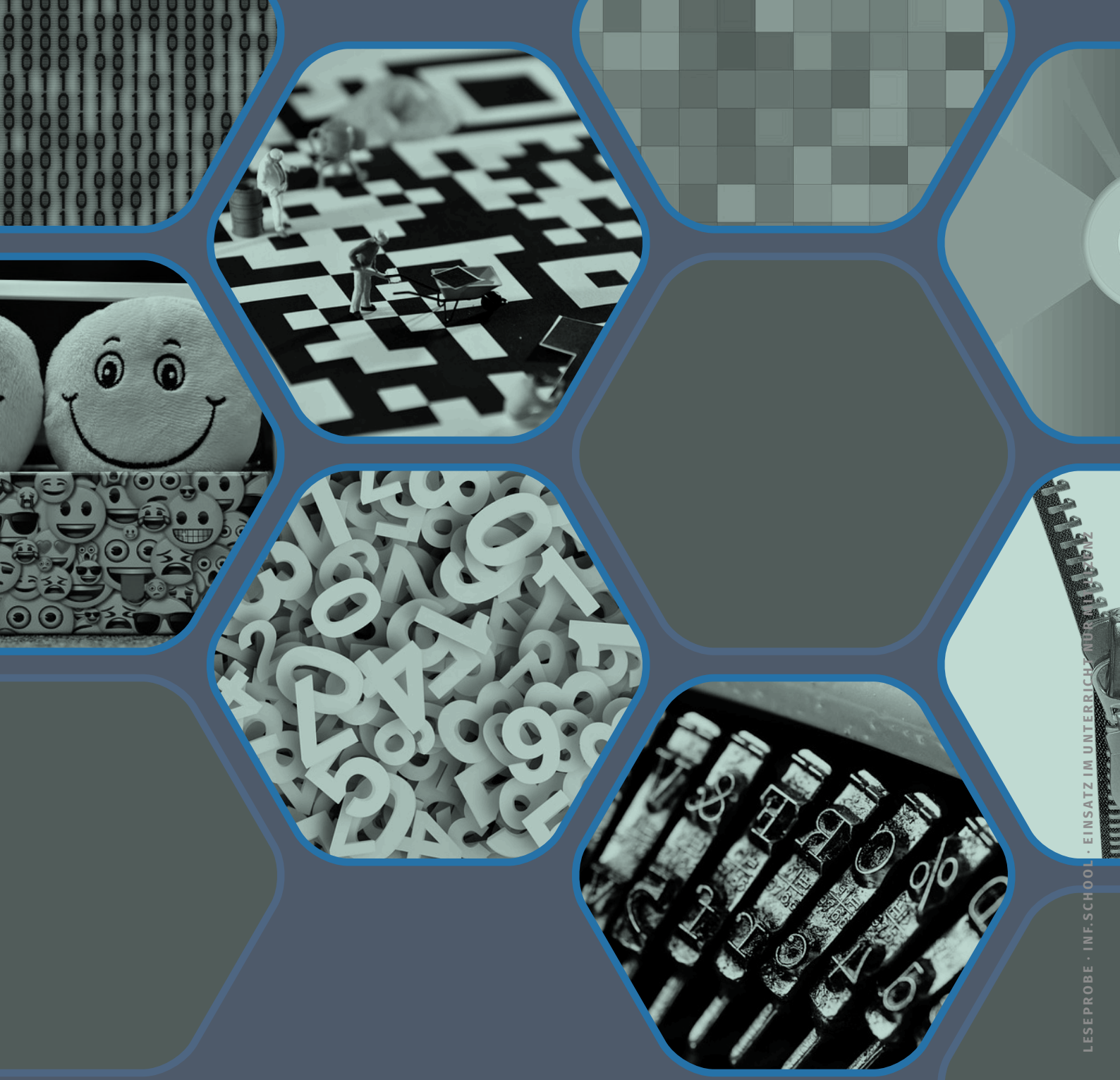


Digitale Daten

Darstellung · Kodierung · Komprimierung

FÜR DEN INFORMATIKUNTERRICHT AM GYMNASIUM

INF school

Ausgabe Oktober 2026

Florian Köhl

LESEPROBE · INF.SCHOOL · EINSATZ IM UNTERRICHT NUR MIT LICENZ

Inhaltsverzeichnis

1 **Information und Daten** 5
Daten in vielen Formen 7
Bits und Bytes - das Computeralphabet 8

2 **Zahlen darstellen** 11
Das Dezimalsystem 11
Das Fünfersystem – (nur) mit links 13
Positionale Zahlensysteme 15
Das Binärsystem 16
Vom Dezimal- ins Binärsystem umrechnen 19
Das Hexadezimalsystem 28

3 **Zeichenkodierungen** 35
Eindeutige Kodierungen 37
Ein erster Standard: ASCII 42
Das Unicode-'Wunder' 48
Das Unicode Transformation Format 50

4 **Digitale Bilder** 55
Pixel 55
Monochrome Bitmaps 58
Graustufen 65
Farbe ins Spiel bringen 70

5 **Textkomprimierung** 79
Präfixfreie Kodierungen 83
Die Huffman-Kodierung 88

LESEPROBE · INF.SCHOOL · EINSATZ IM UNTERRICHT NUR MIT LIZENZ

Über dieses Skript

Es freut mich sehr, dass du[1] dich entschieden hast, dieses Lehrmittel zu verwenden. Ich hoffe, dass es dich auf deinem Weg zur gymnasialen Matura begleiten und unterstützen wird, indem es dein Interesse am Fachgebiet Informatik weckt, dich zur selbstständigen Erkundung der angesprochenen Themengebiete motiviert, und dir die Grundlagen bietet, deinen Weg in der digitalen Zukunft zu beschreiten.

