

Programmering i C

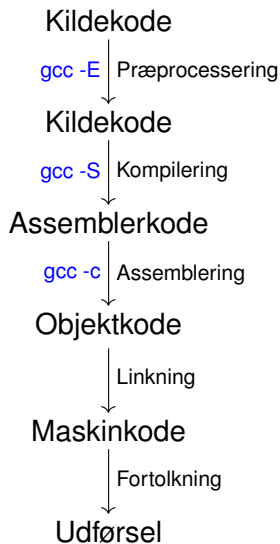
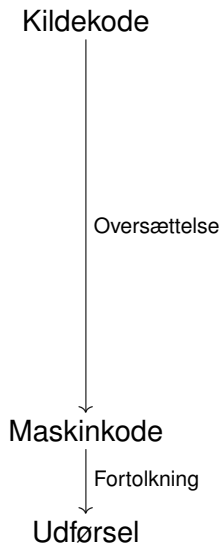
Lektion 2

20 oktober 2008

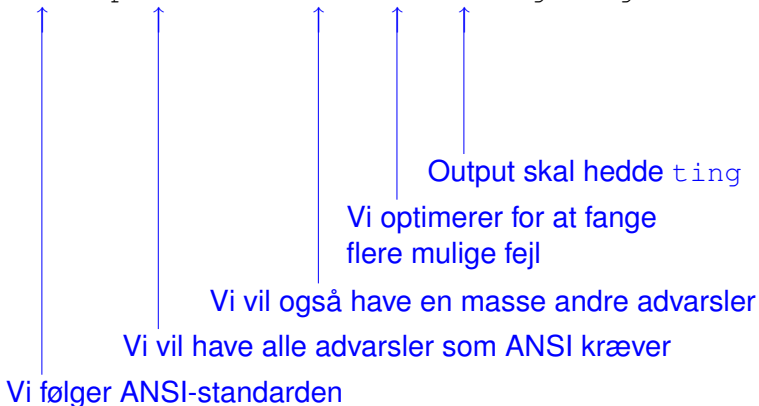
Fra sidst

- 1 Historie
- 2 Processen
- 3 At compilere





```
gcc -ansi -pedantic -Wall -O -o ting ting.c
```

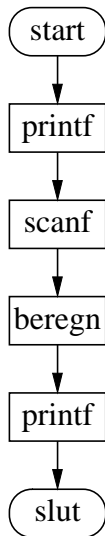


Kontrolstrukturer

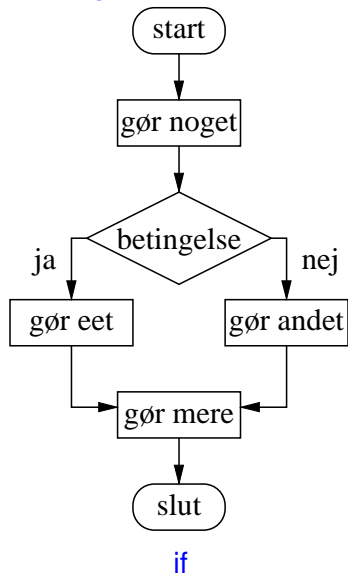
- 4 Sekventiel kontrol
- 5 Logiske udtryk
- 6 Short circuit evaluering
- 7 Udvælgelse af kommandoer

```
#include <stdio.h>
```

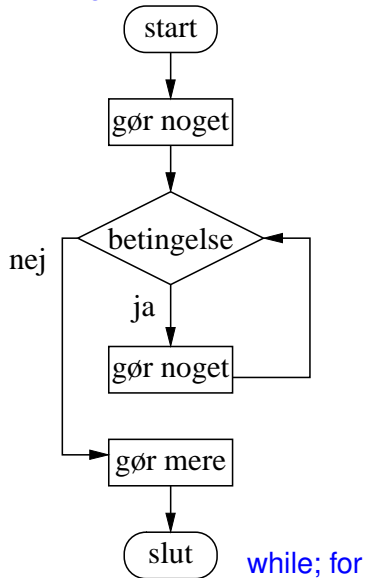
```
int main( void ) { /* seconds.c */  
    long int input, temp, h, m, s;  
    printf( "Giv mig et heltal!\n" );  
    scanf( "%ld", &input );  
    h= input/ 3600;  
    temp= input- h* 3600;  
    m= temp/ 60;  
    s= temp% 60;  
    printf( "\n%ld sekunder svarer til \  
%ld timer, %ld minutter og %ld sekunder\n",  
           input, h, m, s );  
    return 0;  
}
```



Udvælgelse af kommandoer:



Gentagelse af kommandoer:



Udvælgelse: `if(logisk udtryk)`

Gentagelse: `while(logisk udtryk)`

Logiske udtryk:

- $x < y$, $x \leq y$, $x \geq y$, $x > y$, $x != y$, $x == y$
- `!A`, `A && B`, `A || B`, hvor `A` og `B` selv er logiske udtryk
- har værdien *falsk* (0) eller *sandt* (1, i de fleste(!) compilere)
- `&&` har højere prioritet end `||`

⇒ brug parenteser!

