Lernziele LB Modul 114

Inhalt

Prüfungsinformationen

- Der Test auf dem Computer, nicht schriftlich
- etwa 10 Aufgaben beim Test
- 90min Zeit hat man für die Prüfung
- Zu verwendende Programme: Windows 10, Ubuntu, Microsoft Office, Gpg4win
- doppelt A4-Prüfungsspick

Zahlensysteme

Positionsbasierte und nicht Positionsbasierte

Die Position links/rechts gibt an, das eine Zahl ein vielfaches mehr «zählt», je nach Stelle. Mit 847 zählt die 2. Stelle von links (8) 100 Mal (10x10) mehr als die Stelle ganz links (7). Die 1. Stelle von links (4) zählt 10 mal mehr.

Nicht positionsbasierte Zahlensysteme: Strichliste oder römischen Zahlen (Erweitert: warum verwenden diese so wenig: Versuche mal zwei römischen Zahlen zu addieren….)

Dezimales Zahlensystem

Das Dezimale Zahlensystem ist ein Positionssystem zur Darstellung von Zahlen, das auf der Basis 10 beruht. Das bedeutet, dass es zehn verschiedene Ziffern gibt, von 0 bis 9, die verwendet werden, um Zahlen darzustellen. Jede Ziffer hat einen bestimmten Stellenwert, abhängig von ihrer Position in der Zahl.

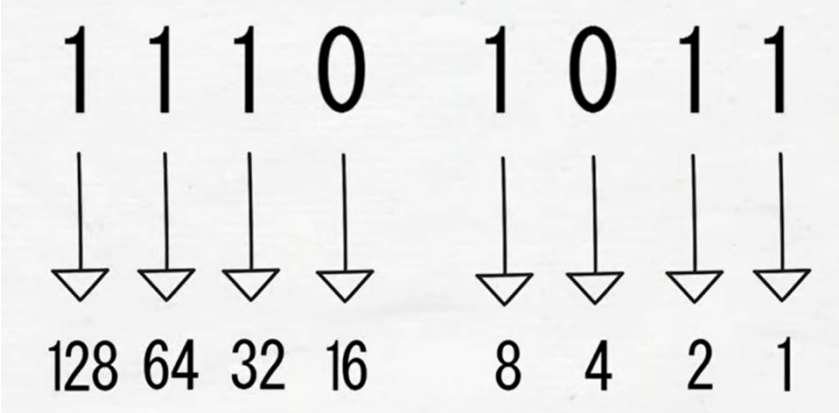
Die rechte Seite einer Dezimalzahl repräsentiert die Dezimalstellen, beginnend mit 10^0 (Einer), 10^1 (Zehner), 10^2 (Hunderter) und so weiter. Die linke Seite der Dezimalzahl repräsentiert die Potenzen von 10 mit zunehmender Exponentenwerte.

Zum Beispiel: Die Zahl 235,7 bedeutet 2 * 10^2 (200), plus 3 * 10^1 (30), plus 5 * 10^0 (5), plus 7 * 10^(-1) (0,7). Zusammen ergibt das 235,7.

Das Dezimale Zahlensystem wird in vielen Bereichen des täglichen Lebens verwendet und ist das am weitesten verbreitete Zahlensystem in der Welt. Es wird oft als "Basis 10" bezeichnet, um den zehn Ziffern Rechnung zu tragen.

Binäres Zahlensystem

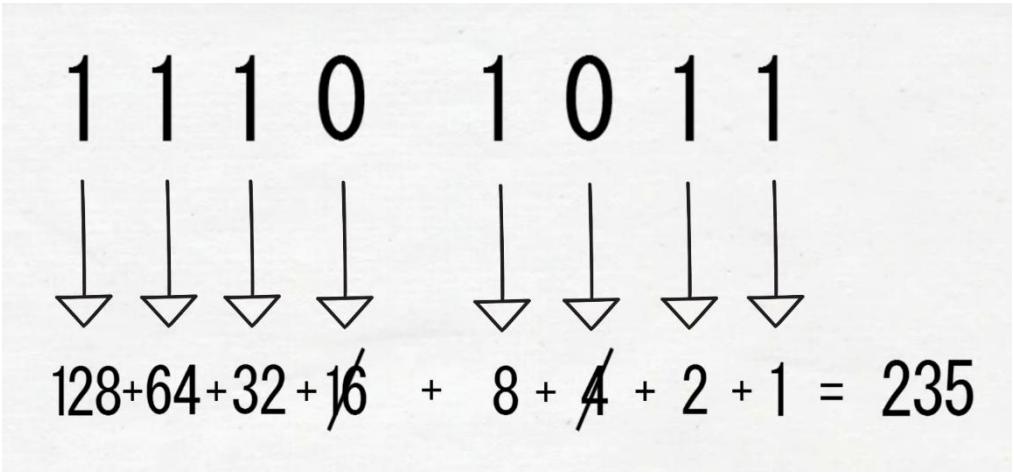
Positionsbasiertes System mit 2 Zahlen: 0 und 1. Jede Stelle nach links ist daher 2x mehr Wert als die vorherige (1, 2, 4, 8, 16…)



