**IMT1031 – Funksjoner**Dato: 28.09.2018 – 03.10.2018  
Foreleser: Frode Haug

**Stoff:**

* **Deklarere og Definere Funksjoner**
  + Deklarere
  + Definere
* **Parametere**
  + Parametere/Argumenter
* **Return**
* **Referanse**
* **Inline Funksjoner**
* **Overloading**
* **Default Parametre**

**Deklarere og Definere funksjoner**

**Deklarere**

void funksjonsNavn(); //Deklarert

Vi skal deretter definere funksjonen senere i koden, f.eks etter main funksjonen. Det beste er å skrive etter main, dette gjør koden mer lesbar.

**Definere**

void funksjonsNavn()

{

//*Kode*

}

**Parametere**

Parametere skal legges inn i () etter funksjonsnavnet, disse deklareres men funksjonen i starten.

I eksemplet til Frode skrev han ikke variablenavn til parameterne:

*void repchar(char, int);*

Variabelnavnene gis når funksjonen defineres senere, datatypene er bare for å fortelle programmet hvilke datatyper som kommer senere.

Definering av parametere

*void repchar(char ch, int n)*

*{*

*//Kode som kan bruke variable ch og n*

*}*

I ettertid sa Frode at du kan gi navn til variablene i starten også, men det trengs ikke, så det er ok å kopiere funksjonen når den defineres og legge den i starten sånn at den er deklarert.

**Parameter/Argumenter**

= data overført fra kallstedet til funksjonen

= verdier/data som er input til en funksjon

**verdioverførte** parametrene er det samme som definisjonen av dem og initialiseringen til de medsendte verdier.

**Reference**

void function(float, float&, float&) s.182-183

ved å legge til & sier vi til programmet at original variabelen skall oppdateres direkte.

**Overloading**

Når ulike funksjoner har samme navn, men forskjellige parametere

**Scope og Storage Class:**

Scope, variabelens gyldighetsområde/synlighet og initialisering

To regler:

En variabel er synlig:

1. Fra det sted den er definert inni en {}-blokk, og ut blokken.
2. Fra det sted den er definert i en fil, og alle de fysisk etterfølgende funksjoner/blokker.

Regel 1 omhandler **lokale** (automatic) variable:

* Levetid: fram til avsluttende ‘}’
* Synlighet/scope: regel 1
* Ingen initialisering

**Static**

Eks:

void x(int data) {

static int count = 0;

count ++;

}

Static kjører koden bare 1 gang i programmet, dersom vi kjører funksjonen igjen vil ikke koden back static leses.