UNIVERSITETET I OSLO
DET MATEMATISK-NATURVITENSKAPELIGE FAKULTET

Programmeringsspråket C

Bakgrunn

- Implementasjon av Unix ved AT&Ts laboratorium i Palo Alto 1960–75.
- Navnet kommer fra BCPL → B → C.
- Opphavsmannnen heter *Dennis Ritchie*.
- ANSI-standard i 1988; omtrent alle følger den nå.
- I 1999 kom C99, men ikke alle følger den. Vi vil derfor stort sett ignorere den.

Formål:

- Kunne programmere oversiktlig; lettlest kode.
- Tilgang til maskinens ressurser.
- Lite maskinavhengige programmer.
- Kompakte programmer.
- Raske programmer.

INF1070

Cs fortrinn

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

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

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 av og til 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 språket 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 med følgende lille eksempel:

```
#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
./hallo
```

Forklaring

Program

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

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

Hovedprogrammet

«Hovedprogrammet» er en funksjon ved navn main:

Java	C
public static void main(. . .) {	int main(void)
⋮	{
}	⋮
	}

Store og små bokstaver

Det er forskjell på store og små bokstaver i C. MAIN, Main og main er tre helt ulike navn.

Funksjoner

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

- *type* på returverdien. Hvis ingen returverdi, skrives void.
- *navn* på funksjonen.
- *parameterliste* med typeangivelse av hver parameter.

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

  Returverdien angis med en return-setning.

Tekstkonstanter

Tekstkonstanter skrives med " foran og bak.

Java	C
"En tekst"	"En tekst"

I C kan 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

På samme måte som Java har C diverse heltallstyper:

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

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

Operatorer

Aritmetiske operatorer

C har de vanlige aritmetiske operatorene:

+	Addisjon
-	Subtraksjon
*	Multiplikasjon
/	Divisjon
%	Modulo (rest ved divisjon)

Disse kan også brukes til oppdatering av variable:

Koden . . .	**. . . gir det samme som . . .**
a += x;	a = a + x;
a -= x;	a = a - x;
⋮	⋮

Sammenligninger

Sammenligningsoperatorene er også de samme som i Java.

==	Likhet
!=	Ulikhet
<	Mindre enn
<=	Mindre enn eller lik
>	Større enn
>=	Større enn eller lik

Disse operatorene gir 1 om sammenligningen holder og 0 ellers.

NB! Ikke bland sammen = og ==!

Logiske verdier

Før C99 fantes ingen type boolean i C! I stedet brukes heltall der 0 er **false** og alle andre verdier er **true**.

Logiske operatorer

! a	1 om a = 0 0 ellers
a && b	1 om a ≠ 0 og b ≠ 0 0 ellers
a || b	1 om a ≠ 0 eller b ≠ 0 0 ellers

Maskeoperatorer

~	not
&	and
|	or
^	xor

NB! Det er forskjell på logiske og maskeoperatorer! For eksempel er

1 && 4 gir 1
1 & 4 gir 0

INF1070

Et litt større eksempel

```
#include <stdio.h>

/* Hvor mange hele fot utgjør disse tommene? */
int finn_fot(int tommer)
{
  return tommer/12;
}

/* Hvor mange rene tommer er det i dette målet? */
int rene_tommer(int tommer)
{
  int fot = tommer/12;

  return tommer - 12*fot;
}

int main(void)
{
  int tom;

  printf("Angi et mål i tommer: ");
  scanf("%d", &tom);
  while (tom != 0) {
    if (tom > 0 && tom <= 999)
      printf("%5d tommer = %d fot %d tommer\n",
             tom, finn_fot(tom), rene_tommer(tom));
    else
      printf("Programmet godtar kun verdier 0-999!\n");

    printf("Gi et nytt mål i tommer (0 for avslutning): ");
    scanf("%d", &tom);
  }
}
```

Kommentarer

Kommentarer omgis av /* og */. De kan stå hvor som helst.

Deklarasjon av variable

Variable kan deklareres først i en funksjon eller mellom funksjonene. En deklarasjon består av tre deler:

- Variabelens *type*.
- Variabelens *navn*. Flere variable kan deklareres, adskilt av komma.
- En *initialverdi* for sistnevnte variabel; kan droppes. En variabel uten angitt initialverdi får en *tilfeldig* startverdi.

Innlesning

Til innlesning brukes scanf. Første parameter angis hva som skal leses inn: %c for tegn, %d for heltall og %f for flyt-tall.

Legg merke til & foran variabelnavnet; den *må* være der.

Setninger

Tilordning

I C og Java brukes = for tilordning.

If-setninger

If-setninger er som de fleste andre språk.

While-løkker

While-løkker er også som i de fleste andre språk.

Lagring av tegn

C har ingen egen type for å lagre *tegn* (som char i Java). I stedet benyttes heltall.

Hvilken koding som brukes, vil variere fra én maskin til en annen. I den vestlige verden brukes stort sett nå **ISO 8859-1**, også kjent som **ISO Latin-1**; om noen år blir det **Unicode**.