

# Hardwarenær programmering E16

Uge 3, funktioner, enums og type cast

Afleveret 15. September 2016

David Bjerre Bjørklund  
s113070

# Formål og krav

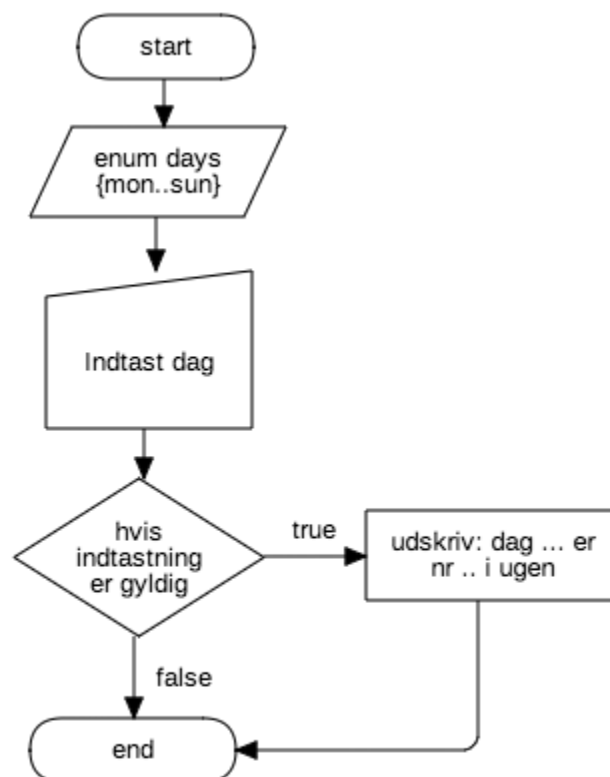
Ugens opgaver omhandlende enums, funktioner og typecast.

I første program indtaster brugeren navnet på en ugedag, herefter returnere programmet månedens tal. Programmet skal benytte enum.

Anden og tredje opgave består i at lave en simpel lommeregner. To tal indtastes og brugeren vælger om de skal multipliceres eller adderes. Resultatet udskrives. Første gang skrives programmet uden brug af funktioner, anden gang skrives programmet med.

Sidste opgave omhandler typecasting. Altså konvertering fra en 'højere' til en 'lavere' type.

## Enums, enums, enums!!! (opgave 1)



Denne opgave benytter enums. Mere præcist kræver denne opgave at et char-array "konverteres" til en enum reference, der så refererer til et tal. C indeholder ikke nogen direkte måde at gøre dette, så dette implementeres som et opslag. Herefter benyttes enum på sædvanlig vis. Ovenstående flowchart viser i grove træk algoritmen.

Enum kan beskrives som en type der benytter ord, som reference. Udover at gøre koden mere læselig har enums ingen praktisk funktion, på dette niveau.

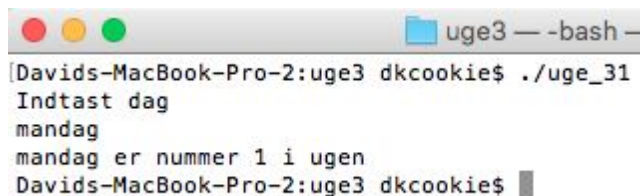
Til sammenligning benytter vi `strcmp(char*,char*)` fra biblioteket `string.h` der kan sammenligne to `char`-arrays, fremover refereret som strings, og returnere et tal forskelligt fra nul, når strengene er forskellige og 0 når de er ens. Derfor negerer vi returværdien fra `strcmp()`.

```
// Declare enum
enum days retday;
//uses strcmp from string.h to compare input to predefined strings.
if(!(strcmp(input,"mandag")))
{
    retday = mon;
    //Return enum with corresponding value
    return retday;
}
```

Resten af programmet er vedlagt som bilag.

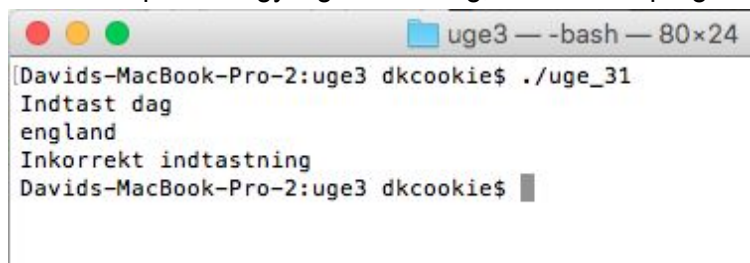
## Afprøvning

Programmet er testet ved at efterprøve alle ugedagene, og validere at disse stemmer overens med dagens nr.



```
David's-MacBook-Pro-2:uge3 dkcookie$ ./uge_31
Indtast dag
mandag
mandag er nummer 1 i ugen
David's-MacBook-Pro-2:uge3 dkcookie$
```

