Representasjon av tekst (enkelttegn), Seksjon 4.3 i komp.

I en datamaskin er alt 0 og 1
Ved hjelp av dette kan vi representere heltall og reelle tall (flyttall).
Hvordan kan vi bruke tall til å representere tekst?

I en datamaskin representeres enkelttegn ved hjelp av heltalls koder og en tabell som forteller hvilken heltallskode som svarer til hvilket tegn. Ved inalesing oversettes tegnene til korrekte koder og ved utskrift skrives kodeni som riktig tegn.

a - 1
b - 2
c - 3
d - 4
e - 5

b a d
2 1 4

ASCII - tabellen.
I datamaskinens barndom hadde hver leverandør hver sin tekststandard.
ASCII - tabellen ble etterhvert en standard.
Første gang publisert i 1963, siste gang oppdatert i 1986.
ASCII består av 128 tegn og kan dermed kodes med 7 bits ($2^7 = 128$).
Etterhvert ble det vanlig å legge til en 0 foran slik at tegnene kunne representeres med 1 byte (= 8 bits)

Etterhvert ble det behov for å kunne representere tegn utover de som fins i ASCII. Mot slutten av 80-tallet kom såkalte ISO-Latin tegnsett.

ISO-Latin 1 – vest Europa
ISO-Latin 2 – sentral Europa
ISO-Latin 5 – Tyrkisk.

Alle disse bruker koder med 8 bits som brukes fullt ut, altså $2^8 = 256$ tegn. ASCII ligger i de første 128 kodene. Iso Latin fungerte greit internt i deler av den vestlige verden, men kommunikasjon på tvers av de ulike områdene var stadig vanskelig.

Unicode.

Man trengte en tabell som kunne inneholde alle verdens tegn. Dataselskaper gikk og dannet en organisasjon som het Unicode. Unicode har koder fra 0 - 1114111 - litt over en million tegn. Men ikke på langt nær alle koder er i bruk (bare ca 100 000 som er i bruk).

Med UTF-32 lagres unicode-kodene fullt ut. Da trenger vi 4 bytes til hvert tegn. Dette er problematisk for vestlige tekster - en fil tar da opp 4 ganger så mye plass som hvis brukt 150-årene!

Løsning: Man lagrer kodene ved hjelp av et variabelt antall bytes.

UTF-8.

Fact 4.10. Hvis koden er

1. $c = (d_6 d_5 d_4 d_3 d_2 d_1 d_0)_2$ i intervallet (0-127) lagres den i en byte som $65 = 64 + 1$

   $0 d_6 d_5 d_4 d_3 d_2 d_1 d_0$ 01000001

2. Hvis $c = (d_{10} d_9 d_8 d_7 d_6 d_5 d_4 d_3 d_2 d_1 d_0)_2$ er i intervallet 128-2047 lagres den i 2 bytes som

   $110 d_{10} d_9 d_8 d_7 d_6$ $10 d_5 d_4 d_3 d_2 d_1 d_0$

3. Hvis $c = (d_{15} d_{14} \cdots d_1 d_0)_2$ er i intervallet 2048-65535 lagres koden som

   $1110 d_{15} d_{14} d_{13} d_{12}$ $10 d_{11} d_{10} d_9 d_8 d_7 d_6$ $10 d_5 d_4 d_3 d_2 d_1 d_0$

4. Hvis $c = (d_{20} d_{19} \cdots d_0)_2$ er i intervallet 65536 - 1114111 lagres med 4 bytes

   $11110 d_{20} d_{19} d_{18}$ $10 d_{17} d_{16} d_{15} d_{14} d_{13} d_{12}$ $10 d_{11} \cdots d_6$ $10 d_5 d_4 d_3 d_2 d_1 d_0$

Datamaskiner må tet enhver tid
vite hvordan 0'ene og 1'ene eller
leser skal tolkes!