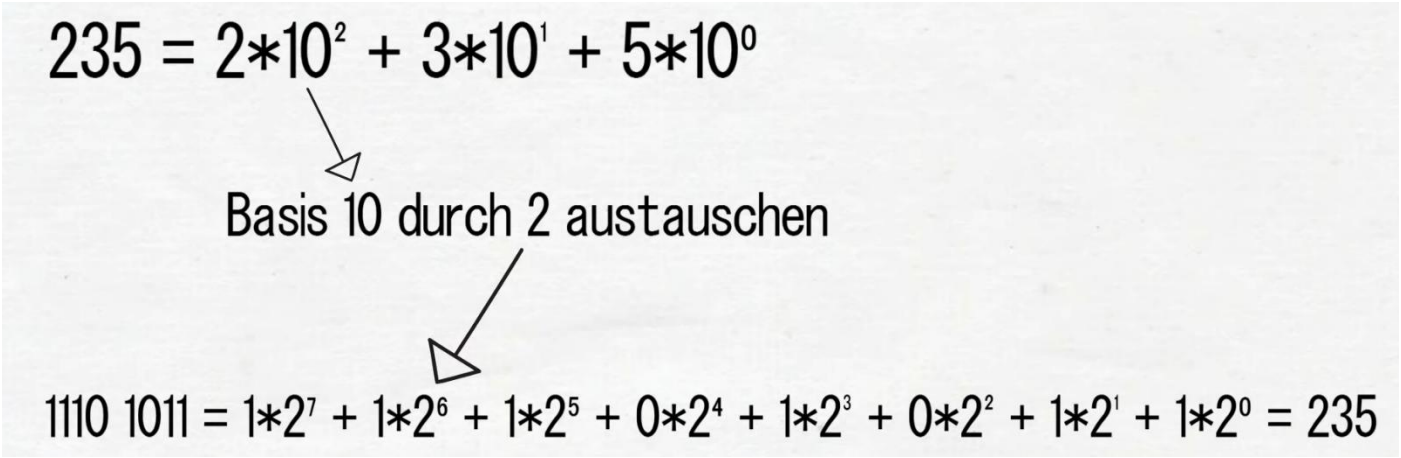
Die Informatik verwendet das binäre Zahlensystem, da fast alle Rechner mit Elektrizität funktionieren. Es ist viel einfacher zu messen, ob Strom fliesst oder nicht (0/10) als zum Beispiel 10 verschiedene Stromstufen messen zu müssen (10er System).

Binär zu Dezimal

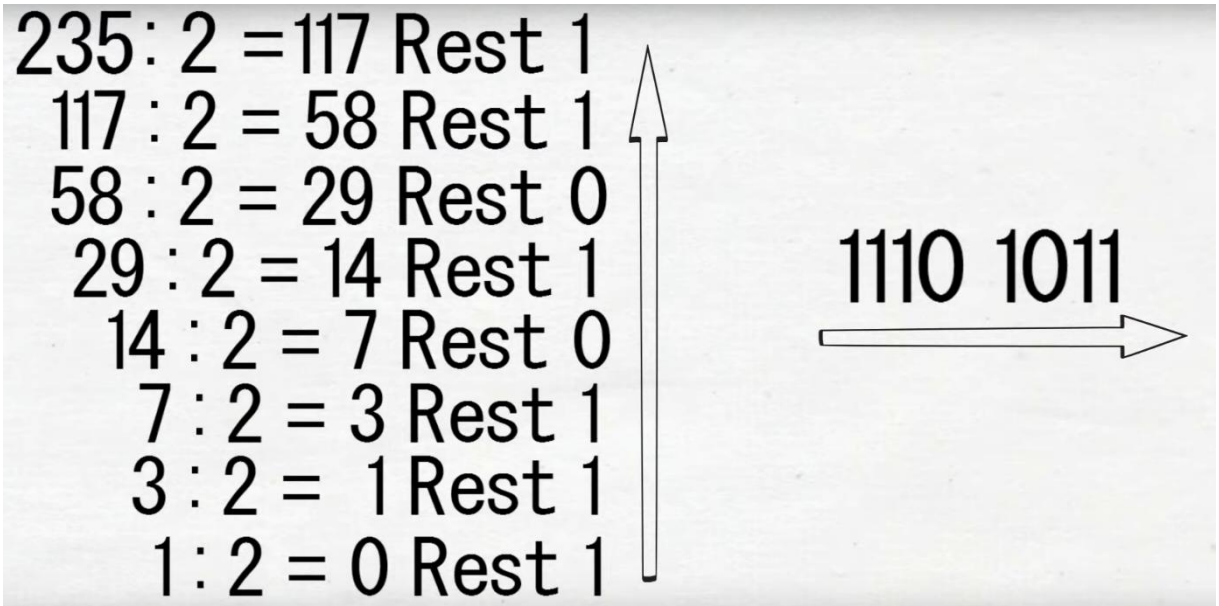
1. Variante (kleiner Zahl)



2. Variante (grössere Zahl)



Dezimal zu Binär



Addition zweier binären Zahlen

Natürlich, hier ist ein Beispiel für die Addition von zwei binären Zahlen:

Angenommen, wir wollen die binären Zahlen `1101` und `101` addieren. Hier sind die Schritte:

```
markdown
   1101
+   101
-------
  10010
```

1. Beginne mit der Addition der rechten Ziffern: `1 + 1 = 10`. Schreibe das `0` unten und trage das `1` als Übertrag nach links.
2. Addiere die nächsten Ziffern, einschließlich des Übertrags: `1 + 0 + 1 + 1 = 11`. Schreibe das `1` unten und trage das `1` als Übertrag nach links.
3. Addiere die linken Ziffern: `1 + 1 + Übertrag = 1 + 1 + 1 = 11`. Schreibe das `1` unten und trage das `1` als Übertrag nach links.
4. Da keine weiteren Ziffern vorhanden sind, schreibe den verbleibenden Übertrag oben auf.