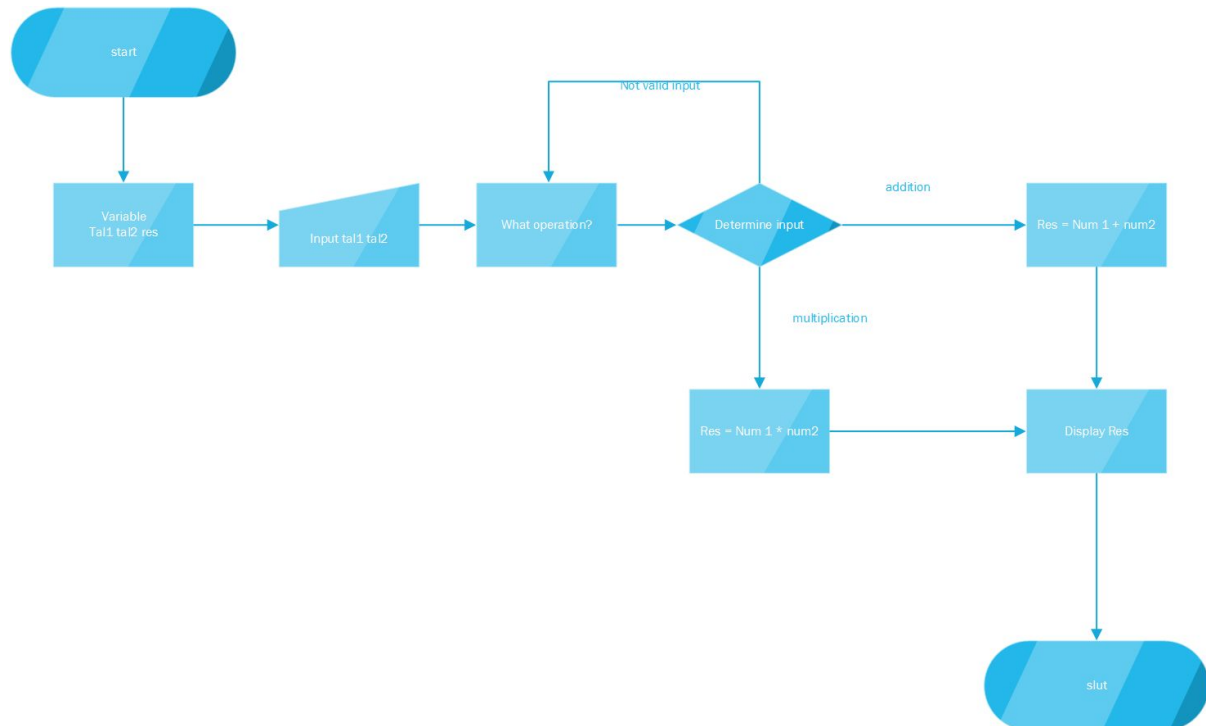
Herefter afprøves ugyldig indtastning, som i dette program, blot afslutter.



```
David's-MacBook-Pro-2:uge3 dkcookie$ ./uge_31
Indtast dag
england
Inkorrekt indtastning
David's-MacBook-Pro-2:uge3 dkcookie$
```

# Lommeregner (opgave 2 og 3)

Lommeregneren benytter ikke funktionalitet udover hvad der er dokumenteret i kommentarerne. Der henvises til koden som ligger i bilag. Udover funktioner er programmet identisk



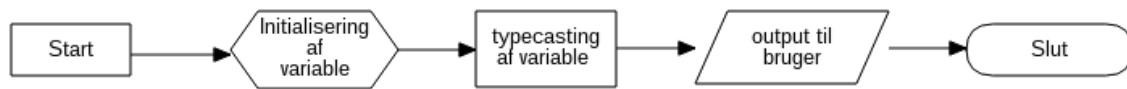
## Afprøvning

Programmet skal tage to inputs og levere et output. Dette testes for addition og multiplikation.

```
u3e3 — -bash —
[Dauids-MacBook-Pro-2:uge3 dkcookie$ ./uge_33
Velkommen til lommeregneren!
Indtast det første tal: 10
Indtast det andet tal: 30
10.00
30.00
Should they be:
1: Added together?
2: Or multiplied?
1
40.00
[Dauids-MacBook-Pro-2:uge3 dkcookie$ ./uge_33
Velkommen til lommeregneren!
Indtast det første tal: 10
Indtast det andet tal: 30
10.00
30.00
Should they be:
1: Added together?
2: Or multiplied?
2
300.00
Dauids-MacBook-Pro-2:uge3 dkcookie$
```

```
input.txt
output.txt
10
40
1
Velkommen til lommeregneren!
Indtast det første tal:
Indtast det andet tal: 10.00
40
2
40.00
Should they be:
1: Added together?
2: Or multiplied?
50.00
```