Der Ursprung dieses Dokuments ist während meiner Ausbildung zur Gymnasiallehrperson an der ETH Zürich entstanden, und wird laufend mit aktuellen Erkenntnissen erweitert. Es soll dir vermitteln, wie wir Informationen auf Computern abspeichern und abrufen können. Wir erforschen dies anhand drei verschiedener Datenarten: Zahlen, Text und Bilder.

Die Flughöhe versuche ich so hoch wie möglich, und so niedrig wie nötig zu halten. Ein intuitives Verständnis der Materie ist mir wichtig, genauso aber soll man wichtige Konzepte im Detail anschauen und beleuchten. Zudem sollen für interessierte SuS weiterführende Informationen für die eigenständige Vertiefung zur Verfügung gestellt werden.

Feedback

Egal ob du zu den Lehrpersonen oder zu den Lernenden gehörst: Wenn du Fehler oder Ungereimtheiten findest, Verbesserungsvorschläge hast, oder andere Ideen hast, wie man dieses Lehrmittel verbessern könnte, dann melde dich ungeniert bei mir! Ein kurzes Mail an feedback@koehlf.ch genügt. Davon lebt dieses Skript, dadurch wird es besser, und du kriegst nebenbei auch einen Eintrag in der *List of bug hunters*.

Danke!

Vielen Dank an Giovanni Serafini, welcher mich während meinem Lehrdiplomstudium an der ETH Zürich unterstützt und wertvollen Input geliefert hat, welche in diese Ausgabe eingeflossen ist. Weiter möchte ich allen Lehrpersonen und SchülerInnen einen grossen Dank aussprechen, die es durch ihr Feedback ermöglichen, dieses Skript ständig zu verbessern.

Ein grosses Dankeschön geht auch an die ganze LaTeX-Community auf StackExchange, ohne welche ich dieses Skript niemals in dieser Form hätte schreiben können.

Zu guter Letzt möchte ich mich bei meinen Eltern bedanken, die mich gross[2] gezogen haben, mich jederzeit in all meinen Zielen unterstützt haben und immer an mich geglaubt haben. Tusen takk! ♥

©2026 Florian Köhl

[1] bzw. deine Lehrperson
[2] sehr gross

LESEPROBE · INF.SCHOOL · EINSATZ IM UNTERRICHT NUR MIT LIZENZ

Trotz sorgfältiger Recherche und mehrmaligem Durchlesen schleichen sich natürlich auch in diesem Skript Fehler ein. Danke an alle aufmerksamen Schüler*innen und Lehrpersonen, die durch das Melden von Fehlern oder Ungereimtheiten helfen, dieses Skript zu verbessern!

- Alexia Tuena
- Hugo Vetter
- Joel Spiess
- Leana Troller
- Livio Abegg
- Rayen Ajlani
- Seung Hee Ma
- Jorin Schaffner, Serfin Dohner & Nicola Dietiker

LESEPROBE · INF.SCHOOL · EINSATZ IM UNTERRICHT NUR MIT LIZENZ

2

Zahlen darstellen

There are only 10 kinds of people in the world.
Those who understand binary and those who don't.

Das Dezimalsystem

Das **Dezimalsystem** kennst du bestimmt. Eine Zahl, welches in diesem Zahlensystem geschrieben ist, setzt sich – von rechts nach links gelesen – aus Einern, Zehnern, Hundertern, Tausendern usw. zusammen. Jede dieser Stellen ist durch eine Ziffer von 0 bis 9 beschrieben. Um den Wert der Zahl zu bestimmen, rechnen wir jeweils die Anzahl an einer Stelle mal deren Wertigkeit, und addieren diese Zwischenergebnisse dann zusammen.

BEISPIEL: DEZIMALSYSTEM

Betrachte die Dezimalzahl 3172. Sie besteht aus 2 Einern, 7 Zehnern, einem Hunderter und 3 Tausendern. Der Wert ergibt sich also als

$$3 \cdot 1000 + 1 \cdot 100 + 7 \cdot 10 + 2 \cdot 1 = 3172$$

Im Zentrum steht beim Dezimalsystems die Zahl 10. Sie bestimmt einerseits, **wie viele Ziffern** wir pro Stelle zur Verfügung haben. Andererseits bestimmt sie auch die Wertigkeit der einzelnen Stellen. Diese sind nämlich durch **Potenzen der Zahl** 10 gegeben. So hat die Stelle ganz rechts im Dezimalsystem die Wertigkeit 1, oder anders ausgedrückt 10^0. Die zweite Stelle – also die Zehner – haben die Wertigkeit 10^1, während die dritte Stelle (die Hunderter) die Wertigkeit 10^2 haben. Jede Stelle ist also 10 mal mehr 'Wert' als die Stelle zu ihrer Rechten.