Das Ergebnis der Addition von `1101` und `101` ist `10010` im binären System. Beachte, dass dies eine Übertragung nach links erforderlich gemacht hat, und die resultierende Summe hat eine Ziffer mehr als die ursprünglichen Zahlen.

Oktales Zahlensystem

Das oktale Zahlensystem, auch als Basis-8-System bekannt, verwendet 8 Symbole (0, 1, 2, 3, 4, 5, 6, 7) zur Darstellung von Zahlen. Im Gegensatz zum dezimalen System, das 10 Symbole (0-9) verwendet, hat das oktale System eine Basis von 8.

Um eine Zahl im oktalen System darzustellen, schreibst du sie mit den oben genannten Oktalziffern.

Dezimal in Oktal

Dezimalzahl: 117

117:8 = 14 (+ Rest) Rest= 117 – (8 * 14) = 5

14:8= 1 (+Rest) Rest= 14-(8 * 1) = 6

1:8= 0 (+Rest) Rest= 1-(8*0) = 1

Reste jetzt zusammenstellen für die oktale Zahl= 165

Oktal in Dezimal

Oktalzahl: 165

165 = 1*8^2 + 6*8^1 + 5*8^0 -> 64 + 48 + 5=117

Addition zweier oktalen Zahlen

Angenommen, wir wollen die oktalen Zahlen `35` und `27` addieren. Hier sind die Schritte:

```
markdown
   35
+  27
-------
   64
```

1. Beginne mit der Addition der rechten Ziffern: `5 + 7 = 12`. Schreibe das `2` unten und trage das `1` als Übertrag nach links.
2. Addiere die nächsten Ziffern, einschließlich des Übertrags: `3 + 2 + 1 = 6`. Schreibe das `6` unten.

Das Ergebnis der Addition von `35` und `27` ist `64` im oktalen System. Beachte, dass im oktalen System die größte Ziffer 7 ist, daher wird jede Übertragung von mehr als 1 zum nächsten Ziffernplatz zurückgeführt.

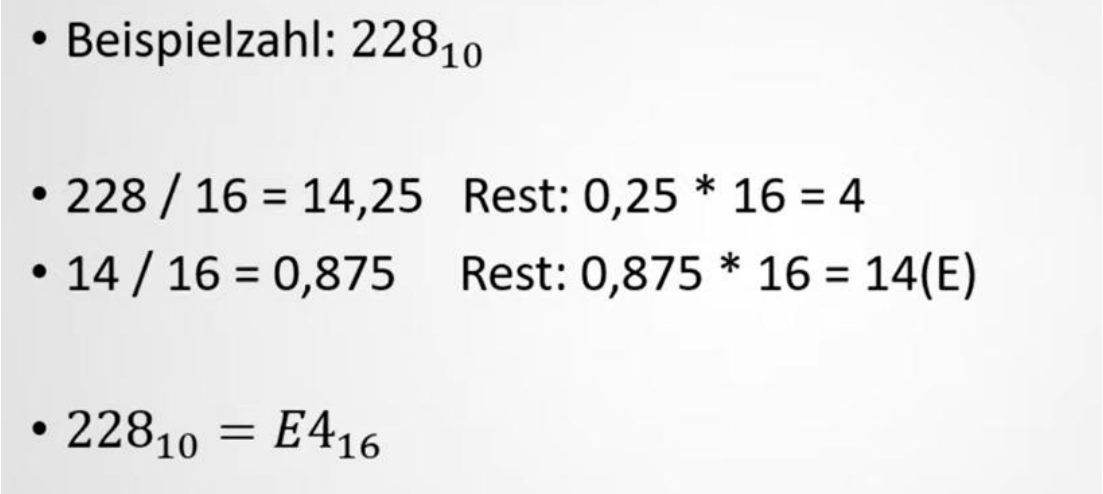
Hexadezimales Zahlensystem

Positionsbasiertes System mit 16 Zahlen (0-F). Jede Stelle nach links ist daher 16x mehr Wert als die Vorherige (1, 16, 256, 4096…)

Mit eine zweistellige Hexadezimalzahl können 256 Werte dargestellt, was genau ein Byte entspricht. Man «verschwendet» also kein Platz wie wenn man 0-255 mit dem Dezimalsystem darstellt.

Hexadezimal Reihenfolge (16 Zahlen): 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F

Dezimal zu Hexadezimal



Hexadezimal zu Dezimal

Natürlich, ich werde dir ein Beispiel geben, wie man eine hexadezimale Zahl in eine dezimale Zahl umwandelt.

