>>> C-Programmering for begyndere
>>> Del 3 - Funktioner, arrays, datarepræsentationer

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 gennemgik operatorer, og deres betydning for if-else statements:

```
1       if([betingelse]){
2          handling();
3       }
4       else if({betingelse2]){
5          handling2();
6       }
7       else{
8          handling3();
9       }
```

>>> Hvad lavede vi sidste gang?

* Operatorerne selv evalueredes blot til Boolske udtryk som `true` og `false`:

```
1       1 > 2; // false
2       3 >= 3; // true
3       12 > 2 && 12 < 11; // false
4       12 > 2 || 12 < 11; // true
```

>>> Hvad lavede vi sidste gang?

* På samme måde kunne de også bruges som betingelser i løkkerne
* Her var der while() og for() -løkkerne:

```
1       int i = 0;
2       while(i <= 10){
3          printf(  % d , i);
4          i++;
5       }
6
7       for(int j = 0; j <= 10, j++){
8          printf(  % d , j);
9       }
```

>>> Hvad lavede vi sidste gang?

* Til sidst kiggede vi på switch-cases:

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

>>> Dagens program

* I dag kigger vi på mere avanceret brug af C:
* Funktioner
  * Genopfriske argumenter og retur-værdier
  * Header- og implementationsfiler
  * Hvad skal der til for at lave et library?
* Arrays
  * Lister over variable, refereret til ved indeks
  * Nogle af jer har forsøgt sig med char array AKA strings
* Anderledes datarepræsentation:
  * typedef og enum
  * Klynger af data i structs
  * Flerformet data i unions

* Vi har ofte brugt funktioner:
  * pow(), sqrt(), rand(), printf() og scanf()
* De ligner alle i skrift lidt matematiske funktioner:

$$printf(x) \tag{1}$$

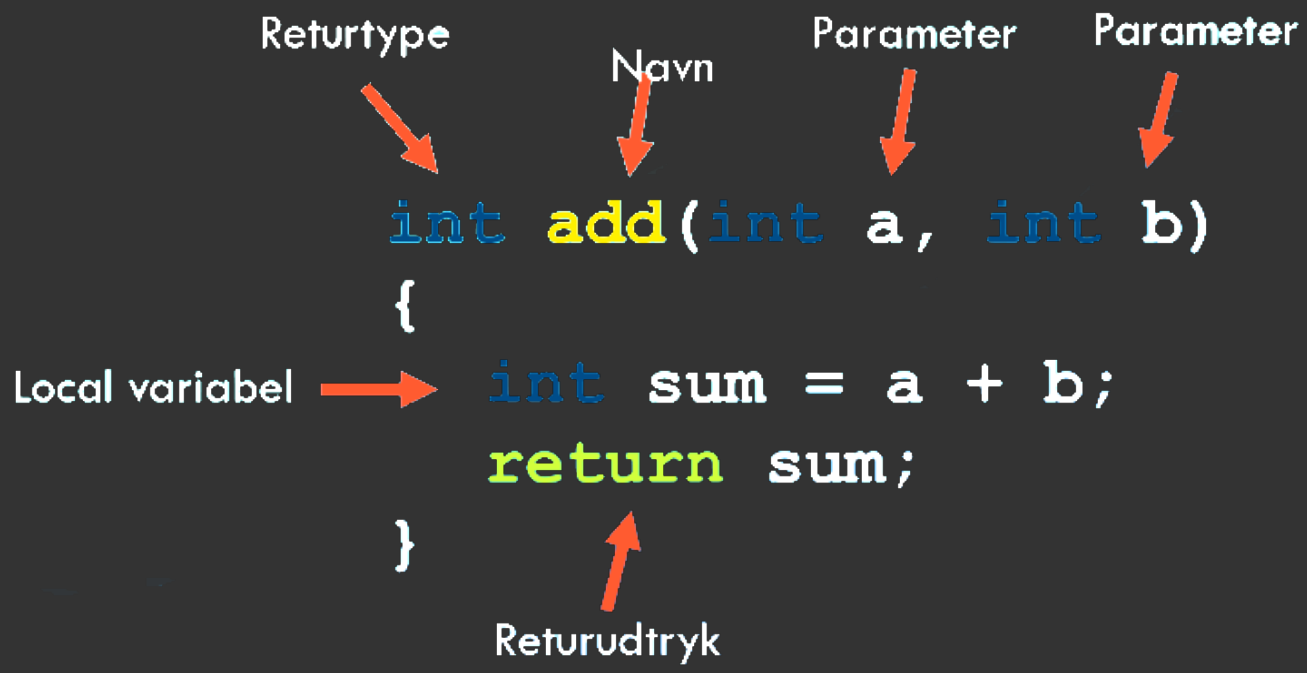
* Og vi kan selvfølgelig også designe vores egne!
* Vi husker tydeligt en funktionsdefinition, vi har med hver gang:

```
1     int main(int argc, char ** argv){
2        return 0;
3     }
```

>>> Funktioner

* I funktionerne er der en række begrber vi skal genopfriske/kende:
  * returtype, argumenttype/parameter, returværdi og lokale variable:

```
add(a, b);
```



* Så først defineres funktionen, og derefter kan den kaldes som alt andet i main:

```
1       int add( int a, int b ){
2          int result = a + b;
3          return result;
4       }
5
6       int main(void){
7          int a = 3;
8          int b = 2;
9          printf(  %d + %d = % d , a, b, add(a,b));
10      }
```

* Noget andet fedt man kan med sine funktioner, er at kalde dem rekursivt:

```
1       int rekursion(int antalGange){
2          if(antalGange < 0){
3          return 0;
4          }
5          printf(   d   , rekursion(antalGange-1));
6          return 0;
7       }
```

>>> 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!