LESEPROBE · INF.SCHOOL · EINSATZ IM UNTERRICHT NUR MIT LIZENZ

Allgemein können wir eine vierstellige Dezimalzahl als $d_3 d_2 d_1 d_0$ schreiben. d_0 bezeichnet die Stelle ganz rechts, d_1 die zweite Stelle von rechts, usw. Der Wert einer solchen Zahl ist dann also

$$d_3 \cdot 1000 + d_2 \cdot 100 + d_1 \cdot 10 + d_0 \cdot 1 = d_3 \cdot 10^3 + d_2 \cdot 10^2 + d_1 \cdot 10^1 + d_0 \cdot 10^0$$

Im Beispiel oben haben wir also $d_3 = 3$, $d_2 = 1$, $d_1 = 7$ und $d_0 = 2$. Wenn wir diese Zahlen in die Formel oben einsetzen, erhalten wir wieder das gleiche Resultat.

Wir können dies auch allgemeiner formulieren. Der Wert einer n-stelligen Dezimalzahl, welche durch ihre Stellen $d_{n-1} d_{n-2} \ldots d_2 d_1 d_0$ gegeben ist, können wir als

$$\sum_{i=0}^{n-1} d_i \cdot 10^i$$

schreiben. Das sieht auf den ersten Blick abschreckend aus, entspricht aber genau dem, was wir oben beschrieben haben. d_0 steht für die Anzahl Einer. Diese werden mit 10^0, also 1 multipliziert. Dazu zählen wir die Anzahl Zehner (d_1), die wir zuerst noch mit $10^1 = 10$ multiplizieren. Danach wir die Hunderter (d_2 multipliziert mit $10^2 = 100$) usw.

LESEPROBE · INF.SCHOOL · EINSATZ IM UNTERRICHT NUR MIT LIZENZ

Das Fünfersystem – (nur) mit links

Wieso zählen wir heutzutage eigentlich (meist) in der Basis 10? Die Antwort, die man darauf am häufigsten findet, hat mit der menschlichen Anatomie zu tun. Wir haben 10 Finger, und diese Finger werden zum (Auf)Zählen verwendet. Was wäre jedoch, wenn wir nur mit einer Hand zählen würden? Also statt dem Dezimalsystem (Zehnersystem) ein Fünfersystem verwenden würden? Wir wollen das im Folgenden kurz untersuchen.

Das Fünfersystem verwendet im Gegensatz zum Dezimalsystem als **Basis** die Zahl **5**. Das wirkt sich sowohl auf die Anzahl unterschiedlicher Ziffern oder Zeichen aus, die wir nützen können, beeinflusst aber auch die Werte der einzelnen Stellen.

Der erste Unterschied ist, dass wir im Fünfersystem nur 5 Ziffern zur Verfügung haben: 0, 1, 2. 3 und 4. Die ersten 5 Zahlen (wenn wir von 0 an beginnen) sind also stellen wir also gleich dar wie im Dezimalsystem. Nach der 4 kommt im Fünfersystem allerdings die 10: Da wir keine weiteren Ziffern mehr haben, müssen wir mit der nächsten Stelle arbeiten. Die zweite Stelle (von rechts) bezeichnet die *Fünfer*, oder anders ausgedrückt die Basis hoch eins (5^1). Die nächste Stelle steht dann für die 25er, also die Basis im Quadrat, gefolgt von den 125ern – die Basis hoch drei (5^3).

★ **AUFGABE 4:** FÜNFERSYSTEM

(a) Stelle die Dezimalzahl 32 im Fünfersystem dar. Überlege dir dafür, wie viele Einer, Fünfer und 25er du dafür benötigst bzw. zur Verfügung hast.

(b) Welcher Dezimalzahl entspricht die Fünferzahl 123?

LESEPROBE · INF.SCHOOL · EINSATZ IM UNTERRICHT NUR MIT LIZENZ

★★ AUFGABE 5: FÜNFERPOTENZEN SUMMIEREN

Schaffst du es, den Wert einer im Fünfersystem geschriebenen n-stelligen Zahl $d_{n-1}d_{n-2}\ldots d_2d_1d_0$ mathematisch mit Hilfe des Summenzeichens $\sum$ aufzuschreiben?

LESEPROBE · INF.SCHOOL · EINSATZ IM UNTERRICHT NUR MIT LIZENZ

Positionale Zahlensysteme

Das Dezimalsystem ist ein Spezialfall der allgemeineren Notation eines **positionalen Zahlensystems**. Positional bedeutet hier, dass die Position der Ziffer deren Wertigkeit beeinflusst. Eine Ziffer d an der letzten Stelle ist *weniger Wert* als d an der zweitletzten Stelle.

Ein positionales System definiert eine **Basis**. Diese Basis bestimmt einerseits, **wie viele Ziffern** wir pro Stelle zur Verfügung haben. Andererseits bestimmt sie auch die **Wertigkeit** der einzelnen Stellen. Ein Zahlensystem mit Basis B hat im Allgemeinen B verschiedene Ziffern, nämlich $0, 1, 2 \ldots B-1$. Die Wertigkeit der verschiedenen Stellen hängt ebenfalls von B ab. Die letzte Stelle hat die Wertigkeit B^0, also 1, die zweitletzte B^1, gefolgt von B^2 usw.

Wenn wir also den Wert einer solchen Zahl $d_{n-1}d_{n-2}\ldots d_2d_1d_0$ bestimmen möchten, rechnen wir

$$\sum_{i=0}^{n-1} d_i \cdot B^i$$

Setzen wir statt B die Zahl 10 ein, so erhalten wir wieder die bekannte Formel für das Dezimalsystem, wenn wir $B = 5$ wählen, so entsteht das Fünfersystem.