Angenommen, wir haben die hexadezimale Zahl `1A3`. Hier sind die Schritte, um sie in eine dezimale Zahl umzuwandeln:

1. Schreibe die hexadezimale Zahl auf, und ordne den entsprechenden Dezimalwert jeder Hexadezimalziffer zu:

- 1A3 = (1 * 16^2) + (10 * 16^1) + (3 * 16^0)

2. Berechne die Werte für jede Potenz von 16:

- 16^2 = 256

- 16^1 = 16

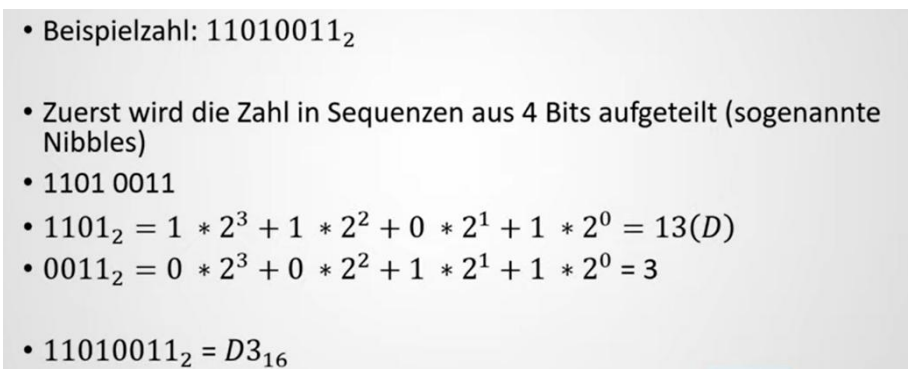
- 16^0 = 1

3. Multipliziere die entsprechenden Ziffern mit ihren Potenzen von 16 und addiere die Ergebnisse:

- (1 * 256) + (10 * 16) + (3 * 1) = 256 + 160 + 3 = 419

Daher entspricht die hexadezimale Zahl `1A3` der dezimalen Zahl `419`.

Binär zu Hexadezimal



Addition zweier oktalen Zahlen

Angenommen, wir wollen die oktalen Zahlen `45` und `23` addieren. Hier sind die Schritte:

```
markdown
   45
+  23
-------
  110
```

1. Beginne mit der Addition der rechten Ziffern: `5 + 3 = 10`. Schreibe das `0` unten und trage das `1` als Übertrag nach links.
2. Addiere die nächsten Ziffern, einschließlich des Übertrags: `4 + 2 + 1 = 7`. Schreibe das `7` unten.

Das Ergebnis der Addition von `45` und `23` ist `110` im oktalen System. Beachte, dass jede Übertragung von mehr als 1 zum nächsten Ziffernplatz zurückgeführt wird, da im oktalen System die größte Ziffer 7 ist.

Zahlensysteme verglichen

Dezimal	Binär	Oktal	Hexadezimal	Eigenes
0	0	0	0	#
1	1	1	1	?
2	10	2	2	!
3	11	3	3	?#
4	100	4	4	??
5	101	5	5	?!
6	110	6	6	!#
7	111	7	7	!?
8	1000	10	8	!!
9	1001	11	9	?##
10	1010	12	A	?#?
11	1011	13	B	?#!
12	1100	14	C	??#
13	1101	15	D	???
14	1110	16	E	??!
15	1111	17	F	?!#
16	10000	20	10	?!?
17	10001	21	11	?!!
18	10010	22	12	!##
19	10011	23	13	!#?

Binäre Kodierung von negativen Zahlen und Fliesskommazahlen

Einerkomplement (negative Zahl)

Einerkomplement ist eine Methode zur Darstellung negativer Zahlen in der Binärdarstellung. Um die Einerkomplementdarstellung einer negativen Zahl zu erhalten, invertiert man einfach alle Bits (1 wird zu 0 und umgekehrt). Zum Beispiel:

- Positive Zahl 5 in Binär: `0101`
- Einerkomplement von -5: `1010`

Zweierkomplement (negative Zahl)

Zweierkomplement ist eine effizientere Methode zur Darstellung negativer Zahlen. Um die Zweierkomplementdarstellung einer negativen Zahl zu erhalten, invertiert man alle Bits und addiert anschließend 1 zur resultierenden Binärzahl. Zum Beispiel:

- Positive Zahl 5 in Binär: `0101`
- Zweierkomplement von -5: `1011`

Offset Binary (negative Zahl)

Offset Binary ist eine Methode zur Darstellung sowohl positiver als auch negativer Zahlen, indem ein konstanter Offset zu den Zahlen hinzugefügt wird. Dieser Offset verschiebt den Bereich der darstellbaren Zahlen. Zum Beispiel:

- Offset Binary mit einem Offset von 3: `000` repräsentiert -3, `001` repräsentiert -2, ..., `111` repräsentiert 4.

Fliesskommazahlen

Fliesskommazahlen sind eine Darstellung von Zahlen, die eine Mantisse und einen Exponenten verwendet, um den Zahlenbereich und die Genauigkeit zu erweitern. In der Binärdarstellung besteht eine Fliesskommazahl aus Vorzeichenbit, Mantisse und Exponent. Zum Beispiel:

- `1.101 * 2^3` bedeutet 1.101 im Binär mit einem Exponenten von 3.

Logikschaltungen

1. **AND (UND):**

Ausgang ist wahr (1), nur wenn beide Eingänge wahr sind.

AND

Input		Output
A	B	X
0	0	0
1	0	0
0	1	0
1	1	1

2. **OR (ODER):**

Ausgang ist wahr (1), wenn mindestens einer der Eingänge wahr ist.

OR

Input		Output
A	B	X
0	0	0
1	0	1
0	1	1
1	1	1

3. **XOR (Exklusiv-ODER):**

Ausgang ist wahr (1), wenn genau einer der Eingänge wahr ist, aber nicht beide.

XOR

Input		Output
A	B	X
0	0	0
1	0	1
0	1	1
1	1	0

4. **NAND (NICHT-UND)**

Ausgang ist wahr (1), es sei denn beide Eingänge sind wahr.

