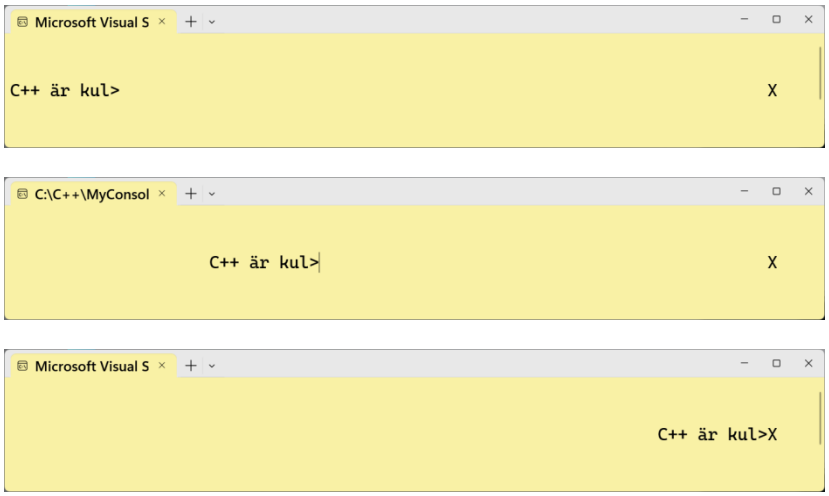
Välkomna alla SUVx-23-are till ***Boiler room fre 1/12, kl 9-12.***

Idag presenteras följande två uppdrag till klassen för dagens Boiler room:

1. Löpande texten – en animation i konsolen

Skriv en C++ Console Application som simulerar en löpande text. Ta som exempel texten **`C++ är kul>`** som ska röra sig horisontellt från konsolfönstrets vänstra kant tills den ”träffar” på ett hinder, t.ex. ett kryss **`X`**. Texten ska börja från vänstra kanten. Krysset ska ligga nära den högra kanten (ca. 70-80 tecken borta). Dessa ögonblicksbilder ska illustrera animationen:



Ledning:

Skriv ut först krysset **`X`** i slutet av en tom rad (fylld med mellanslag). Anteckna hur många mellanslag ni har valt för att placera krysset från konsolens vänstra kant. Gå i samma rad tillbaka till radens början genom att använda escapesekvensen **`\r`** (carriage return). **`\r`** skickar tillbaka markören till början av samma rad, utan att byta rad (till skillnad från **`\n`**). Skriv sedan ut **`C++ är kul>`** som då blir textens initialposition – det som visas i den första ögonblicksbilden ovan. Om ni vill bekanta er mer med **`\r`**:s funktion gör experiment med det i ett annat program.

Rörelsen kan sedan simuleras t.ex. i en **`for`**-loop genom att i varje varv av loopen med ett antal **`\b`** ta bort texten som skrevs ut i förra varvet. Escapesekvensen **`\b`** (backspace) tar bort *ett* tecken till vänster om det aktuella tecknet, precis som tangenten backspace (←). Stega sedan med ett (eller flera) mellanslag, vilket kommer att bestämma rörelsens ”hastighet”. Skriv slutligen om texten **`C++ är kul>`**.

Beräkna antalet varv i **`for`**-loopen genom att ta hänsyn till textens längd och avståndet som kryss **`X`** har från vänstra kanten (som antecknats ovan). Har ni räknat rätt, kommer rörelsen att stoppas strax före krysset **`X`**, utan att ta bort det – liknande den tredje ögonblicksbilden ovan.

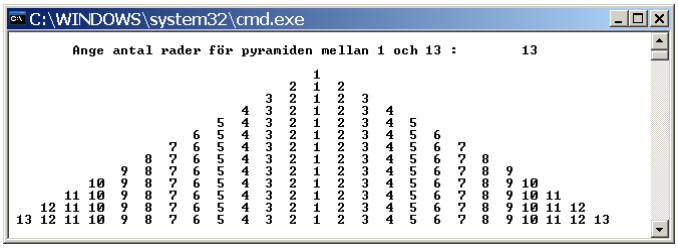
Även om ni gjort allt rätt kommer ni inte ”se” texten att röra sig, eftersom det går så fort, så att ögat inte hinner att se förloppet. Ni måste lägga in en fördröjning, vilket kan göras genom att infoga i loopen t.ex. satsen:

```
Sleep(100);
```

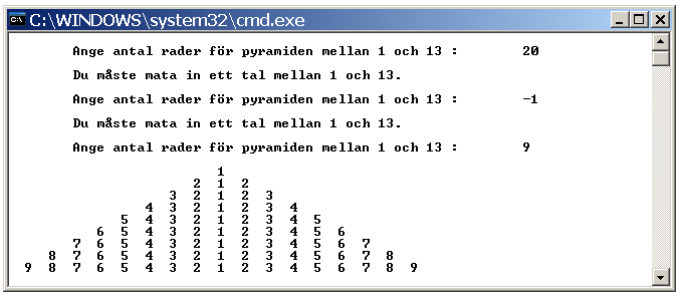
Parameterns enhet är millisekunder. Fördröjningsfunktionen **`Sleep()`** kräver inkluderingen av biblioteket **`windows.h`**.

2. Pyramiden

Slutmålet med detta uppdrag är att utveckla ett program som skriver ut en pyramidliknande figur med tal, som t.ex. ser ut så här:



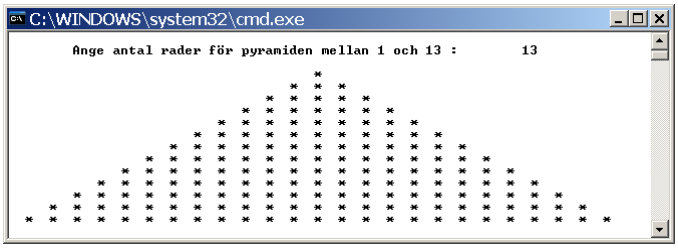
Programmet ska vara så generellt att det skriver ut talpyramider även om man matar in mindre antal rader. Uppmana användaren att hålla sig till talintervallet [1, 13]. Annars ryms talpyramiden inte i konsolen. Så här kan en körning se ut:



Ledning:

Denna ledning är endast en rekommendation och ska inte förhindra att ni använder egna idéer för att lösa problemet. Det finns andra möjliga tillvägagångssätt. Ni kan använda hela eller också delar av denna ledning för att komma igång.

Man kan *börja* med ett program som ritar en pyramid av *stjärnor* istället för tal:



Strunta till att börja med även på hanteringen av felinmatning av antal rader. Jobba med ett fast antal rader. Du kan lägga till det senare.

Använd en nästlad for-sats med en yttre loop och tre inre loopar:

- En för de tomma platserna i pyramiden (mellanslagen),
- En för stjärnorna i pyramidens högra halvan (räknat från den vertikala mittlinjen (symmetriaxeln),
- En för stjärnorna i pyramidens vänstra halvan.

Räkna med att ni måste använda i de inre looparna den yttre loopens räknare och slutvärde. T.ex. kan villkoret i den första inre loop som ritar de tomma platserna, se ut så här:

```
column <= numberOfRows - row;
```

Här är **`column`** den inre loopens, **`row`** den yttre loopens räknare och **`numberOfRows`** hela pyramidens antal rader, t.ex. 13. Då kan den första inre loopen skriva ut tre mellanslag i varje varv. I de två andra inre looparna kan två mellanslag och en **`*`** skrivas ut.

Lycka till!

Hälsningar

Taifun