Övningar till kapitel 1

1.1 Skapa en Console Application och kalla den för AdditionC. Den ska definiera och initiera två heltalsvariabler och producera t.ex. följande utskrift till konsolen:

```
Summan av 9 och 2 är 11
```

9 och **2** ska vara de värden som variablerna blivit inirierade till i programmet.

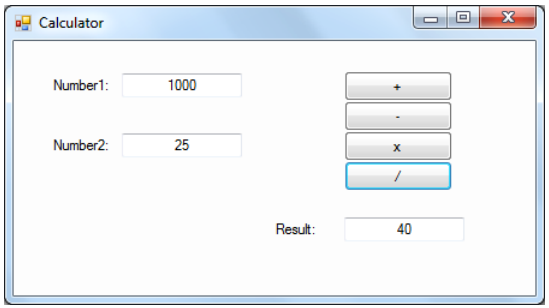
1.2 Skapa en Windows Forms Application och kalla den för AdditionW. Den ska göra samma sak som lösningen i övning 1.1, bara att utskriften inte hamnar i konsolen utan i en MessageBox och visas när man klickar på en knapp (med texten *Visa MessageBox*) i formfönstret. Förse MessageBoxen med rubriken *Windows Addition*.

1.3 I både övn 1.1 och 1.2 är heltalsvärdena 9 och 2 hårdkodade. Vidareutveckla dessa övningar genom att skapa ett användarvänligt, interaktivt grafiskt gränssnitt där man kan mata in vilka tal som helst och få summan utskriven i en MessageBox när man klickar på en knapp med texten *Addera*. Välj lämpliga rubriker för formen och MessageBoxen. Kalla projektet för Addition.

1.4 Skapa en Windows Forms Application och kalla den Division. Modifiera lösningen i övn 1.3 så att beräkningens resultat inte skrivs ut till en MessageBox utan placeras i ett textfält som läggs till i formen. Välj den här gången division som räkneoperation.

1.5 Skapa en Windows Forms Application och kalla den för SafeDivision. Skapa samma grafiska gränssnitt som i projektet Division (övn 1.4). Applikationen ska genomföra säker division, dvs ta hand om en eventuell division med 0. Modifiera koden i Form1.cs genom att införa ett *egengenererat undantag* för fallet att användaren matar in 0 i det andra textfältet. Styr meddelandena från undantagshanteringen till en MessageBox.

1.6 Vidareutveckla övningsserien 1.1-1.5 till en komplett kalkylator som inkluderar de fyra räknesätten. Det grafiska gränssnittet kan se ut som bilden till höger. Förse divisionen med en undantagshantering (sid 28) som vid division med 0 skriver ut ett felmeddelande till en MessageBox.



1.7 ***Grafiska applikationer (projekt)*** Gå igenom dina konsolapplikationer som du skrivit hittills. Undersök vilka av dem som är lämpliga för att skriva om dem till grafiska applikationer. Integrera all inläsning från och utskrift till konsolen helt och hållet i en grafisk miljö. OBS! En befintlig konsolapplikation kan inte laddas i Visual Studio och göras om till en Windows Forms Application. Man måste skapa en ny Windows Forms Application och förse den både med grafik och kod som gör samma sak som den ursprungliga konsolapplikationen. Skillnaden är bara att användaren kommunicerar med programmet via ett grafiskt gränssnitt istället för via konsolen. Har du inga konsolapplikationer fortsätt här.

De projektuppgifter som nu följer är konsolapplikationer. Repetera hur man skapar en C# Console Application i ***Appendix***, sid 273.

1.8 *Gissa tal – ett spel (projekt)*

Skriv en Console Application som slumpmässigt genererar ett heltal mellan 1 och 100. Låt användaren i flera försök gissa detta hemliga tal. För att stödja gissningen, låt programmet efter varje gissningsförsök skriva ut, om det gissade talet var mindre eller större än programmets hemliga slumptal. Låt användaren försöka igen. Gissningen ska pågå tills man gissat rätt. Vid rätt gissning skriv ut ett ”Grattis!”-meddelande följt av ett datorljud, t.ex. med `\a` . Förse programmet med ytterligare två funktionaliteter:

a) Vid rätt gissning skriv även ut ett meddelande om antalet gissningsförsök.
b) Ge möjligheten att avsluta spelet och få reda på programmets hemliga tal, vilket kan ske genom att mata in t.ex. `0` .