WEITERE ZAHLENSYSTEME

Neben dem bei uns dominanten Dezimalsystem finden oder fanden diverse andere Zahlensysteme Verwendung. Das **römische Zahlensystem** beispielsweise ist kein klassiches positionales System. Es besteht aus den Ziffern `I` (Wert 1), `V` (5), `X` (10), `L` (50), `C` (100), `D` (500) und `M` (1000). Eine Zahl wird ganz einfach als Summe dieser Ziffern gebildet, wobei man die grössten zu Beginn nennt. 15 wird also beispielsweise als `XV` geschrieben. Eine kleine Ausnahme gibt es aber: Steht eine kleinere Ziffer *vor* einer Ziffer mit grösserem Wert, so wird sie von letzterer abgezogen. Die Zahl 4 wird also als `IV` dargestellt, `CD` bezeichnet die Dezimalzahl 400.
Noch vor dem römisches System dominierte im mesopotamischen Reich das **babylonische System**. Dieses war eines der ersten positionalen Systeme und basierte auf der Zahl 60. Relikte davon finden wir heute noch im Alltag: Eine Stunde unterteilen wir beispielsweise in 60 Minuten.

Manche Leute plädieren gar für die Adaptierung des Zwölfersystems:

LESEPROBE · INF.SCHOOL · EINSATZ IM UNTERRICHT NUR MIT LIZENZ

3 Zeichenkodierungen

Binary (He)art

Quelle: xkcd, `https://xkcd.com/99/`

Nachdem wir nun wissen, wie wir (ganze) Zahlen als Folge von Nullen und Einsen darstellen können, wollen wir in diesem Abschnitt erforschen, wie wir damit auch Text repräsentieren können.

Wir lernen dabei, auf was wir achten müssen, wenn wir Texte auf diese Weise ausdrücken möchten, und welche Lösungen sich im Laufe der Zeit dafür etabliert haben.

Bevor wir loslegen, wollen wir noch kurz definieren, was wir denn unter einem Text verstehen.

LESEPROBE · INF.SCHOOL · EINSATZ IM UNTERRICHT NUR MIT LIZENZ

TEXTE UND ALPHABETE

Unter einem **Alphabet** verstehen wir eine endliche Menge von Symbolen. Beispiele für Alphabete sind die lateinischen Grossbuchstaben {A, B, C, ..., X, Y, Z } oder das Computeralphabet, welches nur aus zwei Symbolen besteht: {0, 1 }
Ein **Text** ist nichts anderes als eine Folge von Symbolen aus einem bestimmten Alphabet. Wir benutzen der Abwechslung halber auch oft die Begriffe Botschaft oder Nachricht.

Da unsere Texte nicht nur aus Buchstaben, sondern auch Satzzeichen oder Sonderzeichen bestehen, verwenden wir in diesem Abschnitt den allgemeineren Ausdruck **Zeichen**, die all diese Kategorien umfasst.

DENKANSTOSS: ZEICHEN ABBILDEN

Wie würdest du Zeichen abbilden, damit wir sie auf einem Computer speichern können?

Eine der wohl intuitivsten Ideen, um das vorliegende Problem zu lösen, ist, einfach jedem Zeichen eine Zahl zuzuweisen. Dem Buchstaben A können wir beispielsweise die Zahl 1 zuordnen, B weisen wir die Zahl 2 zu usw.

Eine solche Abbildung von Zeichen zu Zahlen und umgekehrt ist ein Beispiel einer Kodierung, oder kürzer Code.

KODIERUNG

Eine **Kodierung**, auch **Code** genannt, ist eine injektive Abbildung[1] zwischen zwei Alphabeten $\mathcal{A}$ und $\mathcal{B}$, die jedem Symbol aus $\mathcal{A}$ ein Symbol aus $\mathcal{B}$ zuordnet.
Handelt es sich bei $\mathcal{A}$ um Zeichen, so sprechen wir von einem **Zeichencode**.

Wenn wir also Zeichen mittels einem solchen Code in beispielsweise Zahlen umwandeln, so nennen wir diesen Vorgang **Kodieren**. Den umgekehrten Weg, also das Umwandeln von Zahlen in Zeichen, bezeichnen wir als **Dekodieren**.

Ein einzelnes kodiertes Zeichen nennen wir **Code-Wort**. Beim Dekodieren können wir die empfangene Nachricht also Code-Wort für Code-Wort durchgehen und zurückübersetzen.

Die wohl naheliegendste Art, Zeichen zu kodieren, ist durch eine simple Aufzählung: Wir nummerieren alle Zeichen einfach durch. Da wir diese Zeichencodes danach auf einem Computer speichern möchten, nutzen wir dafür das Binärsystem. Der Einfachheit halber beschränken wir uns hier auf die Grossbuchstaben des lateinischen Alphabets.

Das Code-Wort für den Buchstaben A ist hier also `0`, jenes für das Zeichen L lautet `1011`. Mit dieser Kodierungstabelle würden wir also HANS als `111 0 1101 10010` **kodieren**. Analog können wir die Zahlenfolge `1100 10100 10010 10011 100 10001` **dekodieren** und erhalten MUSTER.

[1] Eine Abbildung heisst injektiv, wenn es zu jedem Element der Zielmenge entweder genau ein oder gar kein Element aus der Ursprungsmenge gibt. Für eine injektive Funktion f gilt also: Falls $f(x) = f(y)$, so ist $x = y$.