NAND

Input		Output
A	B	X
0	0	1
1	0	1
0	1	1
1	1	0

5. **NOR (NICHT-ODER):**

Ausgang ist wahr (1), wenn beide Eingänge falsch sind.

NOR

Input		Output
A	B	X
0	0	1
1	0	0
0	1	0
1	1	0

6. **XNOR (Exklusiv-NICHT-ODER):**

Ausgang ist wahr (1), wenn beide Eingänge gleich sind, entweder beide wahr oder beide falsch.

XNOR

Input		Output
A	B	X
0	0	1
1	0	0
0	1	0
1	1	1

7. **NOT (NICHT):**

Ändert den Zustand des Eingangs; wenn Eingang wahr ist, wird der Ausgang falsch und umgekehrt.

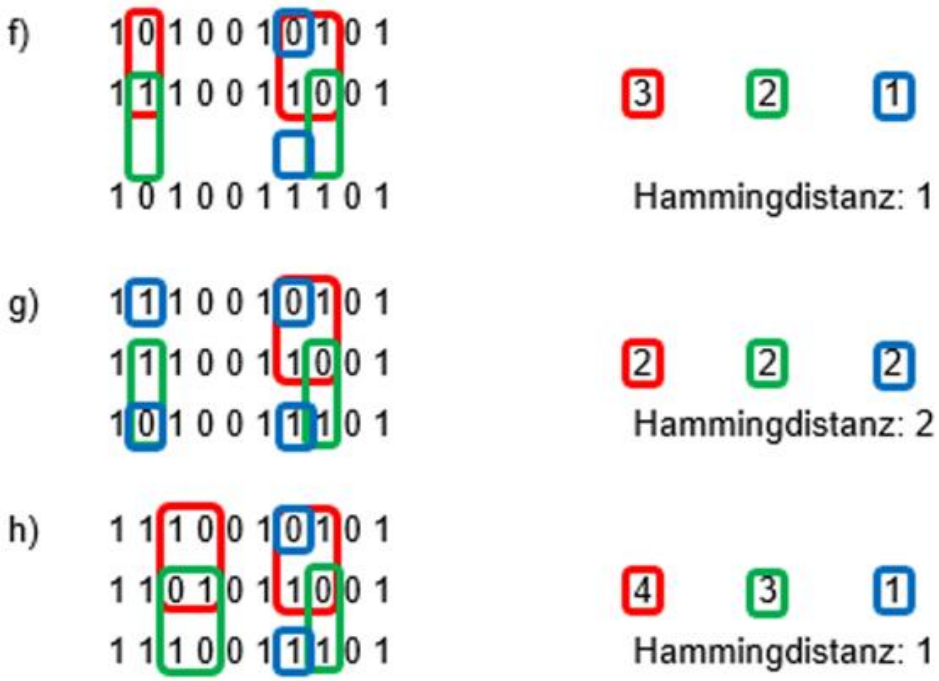
NOT

Input		Output
A		X
0		1
1		0

Diese logischen Schaltungen werden in der Digitaltechnik verwendet, um komplexe Aufgaben auszuführen, indem sie Bits (0 oder 1) entsprechend den oben genannten Regeln kombinieren und manipulieren.

Fehlererkennung (Hamming herausfinden)

Nur die Unterschiede zusammenzählen.



Codes

Dezimalcode

Das Dezimalsystem verwendet zehn Symbole (0-9) zur Darstellung von Zahlen. Jede Stelle im Dezimalsystem repräsentiert eine Potenz von 10 (10^0, 10^1, 10^2 usw.), was die Grundlage für die Umrechnung von Binärzahlen in Dezimalzahlen ist.

8-Bit Zeichensätze

8-Bit-Zeichensätze repräsentieren Zeichen mithilfe von 8 Bits (1 Byte). Dies ermöglicht die Darstellung von 256 verschiedenen Zeichen. Beispiele für 8-Bit-Zeichensätze sind ISO-8859-1, Windows-1252 und UTF-8.

Unicode und UTF-Zeichensätze

Unicode ist ein internationaler Standard, der jedem Zeichen in allen Schriftsystemen einen eindeutigen Codepunkt zuweist. UTF (Unicode Transformation Format) ist eine Codierung, die es ermöglicht, Unicode-Zeichen in Bytes zu speichern. Beispiele sind UTF-8, UTF-16 und UTF-32.

BCD

BCD ist eine Darstellung von Dezimalzahlen mithilfe von Binärzahlen. Jede Dezimalziffer wird in 4 Bits dargestellt. Dies erleichtert die Umwandlung von Dezimalzahlen in Binärform und wird häufig in der Computertechnik verwendet.

2 aus 5 Code und 1 aus 10 Code

Ein 2 aus 5 Code ist eine binäre Codierungstechnik, bei der zwei von fünf Elementen aktiviert sind, um Dezimalziffern zu repräsentieren. Es wird oft in Barcode-Systemen verwendet.

Ein 1 aus 10 Code ist eine ähnliche Codierungstechnik, bei der eines von zehn Elementen aktiviert ist, um Dezimalziffern zu repräsentieren. Es wird ebenfalls in Barcode-Systemen verwendet.

1D Code- EAN Strichcodes

EAN ist ein Strichcodesystem, das zur Identifikation von Produkten im Einzelhandel verwendet wird. Es besteht aus einer 13-stelligen Zahl, die auf Produkten und Verpackungen zu finden ist.

