

INF105 - 04
Unicode, floating point, dataformater

David Grellscheid

Q&A

- Kan du legge ut presentasjonen for forelesningene før forelesningen? Lettere å følge med når noe er litt kjent før man kommer inn i forelesningen



Heltall - unsigned

- 8 bit kan holde 256 ulike verdier
- unsigned int 8bit: 0 .. 255
- unsigned int 16bit: 0 .. 65535
- ...
- Integer overflow!
- f.eks med 8bit: ..., 253, 254, 255, 0, 1, 2, ...
- overflow etter 0xFF

Heltall - signed

- 8 bit kan holde 256 ulike verdier
- Valg: første bit holder fortegn: 0 = + 1 = -
- 0 ..127 som vanlig, men hvilket negativt tall nå?
- Vanlig valg: -128, -127, ... (Two's complement)
- signed int 8bit: -128 .. 127
- signed int 16bit: -32768 .. 32767
- Integer overflow f.eks med 8bit: ..., 125, 126, 127, -128, -127, -126, ...
- overflow etter 0x7F !

Tekst

- ASCII - 7 bit, nok til A-Z, a-z, 0-9, .,;:"()!? osv.
- Ingen rom til tegn fra andre språk
- ASCII bruker bare 0x00 - 0x7F, kan bruke 0x80-0xFF til andre tegn,
  om vi fortsetter med 1 byte per tegn
- => codepages, trenger ulike til vest-eur., øst-eur., thai, gresk, kyrillisk,...
- samme byte blir brukt til ulike tegn på ulike codepages

Comparison of the various parts (1–16) of ISO/IEC 8859

Binary	Oct	Dec	Hex	1	2	3	4	5	6	7	8	9	10	11	13	14	15	16
1110 0110	346	230	E6	æ	ć	ĉ	æ	ц	ъ	ζ	ŧ	æ		ŋ	ę	æ		

Part 15	*Latin-9*	1999	-	A revision of 8859-1 that removes some little-used symbols, replacing them with the euro sign *€* and the letters *Š*, *š*, *Ž*, *ž*, *Œ*, *œ*, and *Ÿ*, which completes the coverage of French, Finnish and Estonian.
Part 16	*Latin-10 South-Eastern European*	2001	SR 14111 (1998)	Intended for Albanian, Croatian, Hungarian, Italian, Polish, Romanian and Slovene, but also Finnish, French, German and Irish Gaelic (new orthography). The focus lies more on letters than symbols. The generic currency sign is replaced with the euro sign.

https://en.wikipedia.org/wiki/ISO/IEC_8859

Unicode

- mange flere enn 256 symboler er i bruk
- Unicode-prosjekt: et **codepoint** til hver **glyph** som finnes
- **0x000000 - 0x10FFFF:** rom til litt over 1 million glyphs

Unicode representation

Each glyph has one number from 0x0 to 0x10ffff

glyph	unicode point	name
A	U+0041	*Latin Capital letter A*
æ	U+00e6	*Latin Small letter Æ*
ć	U+0107	*Latin Small letter C with acute*
ц	U+0446	*Cyrillic Small Letter Tse*
ن	U+0646	*Arabic Letter Noon*
و	U+0648	*Arabic Letter Waw*
ر	U+0631	*Arabic Letter Reh*
स	U+0938	*Devanagari Letter Sa*
ღ	U+10E6	*Georgian Letter Ghan*
人	U+4EBA	*CJK Unified Ideograph-4EBA*
𐏈	U+103C8	*Old Persian Sign Auramazdaa*
🤓	U+1F913	*Nerd Face*
🇿🇦	-	-
🇿	U+1F1FF	*Regional Indicator Symbol Letter Z*
🇦	U+1F1E6	*Regional Indicator Symbol Letter A*

Unicode

- mange flere enn 256 symboler er i bruk
- Unicode-prosjekt: et **codepoint** til hver **glyph** som finnes
- **0x000000 - 0x10FFFF:** rom til litt over 1 million glyphs