LESEPROBE · INF.SCHOOL · EINSATZ IM UNTERRICHT NUR MIT LIZENZ

0	A	101	F	1010	K	1111	P	10100	U	11001	Z
1	B	110	G	1011	L	10000	Q	10101	V		
10	C	111	H	1100	M	10001	R	10110	W		
11	D	1000	I	1101	N	10010	S	10111	X		
100	E	1001	J	1110	O	10011	T	11000	Y		

Tabelle 3.1: Einfache Kodierungstabelle
Die Buchstaben (rechte Spalte) und die entsprechenden Code-Wörter (linke Spalte)

Das funktioniert soweit ja ganz gut, wenn wir selbst für uns ein paar Zeichen kodieren. Was geschieht aber, wenn wir diese kodierten Zeichen mit anderen Personen austauschen möchten? Beispielsweise in Form einer Chat-Nachricht, oder einem Email? Dann können wir nur hoffen, dass der Empfänger der Nachricht die gleiche Kodierungstabelle wie wir verwendet, anonsten wird seine dekodierte Nachricht nicht viel Sinn ergeben! Hätten wir beispielsweise nicht bei `0`, sondern bei `1` angefangen zu zählen, würden wir die obige Zahlenfolge `1100 10100 10010 10011 100 10001` plötzlich als NVTUFS dekodieren und würden uns fragen, was der Sender denn damit meint.

Beim Austauschen von kodierten Nachrichten (beispielsweise über das Internet) ist es also wichtig, dass der Sender und der Empfänger der Nachricht den gleichen Code verwenden! Ansonsten kann es sehr schnell zu Missverständnissen kommen!

DENKANSTOSS: VERWECHSLUNGSGEFAHR?

Siehst du ein weiteres Problem bei der eben vorgestellten Kodierung? Kannst du dieses anhand eines Beispiels verdeutlichen?

Eindeutige Kodierungen

Ein Problem, das wir elegant umschifft haben, ist das Erkennen von einzelnen Zeichen innerhalb einer Bitfolge. In den Beispielen vorher haben wir zwischen Zeichen jeweiles einen Leerschlag gesetzt. Für uns Menschen scheint dies absolut logisch und intuitiv, Computer stellt dies allerdings vor ein Problem: diese kennen nämlich auf dieser logischen Ebene keine Leerzeichen! Alles, was sie kennen, sind Nullen und Einsen – wir müssen also ohne Trennzeichen wie etwa Leerschläge auskommen! Dass dies etwas mehr als ein simples Löschen Letzterer benötigt, soll folgendes Beispiel aufzeigen:

BEISPIEL: MIXED (UP) SIGNALS

Alice möchte Bob gerne eine Nachricht übermitteln. Als Kodierung verwendet sie diejenige aus Tabelle 3.1, welche Bob ebenfalls zur Hand hat. Was Alice Bob gerne endlich mal sagen möchte, ist `ILOVEYOU`.

LESEPROBE · INF.SCHOOL · EINSATZ IM UNTERRICHT NUR MIT LIZENZ

Sie übersetzt also alle Zeichen einzelnen und fügt diese anschliessend zu einem Bitstring – der zu übermittelnden Nachricht – zusammen. Das I wird zu 1000, das L zu 1011 und so weiter. Schlussendlich erhält sie also die Nachricht

1000101111101010110011000111010100

und übermittelt diese an Bob. Dieser freut sich extrem, eine Nachricht von Alice bekommen zu haben. Nur hat er seine liebe Mühe, die Botschaft zu entschlüsseln. Schon beim ersten Buchstaben gerät er ins Stocken…wo endet denn dieser? Handelt es sich um ein B, gefolgt von drei A? Oder ein C gefolgt von zwei A? Oder ein E gefolgt von nur einem A? Oder doch ein I? Oder etwa ein R? Alle diese Varianten wären möglich, je nach dem wo man sich die Leerschläge "denkt":

```
1 0 0 0 1...
10 0 0 1...
100 0 1...
1000 1...
10001...
```

Dass er die richtige Kombination von Buchstaben trifft, ist extrem unwahrscheinlich – es gibt einfach zu viele davon! Eine Mögliche wäre beispielsweise

```
10001 0 1111 101 0 101 100 11000 111 0 10 100
```

und das bedeutet so viel wie RAPFAFEYHACE. Bob wird daraus allerdings nicht schlau. Da er nicht alle Möglichkeiten durchprobieren möchte, gibt er enttäuscht auf.

Wie das letzte Beispiel zeigt, ist eine Nachricht, die mittels dem Zeichencode aus Tabelle 3.1 kodiert wurde, **nicht eindeutig dekodierbar**. Eine Nachricht kann also vom Empfänger anders interpretiert werden, als dies vom Sender gedacht war!

Zeichencodes, bei denen dieses Problem nicht auftritt, nennen wir demzufolge **eindeutig dekodierbar**, und wollen uns als Nächstes anschauen, wie wir unseren naiven Zeichencode in einen solchen umwandeln können.

DENKANSTOSS: ZEICHENCODES EINDEUTIG MACHEN

Was für Möglichkeiten siehst du, den Zeichencode aus Tabelle 3.1 eindeutig zu machen?

Eine Art von Zeichencodes, die zu den eindeutig dekodierbaren gehören, sind die so genannten **Zeichencodes mit fixer Länge** (engl. fixed length encodings). Bei diesen hat jedes Code-Wort eine fixe Länge, und diese Länge ist für alle Code-Wörter gleich. Das Dekodieren ist danach ganz einfach. Wenn wir wissen, wie lange die einzelnen Code-Wörter sind, so können wir diese ganz einfach identifizieren und übersetzen.