2D/3D Codes – QR-Codes

QR-Codes sind zweidimensionale Matrix-Barcodes, die eine breite Palette von Informationen speichern können. Sie werden in vielen Anwendungen wie Werbung, Verpackungsetiketten und mobilen Anwendungen eingesetzt.

Bildkodierung

Bitmap (Rastergrafiken)

- **Merkmale:**
    - Bild besteht aus einer Rastermatrix von Pixeln.
    - Jedes Pixel speichert Farbinformationen.
    - Gängige Formate: BMP, GIF, JPG, PNG.
- **Vorteile:**
    - Gute Darstellung von Fotografien und komplexen Grafiken.
    - Pixelbasierte Bearbeitung möglich.
- **Nachteile:**
    - Skalierung kann zu Qualitätsverlust führen.
    - Größere Dateigrößen.

Vektoren

- **Merkmale:**
    - Verwendung von mathematischen Formeln für Grafikdarstellung.
    - Unbegrenzte Skalierbarkeit ohne Qualitätsverlust.
    - Gängige Formate: SVG, AI, EPS.
- **Vorteile:**
    - Hohe Qualität bei Skalierung.
    - Kleinere Dateigrößen für einfache Grafiken.
    - Bearbeitung von Formen und Pfaden.
- **Nachteile:**
    - Komplexere Darstellung von Fotografien.
    - Nicht ideal für komplexe Details.

Farbsysteme

- **RGB (Rot, Grün, Blau):**
    - Lichtadditives Farbsystem für digitale Anwendungen.
    - Kombination der Farbanteile erzeugt unterschiedliche Farben.
- **CMYK (Cyan, Magenta, Yellow, Black):**
    - Subtraktives Farbsystem für den Druck.
    - Kombination von Farben absorbiert Licht und erzeugt Druckfarben.
- **Hexadezimal (Web):**
    - Darstellung von RGB-Farben durch sechsstellige Hexadezimalzahlen.
    - Beispiel: #RRGGBB.

Grafiksysteme

BMP (Bitmap):

- **Merkmale:**
    - Unkomprimiertes Rastergrafikformat von Microsoft.
    - Gute Qualität, aber große Dateigrößen.

GIF (Graphics Interchange Format):

- **Merkmale:**
    - Verlustfreie Komprimierung für Grafiken mit begrenzter Farbpalette.
    - Unterstützt Animation und Transparenz.

JPG (Joint Photographic Experts Group):

- **Merkmale:**
    - Verlustbehaftete Kompression für Fotografien.
    - Geringere Dateigrößen mit Qualitätsverlust.

PNG (Portable Network Graphics):

- **Merkmale:**
    - Verlustfreie Kompression für Grafiken und Fotos.
    - Unterstützt Transparenz und verschiedene Farbtiefen.

SVG (Scalable Vector Graphics):

- **Merkmale:**
    - XML-basiertes Vektorformat.
    - Skalierbar ohne Qualitätsverlust.
    - Ideal für Webgrafiken und interaktive Inhalte.

Kompression

Verlustbehaftete Kompression

Hier sind einige der gängigen verlustbehafteten Kompressionsalgorithmen:

1. **JPEG (Joint Photographic Experts Group):**
    - Häufig verwendet für Fotos und Bilder. Es basiert auf der Diskretisierung von Farben und der Verwendung von Frequenzbereichstransformationen.
2. **MPEG (Moving Picture Experts Group):**
    - Für die Kompression von Audio- und Videodateien. Die MPEG-Technologie umfasst verschiedene Standards wie MPEG-1, MPEG-2, und MPEG-4, die für unterschiedliche Anwendungen verwendet werden.
3. **MP3 (MPEG Audio Layer III):**
    - Eine spezielle Anwendung von MPEG für die verlustbehaftete Audiokompression. Es verwendet Psychoakustik, um unhörbare Töne zu eliminieren.
4. **H.264 (Advanced Video Coding):**
    - Ein weit verbreiteter Videokompressionsstandard, der in vielen Anwendungen wie Blu-ray-Discs, Internet-Streaming und Videoübertragungen verwendet wird.
5. **AAC (Advanced Audio Coding):**
    - Ein weiterer Audio-Kompressionsstandard, häufig in Verbindung mit dem MPEG-2- und MPEG-4-Standard verwendet.
6. **WebP:**
    - Ein von Google entwickeltes Format für verlustbehaftete Kompression von Bildern. Es wurde speziell für die effiziente Darstellung von Bildern im Web entwickelt.
7. **FLAC (Free Lossless Audio Codec):**
    - Obwohl "lossless" im Namen, ist FLAC im Grunde verlustfrei, da es eine Kompression ohne Qualitätsverlust für Audiodateien bietet.

Es ist wichtig zu beachten, dass bei der verlustbehafteten Kompression Informationen verloren gehen, was zu einem Qualitätsverlust führen kann. Die Wahl des Komprimierungsverfahrens hängt von der Art der Daten und den Anforderungen an die Qualität ab.

Verlustfreie Kompression

Hier sind einige der bekanntesten verlustfreien Kompressionsalgorithmen:

1. **Lempel-Ziv-Welch (LZW):**

    - Ein adaptiver Algorithmus, der auf der Wiederholung von Zeichenketten basiert. Wird oft in GIF- und TIFF-Dateiformaten verwendet.

2. **Deflate (ZIP):**
    - Kombination von LZ77 und Huffman-Codierung. Wird in ZIP-Archiven und im PNG-Dateiformat verwendet.