## Konverterer mellem typer (opgave 4)



Typecasting benyttes når en type skal konverteres til en anden. Man kan forfremme lavere variabler til højere, f.eks. float til double og ligeledes degradere fra højere til lavere variabler, f.eks. fra float til int. Hertil bemærkes der at ved sidstnævnte, går information tabt.

Sidste del omkring null-pointere, viser at disse kan castet som alle andre typer, men dette er udover omfanget af denne opgave.

## Afprøvning

```
David's-MacBook-Pro-2: uge3 — -bash — 80x24
[David's-MacBook-Pro-2: uge3 dkcookie$ ./uge_34
We start our typecasting with some simple integers
x = y + 6;
y = (int)x/4;

then we show that typecasting is not necessary, when going from a simple to an advanced type
a = x / 3.2;
b = x + y;

Now we try with char and unsigned
This is our char: A
now we print it as unsigned: 0x00000041

Here we use null pointers, which is an amazing type who can be typecasted to all types
Now our void pointer points to the float: 1.875000
Now our void pointer points to the char: A
Now our void pointer points to the int: 6
David's-MacBook-Pro-2: uge3 dkcookie$
```

## Appendix Fase 1

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

// Enum is declared
enum days {mon=1, tue, wed, thu, fri, sat, sun};

// Prototype for function scan_input
enum days scan_input(char* input);

int main(){

    // Declare char array (string)
    char input[8];

    // Ask for input
    printf("Indtast dag\n");
    scanf("%s",input);

    // Translate string input to enum reference
    enum days scanDays = scan_input(input);

    // If an error has occurred, returns an error, and closes the program.
    // That is if scan_input returns 0;
    if(!scanDays)
    {
        printf("Inkorrekt indtastning\n");
        return 1;
    }

    // Udskriv brugerens input samt denne dag nr.
    printf("%s er nummer %d i ugen\n",input,scanDays);

    return 0;
}

/* Function enum days scan_input(char* input)
Arguments: takes a string
Returns enum reference corresponding to string.
*/
enum days scan_input(char* input)
{
    // Declare enum
    enum days retday;
    //uses strcmp from string.h to compare input to predefined strings.
    if(!(strcmp(input,"mandag")))
    {
        retday = mon;
        //Return enum with corresponding value
        return retday;
    }
    else if(!(strcmp(input,"tirsdag")))
    {
        retday = tue;
        return retday;
    }
    else if(!(strcmp(input,"onsdag")))
    {
        retday = wed;
        return retday;
    }
}
```

```
else if(!(strcmp(input,"torsdag")))
{
    retday = thu;
    return retday;
}
else if(!(strcmp(input,"fredag")))
{
    retday = fri;
    return retday;
}
else if(!(strcmp(input,"lørdag")))
{
    retday = sat;
    return retday;
}
else if(!(strcmp(input,"søndag")))
{
    retday = sun;
    return retday;
}
else
{
    // Return an 'error'
    return 0;
}
}
```



## Appendix Fase 2

```
/*
```

```
Uge 3 opgave 2
```

```
This is a simple calculator taking two numbers and multiply or add them together.
```