Wir wollen unseren naiven Zeichencode aus Tabelle 3.1 in einen solchen Zeichencode fixer Länge umwandeln. Dafür füllen wir alle Zahlen einfach von links mit Nullen auf, bis alle Code-Wörter dieselbe Länge haben. Dies resultiert in folgender Tabelle:

LESEPROBE · INF.SCHOOL · EINSATZ IM UNTERRICHT NUR MIT LIZENZ

00000	A	00101	F	01010	K	01111	P	10100	U	11001	Z
00001	B	00110	G	01011	L	10000	Q	10101	V		
00010	C	00111	H	01100	M	10001	R	10110	W		
00011	D	01000	I	01101	N	10010	S	10111	X		
00100	E	01001	J	01110	O	10011	T	11000	Y		

Tabelle 3.2: Eindeutige Kodierung mit fixer Länge
Die Buchstaben (rechte Spalte) und die entsprechenden Code-Wörter (linke Spalte)

Mit diesem Zeichencode wird also HANS als 00111000000110110010 kodiert - die vier Zeichen werden je einem Code-Wort mit einer Länge von 5 Bits zugeordnet, die gesamte Nachricht ist also 20 Bits lang.

Nun können wir auch das zweite Beispiel eindeutig dekodieren! Bei der Nachricht 011001010010010100110010010001 wissen wir, dass jedes Zeichen jeweils mit 5 Bits kodiert wurde und können diese also eindeutig unterscheiden:
01100 10100 10010 10011 00100 10001 wird folglich zu MUSTER.

AUFGABE 21: LIEBESBOTSCHAFTER

Kannst du Alice helfen, ihre Botschaft aus obigem Beispiel (ILOVEYOU) mittels dem Zeichencode aus Tabelle 3.2 so zu kodieren, dass Bob die Nachricht eindeutig dekodieren kann?

LESEPROBE · INF.SCHOOL · EINSATZ IM UNTERRICHT NUR MIT LIZENZ

4 Digitale Bilder

In diesem Kapitel befassen wir uns mit digitalen Bildern. Woraus bestehen diese, und wie werden diese auf einem Computer gespeichert? Und wie landen sie schliesslich farbig ausgedruckt auf einem Blatt Papier, oder digital auf deinem Monitor?

Pixel

Im Allgemeinen unterscheiden wir zwei Arten von digitalen Bildern: **Vektorgrafiken** und **Rasterbilder**. Bei ersteren wird der Inhalt einer Grafik durch geometrische Figuren wie Kreise, Polygone oder Pfade beschrieben und wird häufig für Logos oder Diagramme verwendet. Wir beschränken uns hier aber auf Rasterbilder.

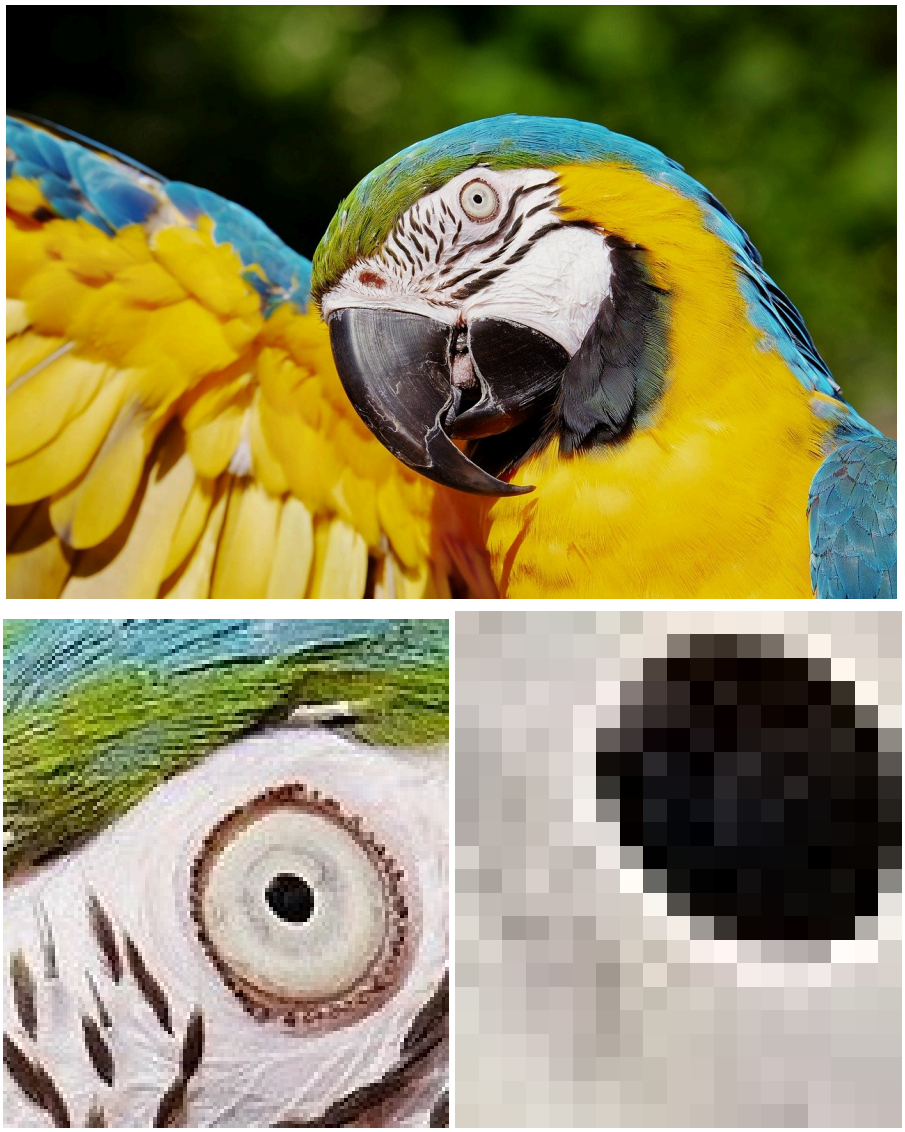
Digitale Rasterbilder bestehen aus (sehr) vielen Bildpunkten, so genannten Pixeln, die rasterförmig angeordnet sind und denen eine bestimmte Farbe zugeordnet ist.