(Hvad er værdien af `3==5 || 1==1 && 1==2 ?...`)

[oper.c]

```
# include <stdio.h>
```

```
int main( void ) { /* lighed.c */  
    int a, b, lig;
```

```
    printf( "Vi sammenligner to tal.\n\  
Output 0 betyder at de er forskellige.\n\  
Må jeg bede om to heltal?\n" );  
    scanf( "%d %d", &a, &b);
```

```
    lig= a== b; /* bedre med parenteser... */
```

```
    printf( "\nOutput: %d\n", lig );  
    return 0;  
}
```

Observation:

- Hvis A er falsk, da er $A \& \& B$ også falsk
- Hvis A er sandt, da er $A || B$ også sandt

⇒ i udtrykket $A \& \& B$ beregnes B kun hvis A er sandt
– og i udtrykket $A || B$ beregnes B kun hvis A er falsk

- Smart, men kilde til fejl

```
# include <stdio.h>
```

```
int main( void) { /* lighed2.c */
```

```
    int a, b;
```

```
    char lig;
```

```
    printf( "Vi sammenligner to tal.\n\n\  
Må jeg bede om to heltal?\n");
```

```
    scanf( "%d %d", &a, &b);
```

```
    (a== b) &&( lig= ' ');
```

```
    (a!= b) &&( lig= 'u');
```

```
    printf( "%d er %clig %d\n", a, lig , b);
```

```
    return 0;
```

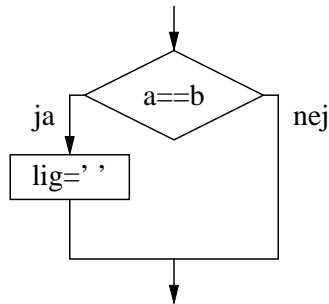
```
}
```

Udvælgelse med &&:

- `(a == b) &&(lig= ' ');`
- kryptisk...

Udvælgelse med if:

- `if(a == b) lig= ' ';`
- det var bedre!

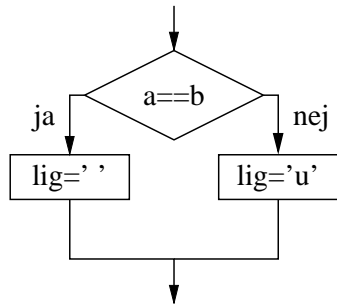


Udvælgelse med &&:

- `(a== b) &&(lig= ' ');`
- kryptisk...

Udvælgelse med if:

- `if(a== b) lig= ' ';`
- det var bedre!
- `if(a==b) lig= ' ';`
`else lig= 'u';`



if (udtryk) kommando1; **else** kommando2;

- først beregnes **udtryk**
- hvis **udtryk** er sandt, udføres **kommando1**
- hvis **udtryk** er falsk, udføres **kommando2**

```
# include <stdio.h>
```

```
int main( void) { /* lighed2.c */
```

```
    int a, b;
```

```
    char lig;
```

```
    printf( "Vi sammenligner to tal.\n\n\  
Må jeg bede om to heltal?\n");
```

```
    scanf( "%d %d", &a, &b);
```

```
    if( a== b) lig= ' ';
```

```
    else lig= 'u';
```

```
    printf( "%d er %clig %d\n", a, lig, b);
```

```
    return 0;
```

```
}
```

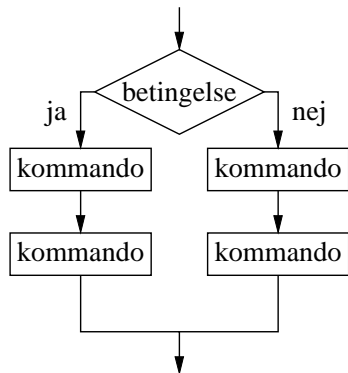
Kontrolstrukturer, 2.

