Programmeringsspråket C

- Laget til implementasjon av Unix ved AT&Ts Bell labs i Palo Alto 1969–73.
- Navnet kommer fra BCPL → B → C.
- Opphavsmannnen heter *Dennis Ritchie*.
- ANSI-standard i 1988; omtrent alle følger den nå.
- I 1999 kom C99 og i 2011 kom C11, men ikke særlig mange følger dem fullt ut. Vi vil derfor stort sett ignorere dem.

Formål

- Kunne programmere oversiktlig; lettlest kode.
- Tilgang til maskinens ressurser.
- Lite maskinavhengige programmer.
- Kompakte programmer (dvs lite kode).
- Raske programmer.
- Korte programmer (dvs få linjer).

Cs fortrinn

- Mulig å skrive raske programmer.
- Gode muligheter for strukturering av data og program.
- Svært kompakt kode:

Simula	**C**
n := n+1;	A[++n] *= 3.1;
A[n] := A[n]*3.1;	

- Mulig å skrive elegante, oversiktlige og portable programmer.
- Fast standard (ANSI C) fra høsten 1988.
- Finnes overalt.

Cs svake sider

- Ofte lite portable hvis man ikke tenker på det mens man koder; bedre etter ANSI C.
- C tilbyr programmereren større frihet. Kompilatoren vil dog oppdage færre av de feil programmereren gjør.
- Muligheter for kryptisk kode:

```
A[*(*x)++ = y] += 4;
```

Sagt om C

Å programmere i Java er som å kjøre en Volvo stasjonsvogn; den duver rolig av gårde på veien, men man kommer trygt frem.

Å programmere i C er som å kjøre en Ferrari; den kan gå uhyggelig fort i svingene, men man havner noen ganger i grøften.

— ukjent opphavsmann

I C er det viktigere at det går fort enn at svaret blir riktig!

— Dag Langmyhr

En skrivefeil i C er ingen feil; det er bare et annet program.

— enda en ukjent meningsytrer

Hvorfor er det nyttig å lære C?

Det er flere grunner:

- C er blant de aller mest utbredte språk i dag.
- C brukes i svært mange større programmeringsprosjekter.
- C og Unix er uløselig knyttet sammen.
- Med C kan man skrive raskere kode enn med de fleste andre språk.
- Med C kan man skrive svært kompakt kode (dvs bruke lite minne).
- Programmering i C gir en følelse av hvorledes datamaskinen fungerer.

Et minimalt eksempel

«Alle» lærebøker i programmering har dette eksemplet:

```
#include <stdio.h>

int main(void)
{
  printf("Hallo, alle sammen!\n");
}
```

(Det var Kernighan & Ritchies første bok om C som startet denne moten!)

I Java ser programmet slik ut:

```
class Hello {
  public static void main(String args[]) {
    System.out.println("Hallo, alle sammen!");
  }
}
```

Kompilering

Følgende kommando kan brukes for å kompilere programmet:

```
gcc hallo.c -o hallo
```

Det kompilerte programmet kjøres med

```
hallo
```

eller helst

```
./hallo
```

Program

Et program er en liste av deklarasjoner av variabler og funksjoner:

Java	**C**
⟨Klasse-deklarasjoner⟩	⟨Deklarasjoner⟩

Hovedprogrammet

«Hovedprogrammet» er en funksjon ved navn `main`:

Java	**C**
public static void main(String arg[]) {	int main(void)
	{
⋮	⋮
}	}

Funksjoner

En C-funksjon ligner veldig på en metode i Java. Den består alltid av fire deler:

1. *type* på returverdien; hvis ingen, skriv `void`
2. *navn* på funksjonen
3. *parameterliste* med typeangivelse av hver parameter

   Til forskjell fra Java: hvis det ikke er noen parametre, skrives `void`.
4. *kroppen* som er selve funksjonen. Den er omsluttet av { og }

Tekstkonstanter

Tekstkonstanter skrives med " foran og bak.

Java	C
"En tekst"	"En tekst"

I C må vi ofte vi legge inn spesialtegn i teksten; det vanligste er \n som angir linjeskift.

Java	C
"Hei!\n"	"Hei!\n"

Utskrift

Utskrift skjer via kall på funksjonen `printf`. Eventuelt linjeskift legges inn i teksten.

Java	**C**
System.out.print("Hei, ");	printf("Hei, ");
System.out.println("dere!");	printf("dere!\n");

Utskrift av tall

Med %d i teksten kan man angi at det skal settes inn et tall. Dette tallet må komme senere i parameterlisten.

Java	**C**
System.out.println(a + " og " + b);	printf("%d og %d\n", a, b);

Heltall i C

C har litt andre heltallstyper enn Java:

Navn	Alternativt	Ant byte	Java
`signed char` `unsigned char`	`char` † `char` †	1 1	`byte`
`short` `unsigned short`	`signed short`	2 2	`short`
`int` `unsigned int`	`signed int` `unsigned`	2–4 2–4	
`long` `unsigned long`	`signed long`	4 4	`int`

† Standarden sier at det er udefinert om «`char`» betyr «`signed char`» eller «`unsigned char`», så det varierer.