PIXEL UND AUFLÖSUNG

Ein **Pixel** (von engl. picture element) gibt die Farbinformation eines Bildes an einer bestimmten Stelle wieder.
Die Anzahl Pixel in einem Bild nennt man **Auflösung** eines Bildes.

Die Auflösung geben wir oft auch als **Breite x Höhe** an. Je höher die Auflösung, desto schärfer erscheint das Bild, da mit mehr Pixel feinere Konturen und Details abgebildet werden können.

LESEPROBE · INF.SCHOOL · EINSATZ IM UNTERRICHT NUR MIT LIZENZ



Ein Ara Papagei.
Oben: Gesamtansicht
Unten: vergrösserter bzw. stark vergrösserter Ausschnitt der Augenpartie

BEISPIEL: AUFLÖSUNG

Ein Bild, welches 5000 Pixel breit und 2000 Pixel hoch ist, hat eine Auflösung von 5000 x 2000 Pixel, oder anders ausgedrückt 10 Megapixel (10 Millionen Pixel).

LESEPROBE · INF.SCHOOL · EINSATZ IM UNTERRICHT NUR MIT LIZENZ

AUFGABE 30: AUFLÖSUNG BERECHNEN

Ein Bild hat eine Höhe von 800 und eine Breite von 600 Pixel. Berechne die Auflösung.

AUFGABE 31: HÖHE BERECHNEN

Ein Bild hat eine Auflösung von 800 Pixel. Du weisst, dass es 40 Pixel breit ist. Wie hoch ist es?

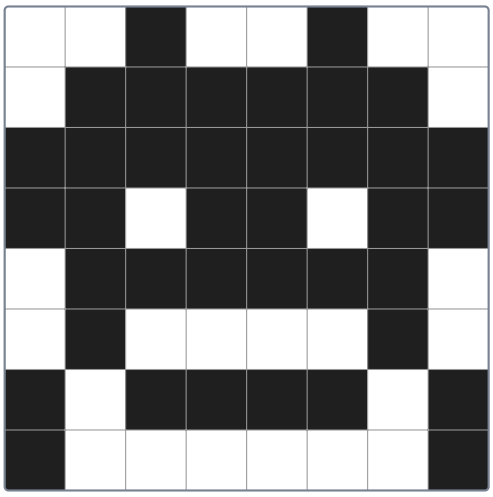
AUFGABE 32: BREITE BERECHNEN

Ein Bild hat eine Höhe von 8000 Pixel. Die Auflösung des Bildes beträgt 48 MegaPixel. Wie breit ist das Bild?

LESEPROBE · INF.SCHOOL · EINSATZ IM UNTERRICHT NUR MIT LIZENZ

Monochrome Bitmaps

Die erste Art von Rasterbilder, die wir genauer anschauen wollen, sind **monochrome Bitmaps**. Der Begriff *monochrom* kommt aus dem Griechischen und bedeutet *einfarbig*. Wir verstehen darunter Rasterbilder, die nur aus Abstufungen einer Farbe bestehen: Schwarz und Weiss. Betrachte dazu das folgende Beispielbild eines kleinen Pixelmonsters:



Das Bild hat eine Auflösung von 8 x 8 Pixel, und jedes Pixel ist entweder schwarz oder weiss. Wie können wir dieses Bild in binärer Form auf dem Computer darstellen?

Der erste Schritt, den wir dafür unternehmen, ist folgender: Wir repräsentieren jedes Pixel durch 1 Bit. Hat dieses den Wert 1, so ist das Pixel weiss (*Licht an*), bei einer 0 ist das Pixel schwarz (*Licht aus*):

1	1	0	1	1	0	1	1
1	0	0	0	0	0	0	1
0	0	0	0	0	0	0	0
0	0	1	0	0	1	0	0
1	0	0	0	0	0	0	1
1	0	1	1	1	1	0	1
0	1	0	0	0	0	1	0
0	1	1	1	1	1	1	0

Leider ist der Arbeitsspeicher eines Computers nicht zweidimensional aufgebaut, wir müssen also diese Pixel als eine eindimensionale Folge von Bits darstellen können. Der naheliegendste

LESEPROBE · INF.SCHOOL · EINSATZ IM UNTERRICHT NUR MIT LIZENZ

Ansatz ist, diese einfach Reihe für Reihe aneinander zu knüpfen. Wir beginnen dafür mit der ersten Reihe:

1	1	0	1	1	0	1	1
1	0	0	0	0	0	0	1
0	0	0	0	0	0	0	0
0	0	1	0	0	1	0	0
1	0	0	0	0	0	0	1
1	0	1	1	1	1	0	1
0	1	0	0	0	0	1	0
0	1	1	1	1	1	1	0

