Observera att det även här sker en implicit typomvandling från `int` till `double`, eftersom det är vad metoden skall returnera.

Exekveringen fortsätter från anropspunkten, dvs att variabeln `y` nu tilldelas värdet 98, men eftersom `y` är av typen `char` (som är mindre än en `double`) måste vi göra en explicit typomvandling från `double`, när vi tar hand om detta returvärde:

```
char y = (char) Test('a');
```

När vi sedan skriver ut `y`, är det således en `char` med Unicode-värdet 98 som kommer att skrivas ut: 'b'!

```
System.Console.WriteLine( y );  // y har värdet 'b'
```

I nästa anropsexempel skickar vi helt enkelt in en `int`. Observera att anropet i sig ”får samma värde” som vad metoden returnerar, vilket gör att vi även kan använda metodanrop som delar av uttryck:

```
System.Console.WriteLine( Test(72) + 2 );
```

I ovanstående anropsexempel skickar vi in 72, vilket metoden räknar upp till 73, varefter vi adderar ytterligare 2, så det som skrivs ut blir: 75

4.10.3 Metodnamn, signaturer och överbelastning

Metodnamn följer samma regler som variabelnamn, dvs att vi kan använda alla bokstäver, siffror och understrykningstecknet. Praxis säger att metodnamn i C# skall börja med stor bokstav, medan andra språk har andra konventioner för namnsättning.

Metodnamnet, tillsammans med de formella parametrarna och returtypen, kallas för metodens **signatur**.

Du kan i ett och samma C#-program ha flera metoder med samma namn, så länge deras signaturer skiljer sig åt. Poängen med detta är att Du kan vilja ha metoder som i grunden är avsedda för samma ändamål, men att Du kan vilja anropa dem med andra parametertyper eller ett annat antal parametrar vid olika tillfällen.

Denna teknik kallas för att överbelasta metoden. Ett typiskt sådant exempel är `WriteLine` som vi använder vid utskrift mot skärmen. Där finns ett stort antal överbelastade metoder för `WriteLine` så Du i princip skall kunna skicka in vad som helst, och det skall ändå komma ut på skärmen som text.