Ett exempel på en omgång av Gissa tal-spelet kan se ut så här:

```
Gissa ett heltal mellan 1 och 100 (Avsluta med 0):      50

                För LITET, försök igen!                 75

                För LITET, försök igen!                 87

                För STORT, försök igen!                 81

                För LITET, försök igen!                 84

                Grattis, du har gissat rätt efter 4 försök.
```

Eller om spelaren blivit trött och vill avluta genom att mata in t.ex. `0` :

```
Gissa ett heltal mellan 1 och 100 (Avsluta med 0):      0

                Avbrott: Programmets hemliga tal var    66
```

Ledning:

a) Hantering av slumptal i C#:
För att slumpmässigt generera ett heltal mellan 1 och 100 kan man skriva:

```
Random r = new Random();
int secret = r.Next(1, 101);
```

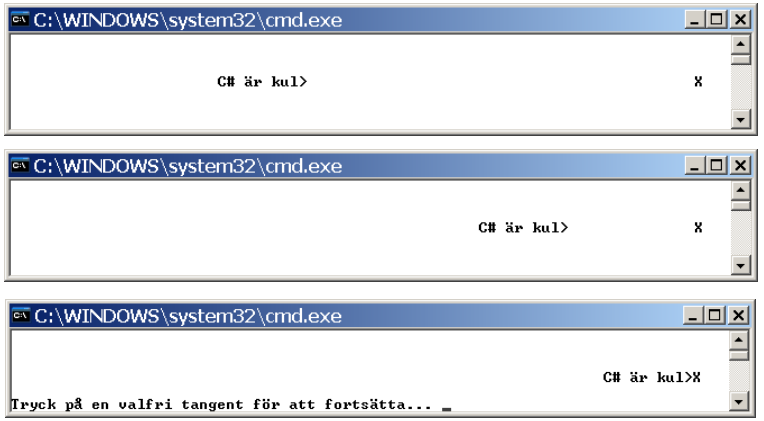
Första raden skapar ett objekt av klassen **`Random`**. Variabeln **`r`** av typ **`Random`** refererar till detta objekt. I den andra raden anropas metoden **`Next()`** som är definierad i klassen **`Random`**. Därför måste anropet ske med referensvariabeln **`r`** via punktnotation. Parametrarna **`1`** och **`101`** bestämmer att metoden returnerar ett heltal mellan 1 och 100. Returvärdet tildelas heltalsvariabeln **`secret`**, programmets hemliga slumptal.

b) Resten består huvudsakligen av en loop som tillåter spelaren gissa upprepade gånger. I loopen kan en kontrollstruktur användas som är lämplig för flervägsval, för att skilja mellan de olika alternativen. Du kan styra loopens förlopp samt avslutning t.ex. med en logisk variabel av typ **`bool`**.

Extrauppgift:

Fundera och testa på en *spelstrategi* som kan minimera antalet gissningsförsök. I körexemplet ovan med bara 4 försök har en sådan strategi använts. Hur skulle du beskriva den?

1.9 ***Löpande texten (projekt)*** Skriv ett program som simulerar en löpande text, t.ex.: **`C# är kul>`** som horisontellt rör sig i konsolfönstret tills den ”träffar” på ett hinder, t.ex. ett kryss i form av ett **`X`**. Så här kan ett körresultat se ut:



Ledning:

a) Skriv ut med hjälp av ett antal mellanslag krysset i slutet av en rad i konsolen utan radbyte. Anteckna antalet mellanslag krysset har avstånd från konsolens vänstra rand. Stanna på samma rad, gå med hjälp av escapesekvensen **`\r`** (car-

riage return) till början av raden och skriv ut texten **`C# är kul>`**. Gör experiment med **`\r`** för att bekanta dig med dess funktion. Så här borde ett körresultat se ut:

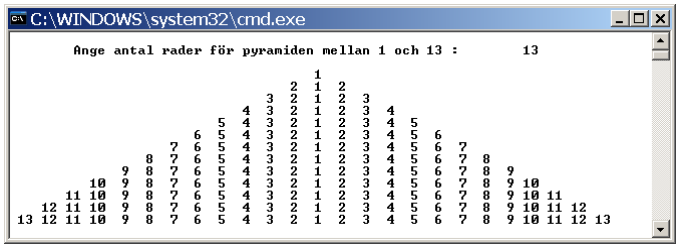
b) Skriv en **`for`**-loop. Ta bort (dvs skriv ut) i varje varv av loopen med 10 styck **`\b`** texten **`C# är kul>`** som ritats i förra varvet (initialt texten ovan), följt av ett eller flera mellanslag (vilket påverkar rörelsens ”hastighet”). Skriv sedan om texten **`C# är kul>`**. Välj som antal varv i loopen kryssets avstånd från konsolens vänstra rand (antecknat i a) minus textens längd – i det föreslagna exemplet 10. Då kommer rörelsen att stoppas strax innan texten ”träffar” på **`X`**.

Även om du gjort allt rätt kommer du inte se texten att röra sig om du inte lägger in en fördröjning i loopen, eftersom allt går så fort och ögat inte hinner se något förlopp. Fördröjningen kan du åstadkomma genom att lägga in i loopen satsen:

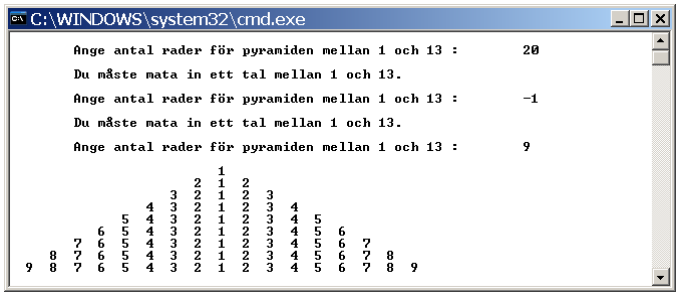
```
System.Threading.Thread.Sleep(100);
```

Detta ger en fördröjning på 100 milisekunder i varje varv av loopen.

1.10 ***Pyramiden (projekt)*** Slutmålet med detta projekt är att utveckla ett program som skriver ut en pyramidliknande figur med tal, t.ex. så här:

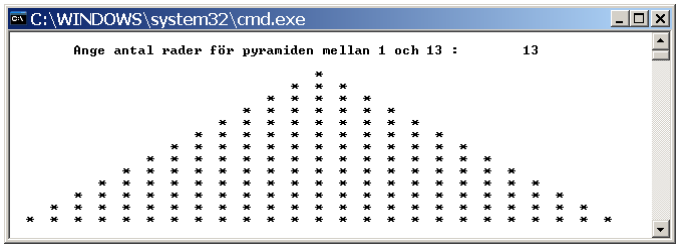


Programmet ska vara så generellt att det skriver ut talpyramider även om man matar in mindre antal rader. Men om användaren inte följer ledtextens instruktion att mata in tal mellan 1 och 13 ska programmet inte skriva ut talpyramiden utan uppmana användaren att hålla sig till det föreskrivna talintervallet [1, 13]. Anledning till denna restriktion är att talpyramiden inte ryms i konsolen om man överskrider detta intervall . Så här kan då en dialog t.ex. se ut:



Tips till Pyramiden:

För att komma igång med talpyramiden, börja med att skriva ett program som ritar en stjärnpyramid:



Strunta till att börja med även på hanteringen av felinmatning av antal rader och 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 du 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;
```

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

Observera att alla dessa tips inte ska förhindra att du använder dina egna idéer för att lösa projektuppgiften. Det finns inte endast ett tillvägagångssätt. Uppgiften kan lösas på väldigt många olika sätt.