Screencast

Variabler och tilldelning

Sammanfattning

- I matematiken (och i den rena funktionella programmeringen) binds ett variabel till ett värde som den sedan behåller (Vi kan implementera sådana variabler med substitution)
- I (de flesta) imperativa programspråk kan en variabels värde variera under programmets körning
- Vi kan inte längre implementera variabler med substitution eftersom ett värde på en variabel potentiellt påverkas av ”allt som sker före läsningen av dess värde”

- C är statiskt typat, det innebär (bl.a.) att alla variabler har en/måste ha en typ vid kompilering
- En variabels typ styr vilka värden som kan lagras i den
- Vid tilldelning mellan variabler måste typer respekteras
- C tillåter viss implicit konvertering mellan värden av olika typ

– En funktions (formella) parametrar är också variabler; i C kan dessa variabler också tilldelas och ändras

– Vid anrop av en funktion kopieras argumenten och de kopierade värdena sparas i ”parametervariablerna” hos funktionen

```
1   int max(int a, int b) {
2     return (a < b) ? b : a;
3   }
```

– Varje funktionsanrop har sin egen kopia av funktionens variabler

```
1   void swap(int c, int d) {
2     int temp = c;
3     c = d;
4     d = temp;
5   }
6
7   int x = 7;
8   int y = 0;
9   swap(x,y);
10  // vad är värdet på x nu?
```

– I C är varje variabel synlig endast i ett visst ”scope” (räckvidd). Till exempel kan a bara läsas i max och c bara läsas i swap och x bara läsas i main i programmet vi skrev.

– Variablers namn är oerhört viktiga. Programmet vi skrev var ett nonsensprogram och variablernas namn var också nonsens. Ett bra namn på en variabel är t.ex. `vikt_i_ton` eller `firstName` eftersom dessa namn beskriver vad man kan föräntas finna i variablerna samt hur man skall tolka dem (t.ex. ton kontra kg).

– Inkremeteringsoperatorerna `++` (och `--`) ökar (respektive minskar) en variabels värde med 1.

```
1    x++ // ökar x's värde med 1 men returnerar det ursprunliga värdet
2    ++x // ökar x's värde med 1 och returnerar resultatet
```

  Tips: använd alltid `++x` för enklare resonering

– Vanliga operationer som `x = x + y` kan skrivas `x += y`. Detta fungerar med många olika binära operatorer.