- Enkel koding: UTF-32: bruk 4 bytes til hver glyph.
- Innholdet til bytes tilsvarer codepoint-nummer

UTF-32 vs UTF-8

- Enkel koding: UTF-32: bruk 4 bytes til hver glyph.
- Ulempe: 4x større filer enn ASCII
- ikke kompatibel med ASCII-filer
- Løsning: Variable length coding
- bruk de første bit-posisjonene som *metadata* (informasjon om kodingen)

UTF-8

Code point ↔ UTF-8 conversion

First code point	Last code point	Byte 1	Byte 2	Byte 3	Byte 4
U+0000	U+007F	0yyyzzzz			
U+0080	U+07FF				
U+0800	U+FFFF				
U+010000	U+10FFFF				

Dekoding kan skje fra enhver tilfeldig posisjon, vi vet hvor vi er i codepoint. Codepoint begynner maks tre bytes bak.

Unicode representation

Each glyph has one number from 0x0 to 0x10ffff

glyph	unicode point	name	utf-8
A	U+0041	*Latin Capital letter A*	"41"
æ	U+00e6	*Latin Small letter Æ*	"c3 a6"
ć	U+0107	*Latin Small letter C with acute*	"c4 87"
ц	U+0446	*Cyrillic Small Letter Tse*	"d1 86"
ن	U+0646	*Arabic Letter Noon*	"d9 86"
و	U+0648	*Arabic Letter Waw*	"d9 88"
ر	U+0631	*Arabic Letter Reh*	"d8 b1"
स	U+0938	*Devanagari Letter Sa*	"e0 a4 b8
ღ	U+10E6	*Georgian Letter Ghan*	"e1 83 a6"
人	U+4EBA	*CJK Unified Ideograph-4EBA*	"e4 ba ba"
𐏈	U+103C8	*Old Persian Sign Auramazdaa*	"f0 90 8f 88"
🤓	U+1F913	*Nerd Face*	"f0 9f a4 93"
🇿🇦	-	-	-
🇿	U+1F1FF	*Regional Indicator Symbol Letter Z*	"f0 9f 87 bf"
🇦	U+1F1E6	*Regional Indicator Symbol Letter A*	"f0 9f 87 a6"

Unicode

- er hele problemet med ulike språk løst ved bruk av Unicode?
- med UTF-8 snakker vi bare om datalagring
- Input? Presentasjon?
- Homoglyphs? Combining Characters?

  CYRILLIC SMALL LETTER O: o U+043E
  LATIN SMALL LETTER O: o U+006F

Tilbake til tall

```
>>> 0.1 + 0.2 == 0.3
False
```

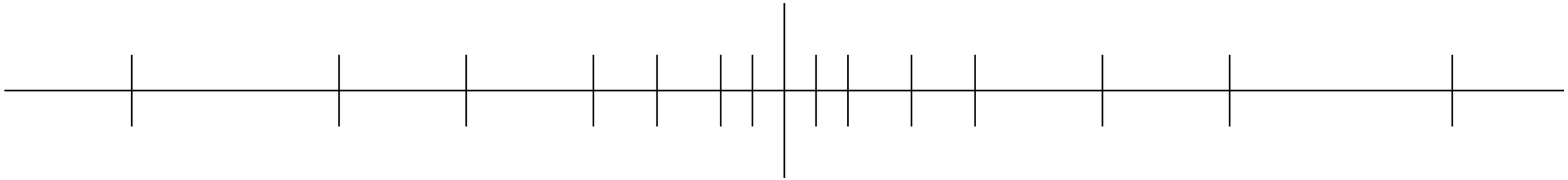
Tilbake til tall

- Hvordan kodere desimaltall?
- En mulighet: fixed point numbers
- Fast posisjon til desimalen. F.eks 8bit signed 5/3: 01011,011
- Hvilke tall kan vi representere?
- 32 heltall fra -16 .. 15, med 8 steg av 1/8 mellom tallene.
- fra 0x00 til 0xFF:
  0, 0.125, 0.25, 0.375, ... 15.875, -16, -15.875, ..., -0.25, -0.125