1	1	0	1	1	0	1	1																										

Dahinter fügen wir nun die zweite Reihe an:

1	1	0	1	1	0	1	1
1	0	0	0	0	0	0	1
0	0	0	0	0	0	0	0
0	0	1	0	0	1	0	0
1	0	0	0	0	0	0	1
1	0	1	1	1	1	0	1
0	1	0	0	0	0	1	0
0	1	1	1	1	1	1	0

1	1	0	1	1	0	1	1	1	0	0	0	0	0	0	1																		

LESEPROBE · INF.SCHOOL · EINSATZ IM UNTERRICHT NUR MIT LIZENZ

Und nun die dritte Zeile:

1	1	0	1	1	0	1	1
1	0	0	0	0	0	0	1
0	0	0	0	0	0	0	0
0	0	1	0	0	1	0	0
1	0	0	0	0	0	0	1
1	0	1	1	1	1	0	1
0	1	0	0	0	0	1	0
0	1	1	1	1	1	1	0

1	1	0	1	1	0	1	1	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0								

Du ahnst sicher, wie es weiter geht. Schlussendlich erhalten wir die folgende Darstellung:

1	1	0	1	1	0	1	1	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0
1	0	0	0	0	0	0	1	1	0	1	1	1	1	0	1	0	1	0	0	0	0	1	0	0	1	1	1	1	1	1	0

Sind wir nun fertig? Leider noch nicht ganz... Um dies zu verdeutlichen, versuchen wir, eine andere solche Bitfolge rückwärts, also von der eindimensionalen in die zweidimensionale Darstellung zu übersetzen:

1	1	1	0	0	1	1	1	1	1	0	0	0	0	1	1	1	0	0	0	0	0	0	1	0	0	1	0	0	1	0	0
0	0	0	0	0	0	0	0	1	1	0	0	0	0	1	1	1	0	0	0	0	0	0	1	0	1	0	1	1	0	1	0

Was fehlt uns? Genau, wir wissen gar nicht, welche Dimensionen das gewünschte Bild hat. Alles, was wir wissen, ist, dass das Bild aus 64 Pixel besteht. Sind es 8x8 Pixel? 4x16 Pixel? 2x32 Pixel? 1x64 Pixel? Oder umgekehrt? Wir wissen es nicht. Diese Information müssen wir also ebenfalls mitkodieren! Wir tun dies in einem so genannten **Header**. Dieser beinhaltet Metadaten des Bildes, wie beispielsweise eben Breite und Höhe. In der Praxis werden noch viele weitere Eigenschaften angegeben, wir beschränken uns hier der Einfachheit halber auf diese zwei. Wir definieren unseren Header also als Bitfolge der Länge 16, wobei die ersten 8 Bits die Breite und die zweiten 8 Bits die Länge repräsentieren.

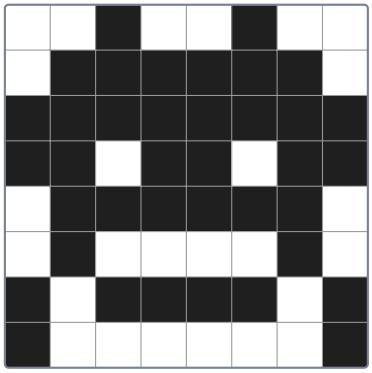
LESEPROBE · INF.SCHOOL · EINSATZ IM UNTERRICHT NUR MIT LIZENZ

HEADER UND METADATEN

Im Header einer Datei sind meist so genannte Metadaten abgelegt, also Informationen über die Datei selbst. Bei Bildern sind dies beispielsweise die Auflösung, die Farbtiefe oder das verwendete Farbmodell, Informationen zu einer allfälligen Komprimierung, uvm.

Bei einigen Dateitypen (wie zum Beispiel bei Bitmaps mit der Dateiendung .bmp) werden diese Metainformationen direkt im Header gespeichert, bei anderen (wie beispielsweise .png oder .jpg) werden sie in so genannten Chunks oder Frames nach dem eigentlichen Header aufgelistet. Dies erlaubt gute Vorwärtskompatibilität und das Auslassen von unnötigen oder nicht relevanten Informationsblöcken.

In unserem Beispiel ist sowohl die Höhe als auch die Breite 8 Pixel, die Binärdarstellung dafür lautet `00001000`. Als Header schreiben wir deshalb `00001000 00001000`. Nach dem Header folgt der **Payload**, also die Folge von Pixeldaten. Unser Pixelmonster



wird also wie folgt repräsentiert:

HEADER

0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0

BREITE HÖHE

PAYLOAD

1	1	0	1	1	0	1	1	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0
1	0	0	0	0	0	0	1	1	0	1	1	1	1	0	1	0	1	0	0	0	0	1	0	0	1	1	1	1	1	1	0

Mit dieser Information können wir also folgende Bitfolge als Bild interpretieren:

0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0

1	1	1	0	0	1	1	1	1	1	0	0	0	0	1	1	1	0	0	0	0	0	0	1	0	0	1	0	0	1	0	0

0	0	0	0	0	0	0	0	1	1	0	0	0	0	1	1	1	0	0	0	0	0	0	1	0	1	0	1	1	0	1	0

LESEPROBE · INF.SCHOOL · EINSATZ IM UNTERRICHT NUR MIT LIZENZ