3. **Huffman-Codierung:**
    - Ein statistischer Codierungsansatz, bei dem häufige Zeichen kürzere Codes erhalten. Wird in Kombination mit anderen Algorithmen wie Deflate verwendet.

4. **Burrows-Wheeler-Transformation (BWT):**
    - Umordnung der Zeichen in einem Text, gefolgt von Run-Length-Encoding oder Huffman-Codierung. Wird in Bzip2 und einige Varianten von ZIP verwendet.

5. **Run-Length-Encoding (RLE):**
    - Kompromisslose Methode, bei der aufeinanderfolgende gleiche Zeichen durch eine einzige Instanz und ihre Anzahl ersetzt werden.

6. **Arithmetische Codierung:**
    - Ein mathematischer Ansatz, bei dem eine kontinuierliche Bruchzahl verwendet wird, um Daten zu repräsentieren. Oft sehr effizient, aber auch rechenaufwendig.

7. **Prediction by Partial Matching (PPM):**
    - Ein adaptiver Algorithmus, der auf statistischen Modellen basiert und versucht, das nächste Zeichen in einer Sequenz vorherzusagen.

Diese Algorithmen werden je nach Art der Daten und den Anforderungen an die Kompression in verschiedenen Kontexten eingesetzt. Je nach Anwendung kann ein bestimmter Algorithmus besser geeignet sein.

Symmetrische Verschlüsselung

Stab-Chiffre

Die Stab-Chiffre ist eine relativ einfache Verschlüsselungsmethode, die zu den Substitutionstechniken gehört. Bei der Stab-Chiffre wird jede Buchstabe des Klartexts durch einen anderen Buchstaben ersetzt. Dieser Ersatz basiert auf einer festen Anzahl von Positionen, die jeden Buchstaben im Alphabet verschieben. Beispielsweise könnte der Buchstabe A durch den Buchstaben D ersetzt werden. Es handelt sich dabei um eine Form der Verschiebechiffre.

Ein Beispiel:

- Klartext: "VERSCHLUESSELUNG"

- Schlüssel: 3142
- Verschlüsselter Text: "VLSEUERSSNGHELC"

Cäsar-Chiffre

Die Cäsar-Chiffre ist eine spezielle Form der Stab-Chiffre und auch als Verschiebechiffre bekannt. Hierbei wird jeder Buchstabe im Klartext um eine feste Anzahl von Positionen im Alphabet verschoben. Der Schlüssel, der die Anzahl der Positionen angibt, wird oft als "Schlüssel" bezeichnet. Diese Chiffre ist einfach zu verstehen und zu implementieren, aber auch leicht zu brechen, da es nur 25 mögliche Schlüssel gibt. Zum Beispiel mit einer Verschiebung von 3:

- Klartext: "VERSCHLUESSELUNG"
- Verschlüsselter Text: "YHUFLJKXHVVDNHQJ"

Vigenère-Chiffre

Die Vigenère-Chiffre ist eine verbesserte Version der Cäsar-Chiffre und fällt unter die polyalphabetischen Substitutionstechniken. Anstatt jeden Buchstaben um eine feste Anzahl von Positionen zu verschieben, verwendet die Vigenère-Chiffre einen Schlüsselwort, um die Verschiebung für jeden Buchstaben im Klartext anzugeben. Das Schlüsselwort wird wiederholt, um die Länge des Klartexts zu erreichen. Dies macht die Vigenère-Chiffre gegenüber der einfachen Cäsar-Chiffre widerstandsfähiger gegenüber Brute-Force-Angriffen. Beispiel:

- Klartext: "VERSCHLUESSELUNG"
- Schlüssel: "KEY"
- Verschlüsselter Text: "VIRKBNYGOBKPWLVG"

Vernam

Die Vernam-Chiffre, auch als One-Time Pad bekannt, ist ein perfektes symmetrisches Verschlüsselungsverfahren, vorausgesetzt, der Schlüssel wird korrekt erzeugt und nur einmal verwendet. Jeder Buchstabe des Klartexts wird mit einem Buchstaben des zufällig generierten Schlüssels kombiniert. Da der Schlüssel so lang wie der Klartext ist und nur einmal verwendet wird, ist die Vernam-Chiffre gegenüber statistischen Angriffen und Brute-Force-Angriffen immun. Allerdings erfordert dies, dass beide Parteien den identischen Schlüssel im Voraus haben. Ein Beispiel mit einem Schlüsselstrom "K":

- Klartext: "VERSCHLUESSELUNG"
- Schlüsselstrom: "XYUJNSWHDCLTRQOL"
- Verschlüsselter Text: "ZFSJQSNFOTXVDFRJ"

Asymmetrische Verschlüsselung

Prinzip der asymmetrischen Verschlüsselung

Die asymmetrische Verschlüsselung, auch als Public-Key-Kryptographie bekannt, basiert auf der Verwendung von zwei Schlüsseln: einem öffentlichen Schlüssel, der zur Verschlüsselung verwendet wird, und einem privaten Schlüssel, der zur Entschlüsselung dient. Diese beiden Schlüssel sind mathematisch miteinander verknüpft, aber es ist praktisch unmöglich, den privaten Schlüssel aus dem öffentlichen Schlüssel abzuleiten. Das bedeutet, dass der öffentliche Schlüssel sicher verteilt werden kann, während der private Schlüssel geheim gehalten wird.