Fixed point

- Fordel: alltid samme *precision* (her: 1/8)
- Ulempe: avkortet *range* (fra -128..127 til -16..15)

Floating point



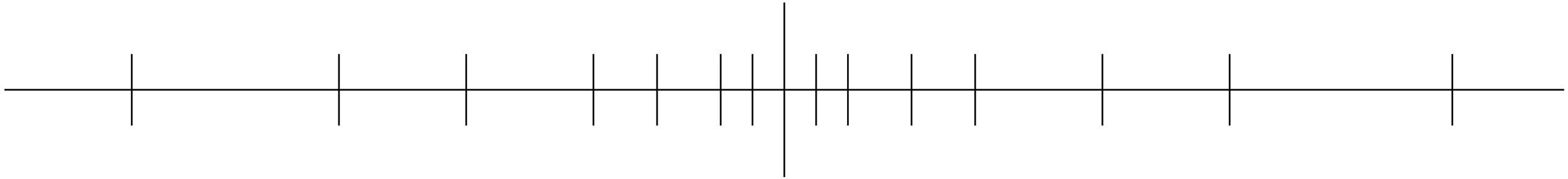
- Høy *precision* nær 0, mindre *precision* jo større tallene blir

$$[\pm]1.[\text{mantisse}] \times 2^{[\text{eksponent}]}$$

significand

fractional part

Floating point - 8bit example



$$[\pm]1.[\text{mantisse}] \times 2^{[\text{eksponent}]-\text{shift}}$$

halvparten-1 av mulig exponent

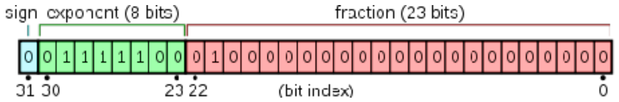
1bit	3bit		4bit
0	110		1100
+	1.75	×	2^{12-7}

1.75× 32 = 56.0

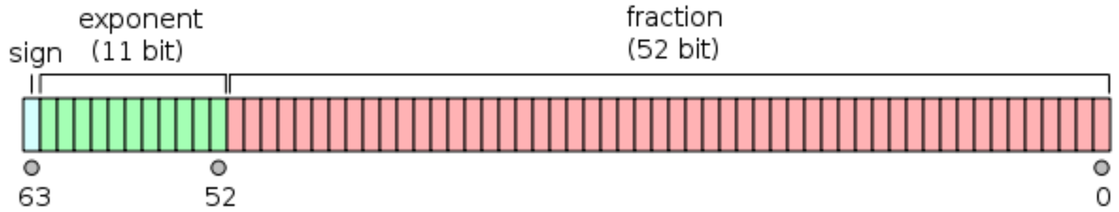
hva er neste tallet?

IEEE 754

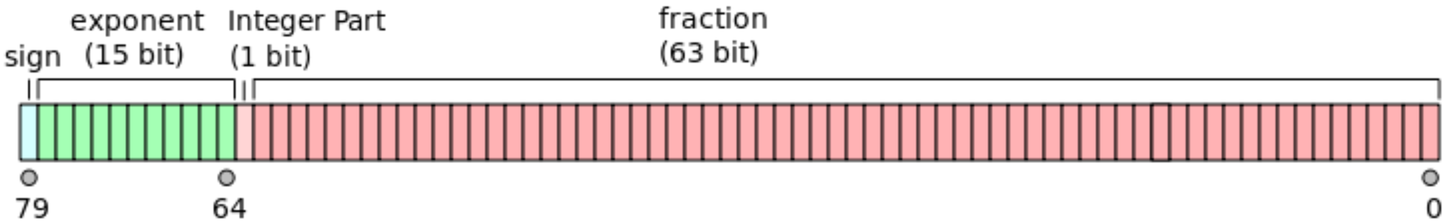
Single-precision floating point (float, 4 bytes)



Double-precision floating point (double, 8 bytes) <- (python `float`)



