Rekursion

Rekursiv kommer från latinet och betyder *återlöpande*. Om man i definitionen av ett begrepp använder begreppet självt så är definitionen rekursiv. Rekursiva tankar kan också användas för problemlösning.

- Rekursiv tanke: reducerar problemet till ett enklare problem med samma struktur
- Basfall: det måste finnas ett fall som inte leder till rekursivt anrop

Exempel

```
Triangeltalet triNumber(n) är summan av de n första heltalen. triNumber(4)=1+2+3+4
```

Fråga: Vad är värdet på triNumber(n)?

```
Rekursivt svar: triNumber(n) = triNumber(n-1) + n ... men <math>triNumber(1)=1.
```

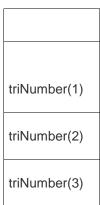
Här följer en rekursiv metod för beräkning av triangeltalen:

```
public int triNumber(int n){
   if(n==1)
     return 1;
   else
     return triNumber(n-1)+n;
}
```

Hur fungerar det?

När man skriver egna rekursiva funktioner bör man lita på att det rekursiva anropet fungerar - man behöver inte analysera anropsgången för varje fall. Men för att förstå varför rekursion kan vara extra minneskrävande är det vara bra att känna till hur programspråken hanterar rekursiva anrop.

- För varje anrop skapas en *aktiveringspost* som innehåller data för anropet, t ex parametrar, lokala variabler och anropspunkt.
- Aktiveringsposten pushas på en stack.
- När det rekursiva anropet är klart poppas aktiveringsposten från stacken, varefter föregående anrop ligger överst på stacken.





- En rekursiv funktion kan alltid omformuleras utan rekursion, men om det finns flera rekursiva anrop i funktionen kan det vara besvärligt.
- För många problem är en rekursiv funktion mycket enklare att formulera och ger kortare kod än utan rekursion. Ofta måste man gå via den rekursiva lösningen i tanken även om man gör en icke-rekursiv lösning.

Dålig rekursionslösning

Ett klassiskt exempel på en dålig rekursionslösning är följande implementation av Fibonaccitalen (där varje tal är summan av de två föregående Fibonaccitalen).

```
int fibonacci(int n) {
    if(n == 0)
        return 0;
    else if(n == 1)
        return 1;
    else
    return fibonacci(n - 1) + fibonacci(n - 2);
}
```

Uppgifter

Fråga: Vilken siffersumma har heltalet n?

Rekursivt svar: Sista siffran plus siffersumman om man stryker sista siffran i n, ...men noll har siffersumman noll. Skriv en rekursiv metod som beräknar siffersumman i ett givet tal n.

Exempelvis har 123 siffersumman 6 (1+2+3) och 9999 har siffersumman 36 (9+9+9+9)

Fråga: Hur många siffror har det positiva heltalet n?

Rekursivt svar: En siffra mer än om man stryker sista siffran i n, ...men tal mindre än tio är ensiffriga. Skriv en rekursiv metod som beräknar antalet siffror i ett givet heltal n.