Das Verschlüsseln von Informationen erfolgt mit dem öffentlichen Schlüssel des Empfängers, und nur der Besitzer des zugehörigen privaten Schlüssels kann die verschlüsselten Daten entschlüsseln. Dies ermöglicht eine sichere Kommunikation, da der öffentliche Schlüssel nicht dazu verwendet werden kann, die Nachrichten zu entschlüsseln, die mit ihm verschlüsselt wurden.

Modulo rechnen

Das Modulo-Rechnen ist eine mathematische Operation, die den Rest einer Division berechnet. Es wird oft mit dem Symbol "mod" dargestellt. In der Modulo-Arithmetik wird der Divisionsrest beibehalten, während der Quotient verworfen wird.

Für zwei ganze Zahlen a und b mit einem positiven Modul n (geschrieben als a mod n oder auch a % n), ist das Ergebnis der Modulo-Rechnung der Rest r der Division von a durch n.

Mathematisch ausgedrückt: a mod n=r

Typische asymmetrische Verschlüsselungen

- RSA
- Diffie-Hellman
- Hash-Verfahren
- MD4 und MD5
- SHA
- Digitale Signaturen

Steganografie

Die Steganografie ist die Kunst und Wissenschaft des Verbergens von Informationen innerhalb unscheinbarer Trägermedien, um die Existenz der Nachricht selbst zu verbergen. Im Gegensatz zur Kryptografie, die sich auf das Verschlüsseln von Informationen konzentriert, versucht die Steganografie, die Tatsache des Vorhandenseins einer Nachricht zu verbergen.

Grundlagen der Steganografie

1. **Trägermedien:** Steganografische Botschaften werden häufig in digitalen Medien wie Bildern, Audiodateien oder Videos versteckt. Diese Medien dienen als Träger für die versteckte Information.

2. **Versteckungsmethoden:** Es gibt verschiedene Methoden, um Informationen zu verbergen. Einige häufige Techniken umfassen das Ersetzen der geringstwertigen Bits

in einem Bild, das Hinzufügen von Rauschen, die Verwendung von unsichtbaren Wasserzeichen oder das Einbetten von Informationen in Audiosignale.

3. **Passwort oder Schlüssel:** Um die versteckte Nachricht zu extrahieren, ist oft ein spezielles Passwort oder ein Schlüssel erforderlich. Dies stellt sicher, dass nur autorisierte Personen die versteckten Informationen extrahieren können.

Verwendung der Steganografie

1. **Datenschutz:** In einigen Fällen wird Steganografie verwendet, um persönliche oder vertrauliche Informationen zu schützen, indem sie in unscheinbaren Medien versteckt werden.
2. **Urheberrechtsschutz:** Steganografie wird in der Medienindustrie eingesetzt, um digitale Wasserzeichen in Bildern, Audiodateien oder Videos einzubetten, um Urheberrechtsverletzungen zu verfolgen.
3. **Geheimdienste und Militär:** In sicherheitsrelevanten Bereichen wird Steganografie zur verdeckten Kommunikation und Informationsübermittlung eingesetzt.

SideQuests

114-1A Einführung

Erste Gedanken zu Codierung, Komprimierung und Verschlüsselung machen.

114-1B Zahlensysteme

Erklärung zu Positionsbasierte und nicht Positionsbasierte Zahlensystem. Auch wird eingeführt, wie binäre, hexadezimale und oktale Zahlensysteme funktionieren.

114-2A Binäre Kodierung

Berechnungen mit binären, hexadezimalen und oktalen Zahlensystemen durchführen.

114-3A Logische Operationen

Wissen wie logische Operationen (AND, OR, XOR, NAND, NOR, NOT) funktionieren.

114-3B Fehlererkennung

Hammingdistanz berechnen.

114-4A ASCII, Unicode, UTF & Co

Mit den unterschiedlichen Zeichencodierungen (8-Bit Zeichensätze (ISO8859-1/15, ANSI) und UTF-Zeichensätze (UTF-8, UTF-16, UTF-32)) auseinandersetzen.

114-4B Binärcodes BCD und X aus Y

Dezimal in BCD und umgekehrt umrechnen sowie auch einen 1 aus 10 und 2 aus 5 Code erstellen.

114-4C 1D Code – EAN Strichcodes

Prüfziffer-Berechnung durchführen.

114-4D 2D/3D Codes – QR-Code

OR-Code erstellen.

114-5A Bildkodierung Vektoren, Bitmap

Einführung zu Bitmap, Vektoren, Farbsystemen sowie den unterschiedlichen Grafikformaten wie BMP, GIF, JPG, PNG und SVG.

114-5B Verlustbehaftete Kompression

Verlustbehaftete Kompressionsverfahren kennen und ausprobieren.

114-5C Verlustfreie Kompression

Verlustfreie Kompressionsverfahren kennen und ausprobieren

114-6A Symmetrische Verschlüsselung

Verschiedene symmetrische Verschlüsselungsverfahren kennen und anwenden.

114-6B Asymmetrsiche Verschlüsselung

Prinzip der asymmetrischen Verschlüsselung verstehen und mit Modulo rechnen.

114-7A Verschlüsselung

Verschiedene Verschlüsselungsverfahren kennen und deren Unterschied.

114-7B Steganografie

Das Prinzip der Steganografie verstehen und selbst verschiedene Botschaften ermitteln und erstellen.

114-7C Einführung Mail- Verschlüsslung

E-Mails verschlüsseln und versenden mit verschiedenen Verfahren.

114-8A Repetitionsaufgabe

Eine grosse Aufgabe bei denen alle vorherigen Themen noch einmal angewendet werden.