Extended-precision floating point (long double, **10** bytes)



https://www.h-schmidt.net/FloatConverter/IEEE754.html

Maths pitfalls

Gaps in representation

2^{23} = 8M available numbers between 1 and 2,
but only 4k numbers between 2048 and 2049

17 000 001 cannot be stored exactly in float32

Almost all $\mathbb{R}$ cannot be represented exactly

```
>>> 0.1+0.2
0.30000000000000004

>>> print '{:10.60f}'.format(0.1)
0.1000000000000000055511151231257827021181583404541015625

>>> print '{:10.60f}'.format(0.2)
0.2000000000000000111022302462515654042363166809082031250000000

>>> print '{:10.60f}'.format(0.3)
0.2999999999999999888977697537484345957636833190917968750000000


>>> print '{:10.60f}'.format(0.1+0.2)
0.3000000000000000444089209850062616169452667236328125000000000
```

```
>>> 0.1 + 0.2 == 0.3
False
```

Maths pitfalls

FP maths is commutative, **but not associative**

Value1	Value2	Value3	Value4	Sum
1.0E+30	-1.0E+30	9.5	-2.3	7.2
1.0E+30	9.5	-1.0E+30	-2.3	-2.3
1.0E+30	9.5	-2.3	-1.0E+30	0

the result of a summation depends on the order of how the numbers are summed up

results may change significantly, if a compiler changes the order of operations for optimisation

prefer adding numbers of same magnitude

avoid subtracting very similar numbers

File formats

- samling av tekst og/eller tall i en avtalt rekkefølge
- Vi skiller
  - plain-text <-> binary
  - closed <-> open

Plain-text / binary

- Plain-text formater kan åpnes og leses som ASCII / UTF-8 text
  - .txt, .csv, .md, .py, .html, .css, .c, ...
  - stort sett selvforklarende, enkel tilgang, krever ingen spesiell applikasjon
  - ineffektiv med store mengder tall (23452 krever 5 ascii-bytes, bare 2 bytes som int16)

Plain-text / binary

- Binærformater
  - .doc, .bmp, .jpg, .mp3, .avi
  - krever ekstern dokumentasjon (er verdiene 32bit? 64bit? heltall? float?)
  - uten dok. knyttet til spesifikke programmer som kan tolke filen
  - åpen format <-> lukket format

Plain-text / binary

- Blandete formater
  - .ppm, .fits
  - ofte plaintext **header** med metadata plus binær **body**
  - kan brukes som *selvforklarende* format

Storytime: docx / odf

- Standardisering

- måleinstrument, vekt, ... men også dataformater
- ulike aktører kan implementere / lese / skrive et format

- *Standards bodies* er åpne til interesserte å delta, mer eller mindre formelt
- Vanligvis representanter av bedrifter, land, NGOs som er relatert til temaet

Storytime: docx / odf

året er 2005



Word, Excel, Outlook, Powerpoint





Storytime: docx / odf

året er 2005



Word, Excel, Outlook, Powerpoint

doc, ppt, xls

lukkede binærformater

Storytime: docx / odf

året er 2005



quickly emerged as a significant competitor to Microsoft Office, reportedly achieving 14% penetration in the large enterprise market by 2004

https://en.wikipedia.org/wiki/OpenOffice.org

åpen v1.0 fra 2002

.odf

kan ~lese .doc

Storytime: docx / odf

året er 2005



åpen v1.0 fra 2002

ISO/IEC 26300
(pub. 2006)



.odf

(722 sider A4)

kan ~lese .doc

Storytime: docx / odf

Desember 2005



ECMA TC45

formal standard for office productivity applications that is fully compatible with the Office Open XML Formats, submitted by Microsoft

(6000 sider A4)

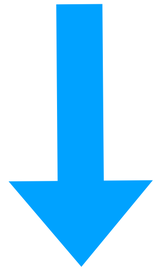
pub. som ECMA-376 i des. 2006

Storytime: docx / odf

2006



ECMA-376



ISO

Storytime: docx / odf

the ISO held a ballot that closed September 2007. This has been observed to be perhaps the most controversial and unusual ISO ballot ever convened, both in the number of comments in opposition, and in unusual actions during the voting process

