>>> C-Programmering for begyndere
>>> Del 2 - Operatorer, conditionals, loops og forgreninger

Name: Jacob B. Pedersen[†] og Jakob S. Nielsen[‡]
Date: 16. april 2018

[†]jacob.bp@mvb.net
[‡]jakob990@gmail.com

>>> Indhold

>>> Hvad lavede vi sidste gang?

* Vi fik sat vores første program op
  * Hello World
* Stiftede bekendtskab med et par datatyper
  * `int`, `float`, `char`, `bool`
* Vi tog input, og skrev ud på konsollen
  * `scanf()` og `printf()` fra `stdio.h`

>>> Hello World

* Begynderprogrammet
* Indeholdt main(), stdio.h, printf() og en return 0

```
1  #include <stdio.h>
2
3  int main(void)
4  {
5     printf("Hello world\n");
6     return 0;
7  }
```

>>> Datatyper

* Et lille cheatsheet over datatyperne i C:

```c
int i; // Integer/heltal - *mindst* 16 bits
       // Fra [-32767 til +32767]
float f; // Floating point decimal - 32 bits
       // *mindst* 6 betydende cifre
char ch; // Character - *mindst* 8 bits
       // ASCII dækker fra 0-128 forskellige tegn
bool b; // Boolean - true/false - *mindst* 8 bits
       // Kun 0 anses som Boolsk "false" ved tilskrivning
       // Skal hentes med #include <stdbool.h>
```

>>> Datatyper

* Lidt flere udvidede datatyper:

```
1       unsigned int ui; // Integer/heltal - *mindst* 16 bits
2            // [0 til +65535]
3       short int f; // Short integer - *mindst* 16 bits
4            // [ 32767  til +32767]
5       long int li; // Long integer - *mindst* 32 bits
6            // [ 2147483647  til +2147483647]
7       double db; // Double precision float - *mindst* 64 bits
8            // *mindst* 10 betydende cifre
9       long double ldb; // Double precision float - *mindst* 128
           bits - *mindst* 10 betydende cifre
```

>>> Input og output

* Til input og output bruges #include <stdio.h>
  * scanf() tager input, eks:

```
1   scanf("%d %d", &variable1, &variable2);
```

  * printf() skriver output, eks:

```
1   printf("Variable 1: %d Variable 2: %d", variable1,
        variable2);
```

* Vi bruger format specifiers til at fortælle printf() og scanf() hvilken datatype det skal fortolkes som

```
1   "%d" // Integer format specifier - printer/scanner heltal
2   "%f" // Float format specifier - printer/scanner kommatal
3     "%.2f" // Float med 2 decimaltal
4   "%c" // Char format specifier - printer/scanner skrifttegn
5   "%s" // String format specifier - printer/scanner
        skriftstrenge
```

>>> Dagens program

* I dag vil vi kigger på følgende:
* Operatorer
  * `<`, `>`, `==`, `!=`, `&&`, `||`, `++`, `--`
* If-else udtryk
  * `if()`, `else if()`, `else()`
* Løkker
  * `while()`, `for()`
* Switch-cases
  * `switch()`

>>> Operatorer

* Vi kender allerede de basale, vi brugte til udregninger mm.
  * +, -, *, /
  * =, !
* Men vi har også brug for at sammenligne værdier
* Vi kender det f.eks. fra ligninger og uligheder
  * >, <, >=, <=
* Når C ser dette:

```
1     12 > 10
```

* Erstattes det med en Boolsk værdi (sandt eller falsk):

```
1     true
```

>>> Operatorer

* Hvilke operatorer har vi så?

```
1       == // Lighed - er værdierne lig hinanden?
2       != // Ikke lig - er værdierne forskellige?
3       > // Større - Er værdien til venstre større?
4       < // Mindre - Er værdien til venstre mindre?
5       >= // Større/lig - Er værdien til venstre større/lig?
6       <= // Mindrel/lig - Er værdien til venstre mindre/lig?
7       || // Or - Bruges til at samle udtryk (er A > B ELLER er
            C > B)
8       && // And - Bruges til at samle udtryk (er A > B OG er C
            > B)
```

>>> Operatorer

* Foruden sammenligningerne er der også et ekstra sæt fikse operatorer til aritmetik og tilskrivning:

```
1       += // Tilskriver venstrehåndsværdien dens egen +
            højrehåndsværdien
2       -= // Tilskriver venstrehåndsværdien dens egen -
            højrehåndsværdien
3       ++ // Inkrementerer en variabel eks:
4          i++; // Evaluerer i, og inkrementerer med n efter
5          ++i; // Inkrementerer først, og evaluerer herefter i
6       -- // Dekrementerer en variabel på samme måde
```

>>> If-Else udtryk

* Programmer skal også kunne noget intelligent
* De skal have indbygget logik!
* Den mest basale måde, er at bruge conditionals:
    * `if()`
    * `else if()`
    * `else()`

>>> If-Else udtryk

* Hvis vi skal illustrere det:

```
1       if ([sandt/falsk udtryk])
2       {
3          handling();
4       }
5       else if ([andet udtryk])
6       {
7          andenHandling();
8       }
9       else
10      {
11         tredjeHandling();
12      }
```

>>> Løkker

* En anden vigtig kontrolstruktur er loopet. Løkken.
* Hvor et stykke af programmet gentager sig, mens betingelser er opfyldte
* I C er der to basale løkketyper:
  * `while()` og `for()`

>>> Løkker

* En while() fungerer således:

```
1       while([betingelse])
2       {
3          handling();
4       }
```

* En for() fungerer således:

```
1       for(int j = 0; j < 5; j++)
2       {
3          handling();
4       }
```

* herudover eksisterer to brugbare nøgleord:

```
1       continue; // Springer videre til næste loopiteration
2       break; // Bryder ud af loop (og scope)
```

>>> Switch-cases

* Sidste ting i dag: Switch-case
* Før forgrenede vi vha. if-else udtryk
* switch() kigger dog ikke på en evaluering
  * Den betragter værdierne dens input-variabel har
  * Og sender programmet til et sted i koden, på baggrund deraf
  * Det virker godt til forgrening på baggrund af brugerinput!

>>> Switch-cases

* En switch() fungerer således:

```
1       switch(variabel)
2       {
3          case 1:
4          handling();
5          break;
6
7          case 2:
8          andenHandling();
9          break;
10
11         default:
12         tredjeHandling();
13         break;
14      }
```

>>> Kreative Opgaver

* Det var det for nu!
* Der ligger som sidst kreative opgaver tilgængelige:
  * ../Del_2/Exercises/C_exercises_2_dansk.pdf
* Der er hjælp at hente her på workshoppen
* God arbejdslyst! - Happy Hacking!