```
Gruppe "taem awesome" (ja det er stavet forkert med vilje)
```

```
s153460 Jonas Ladefoged Holm
```

```
s113070 David Bjerre Bjørklund
```

```
s164920 Markus Visvaldis Ingemann Thieden
```

```
*/
```

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

```
#include <math.h>
```

```
int main (void){
```

```
    //Declare variables
```

```
    printf("Welcome to the calculator!\n");
```

```
    double tal1;
```

```
    double tal2;
```

```
    double res;
```

```
    // User input
```

```
    printf("Input first number: ");
```

```
    scanf(" %lf", &tal1);
```

```
    printf("Input second number: ");
```

```
    scanf(" %lf", &tal2);
```

```
    printf("%.2lf\n", tal1);
```

```
    printf("%.2lf\n", tal2);
```

```
    printf("Should they be:\n1: Added together? \n2: Or multiplied?\n");
```

```
    // To exercise in the functionality of enums...
```

```
    enum brugerValg {intet, multiplied, added};
```

```
    enum brugerValg currentValg = noget;
```

```
    while (currentValg != multiplied && currentValg != added){
```

```
        int valg;
```

```
        scanf(" %d", &valg);
```

```
        // Training the use of enums...
```

```
        if (valg == 1){
```

```
            currentValg = added;
```

```
        } else if (valg == 2) {
```

```
            currentValg = multiplied;
```

```
        } else {
```

```
            printf("Wrong answer try again!\n");
```

```
        }
```

```
    }
```

```
    // Doing the calculations
```

```
    switch (currentValg) {
```

```
        case added:
```

```
            res = tal1 + tal2;
```

```
            break;
```

```
        case multiplied:
```

```
        res = tal1 * tal2;
        break;
    case intet:
        break;
}

// Print result
printf("%.2lf\n",res);

return(0);
}
```

## Appendix Fase 3

```
/*  
  
Uge 3 opgave 2  
This is a simple calculator taking two numbers and multiply or add them  
together.  
Gruppe "taem awesome" (ja det er stavet forkert med vilje)  
s153460 Jonas Ladefoged Holm  
s113070 David Bjerre Bjørklund  
s164920 Markus Visvaldis Ingemann Thieden  
*/  
  
#include <stdio.h>  
#include <math.h>  
  
//Function prototypes  
double addition(double a, double b);  
double multiplication(double a, double b);  
  
int main (void){  
  
    printf("Velkommen til lommeregneren!\n");  
  
    double tal1;  
    double tal2;  
    double res;  
  
    printf("Indtast det første tal: ");  
    scanf(" %lf", &tal1);  
  
    printf("Indtast det andet tal: ");  
    scanf(" %lf", &tal2);  
  
    printf("%.2lf\n", tal1);  
    printf("%.2lf\n", tal2);  
  
    printf("Should they be:\n1: Added together? \n2: Or multiplied?\n");  
  
    // Training the use of enums...  
    enum brugerValg {intet, multiplied, added};  
    enum brugerValg currentValg = noget;  
  
    while (currentValg != multiplied && currentValg != added){  
        int valg;  
        scanf(" %d", &valg);  
        if (valg == 1){  
            currentValg = added;  
        } else if (valg == 2) {  
            currentValg = multiplied;  
        } else {  
            printf("Wrong answer try again!\n");  
        }  
    }  
  
    // Doing the calculations  
    switch (currentValg) {  
        case added:  
            res = addition(tal1,tal2);  
            break;  
        case multiplied:
```

```
        res = multiplication(tal1,tal2);
        break;
    case intet:
        break;
}

// Print result
printf("%.2lf\n",res);

return(0);
}

// Takes two arguments and returns sum
double addition(double a, double b){
    return a + b;
}

// Takes two arguments and returns product.
double multiplication(double a, double b){
    return a * b;
}
```

## Appendix Fase 4

```
/*
Uge 3 opgave 4
This is short showcase of typecasting, with examples.
Gruppe "taem awesome" (ja det er stavet forkert med vilje)
    s153460 Jonas Ladefoged Holm
    s113070 David Bjerre Bjørklund
    s164920 Markus Visvaldis Ingemann Thieden
*/

#include <stdio.h>
#include <stdlib.h>

// We initialise our variables
void* typeless;
int x, y = 0;
float a, b = 0;
char ch = 'A';

int main()
{
    // we start with som simple calculations
    printf("We start our typecasting with som simple integers\n");
    printf("x = y + 6;\ny = (int)x/4;\n\n");
    x = y + 6;
    // we use float to redefine a int, this is only possible using
    //   typecasting
    y = (int)x/4;

    // it's not nessesary to typecast a simple type to a more advanced type
    //   a.i. int to float
    printf("then we show that typecasting is not nessary, when going from a
    //   simple to a advanced type\n");
    printf("a = x / 3.2;\nb = x + y;\n\n");
    a = x / 3.2;
    b = x + y;

    // we use chars and typecast it as unsinged int
    printf("\n Now we try with char and unsinged\n");
    printf("This is our char:  %c\n",ch);
    printf("now we print it as unsigned: 0x%08x\n\n",(unsigned int)ch);

    // Here we use null pointer which can be typecasted as all types
    printf("Here we use null pointers, which is an amazing type who can be
    //   typecasted to all types\n");
    typeless = &a;
    printf("Now our void pointer points to the float:  %f\n", *(float*)
    //   typeless);
    typeless = &ch;
    printf("Now our void pointer points to the char:  %c\n", *(char*)
    //   typeless);
    typeless = &x;
    printf("Now our void pointer points to the int:  %d\n", *(int*)typeless)
    //   ;

    return 0;
}
```