- 8 Kommandoblokke; scope
- 9 Udvælgelse med if, 2.
- 10 Udvælgelse med switch
- 11 Gentagelse med while
- 12 Gentagelse med for

Problem: Vil gerne udvælge mellem to
blokke af kommandoer

Løsning: **Sammensætning** af
kommandoer:

```
if ( a== b )  
{  
    c= 1;  
    d= 2;  
} } blok  
  
else  
{  
    c= 7;  
    d= 5;  
} } blok
```



- blok = antal kommandoer omkranset af { og }
- en blok behandles som **én** kommando
- blokke kan indlejres i hinanden
- i starten af en blok kan variabelerklæringer forekomme
- !! disse variable er **lokale** for blokken (deres **scope** er blokken)

```
#include <stdio.h>
int main(void){ /* blok.c*/
    int a=5;
    printf("Før: a==%d\n",a);

    { /*en blok*/
        int a=7; /* deklARATION*/
        printf("I: a==%d\n",a);
    }

    printf("Efter: a==%d\n",a);

    return 0;
}
```

```
#include <stdio.h>
int main(void){ /* blok2.c*/
    int a=5;
    printf("Før: a==%d\n",a);

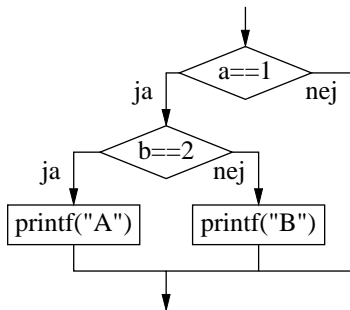
    { /*en blok*/
        a=7; /* assignment!*/
        printf("I: a==%d\n",a);
    }

    printf("Efter: a==%d\n",a);

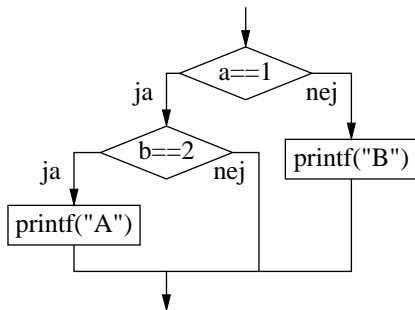
    return 0;
}
```

“Dangling else”-problemet:

```
if ( a== 1 )
    if ( b== 2 )
        printf ( "A" );
    else
        printf ( "B" );
```



```
if ( a== 1 ) {
    if ( b== 2 )
        printf ( "A" );
}
else
    printf ( "B" );
```



- en **else** knytter sig altid til den **inderste if**
- brug kommandoblokke for at undgå tvivl!

Hvad hvis der er flere end to valgmuligheder? Brug switch !

```
#include <stdio.h>
```

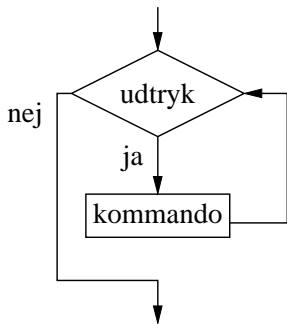
```
int main( void ) { /* switch.c */  
    int a;  
    char * dyr;  
    printf( "Giv mig et heltal!\n" );  
    scanf( "%d", &a);  
  
    switch( a) {  
    case 1: dyr= "hest"; break;  
    case 2: dyr= "gris"; break;  
    case 3: dyr= "abe"; break;  
    default: dyr= "ko"; break;  
    }  
  
    printf( "\nDu er en %s!\n", dyr );  
    return 0;  
}
```

```
switch( udtryk ) {  
  case const1: command1;  
  case const2: command1;  
  ...  
  case constN: commandN;  
  default: command;  
}
```

- først beregnes **udtryk**. Resultatet skal være et heltal eller noget der ligner (f.x. en `char`)
- **udtryk** = **const_i** \Rightarrow **command_i** udføres. Herefter udføres **command_{i+1}** osv.
- **udtryk** \neq **const_i** for alle $i \Rightarrow$ default-kommandoen udføres, og herefter de efterfølgende! Hvis der ingen **default** er, gøres ingenting.
- man ønsker næsten altid at afslutte et **case** med en **break**-kommando; så springes de efterfølgende kommandoer over.

while (udtryk) kommando ;

- først beregnes **udtryk**
- hvis **udtryk** er sandt, udføres **kommando**, og løkken startes forfra
- hvis **udtryk** er falsk, afsluttes løkken



```
#include <stdio.h>
int main( void ) { /* while.c */
    int h= 0;
    while( h!= 1234) {
        printf("Indtast det hemmelige heltal: ");
        scanf( "%d", &h);
    }
    printf( "\nHurra!\n");
    return 0;
}
```

for(*init*; *condition*; *update*) *kommando*;

(den mest generelle løkkekonstruktion i C)

- 1 først udføres *init*
- 2 så beregnes *condition*, og hvis den er falsk, afbrydes
- 3 *kommando* udføres
- 4 *update* udføres, og vi springer tilbage til trin 2.

```
#include <stdio.h>
```

```
int main( void) { /* for.c */
```

```
    int i= 1;
```

```
    printf( "%d elefant\n", i);
```

```
    for( i= 2; i<=10; ) {
```

```
        printf( "%d elefanter\n", i);
```

```
        i++;
```

```
    }
```

```
    return 0;
```

```
}
```