedeutet Schwarz. So stehen über 16 Mio. Farben zur Auswahl ( $=256^3$ )!

Also: # 00 FF 00 ist reinstes ,

d.h.  $0 \hat{=}$  „aus“,  $FF \hat{=}$  „an“.