- Several sections require the implementor to clone the non specified behaviour of a proprietary product:
  - Section 2.15.3.6 page 2161, autoSpaceLikeWord95
  - Section 2.15.3.31 page 2209, lineWrapLikeWord6

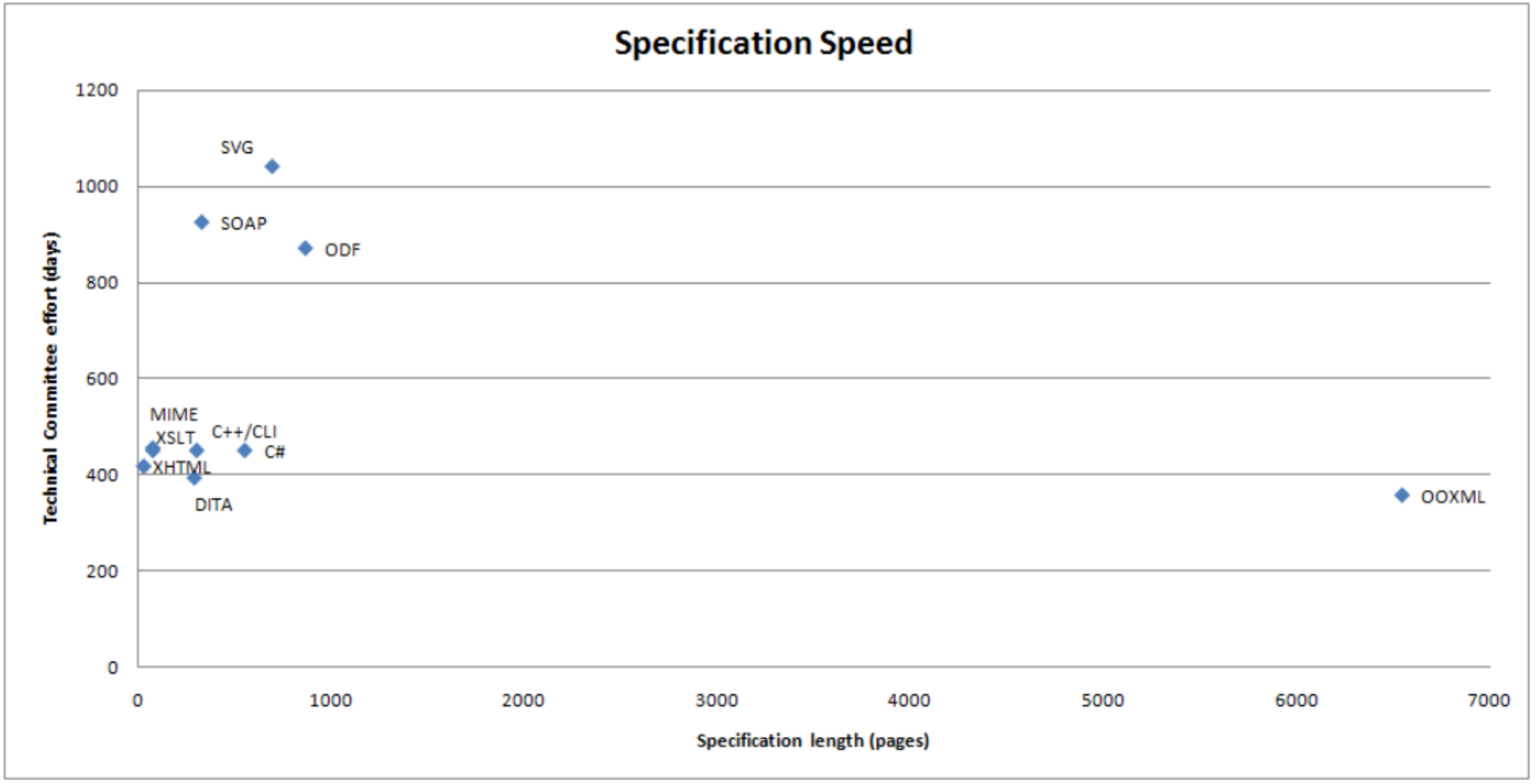
https://en.wikipedia.org/wiki/Standardization_of_Office_Open_XML

Storytime: docx / odf

Norway's vote was decided by Standard Norge; the mostly opposing viewpoints of the technical committee resulted in a disapproval vote in the 2007 ballot. However, the administration of Standard Norge changed Norway's vote to "approval" in 2008 even if the majority of the committee argued in favor of keeping its "disapproval" vote.

