

Lab 5: Rekursion

I denna laboration kommer vi att titta lite närmare på hur rekursion fungerar.

Mål

Du ska efter denna laboration kunna:

- förstå hur rekursion går till
- använda rekursion
- förstå när man skall undvika rekursion
- veta vad begreppet ändrekursion betyder
- använda iteration i kombination med rekursion

Tonvikt läggs på:

- struktureringen av problemet
- läsbarhet av programkod

Uppgift 1 (frivillig):

Du skall skriva den rekursiva implementationen av funktionen som har deklaration enligt följande:

```
function Factorial(N : in Natural) return Positive;
```

Funktionen skall beräkna $N!$ utifrån det N som kommer in som parameter. För att visa att funktionen fungerar krävs förstas ett huvudprogram som anropar funktionen.

Uppgift 2 (frivillig):

Du skall skriva de rekursiva implementationen av funktionen som har deklaration enligt följande:

```
function Power(X : in Float;  
              N : in Integer) return Float;
```

Funktionen skall beräkna X^N .

Uppgift 3:

Du skall skriva en funktion som beräknar det N :te talet i Fibonacci-serien. Indata till funktionen skall vara N och funktionen skall givetvis vara rekursiv. Funktionen skall heta `Fib`.

Definition av Fibonacci-serien:

```
Fib(1) = 1  
Fib(2) = 1  
Fib(N) = Fib(N - 1) + Fib(N - 2)
```

Serien börjar alltså som följer:

```
1 1 2 3 5 8 13 21 34 ...
```

Testa med ett stort tal på N och se vad som händer. Varför blir det så?

Uppgift 4

I Ada finns en procedur som heter `Get_Line` som läser in en ”rad” från tangentbordet och lagrar i en sträng. Om raden är kortare än strängens längd kommer radslutstecknet (vagnreturtecknet) att tas bort från inmatningsbufferten och i annat fall lämnas det som inte får plats i strängen kvar i bufferten.

I Ada-boken finns en mer fullständig beskrivning av hur `Get_Line` skall bete sig. Din uppgift i denna uppgift är att skapa denna `Get_Line` (och lägga den i en separat fil som heter ”`get_line.adb`”) med kravet att din `Get_Line` skall vara rekursiv. Detta innebär att du alltså INTE skall använda dig av någon form av iterativa loopar (`for`, `while` eller `loop`) i denna uppgift.

För att lösa detta måste man använda sig av en funktion som heter ”`End_Of_Line`” (i `Ada.Text_IO`) som kollar i inmatningsbufferten om nästa tecken är ett radslutstecken. Funktionen tar INTE bort något ur bufferten utan kontrollerar endast om det är ett radslut eller ej. Funktionen returnerar sant eller falskt (`True / False`) och kan därför anropas i ett villkor.

Den ordinarie `Get_Line` stoppar in de nya tecknen i den sträng som kommer som parameter, MEN den ändrar inte på de tecken som ligger i strängens sista del (efter den del som ersätts av nya inmatningen). Den ordinarie `Get_Line` har endast ”`out`” på strängvariabeln, men detta kan inte lösas på det sättet i Ada så din `Get_Line` får ha både ”`in`” och ”`out`” för att kunna bete sig på rätt sätt.

Den andra parametern till `Get_Line` heter `Last` och denna returnerar det index i strängen där det sist inlästa tecknet lagrats. Om det var en rad med endast ett radslutstecken kommer inget att läsas in (dock försvinner radslutstecknet från bufferten) i strängen och för att markera detta sätts `Last` till värdet som motsvarar första indexet i strängen minus ett (d.v.s. ett mindre än första indexet). Om strängen är definierad som ”`String(1 .. 5)`” kommer alltså `Last` att få värdet 0.

För att testa din `Get_Line` finns ett givet huvudprogram på kurshemsidan. Detta heter ”`Test_Get_Line`”. Om man kör detta program skall följande körexempel bli resultaten.

Körexempel 1:

Mata in en sträng (maximalt 5 tecken lång): **Bo**

Du skrev in strängen 'Bo' som var 2 tecken lång.

Mata in en sträng (maximalt 5 tecken lång): **Vera**

Du skrev in strängen 'Vera' som var 4 tecken lång.

Körexempel 2 (man hinner inte mata in något efter andra frågan):

Mata in en sträng (maximalt 5 tecken lång): ***Stina***

Du skrev in strängen 'Stina' som var 5 tecken lång.

Mata in en sträng (maximalt 5 tecken lång): Du skrev in strängen '' som var 0 tecken lång.

Körexempel 3 (man hinner inte heller här mata in något efter andra frågan):

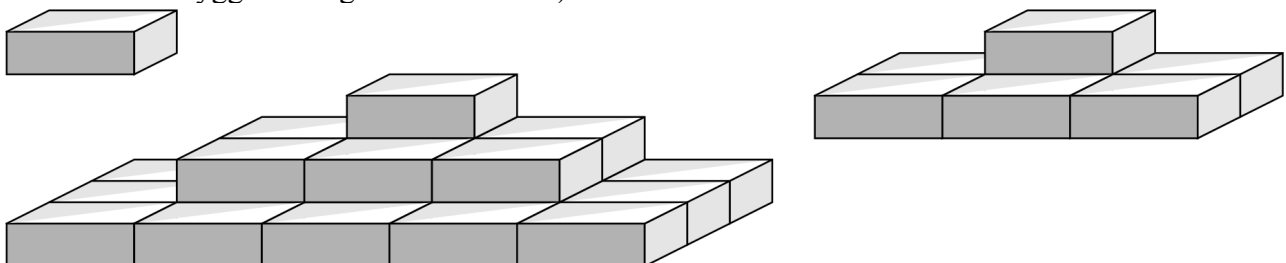
Mata in en sträng (maximalt 5 tecken lång): ***Torbjorn***

Du skrev in strängen 'Torbj' som var 5 tecken lång.

Mata in en sträng (maximalt 5 tecken lång): Du skrev in strängen 'orn' som var 3 tecken lång.

Uppgift 5

När man bygger sitt hus och kommer till den punkt där man skall bygga taket får man lite problem. Man behöver någon form av ställning. Antag att man bygger den av LECA-stenar på så sätt att man kan se det som en trapp oavsett från vilket håll man kommer (lite grann som en kapad pyramid). Se i figuren för att se hur trappen skulle se ut för hushöjderna 2, 3 och 4 (självkärl behövs ingen trappa om man bara bygger ett lager stenar i huset).



Din uppgift är att skriva det program som räknar ut hur många stenar det behövs för att bygga "trappan" givet en viss höjd på huset.

KRAV: Du skall ha en rekursiv funktion som beräknar antalet stenar i ditt program.

Körexempel 1:

Mata in husets höjd (HH = antal stenar i höjddled): 10

Det behövs 525 stenar för att bygga trappan.

Körexempel 2:

Mata in husets höjd (HH = antal stenar i höjddled): 100

Det behövs 651750 stenar för att bygga trappan

Uppgift 6 (frivillig)

Skriv ett program som läser in ett antal rader med tre tecken långa strängar. När inmatningen "---" kommer skall alla tidigare inmatningar skrivas ut i omvänd ordning. Det kan vara godtyckligt många rader att mata in (vilket gör att du inte kan lagra dem i ett fördimensionerat fält).

Körexempel:

Mata in ett antal rader (3 tecken per rad, avsluta med "---"):

aaa

bbb

ccc

Du matade in följande rader:

ccc

bbb

aaa