Membership in the technical committee had risen from 6–7 to 30 members; all of the pre-OOXML members argued in favour of a "no" vote.

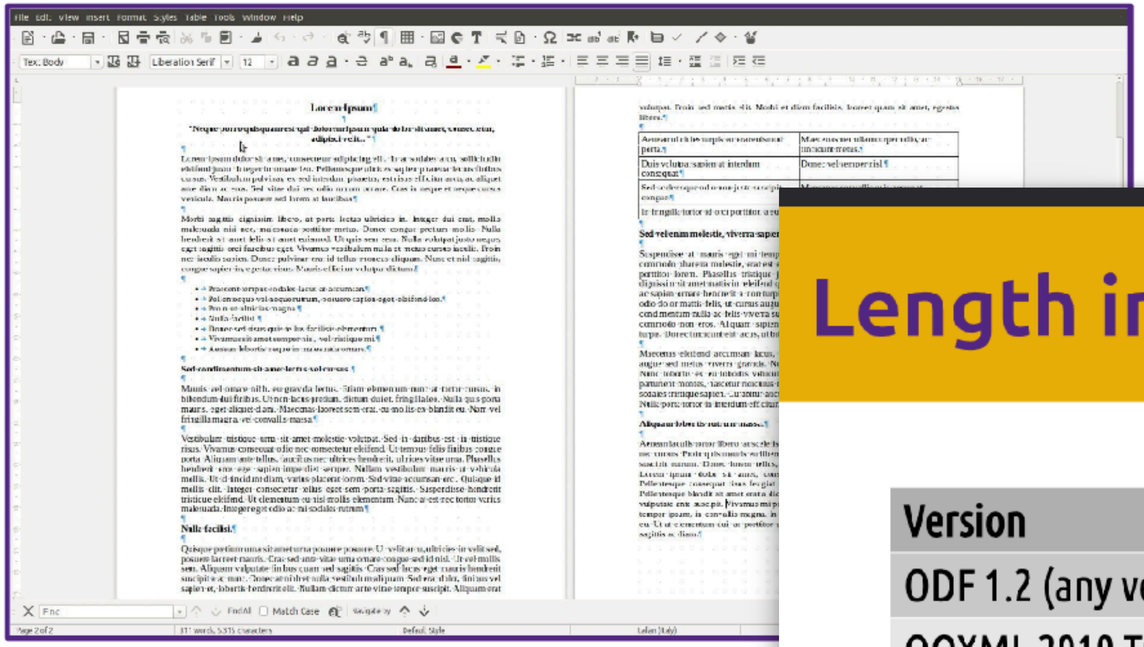




- https://fossforce.com/2026/02/why-ooxml-is-not-a-standard-format-for-office-documents/

- https://blog.documentfoundation.org/wp-content/uploads/2026/02/ODF-Advocacy-OOXML.pdf

Comparison ODF/OOXML



Length in Lines of the XML File

Version	Lines of XML
ODF 1.2 (any version of) LibreOffice	222
OOXML 2010 Transitional (MS Office Windows)	1040
OOXML 2011 Transitional (MS Office MacOS)	12854
OOXML 2013 Transitional (MS Office Windows)	1590
OOXML 2016 Transitional (MS Office Windows)	11667
OOXML 2016 Transitional (MS Office MacOS)	11646
OOXML 2019 Transitional (MS Office Windows)	7085

https://blog.documentfoundation.org/wp-content/uploads/2026/02/ODF-Advocacy-OOXML.pdf